AI Personal CFO

Rekhon

Software Project Blueprint

An offline-first, privacy-first personal finance app for India
that computes every figure on the device, and shows its working

Prepared for	Harish G — project owner and sole maintainer
Date	23 October 2026
Version examined	v0.13.10
Status	Draft
Readiness verdict	**READY WITH RISKS** — 0 blocking questions, 0 critical gaps
Platform	Android, Kotlin, Jetpack Compose; 60 Gradle modules
Storage	Room over SQLCipher, key in the Android Keystore
Deployment	One APK. No server holds user data
Documents	12 linked parts, 5 diagrams
Produced by	`software-project-blueprint` skill

This blueprint is documentation, not application code. It is written from the project's own sources — the SRS, 82 decision records, and the shipped tree at v0.13.10 — so each requirement carries what was actually built against it. Anything concluded rather than read is marked as an assumption, every missing link is a numbered gap rather than an omission, and the readiness verdict is defended from the checklist in Part 11 rather than asserted.

Generated from `docs/project-blueprint/` by `scripts/build_blueprint_pdf.py`. The Markdown is the source of truth; this PDF is built output.

Contents

Part 00 — Master Blueprint **2**

Part 01 — Project Overview and Scope **6**

Part 02 — Requirements and Acceptance Criteria **13**

Part 03 — Business Flows and Use Cases **45**

Part 04 — Data Dictionary and Database Design **106**

Part 05 — Application Architecture **135**

Part 06 — API Specification **167**

Part 07 — Test Strategy and Cases **185**

Part 08 — Development and Delivery Plan **196**

Part 09 — Risks, Decisions, Assumptions and Open Questions **207**

Part 10 — Traceability Matrix **215**

Part 11 — Completeness Checklist **225**

Appendix A — Architecture and Gap Report **229**

Part 00 — Master Blueprint

Summaries only. Every detail lives in a linked document, and a fact lives in exactly one place. ***What this blueprint is.*** *An as-built specification of a product that already exists, written from the project's own sources — the SRS, 80 ADRs,* `README.md`*,* `CLAUDE.md` *and the shipped code at* `v0.13.10` *— rather than from an interview. Where the SRS and the code disagree, the code plus its ADR is recorded as the current truth and the disagreement is named (*`ASM-002`*).*

1. In one paragraph

An Android app that acts as a full-time CFO for one person or household in India, and runs **entirely on the phone**. It records money in three taps, reads receipts and (with explicit permission) bank alerts on-device, classifies every rupee, and then does the work a ledger does not: forecasts the next 90 days and names the days the money runs short, sizes an emergency fund from real essential spend, computes what a goal needs each month, splits an EMI to the paise, measures a holding's money-weighted return, decides where the next surplus rupee should go, and answers "can I afford this?" with the gate-by-gate trace that produced the verdict. Thirty deterministic engines compute every figure; a language model may only put those figures into sentences, and a guardrail refuses any sentence containing a number the engines did not produce. There is no account, no sign-up and no server holding user data: the database is SQLCipher-encrypted under a key in the Android Keystore, consent is per-feature and revocable, and the source is public so the privacy claim can be checked rather than believed.

2. Problem and users

- **Problem.** Money is managed reactively: the month-end balance is a surprise, the picture is fragmented across SMS, UPI, cards, receipts, gold and cash, advice is expensive or conflicted, and users will not hand bank credentials to a server. Detail: 01 §1.
- **Users.** One person, on one device — the four SRS personas, from a first-salary professional to a cash-heavy shop owner — plus an optional household member or business book as a second profile. `STK-001`…`STK-007` in 01 §3.
- **Today's workaround and its cost.** Mental arithmetic and the bank's SMS balance; a spreadsheet reconciled monthly; a ledger app that records but cannot answer a question.

3. Success measures

ID	Goal	Measure	Target
GOAL-001	Be the single source of financial truth	Share of monthly spend captured	$\geq$ 90% of spend for $\geq$ 80% of active users by month 3
GOAL-002	Deliver advice users act on	Recommendations followed or marked helpful	$\geq$ 40%
GOAL-003	Form a habit	Day-30 retention; sessions per week	$\geq$ 35%; median $\geq$ 4
GOAL-004	Earn trust through transparency	Trust score; data complaints	$\geq$ 4.5; $<$ 1%
GOAL-005	Sustain the product after v1	Freemium conversion	**No target yet** — `OQ-006`
GOAL-006	Keep support cost low	Self-serve resolution	$\geq$ 90%
GOAL-007	Make the privacy claim checkable	Source published under an inspectable licence	AGPL-3.0 with a linking exception

Detail, with the SRS mapping: 01 §2.

4. Scope

In. Capture (manual, OCR, opt-in SMS, import, recurring) · eleven account types and daily net worth · categorisation by merchant, by nature and by stream · budgets with pace, projection and alerts · goals, an emergency fund and an eight-stage surplus waterfall · cards, loans, holdings with XIRR, allocation and concentration · vehicles and appliances with service prediction · a 90-day forecast with crunch days · a health score, a ranked insight feed and a notification policy · a purchase advisor with a kept trace · simulators, tax, insurance and market signals · a chat assistant behind a numeric guardrail · app lock, privacy blur, consents, export/import, E2EE backup, crypto-shredding erase, a widget, four languages, household and business profiles.

Out. Moving money (`P-07`) · bank aggregation by credentials (`P-01`) · Account Aggregator (designed, unbuilt —

OQ-001) · regulated investment advice (not SEBI-registered) · a cloud service holding plaintext data · social features, ads or data resale · iOS (blocked on Money — OQ-002) · tax filing, insurance sales, loan origination · children's data. Reasons for each: 01 §6.

5. Architecture in three sentences

A single Android APK holds 60 Gradle modules whose dependencies run one way — feature → domain → data/core — with every engine in pure Kotlin and no Android imports, so each is testable without a device. Inside it the AI is a six-layer pipeline (data → analytics → rules → predictions → decisions → language) in which determinism increases downward and fluency upward: nothing above the decision layer may produce a figure, and nothing below the language layer may produce a sentence. There is no server-side runtime in the product's path at all — the one specified network endpoint ships with a blank base URL — which is P-01 and P-04 expressed as an architectural property rather than an intention.

Detail: 05 Architecture.

6. Data store, and why

Room over SQLCipher (AES-256) in WAL mode, with the key wrapped in the Android Keystore (StrongBox where available). The figures the app computes are relational aggregates — a derived balance, category spend, a forecast window — so a document store was rejected; and the storage must be unreadable to anyone holding the device, so plain SQLite was rejected. Room was preferred over Realm and ObjectBox because its migration story is explicit and testable, which the "destructive migrations are forbidden" rule depends on. Forty tables, 435 columns, schema version 33, 32 non-destructive migrations, each with a test against a real pre-change fixture.

Detail: 04 §13.

7. Delivery shape

Six SRS phases, delivered as thirteen epics and 89 issues, each closed against an eight-item Definition of Done. MS-001…MS-005 are delivered; MS-006 is partly delivered (five of seven expansion features); MS-007 — public-beta readiness — has not started. 25 tasks are open, each tied to the gap or question that opened it, and the critical path to a public beta is two of them: Play submission for READ_SMS, and a performance benchmark.

Detail: 08 Delivery.

8. Top five risks

RISK	Risk	Exposure	Mitigation in one line
RISK-001	One maintainer — no second reviewer, no succession	25	Mechanical enforcement instead of human review, and a public source tree
RISK-003	The device is the only copy of the data	15	E2EE backup, a restore drilled on real hardware before every release, and a UI that states irrecoverability in advance
RISK-002	READ_SMS may be refused at Play submission	12	Gated behind an explicit consent, argued in an ADR, and nothing depends on it
RISK-008	Three performance targets that nothing measures	12	Architecturally sound and architecturally tested — but unmeasured, and recorded as such
RISK-010	Knowledge bases go stale — tax changes every Budget	12	Versioned data rows, cited by id and version; an engine states what it could not estimate

Full register, with RISK-001…RISK-015: 09 §1.

9. Readiness verdict

READY WITH RISKS — copied verbatim from 11 Completeness Checklist.

Thirteen checks PASS, five PARTIAL, none FAIL. **0 blocking open questions** and **0 Critical gaps** of 19. Two gaps are High: the specified backend is not built (GAP-001) and three performance targets are unmeasured (GAP-004). The verdict is not READY because test traceability is PARTIAL — two requirements have no test and four non-functional targets have no measurement, and each is listed rather than omitted.

The nine questions that must be answered first are in 11 §Answer these.

10. Counts

Computed from the detailed documents, not typed.

Family	Count	Declared in
Business goals `GOAL`	7	01
Stakeholders `STK`	7	01
Constraints `CON`	14	01
Functional requirements `FR`	95	02
Non-functional requirements `NFR`	20	02
Business rules `BR`	28	02
Validation rules `VAL`	22	02
Acceptance criteria `AC`	115	02
Business flows `FLOW`	10	03
Use cases `UC`	55 (117 alternate · 107 exception · 100 failure paths)	03
Data entities `ENT`	40 (435 columns)	04
Components `CMP`	60	05
Integrations `INT`	11	05
API operations `API`	12 (1 implemented client-side)	06
Test cases `TC`	124	07
Milestones `MS`	7	08
Epics `EPIC`	13	08
User stories `US`	89	08
Open tasks `TASK`	25	08
Risks `RISK`	15	09
Assumptions `ASM`	12	09
Dependencies `DEP`	13	09
Open questions `OQ`	8 (**0 blocking**)	09
Traceability gaps `GAP`	19 (**0 Critical**, 1 closed)	10
Decision records digested	33 in full, 80 indexed — **generated from `docs/adr/`**	05 §10

From the code, for comparison: 60 Gradle modules · 885 Kotlin files · 40 Room entities at schema version 33 · 34 DAOs · 32 migrations · 30 engines · 13 feature modules with 25 screens · 21 typed routes · 9 background workers · 6 custom lint detectors · 80 ADRs · 3,986 tests in 386 test classes · 115 committed screenshot renders.

One of those counts was wrong until this blueprint checked it. The repository had said "82 ADRs" in several places, read off the highest id rather than counted: the numbers 0031 and 0032 were never written, and nothing references them. There are 80 records, and `ADR-0082` is the newest.

11. Document index

Doc	Covers	Status
01 — Project Overview and Scope	The problem and its cost, the goals, the stakeholders, the system context, what is in and out with a reason each, the constraints, and the glossary	Complete
02 — Requirements and Acceptance Criteria	95 `FR` and 20 `NFR` with their SRS ids and **as-built status**, 28 `BR`, 22 `VAL`, 115 `AC`	Complete
03 — Business Flows and Use Cases	10 flows as step tables with the actor, the system and the ids per step; 55 use cases, each with pre- and postconditions, the entities it touches, and alternate, exception and failure paths as tables — 324 paths in all	Complete

Doc	Covers	Status
04 — Data Dictionary and Database Design	40 entities and 435 columns generated from the schema, relationships, keys, indexes, transactions, retention, migrations, and the 35 SRS tables that do not exist	Complete
05 — Application Architecture	The style and why, the AI pipeline's six layers, 60 components, where each rule is **enforced**, 11 integrations, security, deployment, observability, reliability, and §10's decision digest — 33 records in full plus all 80 indexed, generated from `docs/adr/`	Complete
06 — API Specification	The client-side HTTP contract: 12 operations, each with its request table, response body, field semantics, status codes and the client's behaviour on every failure; the error envelope, caching, idempotency, versioning, an OpenAPI skeleton — and what ships instead	**Specification complete, backend not built (`GAP-001`)**
07 — Test Strategy and Cases	Twelve test levels, the one command that runs them, and 124 test cases naming the real test classes	Complete, with 5 requirements untested and listed
08 — Development and Delivery Plan	7 milestones, 13 epics, 89 stories, 25 open tasks, the critical path, the Definition of Done, CI, release and rollback, the 32 schema migrations and what guards them, and what the plan deliberately does not say	Complete
09 — Risks, Decisions, Assumptions and Open Questions	15 risks with exposure, how the register sorts and which three decide the verdict, what would reopen each decision, 12 assumptions with their cost if wrong, 13 dependencies, 8 open questions, and the eight findings writing this blueprint produced	Complete
10 — Traceability Matrix	The ID graph forward and backward, the two ID conventions and why both exist, coverage computed, SRS coverage, and 19 named gaps of which one is now closed	Complete
11 — Completeness Checklist	18 checks with evidence by ID, the verdict, and the nine questions that gate `READY`	Complete

Five figures are embedded in the documents and in the PDF: the system context (01 §4), capture-to-insight and the insight lifecycle (03), the schema's spine (04 §2), and the module graph with the AI pipeline (05 §1.1). Each is generated from the `.dot` source beside it, never hand-drawn.

Alongside this set. `../architecture-and-gap-report.md` compares the SRS against what was built, including findings against the project's own rules. `../blueprint-summary.md` is the short edition. The diagrams are in `diagrams/`, and `python3 scripts/build_blueprint_pdf.py` renders the whole set as one PDF.

Upstream sources this blueprint is written from. `../init/AI_Personal_CFO_SRS_v1.7.pdf` (the binding blueprint) · `../../CLAUDE.md` (the binding rules) · `../adr/` (80 decisions) · `../../DECISIONS.md` (the index) · `../../FLOW.md` (traced call paths) · `../issues/` (89 work items) · `../../CHANGELOG.md` (what shipped when).

Part 01 — Project Overview and Scope

← Master Blueprint · Related: 02 Requirements · 05 Architecture · 09 Risks & Open Questions

> ***Where these facts come from.*** *This blueprint is written* from *the project's own authored sources, not from an interview:* `docs/init/AI_Personal_CFO_SRS_v1.7.pdf` *(the binding blueprint),* `README.md`*,* `CLAUDE.md`*, the 82 records in* `docs/adr/`*, and the shipped code at* `v0.13.10`*. A statement traceable to one of those is* ***Confirmed*** *and cites its section. A statement that is mine — an inference, a reading, a filled gap — is marked* `[ASSUMED]` *and carries an* `ASM-xxx` *in 09.*

1. Problem statement

Confirmed — SRS §1.2.

Most people in India manage money reactively. The salary arrives, spending happens, and the month-end balance is a surprise. Five specific failures follow:

1. **No forward view.** Existing apps are ledgers. They render what already happened as charts, and answer neither "can I afford this bike?" nor "will I run out of money before the 28th?"
2. **A fragmented picture.** The real balance sheet is scattered across bank SMS, UPI apps, credit-card statements, paper receipts, gold, vehicles and cash. No single place holds all of it, so net worth is never actually known.
3. **Advice that is expensive, generic, or conflicted.** Personalised financial advice costs money or arrives commission-driven. What a user needs is advice *computed from their own data*, with the reasoning shown.
4. **Distrust of cloud finance apps.** Handing raw account credentials to a server is the objection that stops adoption. A local-first model removes it.
5. **Figures without working.** A number a user cannot check is a verdict. An app that issues verdicts is one they stop believing (README, "What it does").

1.1 Current state and its cost

Confirmed — SRS §1.2, §3.1–3.4.

Today's workaround	What it costs
Mental arithmetic plus the bank's SMS balance	The month-end balance is a surprise; "how much is safe to spend today" has no answer
A spreadsheet reconciled monthly (persona 2)	One evening a month, and the figures are stale the day after
A ledger app that only records	Categorisation by hand; no forecast; no answer to a purchase question
"Save 20% of salary" advice (persona 3)	Useless on an income that ranges Rs. 40k–Rs. 1.6L; the rule does not fit the person
Keeping a buffer "just in case"	Cash idles at 3% while a goal goes unfunded

2. Business goals

Confirmed — SRS §2.1 (`BG-01`…`BG-06`), restated here with this blueprint's IDs. The ID mapping is recorded in 10 §0.

ID	Goal	Measure	Target	SRS
GOAL-001	Be the user's single source of financial truth	Share of monthly spend captured in the app	≥ 80% of active users capture ≥ 90% of monthly spend by month 3	BG-01
GOAL-002	Deliver advice users act on	Recommendations marked "followed" or "helpful"	≥ 40%	BG-02
GOAL-003	Form a habit	Day-30 retention; sessions per week	≥ 35%; median ≥ 4	BG-03
GOAL-004	Earn trust through transparency	Privacy/trust survey score; data-related complaints	≥ 4.5 average; < 1%	BG-04
GOAL-005	Sustain the product after v1	Freemium conversion to Pro	Not yet targeted — post-v1 (`OQ-006`)	BG-05
GOAL-006	Keep support cost low	Self-serve resolution via in-app help + assistant	≥ 90%	BG-06

ID	Goal	Measure	Target	SRS
GOAL-007	Make the privacy claim checkable rather than marketed	Source published under a licence that permits inspection	AGPL-3.0 with a linking exception, public repository	README "Licence"

Product KPIs (SRS §2.2, all computed on-device; only anonymised counters leave the device, and only with analytics consent, which defaults OFF):

KPI	Target
Activation — first AI insight viewed	within 10 minutes of install
30-day cash-flow forecast error (MAPE)	$\leq$ 15% after 3 months of data
Auto-categorisation not corrected by the user	$\geq$ 92%
Receipt total read correctly	$\geq$ 95%

3. Stakeholders

Confirmed — SRS §3 (personas), README, `LICENSE`, `TERMS.md`.

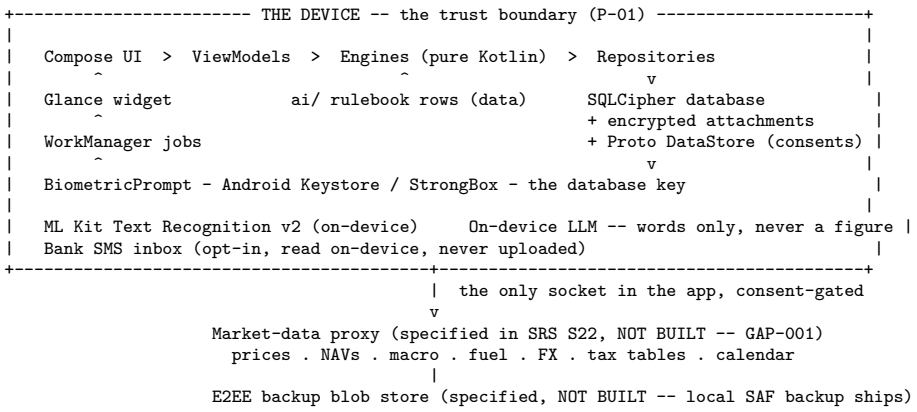
ID	Stakeholder	Role	Interest	Influence	What they need from the system
STK-001	The user (personas "Arjun", "Meera & Ravi", "Suresh", "Lakshmi")	Primary user, and the data principal under the DPDP Act 2023	Knowing where the money goes, what is safe to spend, whether a purchase fits	Decisive — the app is for them and holds their data	A complete picture, a forward view, advice with its working shown, and total control of the data
STK-002	A household member	Secondary profile on the same device	A shared picture without a shared diary	Medium	Aggregation that composes per-profile reads and leaks nothing between members (ADR-0069)
STK-003	Project owner / sole maintainer (Harish G)	Author, architect, reviewer, release manager	Shipping a defensible product solo	Decisive on every technical decision	Gates that fail loudly, documents that cannot drift from the code, and a backlog that is honest
STK-004	Google Play	Distribution channel and policy gatekeeper	Policy compliance, especially `READ_SMS`	Can block release	A justified, gated, declared SMS permission (ADR-0013) and a hardened release build
STK-005	The Indian regulatory frame — DPDP Act 2023, and the SEBI boundary	Compliance context	Purpose-limited consent, data-principal rights; no unregistered investment advice	Can invalidate the product	Local export (access right), erase-all (erasure right), per-feature consent, and an education-not-advice boundary (SRS §44, `TERMS.md`)
STK-006	Open-source contributors	Readers and possible contributors of the published source	Being able to verify the privacy claim, and to build on it	Low on direction, high on credibility	Readable architecture, a decision record, and a contribution path that keeps dual licensing possible
STK-007	A future reviewer or auditor of this codebase	Verification	Whether what the documents promise is what the build enforces	Medium	Enforcement that is a task, not a sentence, and a traceability record that names its gaps

3.1 Actors and roles The app is **single-user and local**. There is no server-side role model, no login, and no multi-tenancy (SRS §1.4, §22.1).

Actor	Kind	What it does
User	Human	Everything: records, reviews, decides. There is one role and it is "the person holding the phone"
Household member	Human	A second profile on the device; every row is scoped by `profile_id` (ADR-0069)

Actor	Kind	What it does
Business book	Human, same person	A second profile holding business transactions (ADR-0073, issue 13.5)
App lock	System	Gates the encrypted database on cold start and after idle; refuses to build the database while locked (ADR-0003)
Background workers	System	Nine WorkManager jobs: net-worth snapshot, insight refresh, scheduled-transaction posting, budget alerts, card alerts, SMS scan, market prices, widget refresh, balance integrity
Market-data proxy	External, optional	The one network path, off by default and inert without a configured host and certificate pins (ADR-0030)

4. System context



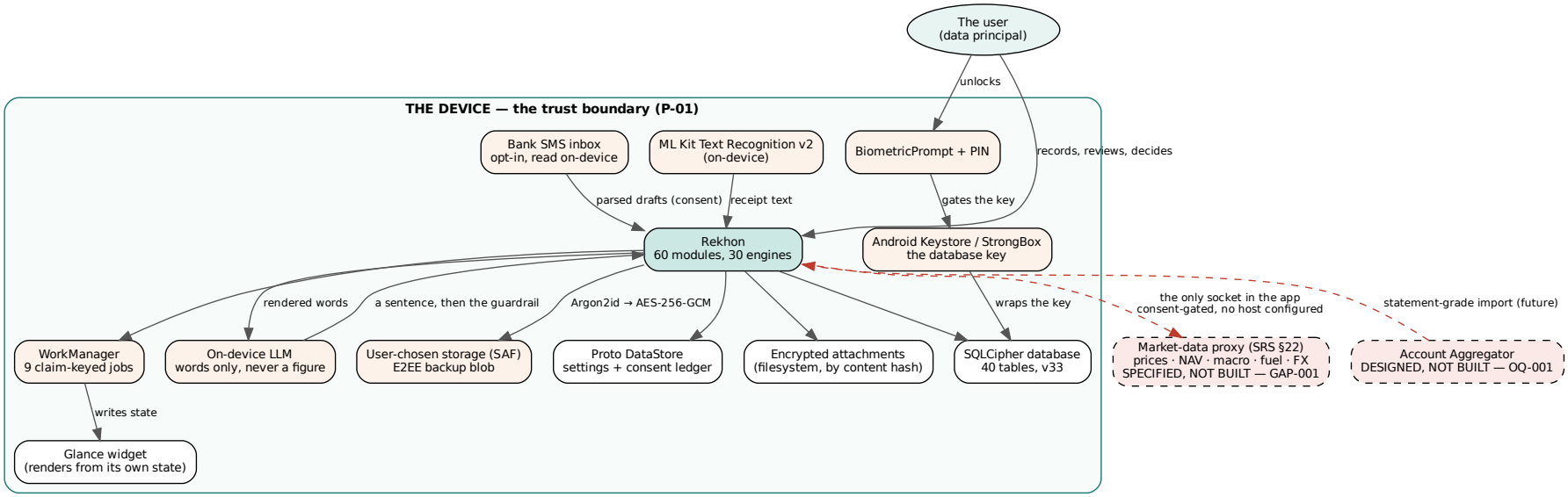


Figure 1 — System context. Everything above the dashed line runs on the device; the two systems below it are specified and unbuilt.

Source: diagrams/context.dot. The picture is generated from it, so the two cannot disagree.

Neighbouring systems are enumerated as `INT-001`...`INT-011` in 05 §4.

---

5. Scope — in

Confirmed — SRS §5, §7–§44; verified against the shipped modules at `v0.13.10`.

Capture. Manual entry in three taps · transfers as one logical record · split lines across N categories · future-dated entries that post themselves · receipt OCR on-device · opt-in bank-SMS drafts · recurring-series proposal · CSV/JSON import batches.

Ledger and taxonomy. Eleven account types · a category editor · merchant→category classification · nature classification (need / want / investment / asset / liability) · stream classification (fixed / semi-fixed / variable) · tags.

Planning. Per-category budgets with suggestions, pace and projection · 80% and 100% alerts · a monthly review · goals with required-monthly maths, feasibility and a prioritisation waterfall · an emergency-fund target and runway coach · the eight-stage financial order of operations.

Wealth. Net worth snapshotted daily · credit-card cycle, utilisation and due dates · loan EMI split and amortisation · investment holdings with dated lots and XIRR · allocation and concentration against a published rulebook · gold, crypto and property · vehicles and appliances with service prediction and running cost.

Intelligence. 90-day cash-flow forecast with P10/P50/P90 and crunch days · seasonality · a 0–1000 financial health score · an insight orchestrator and ranked feed · a notification policy gate · the numeric guardrail · the purchase advisor with a kept decision trace · prepay-vs-invest and debt-payoff simulators · a chat assistant that plans tool calls and receives only words · market opportunity signals with measured hit rates · tax estimation for both regimes · insurance adequacy.

Platform. App lock (biometric + PIN) · privacy blur · FLAG_SECURE on every screen · a consents dashboard with one-tap revoke · export/import · E2EE local backup and restore · crypto-shredding erase · a home-screen widget · four languages · household and business profiles · demo mode.

6. Scope — out, and why

Confirmed — SRS §1.4, §2.3, §27; README "Status".

Excluded	Reason
Moving money — payments, transfers, trades	`P-07`: the app advises and simulates; the user decides. No auto-executed financial action in v1, which also keeps the product outside payment regulation
Automatic bank aggregation by credentials or screen-scraping	`P-01`. Credential handling is precisely the trust objection the product exists to remove (SRS §1.2)
Account Aggregator (AA) integration	Designed for, not built. `transactions.source` reserves the `aa` value and `import_batches` carries statement-grade provenance, so the integration extends the schema rather than changing it (ADR-0074). Tracked as `OQ-001`
Regulated investment advice	The product is not SEBI-registered. Every output is informational and says so (SRS §44, FR-AI-005, `TERMS.md`)
A cloud service holding plaintext financial data	`P-01`. The backup blob is client-side encrypted and the server is blind to it (SEC-005)
Multi-tenant social features, leaderboards, ads, data resale	SRS §1.4. Each would need data to leave the device
iOS	Phase 5, and currently **blocked**: "Android-free" did not mean "KMP-portable", and the blocker is `Money` itself (ADR-0075). Tracked as `OQ-002`
Automated tax filing, insurance sales, loan origination	SRS §2.3 — each is a regulated distribution activity, and §44 draws the education/advice line deliberately
Children's data	DPDP Act 2023: 18+ self-declaration, no processing of children's data (SEC-008)

---

7. Constraints

ID	Constraint	Source	Consequence for the build
CON-001	Android only, minimum API 26	NFR-008	No iOS target; platform APIs below 26 must be guarded
CON-002	The device is the trust boundary; no server holds user financial data	`P-01`, SEC-005	There is no backend to put logic in. Every engine runs on the phone
CON-003	Every core feature must work in airplane mode, and the build and tests must pass with the backend absent	`P-04`	Network is an optional decoration. `offlineSmoke` is a gate
CON-004	Money is `Long` minor units (paise) end to end; no floating point ever touches an amount	MNY-001, NFR-012	`CfoMoneyAsFloatingPoint` fails the build. It also blocks the KMP port (ADR-0075)
CON-005	Timestamps are UTC epoch millis; calendar logic resolves in the profile zone through an injected `Clock`	TIM-001	`CfoWallClockInDomain` fails the build in `:domain:*` and `:core:model`
CON-006	Date-only fields are ISO `LocalDate` strings, never midnight timestamps	TIM-002	Schema columns are named `*_iso_date`
CON-007	An LLM may never produce a figure	`P-03`, AI-ARC-004	The model receives rendered words; the guardrail checks every output before display
CON-008	One developer, no QA team	Project reality	Enforcement must be mechanical: gates, not checklists. 3,986 automated tests stand in for a test team
CON-009	Base APK $\leq$ 40 MB	NFR-005	The on-device model and ML assets are budgeted against this
CON-010	`READ_SMS` carries a Play-policy risk	ADR-0013	The permission ships gated behind an explicit, revocable consent, with the risk written down rather than discovered at submission
CON-011	Dual licence: AGPL-3.0 with a narrow linking exception, plus a commercial option	`LICENSE`, `NOTICE`	Proprietary ML Kit / Play services may be linked; a contributor must grant a relicensing right
CON-012	The SRS is the single source of truth; any deviation needs an ADR	SRS §28, CLAUDE.md	80 ADRs exist because the deviations were recorded rather than absorbed
CON-013	Engines must stay pure Kotlin with no Android imports	ARC-002	Enforced by a convention-plugin TestKit guard in CI
CON-014	SonarQube must be clean before a commit	CLAUDE.md §8.1, ADR-0081	Agent-followed, not build-enforced — stated as such rather than implied

Constraints are repeated as a register in 09 §5.

8. Glossary

Term	Meaning
Safe-to-Spend (STS)	What is left to spend today after reserving upcoming fixed obligations, goal contributions and a 5% buffer (SRS §5.2; ADR-0021)
Financial Health Score (FHS)	A 0–1000 score over weighted pillars, with each pillar's contribution and the one lever that moves it most (SRS §14; ADR-0045)
Engine	A pure-Kotlin, deterministic module that turns typed inputs into a typed result with no Android dependency (SRS §4.3, ARC-002)
Provenance	The record attached to every engine result: `engineId`, `engineVersion`, `inputWindow`, `computedAt`, `confidence`, `evidence` (AI-ARC-003)
Insight	A generated, user-visible card carrying a finding, its evidence, a confidence and at most one action (FR-AI-002)
Guardrail	The check every LLM sentence passes before display: no figure may appear that an engine did not produce (AI-ARC-004; ADR-0048)
Rulebook / `ai/`	The versioned data rows holding every financial threshold. Changing a threshold is a data edit, never a code edit (CLAUDE.md §6)

Term	Meaning
Money	The `@JvmInline value class` wrapping `Long` paise. The only legal representation of an amount (MNY-001)
bps	Basis points — the integer representation of every rate and percentage inside an engine (MNY-002)
Stream classification	Whether an expense stream is FIXED, SEMI_FIXED or VARIABLE (SRS §8.2)
Nature	What a rupee became: NEED, WANT, INVESTMENT, ASSET or LIABILITY (SRS §8.3)
Crunch day	A day on which the forecast's P10 path crosses below zero (SRS §9)
XIRR	The money-weighted return of a holding, computed by bisection over the daily growth factor (ADR-0028)
FOIR	Fixed Obligations to Income Ratio — the Indian obligation metric used in place of a generic debt ratio (SRS §40.1)
Order of operations	The eight-stage waterfall that decides where the next surplus rupee goes (SRS §36)
Snapshot	An immutable daily record of balances and net worth, used for history and forecasting
Soft delete	Setting `deleted_at_utc_millis` instead of removing a row (DB-002)
Crypto-shredding	Destroying the wrapped database key so the ciphertext becomes unrecoverable — the first step of erase-all (SEC-006)
Drift test	A test that fails the build when a typed Kotlin mirror and the `ai/` row it claims to copy disagree
Golden-file test	A frozen input snapshot and its expected result JSON, which an engine must reproduce exactly (SRS §21.5)
DoD	Definition of Done — the eight-item gate every feature passes (SRS §4.2, CLAUDE.md §8)

Part 02 — Requirements and Acceptance Criteria

← Master Blueprint · Related: 01 Overview & Scope · 03 Flows & Use Cases · 07 Tests · 10 Traceability

> ***Two ID columns, on purpose.*** *The left column is this blueprint's handle (`FR-001`), which the traceability matrix and the validator use. The* ***SRS ID*** *column is the project's own requirement identifier (`FR-TXN-004`), which commits, code comments and tests already cite. Neither replaces the other; the mapping is the contract between them and is restated in 10 §0.* ***"As built" is measured, not claimed.*** *The column records what exists in the tree at `v0.13.10`: `Yes` = shipped with tests; `Partial` = shipped with a named limitation, which is linked; `No` = specified and not built, which always carries a `GAP` or `OQ`.*

---

1. Functional requirements

Priority is the SRS's RFC-2119 keyword. A `MUST` blocks release.

1.1 Onboarding and setup — SRS §5.1

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-001	FR-ONB-001	The app MUST be usable after a four-step onboarding: welcome and privacy pledge, local profile with currency format, security setup (biometric/PIN), first account with opening balance	User	MUST	GOAL-001	UC-001	Yes — step order deviates by design (ADR-0002)	Confirmed
FR-002	FR-ONB-002	Onboarding MUST offer an optional quick setup — monthly income, rent/EMI, typical savings — used to seed budget envelopes and the emergency-fund target	User	MUST	GOAL-001	UC-002	Yes (ADR-0004)	Confirmed
FR-003	FR-ONB-003	SMS-parsing permission MUST be a separate, skippable, explained opt-in, with examples of what is read and the statement that parsing is on-device only	User	MUST	STK-005	UC-001	Yes — promoted to its own step (ADR-0002)	Confirmed
FR-004	FR-ONB-004	A demo mode with realistic sample data MUST be available without creating a profile	User	SHOULD	GOAL-003	UC-003	Yes — a second `profile_id`, hard-deleted on exit (ADR-0006)	Confirmed
FR-005	FR-ONB-005	Onboarding MUST complete in under three minutes on the default path	User	SHOULD	GOAL-003	UC-001	Partial — no analytics consent exists to measure it (`ASM-003`)	Confirmed

1.2 Home dashboard — SRS §5.2

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-006	FR-HOME-001	The dashboard MUST show net worth with a 30-day sparkline, Safe-to-Spend today, month-to-date spend against budget, obligations due in the next 14 days, the top three insights, and goal progress chips	User	MUST	GOAL-001	UC-004	Yes	Confirmed
FR-007	FR-HOME-002	Every figure on the dashboard MUST be tappable and navigate to the screen that explains it	User	MUST	GOAL-004	UC-004	Yes — typed routes, one nav graph (ARC-001)	Confirmed
FR-008	FR-HOME-003	A privacy toggle MUST mask every amount instantly, and the state MUST persist for the session	User	MUST	STK-001	UC-005	Yes — fixed-width mask plus `FLAG_SECURE`, not a pixel blur (ADR-0022)	Confirmed
FR-009	FR-HOME-004	The dashboard MUST render fully from local data in under 500 ms on a mid-range device	User	MUST	NFR-001	UC-004	Partial — rendered from local data only; the 500 ms figure is not measured on a device (`GAP-004`)	Confirmed
FR-010	FR-HOME-005	Dashboard cards MUST be reorderable and hideable in an edit mode, stored per profile	User	SHOULD	STK-001	UC-004	No — not built (`OQ-007`)	Confirmed
FR-011	FR-HOME-006	A home-screen widget MUST show Safe-to-Spend and month spend without opening the app, respecting the privacy blur	User	SHOULD	GOAL-003	UC-006	Yes — renders from Glance state, never from Room (ADR-0024)	Confirmed
FR-012	FR-HOME-007	Pull-to-refresh MUST recompute insights and refresh external market data when online; stale external data MUST be labelled "as of <date>"	User	MUST	`P-04`	UC-004	Yes	Confirmed

1.3 Transactions — SRS §5.3

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-013	FR-TXN-001	A transaction MUST support amount, currency, date-time, account, category, subcategory, payee, note, many tags, many attachments, a type (expense / income / transfer), and split lines	User	MUST	GOAL-001	UC-007	Yes	Confirmed
FR-014	FR-TXN-002	Add-transaction MUST be one tap away and completable in three taps for a common expense	User	MUST	GOAL-003	UC-007	Yes — asserted by a Compose test	Confirmed
FR-015	FR-TXN-003	A transfer MUST be a single logical record affecting two accounts atomically; deleting one side MUST delete both	User	MUST	GOAL-001	UC-008	Yes — two legs sharing a `transfer_id`, type derived (ADR-0008)	Confirmed
FR-016	FR-TXN-004	A split MUST allow N lines with independent categories, and the lines MUST sum exactly to the parent amount with no rounding drift	User	MUST	BR-008	UC-009	Yes — child table, largest-remainder distribution (ADR-0009)	Confirmed
FR-017	FR-TXN-005	Auto-categorisation MUST run on every new transaction, and a user correction MUST improve future categorisation and offer to apply to similar past transactions	User	MUST	GOAL-002	UC-010	Partial — correction is stored and reused; "apply to similar past" is not built (`GAP-005`)	Confirmed
FR-018	FR-TXN-006	Recurring detection MUST propose a series when two or more similar transactions match on merchant, amount and cadence, with the user confirming	User	MUST	`P-05`	UC-011	Yes	Confirmed
FR-019	FR-TXN-007	The list MUST support search, filters (date, account, category, type, amount range), day grouping with totals, and paged infinite scroll	User	MUST	GOAL-001	UC-012	Yes	Confirmed
FR-020	FR-TXN-008	Bulk edit MUST support multi-select recategorise, retag and delete, with an undo snackbar and a seven-day soft delete	User	SHOULD	STK-001	UC-012	Partial — soft delete and multi-select ship; the seven-day trash compactor does not (`GAP-006`)	Confirmed
FR-021	FR-TXN-009	Every transaction MUST record its source: manual, OCR, SMS, import or recurring-auto	User, System	MUST	GOAL-004	UC-007	Yes — plus a reserved `aa` value (ADR-0074)	Confirmed
FR-022	FR-TXN-010	Future-dated transactions MUST be supported, excluded from actuals and included in forecasts	User	MUST	FR-063	UC-013	Yes — excluded by date, not by posting stamp (ADR-0010)	Confirmed

1.4 Receipt capture — SRS §5.4

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-023	FR-OCR-001	Camera capture and gallery import MUST both be supported; multi-page receipts MAY be stitched	User	MUST	P-05	UC-014	Partial — capture and import ship; multi-page stitching does not (a MAY)	Confirmed
FR-024	FR-OCR-002	OCR MUST run entirely on-device, and no receipt image may leave the device unless the user explicitly shares it	User	MUST	P-01	UC-014	Yes — ML Kit Text Recognition v2	Confirmed
FR-025	FR-OCR-003	The parser MUST extract total, date, merchant and, best-effort, tax and line items, each with a confidence indicator and editable before save	User	MUST	P-05	UC-014	Partial — total, date, merchant and GST ship (ADR-0077); line items do not (GAP-007)	Confirmed
FR-026	FR-OCR-004	The review screen MUST refuse to save without an amount and a date, and MUST flag low-confidence fields	User	MUST	VAL-010	UC-014	Yes	Confirmed
FR-027	FR-OCR-005	The original image MUST be stored encrypted and linked as an attachment; the user may delete the image and keep the transaction	User	MUST	P-01	UC-014	Partial — stored encrypted and deletable; orphaned blobs are a known defect (ADR-0078, RISK-009)	Confirmed
FR-028	FR-OCR-006	A duplicate guard MUST offer a merge when an existing transaction matches on amount ±1% and date ±1 day, rather than creating a duplicate	User, System	MUST	GOAL-001	UC-014	Yes	Confirmed

1.5 Budgets — SRS §5.5

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-029	FR-BUD-001	Budgets MUST be definable per category or category group, per month, with optional rollover of the unused amount	User	MUST	GOAL-001	UC-015	Partial — per category and per month ship; rollover does not (`GAP-008`)	Confirmed
FR-030	FR-BUD-002	Budget suggestions MUST be offered from three months of history using a median with a seasonality adjustment, accepted one tap per category	User	MUST	GOAL-002	UC-015	Yes	Confirmed
FR-031	FR-BUD-003	The budget screen MUST show spent, remaining, daily pace against safe pace, and the projected end-of-month figure per budget	User	MUST	`P-02`	UC-015	Yes	Confirmed
FR-032	FR-BUD-004	Overspend alerts MUST fire at 80% and 100%, and a pace alert when the projected overshoot exceeds 10% before the 20th	System	MUST	GOAL-002	UC-016	Yes — bands mint `RULE-BUD-ALERT` v1.0, claimed once each (ADR-0019)	Confirmed
FR-033	FR-BUD-005	A monthly budget review flow MUST summarise wins and misses and propose next month's numbers	User	SHOULD	GOAL-002	UC-017	Yes — one card per closed month, keyed by month (ADR-0020)	Confirmed

1.6 Goals — SRS §5.6

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-034	FR-GOAL-001	Goal types MUST include emergency fund (a system goal, always present), purchase, travel, home down-payment, education, retirement, debt payoff and custom	User	MUST	GOAL-002	UC-018	Yes	Confirmed
FR-035	FR-GOAL-002	Each goal MUST carry target amount, target date or open, priority, funding accounts, an expected-return assumption, and a computed required monthly contribution	User	MUST	GOAL-002	UC-018	Partial — every field but the expected return, which the app refuses to invent (ADR-0033)	Confirmed
FR-036	FR-GOAL-003	Goal feasibility MUST be checked against forecast surplus; an infeasible plan MUST show the gap and three levers — extend the date, reduce the target, increase the contribution	User	MUST	P-02	UC-019	Yes	Confirmed
FR-037	FR-GOAL-004	Contributions MUST be linkable to real transactions and accounts, so progress is evidenced rather than self-declared	User	MUST	GOAL-004	UC-020	Yes — evidenced and declared progress are kept separate and never merged (ADR-0036)	Confirmed
FR-038	FR-GOAL-005	Goal conflicts — combined contributions exceeding surplus — MUST trigger a prioritisation assistant	User	SHOULD	GOAL-002	UC-019	Yes — the waterfall ranks and splits the surplus (ADR-0038)	Confirmed

1.7 Accounts and net worth — SRS §5.7

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-039	FR-ACC-001	Account types MUST include bank, cash, credit card, loan, investment, gold, crypto, property, vehicle, receivable and payable	User	MUST	GOAL-001	UC-021	Yes — all eleven	Confirmed
FR-040	FR-ACC-002	A credit card MUST track limit, statement day, due day, current unbilled and last statement amount, and its utilisation MUST feed the health score	User	MUST	GOAL-001	UC-022	Yes	Confirmed
FR-041	FR-ACC-003	A loan MUST store principal, rate, EMI and tenure, and MUST expose an amortisation schedule, a payoff date and a prepayment simulator	User	MUST	GOAL-002	UC-023	Yes — the schedule is derived on read, never stored (ADR-0026)	Confirmed
FR-042	FR-ACC-004	Asset accounts MUST support a valuation history, and market-priced assets MUST revalue from external data when online	User, System	MUST	GOAL-001	UC-024	Partial — the price path and cache ship; the proxy that would serve them does not (`GAP-001`, ADR-0030)	Confirmed
FR-043	FR-ACC-005	Net worth MUST equal assets minus liabilities, be snapshotted daily, and be chartable over 1M / 6M / 1Y / All	User, System	MUST	GOAL-001	UC-025	Yes — the series repairs itself after a back-dated entry (ADR-0012)	Confirmed
FR-044	FR-ACC-006	A reconciliation flow MUST let the user enter the real balance, with the difference posted as an adjustment transaction and never silently mutated	User	MUST	BR-007	UC-026	Yes	Confirmed
FR-045	FR-ACC-007	Archived or closed accounts MUST retain their history and be excluded from active totals	User	MUST	GOAL-001	UC-021	Yes — `archived_at` is distinct from `deleted_at`	Confirmed

1.8 Investments, vehicles and appliances — SRS §5.8, §12

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-046	FR-INV-001	A holding MUST record units, average cost and current price, and MUST show return as both absolute and XIRR	User	MUST	GOAL-001	UC-027	Yes — dated lots; XIRR by bisection over the daily growth factor (ADR-0028)	Confirmed
FR-047	FR-INV-002	The portfolio view MUST show allocation by asset class, a diversification analysis and concentration warnings	User	MUST	GOAL-002	UC-028	Yes — the denominator is the investable accounts, and unpriced holdings are reported as coverage rather than counted as zero (ADR-0029)	Confirmed
FR-048	FR-VEH-001	A vehicle profile MUST capture type, make, model, year, purchase price and date, an odometer log, a fuel log, insurance and PUC expiry, and a service history	User	MUST	GOAL-001	UC-029	Partial — profile, odometer, service and renewals ship; the fuel log does not (GAP-009)	Confirmed
FR-049	FR-VEH-002	The fuel log MUST compute mileage trends, and an anomalous mileage drop MUST raise a maintenance insight	System	MUST	GOAL-002	UC-029	No — depends on FR-048's fuel log (GAP-009)	Confirmed
FR-050	FR-VEH-003	The vehicle screen MUST show the true monthly cost of ownership — fuel, EMI, amortised insurance, service and depreciation	User	SHOULD	P-02	UC-029	Partial — service, renewals and running cost ship; the fuel component is absent with FR-048	Confirmed

1.9 Advisor — SRS §5.9

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-051	FR-AI-001	The advisor hub MUST list the health score with a drill-down, the insight feed with act / snooze / dismiss, the purchase advisor, the simulators and chat	User	MUST	GOAL-002	UC-030	Yes	Confirmed
FR-052	FR-AI-002	Every insight card MUST carry a finding, its evidence, a confidence and at most one recommended action	User	MUST	P-02	UC-031	Yes	Confirmed
FR-053	FR-AI-003	The purchase advisor MUST accept item, price, payment method and urgency, and MUST output a verdict — Comfortable, Stretch or Not now — with the full reasoning trace	User	MUST	GOAL-002	UC-032	Yes — the trace is stored per decision (ADR-0049)	Confirmed
FR-054	FR-AI-004	The chat assistant MUST answer questions about the user's own data through the tool registry, and MUST refuse to fabricate a number when a tool cannot answer	User	MUST	P-03	UC-033	Yes — the model receives words only, never rows (ADR-0053)	Confirmed
FR-055	FR-AI-005	Every AI output MUST carry the disclaimer that it is informational and not regulated investment advice	System	MUST	STK-005	UC-030	Yes	Confirmed

1.10 Settings, security and data — SRS §5.10

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-056	FR-SET-001	Settings MUST include profile and household, a categories editor, a recurring-rules manager, notification preferences with quiet hours, security (biometric, PIN, auto-lock), data (backup, restore, export, import, erase), a consents dashboard, and about/licences	User	MUST	STK-001	UC-034	Partial — all but the recurring-rules *manager* screen, which exists as engine plus repository without a dedicated editor (GAP-010)	Confirmed
FR-057	FR-SET-002	Export MUST produce a complete documented JSON archive plus CSVs, and import MUST round-trip losslessly	User	MUST	STK-005	UC-035	Partial — the JSON archive round-trips losslessly and is pinned by a format test; CSV export is not built (GAP-011, ADR-0023)	Confirmed
FR-058	FR-SET-003	"Erase all data" MUST require biometric plus a typed confirmation, and MUST cryptographically destroy the database key	User	MUST	STK-005	UC-036	Yes — keys first, then files, then a report of what it could not reach (ADR-0060)	Confirmed
FR-059	FR-SET-004	The consents dashboard MUST list every permission and consent with its status, a plain-language purpose and one-tap revoke	User	MUST	P-01	UC-037	Yes — it shows the record, not just the switch (ADR-0059)	Confirmed

1.11 The AI pipeline — SRS §7–§19, §30, §36–§40 Each engine is a `MUST` because the product's claim rests on it, and each is a pure Kotlin module with a frozen golden-file fixture.

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-060	AI-CLSN-001	Stage 1 classification MUST map a merchant to a category with a confidence and the evidence that produced it	System	MUST	GOAL-002	UC-010	Partial — two of three tiers; the two shipped already clear the $\geq$ 92% gate (ADR-0015)	Confirmed
FR-061	—	Stage 2 MUST classify each expense stream FIXED, SEMI_FIXED or VARIABLE from its own history	System	MUST	GOAL-002	UC-038	Yes (ADR-0042)	Confirmed
FR-062	—	Nature classification MUST label every rupee NEED, WANT, INVESTMENT, ASSET or LIABILITY	System	MUST	P-02	UC-038	Partial — fires on account type; only the override is stored, and true spend ships understated by design (ADR-0016)	Confirmed
FR-063	AI-FCT-001	The forecast MUST project the liquid balance 90 days ahead as P10 / P50 / P90 and MUST name every crunch day	System	MUST	GOAL-002	UC-039	Yes — schedules the confirmed and the fixed (ADR-0043)	Confirmed
FR-064	—	Seasonality MUST derive a per-category monthly index from the user's own history before any shared prior	System	MUST	P-08	UC-039	Yes — own history first, lookback divided out (ADR-0044)	Confirmed
FR-065	—	The emergency fund MUST compute a target from essential spend times a multiplier, a runway in months, and a coaching line	System	MUST	GOAL-002	UC-040	Partial — two of §10.1's five multiplier terms; the other three have no field anywhere in the app, and their absence is asserted by a test (ADR-0034)	Confirmed
FR-066	—	Investment intelligence MUST report allocation, diversification and concentration against the published rulebook	System	MUST	GOAL-002	UC-028	Yes	Confirmed
FR-067	—	Vehicle prediction MUST estimate the next service date and its cost range from the personal usage trend	System	MUST	GOAL-002	UC-029	Yes — Theil–Sen on a personal index; it never invents a price (ADR-0052)	Confirmed
FR-068	—	Appliances MUST carry a next service, their consumables, a warranty and a running cost	User, System	SHOULD	GOAL-001	UC-041	Yes — a sibling of the vehicle engine; running cost is the new figure (ADR-0070)	Confirmed
FR-069	—	The purchase advisor MUST run its gates in order, produce one verdict, and keep the trace that produced it	System	MUST	P-02	UC-032	Yes (ADR-0049)	Confirmed
FR-070	—	A buy list MUST hold wishes rather than purchases, with an adaptive interview and a ladder that removes nothing automatically	User	SHOULD	GOAL-002	UC-042	Yes (ADR-0050)	Confirmed
FR-071	—	The health score MUST be 0–1000 over weighted pillars, showing each pillar's contribution and the single strongest lever	System	MUST	P-02	UC-043	Partial — six live signals reweighted on what is present, with no snapshots table; the SRS's five fixed pillars were not the shape that survived (ADR-0045)	Confirmed
FR-072	—	Goal planning MUST compute the required monthly contribution, the ETA and the horizon class from the target and date	System	MUST	GOAL-002	UC-018	Yes — and it assumes no rate of return (ADR-0033)	Confirmed
FR-073	—	External data MUST degrade to cache with a staleness label, never to a blank or a guess	System	MUST	P-04	UC-024	Partial — the degradation path ships and is the only path, because the proxy does not exist (GAP-001)	Confirmed
FR-074	—	A notification policy MUST decide what may interrupt the user and when, honouring quiet hours and per-type caps	System	MUST	STK-001	UC-044	Partial — one gate with caps and quiet hours; the digest is deferred (ADR-0047)	Confirmed
FR-075	—	The orchestrator MUST deduplicate and rank published engine results into one insight feed	System	MUST	GOAL-002	UC-031	Yes (ADR-0046)	Confirmed
FR-076	AI-ARC-004	Every LLM sentence MUST pass a numeric guardrail before display: a figure the engines did not produce MUST never reach the user	System	MUST	P-03	UC-033	Yes — one engine, a rendering allowlist, rounding stops at the rupee (ADR-0048), measured against a frozen set with an absolute threshold (ADR-0054)	Confirmed
FR-077	—	Chat MUST plan tool calls from the question, call only the registry, and pass the model words rather than rows	System	MUST	P-03	UC-033	Yes (ADR-0053, ADR-0079)	Confirmed
FR-078	—	Market signals MUST score only what can be measured, state what cannot, and publish a measured hit rate rather than a claim	System	MUST	P-02	UC-045	Yes (ADR-0055)	Confirmed
FR-079	—	The order of operations MUST send the surplus down the eight-stage waterfall and show where each rupee went	System	MUST	GOAL-002	UC-046	Yes — reads the ai/ FOO file and composes goals (ADR-0037)	Confirmed
FR-080	—	Simulators MUST answer prepay-versus-invest and which-debt-first using the loan engine's own arithmetic	User	MUST	GOAL-002	UC-047	Yes — shared arithmetic, minting nothing (ADR-0051)	Confirmed
FR-081	—	The tax engine MUST compare both regimes and compute capital gains under the post-23-July-2024 rules, and MUST state what it will not estimate	System	MUST	P-02	UC-048	Partial — the KB had no slabs when the engine landed, so the estimate names what it left out (ADR-0072)	Confirmed
FR-082	—	Protection MUST report term and health cover gaps and detect an endowment policy sold as insurance	System	MUST	GOAL-002	UC-049	Yes — the gap is the larger of two readings, and the engine never says surrender (ADR-0071)	Confirmed
FR-083	—	Safe-to-Spend MUST compute what is left to spend this month and name every line it withheld	System	MUST	P-02	UC-004	Yes — income from the declared budget, a labelled 5% buffer (ADR-0021)	Confirmed

1.12 Platform, security and data rights — SRS §21, §23, §27, §34.4

ID	SRS ID	Requirement	Actor	Priority	Source	UC	As built	Status
FR-084	SEC-002	The app MUST lock behind BiometricPrompt class 3 with a PIN fallback, auto-lock on idle, and a doubling lockout after five failed attempts	User	MUST	STK-001	UC-050	Yes — the lock gates a session flag and the database provider refuses to build while locked (ADR-0003)	Confirmed
FR-085	SEC-001	The database key MUST be 256-bit, generated on first run, wrapped by a hardware-backed Keystore key, never written unwrapped, and rotatable without losing the database	System	MUST	P-01	UC-050	Yes — StrongBox where available (ADR-0057)	Confirmed
FR-086	—	Every screen MUST be capture-guarded, because every screen is financial	System	MUST	STK-001	UC-005	Yes — `FLAG_SECURE` app-wide (ADR-0058)	Confirmed
FR-087	SEC-005	Backup MUST be AES-256-GCM under a key derived from a user passphrase by Argon2id, restorable on a fresh device, with irrecoverability made explicit	User	MUST	STK-001	UC-051	Yes — local SAF backup; the blob endpoint is not built (ADR-0039, ADR-0040, `GAP-001`)	Confirmed
FR-088	DRL-001	A backup restore drill MUST run on a real device before any promotion to `stage` or `main`, and a red drill MUST block the release	Maintainer	MUST	GOAL-004	UC-051	Yes — a release gate; the in-app drill waits (ADR-0041)	Confirmed
FR-089	—	Security-relevant events MUST be recorded in an audit log that the export deliberately does not carry	System	MUST	STK-005	UC-052	Yes — `audit_log` has no `profile_id` to scope by, so it is export-excluded by design (ADR-0023)	Confirmed
FR-090	NFR-011	The app MUST speak English, Hindi, Kannada and Tamil, and MUST prove it rather than assert it	User	SHOULD	STK-001	UC-053	Yes — four languages with a test that fails on a missing string (ADR-0056)	Confirmed
FR-091	—	Household mode MUST aggregate from per-profile reads and leak nothing between members	User	SHOULD	STK-002	UC-054	Partial — the row and the composition ship; the mode is off by a constant (ADR-0069)	Confirmed
FR-092	—	A business book MUST be a second profile, not a flag on a transaction	User	SHOULD	STK-001	UC-054	Yes (ADR-0073)	Confirmed
FR-093	—	An import MUST be a batch with statement-grade provenance that a later wipe can verify it removed	User, System	MUST	GOAL-004	UC-055	Yes — `import_batches`, and the wipe is checked rather than trusted (ADR-0078)	Confirmed
FR-094	—	A future-dated transaction MUST post itself when its day arrives, without correctness depending on the worker having run	System	MUST	BR-010	UC-013	Yes (ADR-0010)	Confirmed
FR-095	DB-001	A nightly integrity job MUST verify that every account balance still equals its opening balance plus its transactions, and MUST raise an adjustment prompt on a discrepancy	System	MUST	BR-007	UC-026	Yes	Confirmed

2. Non-functional requirements

`NFR-001`...`NFR-012` are the SRS's own numbers, unchanged. `NFR-013`...`NFR-020` are requirements this project enforces beyond the SRS; each is traceable to the gate that enforces it.

ID	Category	Requirement, with its number	How it is measured	Priority	As built	Status
NFR-001	Performance	Cold start to an interactive dashboard under 2.0 s at P90 on a mid-range device	Device timing run; not yet automated	MUST	Partial — no automated measurement (`GAP-004`)	Confirmed
NFR-002	Performance	Add-transaction local commit under 100 ms	Instrumented timing of the write path	MUST	Partial — same gap as NFR-001	Confirmed
NFR-003	Performance	A full insight recompute over 5 years / 50,000 transactions under 3 s in the background, never blocking the UI	Benchmark over a generated fixture	MUST	Partial — the work is off the main thread by construction; the 50k benchmark is not written (`GAP-004`)	Confirmed
NFR-004	Reliability	Crash-free sessions at or above 99.8%	Play vitals, post-launch	MUST	Not measurable pre-launch (`OQ-005`)	Confirmed
NFR-005	Footprint	Base APK at or below 40 MB	`assembleRelease` artefact size in CI	MUST	Yes — measured by the release job	Confirmed
NFR-006	Battery	Background work only through WorkManager, batched and network-aware, with no wakelocks	Code review plus the nine declared workers	MUST	Yes	Confirmed
NFR-007	Durability	Zero data loss on process death; WAL mode; every write transactional	Room in-memory tests plus the multi-row transaction rule (BR-016)	MUST	Yes	Confirmed
NFR-008	Compatibility	Minimum Android API 26, targeting the latest stable API	`minSdk` in the build	MUST	Yes	Confirmed
NFR-009	Offline	100% of core flows function offline; external data is cached with a TTL and a staleness label	`offlineSmoke` on an emulator in airplane mode (ADR-0067)	MUST	Yes — a gate, on a device	Confirmed
NFR-010	Accessibility	WCAG 2.1 AA equivalent, TalkBack complete, dynamic type to 200%, 48 dp targets, never colour alone	Accessibility scan plus Paparazzi renders at 200% font in light and dark	MUST	Partial — renders and tokens ship; a full TalkBack pass is manual (`GAP-012`)	Confirmed
NFR-011	Localisation	No hardcoded user-facing strings; ICU plurals; four languages	`CfoHardcodedUiString` fails the build; a drift test fails on a missing translation	MUST	Yes	Confirmed
NFR-012	Correctness	Every amount stored as integer minor units; no floating point in money maths, ever	`CfoMoneyAsFloatingPoint` fails the build; `:core:model` holds 100% line coverage	MUST	Yes — and it is the KMP blocker (ADR-0075)	Confirmed
NFR-013	Maintainability	Line coverage at or above 85% on every pure-Kotlin module, and 100% on the money module	`koverVerify` with a real `minBound`, after the vacuous version was found (audit G-01)	MUST	Yes	Confirmed
NFR-014	Maintainability	No new ktlint, detekt or Android lint warning, and six custom detectors that fail the build	`./gradlew unitTests` plus `lintDebug` in CI	MUST	Yes	Confirmed
NFR-015	Security	Every dependency scanned for known vulnerabilities on the versions that actually ship	OSV scan in CI, querying resolved coordinates (SEC-007)	MUST	Yes	Confirmed
NFR-016	Security	Release builds minified, with debuggable and backup flags off, and the log strip verified in the shipped DEX	`verifyReleaseLogStripping` reads the dex rather than trusting the config (ADR-0062)	MUST	Yes	Confirmed
NFR-017	Recoverability	A backup must restore on a fresh device, proven by a drill on real hardware before every promotion	`restoreDrill` on an emulator in CI (ADR-0041)	MUST	Yes	Confirmed

ID	Category	Requirement, with its number	How it is measured	Priority	As built	Status
NFR-018	Privacy	No amount and no personal datum may appear in a log line	`CfoPiiInLogs` fails the build	MUST	Yes	Confirmed
NFR-019	Reproducibility	Every engine result carries its engine version, so an insight produced six months ago still renders and still reproduces	`engine_version` stored with every result (AI-ARC-006); golden files frozen per version (ADR-0064)	MUST	Yes	Confirmed
NFR-020	Verifiability	Every screen in the app has a committed render that the build compares against	`verifyPaparazziDebug` inside `unitTests`; 115 committed renders (ADR-0082)	MUST	Yes	Confirmed

2.1 Requirements without acceptance criteria None. Every `FR-001`…`FR-095` and `NFR-001`…`NFR-020` has at least one `AC` in §5. The coverage is recomputed in 10 §4.

3. Business rules

A business rule is a rule about the money or the user, not about a screen. The right-hand column names who may change it — and for every financial threshold the answer is a data row in `ai/`, never a developer (CLAUDE.md §6).

ID	Rule	Applies to	Owner	Changeable by
BR-001	An amount is `Long` minor units (paise). No `Double` or `Float` may touch a monetary value	Every entity, engine and screen	MNY-001	Nobody — a lint rule fails the build
BR-002	A rate or percentage inside an engine is integer basis points	Every engine	MNY-002	Nobody
BR-003	An instant is UTC epoch millis; calendar logic resolves in the profile's time zone through an injected `Clock`	Every entity and engine	TIM-001	Nobody — a lint rule fails the build in `:domain:*`
BR-004	A date the user picked is an ISO `LocalDate` string, never a midnight timestamp	Every entity	TIM-002	Nobody
BR-005	A user-data row is soft-deleted by setting `deleted_at_utc_millis`; hard purge happens only through erase-all	Every profile-scoped entity	DB-002	Nobody
BR-006	Every row carries `profile_id` and no query may span profiles	Every profile-scoped entity	ADR-0069	Nobody — `ProfileScopingTest`
BR-007	An account balance is derived from its opening balance plus its transactions; it is never mutated ad hoc	ENT-003, FR-044, FR-095	ADR-0007, DB-001	Nobody
BR-008	Split lines sum exactly to the parent amount; the remainder is distributed largest-remainder, not `HALF_EVEN` per line	ENT-005, FR-016	ADR-0009	Nobody
BR-009	A transfer is two legs sharing a `transfer_id`; the type is derived from the leg, never supplied	ENT-004, FR-015	ADR-0008	Nobody
BR-010	A transaction counts as actual by its **date**, not by whether a worker posted it	ENT-004, FR-022, FR-094	ADR-0010	Nobody
BR-011	An LLM never computes. Every figure comes from a deterministic engine; the model only puts figures into sentences	FR-054, FR-076, FR-077	`P-03`	Nobody — a golden guardrail set
BR-012	A figure may be spoken only if it is on the guardrail's rendering allowlist, and rounding stops at the rupee	FR-076	ADR-0048	Maintainer, via an ADR
BR-013	Every engine result carries `engineId`, `engineVersion`, `inputWindow`, `computedAt`, `confidence` and `evidence`	Every engine	AI-ARC-003	Nobody

ID	Rule	Applies to	Owner	Changeable by
BR-014	A consent is per-feature, defaults OFF, is revocable in one tap, and records both grant and revoke times	ENT via DataStore, FR-059	`P-01`, ADR-0059	Nobody
BR-015	Erase destroys the wrapped key first, then the files, then reports what it could not reach	FR-058	SEC-006, ADR-0060	Nobody
BR-016	Every multi-row operation — transfer, split, import batch — runs in one transaction	FR-015, FR-016, FR-093	DB-004	Nobody
BR-017	A schema change ships with a Room migration and a migration test against a real pre-change fixture. Destructive migrations are forbidden	ENT-001… ENT-040	DB-003	Nobody
BR-018	Every financial threshold is a versioned row in `ai/`, cited by id and version in the evidence. A hardcoded financial number in an engine is a defect	Every engine	CLAUDE.md §6	Maintainer, by editing the data row
BR-019	A budget alert fires once per band per month: 80% warning, 100% exceeded	FR-032	`RULE-BUD-ALERT` v1.0, ADR-0019	Data row
BR-020	A card reminder is claimed per statement cycle, not per calendar month, because a cycle can straddle two months	FR-040	`RULE-CC-DUE` v1.0, ADR-0025	Data row
BR-021	Portfolio concentration is measured against the investable accounts, and an unpriced holding is reported as missing coverage rather than counted as zero	FR-047	`RULE-CONC-15-70`, ADR-0029	Data row
BR-022	The emergency-fund target is essential spend times a multiplier; essentials are the median of closed months' NEED spend, with the quick-setup envelope as fallback	FR-065	`RULE-EMF-MULT`, ADR-0034	Data row
BR-023	Surplus flows down the eight-stage order of operations, and the stage that consumed each rupee is shown	FR-079	`ai/rules/financial-order-of-operations.json`	Data row
BR-024	The app never moves money and never executes a financial action	Every feature	`P-07`	Nobody
BR-025	Every advisor output states that it is informational and not regulated investment advice	FR-055	STK-005, SRS §44	Nobody
BR-026	A loan's amortisation is derived on every read from its five terms; no schedule rows are stored	FR-041	ADR-0026	Nobody
BR-027	Goal progress keeps evidenced and declared contributions separate and never merges them	FR-037	ADR-0036	Nobody
BR-028	Revoking MARKET_DATA stops fetching but keeps the cached public price; revoking SMS deletes the drafts	FR-059	ADR-0030	Maintainer, via an ADR

4. Validation rules

ID	Field or input	Rule	Message	Enforced where
VAL-001	Transaction amount	Required; greater than zero; parsed to paise with no floating point	"Enter an amount"	UI, then the repository
VAL-002	Transaction date	Required; a future date is allowed and marks the row future-dated	"Pick a date"	UI

ID	Field or input	Rule	Message	Enforced where
VAL-003	Transaction account	Required; must be an active, non-archived account of this profile	"Choose an account"	UI, DB (FK)
VAL-004	Split lines	At least two lines; every line greater than zero; the sum equals the parent exactly	"Lines must add up to the total"	UI, engine, DB transaction
VAL-005	Transfer	Source and destination must differ, and both must belong to this profile	"Pick two different accounts"	UI, repository
VAL-006	Account opening balance	Required; may be negative for a credit card or a loan	"Enter the balance"	UI
VAL-007	Account type	One of the eleven `AccountType` values	— (not user-enterable)	`:core:model` enum
VAL-008	Account name	Required, non-blank, unique per profile	"Name this account"	UI, unique index
VAL-009	Category name	Required, non-blank, unique per profile	"Name this category"	UI, unique index
VAL-010	Receipt review	Amount and date both required before save; low-confidence fields flagged	"Check the amount and date"	UI
VAL-011	Budget amount	Greater than zero; one budget per category per month	"This category already has a budget this month"	UI, unique index
VAL-012	Goal target	Greater than zero; target date, if given, must be in the future	"Pick a date in the future"	UI, engine
VAL-013	Credit-card statement and due day	1–31; the cycle is derived, not entered	"Pick a day of the month"	UI
VAL-014	Loan terms	Principal greater than zero; rate in bps ≥ 0; tenure 1–600 months	"Check the loan terms"	UI, engine bound
VAL-015	Holding units and price	Units greater than zero; a price must carry the date it was observed	"Enter the units"	UI, engine
VAL-016	Odometer reading	Monotonic per vehicle — a reading may not be lower than an earlier dated one	"That reading is lower than the last one"	UI, repository
VAL-017	PIN	4–8 digits; not a repeated or sequential run	"Choose a stronger PIN"	UI
VAL-018	Backup passphrase	Minimum length enforced, and the recovery screen requires an explicit acknowledgement of irrecoverability	"Confirm you have saved the phrase"	UI
VAL-019	Erase confirmation	Biometric success plus a typed confirmation string	"Type ERASE to confirm"	UI
VAL-020	Import file	Must declare an archive schema version the app can read; an unknown version is refused, not guessed	"This archive was made by a newer version"	Repository
VAL-021	Purchase advisor input	Item name required; price greater than zero; a method from the enum	"Enter a price"	UI
VAL-022	Chat question	Non-empty; a question the registry cannot answer is refused rather than answered	"I can't answer that from your data"	Engine

5. Acceptance criteria

Given / When / Then. One row minimum per `FR` and per `NFR`.

ID	Parent	Given	When	Then
AC-001	FR-001	A fresh install	The user completes onboarding	A profile, a security method and one account with an opening balance exist, and the dashboard opens
AC-002	FR-002	Onboarding, at quick setup	The user enters income, rent and typical savings	Budget envelopes and an emergency-fund target are written as real rows, not held in memory
AC-003	FR-003	Onboarding, at the SMS step	The user skips it	Onboarding completes, no SMS permission is requested, and the app works fully
AC-004	FR-004	No profile exists	The user opens demo mode	Realistic sample data appears under a separate `profile_id`, and leaving demo mode removes every demo row
AC-005	FR-005	The default onboarding path	A user completes it without pausing	Every step is reachable without waiting on I/O; elapsed time is not asserted, because nothing measures it
AC-006	FR-006	A profile with transactions, budgets and goals	The dashboard loads	Net worth, Safe-to-Spend, month-to-date against budget, the next 14 days, three insights and goal chips are all present
AC-007	FR-007	The dashboard	The user taps any figure	The screen that explains that figure opens, by a typed route

ID	Parent	Given	When	Then
AC-008	FR-008	The dashboard with the blur off	The user taps the privacy toggle	Every amount on every screen is replaced by a fixed-width mask, `FLAG_SECURE` is set, and the state survives navigation
AC-009	FR-009	A populated profile, offline	The dashboard loads	It renders from local data with no network call on the path
AC-010	FR-010	The dashboard	The user looks for an edit mode	None exists — this requirement is not built (`OQ-007`)
AC-011	FR-011	The widget on the home screen	The app is locked	The widget still shows its last Safe-to-Spend and net worth from Glance state, touching no database
AC-012	FR-012	Cached market data older than its TTL	The user pulls to refresh while offline	Insights recompute, the market figure keeps its "as of <date>" label, and nothing blanks
AC-013	FR-013	The add-transaction screen	The user fills every field	Amount, currency, date, account, category, payee, note, tags, attachments, type and splits all persist and reload
AC-014	FR-014	The dashboard	The user records a common expense	It takes three taps: amount, suggested category, save
AC-015	FR-015	Two accounts	The user records a transfer and then deletes one leg	Both legs disappear together and both balances return to their prior values
AC-016	FR-016	A Rs. 1,000 bill split three ways	The user saves it	The three lines sum to exactly Rs. 1,000 with no paise lost, and the parent amount is unchanged
AC-017	FR-017	A new transaction at a known merchant	It is saved	A category is suggested with a confidence and the evidence behind it, and a correction is remembered for next time
AC-018	FR-018	Two similar transactions at one merchant	The detector runs	A recurring series is proposed, and nothing is created until the user confirms
AC-019	FR-019	10,000 transactions	The user searches, filters and scrolls	Results page in, day groups carry daily totals, and filters combine
AC-020	FR-020	A multi-selection of transactions	The user deletes them	They disappear from every view, an undo appears, and the rows are soft-deleted rather than removed
AC-021	FR-021	A transaction from each source	The row is read	`source` is one of manual, ocr, sms, import, recurring — and `aa` exists but is unused
AC-022	FR-022	A transaction dated next week	Month-to-date spend and the forecast are computed	It is absent from actuals and present in the forecast
AC-023	FR-023	The scan flow	The user picks the camera or the gallery	Either produces an image the parser receives
AC-024	FR-024	Airplane mode	A receipt is scanned	Text is recognised on-device and no socket is opened
AC-025	FR-025	A photographed receipt	Parsing completes	Total, date, merchant and GST appear with per-field confidence, each editable
AC-026	FR-026	A parse that found no amount	The user tries to save	Saving is refused and the missing field is flagged
AC-027	FR-027	A saved receipt transaction	The user deletes the image	The transaction survives, and the encrypted blob is removed
AC-028	FR-028	A manual transaction for Rs. 500 today	A scanned receipt for Rs. 502 dated yesterday arrives	A merge is offered instead of a second row
AC-029	FR-029	A category and a month	The user sets a budget	One budget exists for that category and month, and a second attempt is refused
AC-030	FR-030	Three months of history	The user asks for suggestions	A median-with-seasonality figure is offered per category and accepted one tap at a time
AC-031	FR-031	A budget mid-month	The screen loads	Spent, remaining, pace against safe pace, and the projected end-of-month figure are all shown
AC-032	FR-032	A budget crossing 80% and later 100%	The alert worker runs twice	Exactly one alert fires per band per month
AC-033	FR-033	A closed month	The review opens	Wins and misses are summarised once for the whole month and next month's numbers are proposed
AC-034	FR-034	A new profile	Goals are listed	The emergency fund is present as a system goal, and the other seven types are offered
AC-035	FR-035	A goal with a target and a date	It is saved	Required monthly is computed; no return rate is assumed, and the screen says so
AC-036	FR-036	A goal needing more than the forecast surplus	Feasibility runs	The gap and three levers are shown
AC-037	FR-037	A transaction into a funding account	It is linked to a goal	Progress rises by that amount, and evidenced progress stays distinguishable from declared progress
AC-038	FR-038	Three goals whose contributions exceed the surplus	The waterfall runs	The surplus is split by priority, and the shortfall is named per goal

ID	Parent	Given	When	Then
AC-039	FR-039	The account editor	The user creates one of each type	All eleven types are creatable and appear under the right section
AC-040	FR-040	A card with a statement day of 25	The cycle is computed	Utilisation, unbilled, the statement amount and the due date are right across the month boundary
AC-041	FR-041	A loan's five terms	The schedule is read twice	Both reads are identical to the paise, and the payoff date matches the schedule's last row
AC-042	FR-042	A gold holding and a cached price	The valuation is read	The value uses the cached price with its observation date; with no price, coverage is reported rather than zero
AC-043	FR-043	Assets and liabilities	The daily snapshot runs	Net worth equals assets minus liabilities, and a back-dated entry repairs the series rather than corrupting it
AC-044	FR-044	A real balance that differs from the derived one	The user reconciles	An adjustment transaction is posted, and no stored balance is edited
AC-045	FR-045	An archived account with history	Totals are computed	Its history remains readable and its balance is excluded from active totals
AC-046	FR-046	A holding with dated lots	XIRR is computed	The same lots always produce the same rate, and the result is money-weighted rather than a cost-to-value ratio
AC-047	FR-047	A portfolio over-weight in one asset class	Allocation runs	The concentration warning cites the rule id and version that fired
AC-048	FR-048	A vehicle	The user records odometer, service and renewals	Each persists; there is no fuel log to record into
AC-049	FR-049	A vehicle with a mileage trend	Mileage is analysed	Nothing happens — the fuel log this depends on is not built (`GAP-009`)
AC-050	FR-050	A vehicle with service and renewal history	Monthly cost is shown	Service, amortised renewals and depreciation are included; fuel is absent and the screen does not imply otherwise
AC-051	FR-051	The advisor hub	It opens	Health score, insight feed, purchase advisor, simulators and chat are all reachable
AC-052	FR-052	Any insight	It is rendered	Finding, evidence, confidence and at most one action are present
AC-053	FR-053	An item, a price, a method and an urgency	The advisor runs	One verdict is produced with a gate-by-gate trace, and the trace is stored
AC-054	FR-054	A question the registry cannot answer	Chat answers	It refuses rather than inventing a figure
AC-055	FR-055	Any advisor screen	It is rendered	The informational-not-advice disclaimer is visible
AC-056	FR-056	Settings	It opens	Every listed group is present except a recurring-rules manager screen (`GAP-010`)
AC-057	FR-057	A populated profile	Export then import into a fresh install	Every row returns identical, asserted field by field by the archive format test
AC-058	FR-058	A populated profile	The user erases everything	Biometric and typed confirmation are required, the key is destroyed first, and the report names what could not be reached
AC-059	FR-059	Two granted consents	The user revokes one	The dependent data path stops immediately, and both grant and revoke times are on the record
AC-060	FR-060	A merchant in the knowledge base	Classification runs	A category, a confidence and the tier that fired are returned
AC-061	FR-061	Six months of a rent payment	Stream classification runs	The stream is FIXED, with the variance that decided it
AC-062	FR-062	A transaction on an investment account	Nature classification runs	It is INVESTMENT, and a user override is stored while the inference is not
AC-063	FR-063	90 days of schedules and history	The forecast runs	P10, P50 and P90 paths are produced and every crunch day is listed with its cause
AC-064	FR-064	Two years of one category	Seasonality runs	A monthly index is derived from the user's own history, with the lookback divided out
AC-065	FR-065	Six closed months of NEED spend	The emergency fund runs	A target, a runway in months and a coaching line are produced from the median, not the mean
AC-066	FR-066	A priced portfolio	Intelligence runs	Allocation, diversification and concentration are reported with their rule citations
AC-067	FR-067	An odometer trend and a service history	Prediction runs	A next-service date and a cost range are produced, and no price is invented
AC-068	FR-068	An appliance with a service interval	The engine runs	Next service, consumables, warranty state and a running cost are produced
AC-069	FR-069	A purchase under consideration	The gates run	Each gate reports pass, warn or fail in order, and one verdict follows from them

ID	Parent	Given	When	Then
AC-070	FR-070	A wish on the buy list	The ladder updates	Nothing is removed automatically, and the interview adapts to the answers given
AC-071	FR-071	A profile with partial data	The score runs	A 0–1000 score, a band, every signal's contribution and the strongest lever are returned, reweighted over what is actually present
AC-072	FR-072	A target and a date	Goal maths runs	Required monthly, ETA and horizon class are returned, with no assumed return
AC-073	FR-073	Cached market data and no network	A priced screen loads	The cache renders with its staleness label, and nothing blanks or guesses
AC-074	FR-074	Several candidate notifications inside quiet hours	The gate runs	Nothing is delivered until quiet hours end, and per-type caps hold
AC-075	FR-075	Results from several engines, some duplicated	The orchestrator runs	The feed is deduplicated by fingerprint and ranked, and the ranking is reproducible
AC-076	FR-076	An LLM sentence containing a figure no engine produced	The guardrail runs	The sentence is refused, and the refusal is measured against the frozen set
AC-077	FR-077	A question about the user's own data	Chat plans	Only registry tools are called, and the model receives rendered words rather than rows
AC-078	FR-078	Five years of cached closes	Signals run	A score and a measured hit rate are produced, and what cannot be measured is stated
AC-079	FR-079	A monthly surplus	The waterfall runs	Each of the eight stages reports what it consumed, and the total equals the surplus
AC-080	FR-080	A loan and an investment assumption	The simulator runs	Both branches use the loan engine's own arithmetic, and the comparison states its assumptions
AC-081	FR-081	A year's income and deductions	The tax engine runs	Both regimes are compared, capital gains follow the post-July-2024 rules, and the output names what it left out
AC-082	FR-082	Declared cover and dependants	Protection runs	Term and health gaps are the larger of the two readings, and no surrender is recommended
AC-083	FR-083	Income, obligations and the month to date	Safe-to-Spend runs	A figure is produced with every withheld line labelled, including the 5% buffer
AC-084	FR-084	A locked app	Five wrong PINs are entered	The lockout starts at 30 s and doubles, and no data path opens
AC-085	FR-085	A first run	The key is created	It is 256-bit, wrapped by a hardware-backed Keystore key, never written unwrapped, and rotatable without data loss
AC-086	FR-086	Any screen	A screenshot is attempted	The system refuses it, and the recents thumbnail shows nothing financial
AC-087	FR-087	A backup and its passphrase	Restore on a fresh device	Every row returns; a wrong passphrase fails closed; restoring into another profile is refused
AC-088	FR-088	A release candidate	The drill runs on a device	A failed restore blocks the promotion
AC-089	FR-089	A security event	It occurs	A row lands in `audit_log`, and the export does not carry it
AC-090	FR-090	Each of the four locales	The app is rendered	Every user-facing string is translated, and a missing one fails the build
AC-091	FR-091	Two profiles in one household	An aggregate is read	It is composed from per-profile reads, and no member's rows reach another's screen
AC-092	FR-092	A business book	It is used	It is a second profile, and no transaction carries a business flag
AC-093	FR-093	An import batch	It is later wiped	Every row it created is gone, and the check proves it rather than assuming it
AC-094	FR-094	A future-dated transaction whose day has arrived	The worker has not run	It still counts as actual, because the date decides
AC-095	FR-095	A tampered balance	The nightly job runs	The discrepancy is detected and an adjustment is proposed, not applied silently
AC-096	NFR-001	A mid-range device	Cold start is timed	A measurement exists — today it does not, which is `GAP-004`
AC-097	NFR-002	The add-transaction path	A save is timed	The commit is local and transactional; the 100 ms figure is unmeasured (`GAP-004`)
AC-098	NFR-003	50,000 transactions	A full recompute runs	It happens off the main thread; the 3 s bound is unmeasured (`GAP-004`)
AC-099	NFR-004	A released build	Sessions are observed	Crash-free rate is read from Play vitals — not available before launch (`OQ-005`)
AC-100	NFR-005	A release build	The APK is measured	It is at or below 40 MB

ID	Parent	Given	When	Then
AC-101	NFR-006	Any background need	Work is scheduled	It goes through WorkManager, batched and network-aware, and `GlobalScope` fails the build
AC-102	NFR-007	A write in flight	The process is killed	No data is lost, and the database reopens consistent
AC-103	NFR-008	An API 26 device	The app runs	It installs and every core flow works
AC-104	NFR-009	Airplane mode on a real device	The end-to-end smoke runs	Every core flow completes, proven by a test that fails if the network is reachable
AC-105	NFR-010	200% font, dark mode	Every screen is rendered	No layout breaks, and colour is never the only signal
AC-106	NFR-011	A new user-facing string	It is hardcoded	The build fails
AC-107	NFR-012	A `Double` named like money	It is declared	The build fails
AC-108	NFR-013	A money test is deleted	Coverage is verified	The build fails below 100% on `:core:model`, and below 85% elsewhere
AC-109	NFR-014	A new lint, ktlint or detekt warning	`unitTests` runs	The build fails
AC-110	NFR-015	A dependency with a known advisory	The scan runs	It fails on the version that actually ships, not on one nobody has
AC-111	NFR-016	A release build	The dex is read	No log call survives, and debuggable and backup flags are off
AC-112	NFR-017	A backup	The drill restores it on a device	A failure blocks the promotion
AC-113	NFR-018	A log line naming an amount	It is written	The build fails
AC-114	NFR-019	An insight produced by an older engine version	It is re-read	It renders, and its stored engine version explains it
AC-115	NFR-020	A screen changes	`unitTests` runs	The committed render no longer matches and the build fails until it is re-recorded deliberately

Part 03 — Business Flows and Use Cases

← Master Blueprint · Related: 02 Requirements · 04 Data · 07 Tests

> ***Path naming.*** *`UC-007-A1` is an alternate path (a different legitimate route), `-E1` an exception (the user or their data is wrong), `-F1` a failure (the system or a dependency is wrong, with the recovery behaviour stated). A use case with no exception and no failure path is incomplete, and none here is left that way.* ***Why the failure paths matter more than usual here.*** *There is no server to retry against and no support desk to call. Every failure is handled on the device or it is handled by the user losing data, so each `-F` path names what survives.* ***How to read a use case.*** *Each one opens with a header table — actor, the requirements it implements, the business rules that bind it, the entities it touches, pre- and postconditions, and how often it is expected to run. Then the main flow as numbered steps, then one table per path family. The* ***Branches at*** *column names the main-flow step the path leaves from, so an alternate is anchored to a point in the flow rather than floating beside it. The failure table's last two columns —* ***Recovery*** *and* ***Data left behind*** *— are the two questions an on-device app with no server has to answer for itself.*

1. Business flows

A flow is a process that crosses screens and usually crosses days. The six the SRS names (§25.2) are F1–F6; four more were added as the project shipped, because the code grew paths the SRS never described — consent revocation, erase, chat, and the release gate itself.

ID	Flow	SRS	Actors	Trigger	Outcome
FLOW-001	Capture to insight	F1	User, classifier, orchestrator	The user taps the FAB	A transaction exists, classified, and either an insight card is published or silence is the correct answer
FLOW-002	Receipt to transaction	F2	User, ML Kit, parser	The user opens Scan	A transaction exists with an encrypted image attached, or the flow degraded to manual entry
FLOW-003	Purchase decision	F3	User, advisor engine	The user opens the Purchase Advisor	A verdict with its full reasoning trace, kept as a record
FLOW-004	Crunch rescue	F4	Forecast worker, notification gate, user	The nightly refresh finds a crunch day	The user knows about a shortfall nine days out, with its causes
FLOW-005	Monthly ritual	F5	User, review and budget engines	The first of the month	The closed month is summarised once, and next month's numbers are proposed
FLOW-006	Backup and restore	F6	User, crypto stack	The user enables backup	An encrypted archive exists off-device, and restores on a fresh phone
FLOW-007	Consent lifecycle	—	User	The user opens the consents dashboard	Every consent's state is visible and revocable, and revoking stops the dependent path at once
FLOW-008	Erase	—	User	The user chooses "erase all data"	The key is destroyed, the files are gone, and anything unreachable is named
FLOW-009	Ask a question	—	User, chat engine, on-device LLM	The user opens Chat	An answer in words, every figure in it produced by an engine and checked
FLOW-010	Release	—	Maintainer	A promotion to `stage` or `main`	A build that passed every gate, or a blocked promotion

FLOW-001 — Capture to insight (SRS F1)

- **Trigger:** the user taps the add-transaction FAB.
- **Actors:** User, the classification engine, the insight orchestrator.
- **Systems touched:** Compose UI, `:data:repository`, SQLCipher, the classification / stream / nature / budget / insight engines, WorkManager.
- **Outcome:** one transaction row exists, classified with its confidence and evidence, and the orchestrator has either published a card or decided correctly to say nothing.
- **Why it is the spine:** every figure in the app derives from `ENT-004`. If this flow is slow or lossy, nothing downstream can be right.

#	Step	Actor	System	IDs
1	Tap the FAB from any screen	User	Shell, nav graph	`FR-014`, `ARC-001`

#	Step	Actor	System	IDs
2	Enter the amount on a numeric-first keypad	User	Add-transaction sheet	FR-013, VAL-001
3	Accept or change the suggested category	User	Classification read	FR-017, UC-010
4	Save — one transaction row, with `source = manual`	System	`:data:repository`, Room	FR-021, UC-007, BR-016
5	Stage 1 maps merchant to category, attaching confidence and evidence	Classifier	`:domain:engines: classification`	FR-060, BR-013
6	Stage 2 scores the stream's variance — FIXED, SEMI_FIXED or VARIABLE	Classifier	Stream engine	FR-061, UC-038
7	Nature is assigned from the account type — NEED, WANT, INVESTMENT, ASSET, LIABILITY	Classifier	Nature engine	FR-062
8	The orchestrator is nudged and debounces, so a burst of entries produces one pass	System	Insight orchestrator	FR-052, ADR-0046
9	Budget and pace rules evaluate against the month's envelopes	System	Budget engine, `ai/rules`	FR-031, FR-032, BR-019
10	Results are deduplicated by fingerprint and ranked	System	Insight orchestrator	ENT-025, UC-031
11	A card is published, or the pass ends with nothing worth saying	System	Insight feed	FR-052, UC-031-E1
12	The dashboard and the widget recompute from local data	System	Home, Glance worker	FR-006, FR-011, UC-006

Where it legitimately ends early. Step 11 producing nothing is a success, not a failure: the orchestrator exists to decide that most days do not need a card. Step 5 failing leaves the transaction uncategorised and the flow continues — capture must never depend on inference (UC-007-F2).

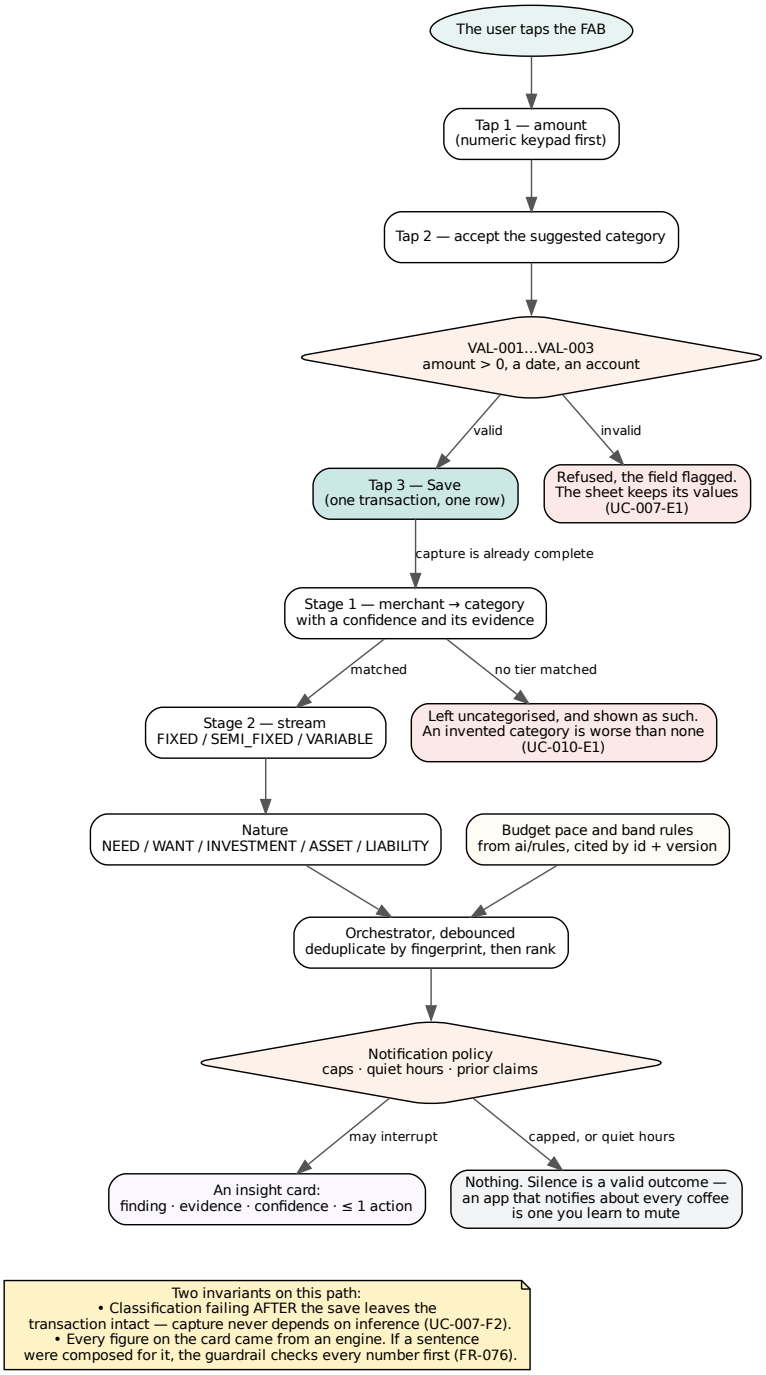


Figure 2 — FLOW-001, capture to insight. The two red boxes are where the path legitimately ends without a card. Source: diagrams/flow-capture.dot.

FLOW-002 — Receipt to transaction (SRS F2)

- **Trigger:** the user opens Scan, from the FAB menu or the receipt tile.
- **Actors:** User, ML Kit text recognition, the receipt parser.
- **Systems touched:** CameraX, ML Kit (on-device), `:domain:engines:receipt`, the encrypted attachment store, `:data:repository`.
- **Outcome:** a transaction with its fields filled from the image, the image stored encrypted and linked, and no byte of the receipt off the device.

- **Privacy note:** FR-024 makes this flow a hard offline path. There is no cloud OCR fallback and the absence is deliberate, not a missing feature.

#	Step	Actor	System	IDs
1	Capture with the camera, or import from the gallery	User	CameraX, SAF	FR-023
2	Recognise text entirely on-device	ML Kit	:ml:ocr	FR-024, P-01
3	Extract total, date, merchant and best-effort tax and lines, each with a confidence	Parser	Receipt engine	FR-025
4	Review: low-confidence fields are flagged, every field is editable	User	Review screen	FR-026, VAL-010
5	Duplicate guard — amount within 1% and date within one day of an existing row	System	Receipt engine	FR-028, UC-014-E2
6	Save the transaction and write the encrypted image as one unit	System	Repository, attachment store	FR-027, BR-016, ENT-015
7	The transaction joins FLOW-001 at step 5 with source = ocr	System	Classification	FR-021, FR-060

Where it legitimately ends early. Step 4 can be abandoned; nothing is written. Step 3 returning nothing useful degrades to manual entry with the image attached, which is still a complete outcome (UC-014-F1).

FLOW-003 — Purchase decision (SRS F3)

- **Trigger:** the user opens the Purchase Advisor, or converts a wishlist item.
- **Actors:** User, the purchase-advisor engine and every engine it reads.
- **Systems touched:** :domain:engines:purchase, forecast, safe-to-spend, goals, budget, emergency fund, ENT-027 and ENT-028.
- **Outcome:** a verdict — Comfortable, Stretch or Not now — with every gate's figures recorded, so the same question asked next month can be compared with this answer.

#	Step	Actor	System	IDs
1	Enter item, price, payment method and urgency	User	Advisor UI	FR-053, VAL-021
2	Gates run in a fixed order, each reading one engine	System	Purchase engine	FR-069, P-08
3	Each gate reports pass, warn, fail — or "cannot assess", when its input is absent	System	Purchase engine	UC-032-F1, ADR-0049
4	One verdict follows from the gate results by a stated rule	System	Purchase engine	FR-053
5	The impact strip shows Safe-to-Spend, the emergency runway and goal ETAs before and after	System	Advisor UI	P-02
6	The trace is stored — one row per gate with the figures it used	System	ENT-027, ENT-028	FR-069, BR-013
7	The user acts: buy, create a goal carrying the price across, or let it go	User	Goals, wishlist	UC-032-A1, P-07

Why the trace is stored and not recomputed. The engines move. A verdict is only defensible if the figures it was given are kept with it (AI-ARC-006).

FLOW-004 — Crunch rescue (SRS F4)

- **Trigger:** the nightly insight refresh finds a day in the next 90 where the P50 liquid balance goes negative.
- **Actors:** the forecast worker, the notification gate, the user.
- **Systems touched:** :domain:engines:forecast, the notification engine, WorkManager, ENT-026, the forecast detail screen.
- **Outcome:** the user learns about a shortfall roughly nine days before it lands, with the obligations that cause it named and moves suggested.

#	Step	Actor	System	IDs
1	The nightly worker projects 90 days as P10 / P50 / P90	System	Forecast engine	FR-063, UC-039
2	Crunch days are identified and each is attributed to the obligations causing it	System	Forecast engine	FR-064
3	A candidate notification is raised at roughly T−9	System	Notification engine	FR-074
4	The gate checks type caps, quiet hours and prior claims for that crunch	System	Notification gate, ENT-026	UC-044, BR-019

#	Step	Actor	System	IDs
5	Delivered, or dropped with the reason recorded	System	WorkManager, notification log	UC-044-E1
6	The user opens the forecast detail and reads the causes	User	Forecast screen	FR-064, P-02
7	Suggested moves are offered — never executed	System	Forecast screen	BR-024, P-07

What is not yet evidenced. Forecast accuracy against actuals is not measured; GAP-003 and ADR-0043 record that the MAPE claim is unbacked, so this flow is honest about projecting but silent about how well.

FLOW-005 — Monthly ritual (SRS F5)

- **Trigger:** the first run after a calendar month closes in the profile's time zone.
- **Actors:** User, the budget-review, budget-suggestion, goal and health-score engines.
- **Systems touched:** ENT-011, ENT-009, ENT-022, the review screen.
- **Outcome:** exactly one review card for the closed month, and next month's budget numbers proposed rather than imposed.

#	Step	Actor	System	IDs
1	The month boundary resolves in the profile's time zone through the injected `Clock`	System	`:core:common`	TIM-001, BR-003
2	One review card is minted, keyed by `(profile, month)`	System	Review engine, ENT-011	FR-033, UC-017-F1
3	Wins and misses per envelope are summarised	System	Budget review	FR-033
4	Next month's numbers are proposed from a three-month median with seasonality	System	Budget suggestion	FR-030, UC-015
5	A goal digest reports each goal's movement and ETA change	System	Goals engine	FR-035, UC-018
6	The health score's change and the reason it moved are shown	System	Health-score engine	FR-071, UC-043
7	The user accepts the proposed numbers, which writes next month's budgets	User	Budget repository	UC-017-A1, BR-016

Why one card and not one per finding. A month that went badly in six envelopes would otherwise interrupt six times. The claim key is the month, so the ritual is a single event (ADR-0020).

FLOW-006 — Backup and restore (SRS F6)

- **Trigger:** the user enables backup in Settings → Data, or restores on a fresh install.
- **Actors:** User, the crypto stack; the maintainer, when it runs as a release gate.
- **Systems touched:** `:sync:backup`, `:core:crypto` (Tink, Argon2id), SAF, the whole schema through the archive serialiser.
- **Outcome:** an AES-256-GCM archive the user holds, which rebuilds the profile on a different phone — or an explicit, early statement that a lost passphrase means lost data.

#	Step	Actor	System	IDs
1	Set a passphrase and acknowledge that it cannot be recovered	User	Backup screen	FR-087, VAL-018
2	Argon2id derives the archive key from the passphrase	System	`:core:crypto`	SEC-005, ADR-0040
3	The archive is serialised with its schema version and every field name pinned	System	Archive serialiser	FR-057, UC-035-F1
4	AES-256-GCM with a unique nonce encrypts it to user-chosen storage	System	`:sync:backup`, SAF	SEC-005
5	On a fresh device: choose the file and enter the passphrase	User	Restore screen	FR-087
6	Decrypt, check the schema version, and replace the profile transactionally	System	Repository	UC-035-E1, BR-016
7	Before any promotion to `stage` or `main`, the drill runs on real hardware	Maintainer	`./gradlew restoreDrill`	FR-088, DRL-001

Why the drill is a gate and not a test. Restore is the one path whose failure is unrecoverable by definition. A red drill blocks the release (ADR-0041).

FLOW-007 — Consent lifecycle

- **Trigger:** the user opens the consents dashboard, or an optional feature asks for a consent.
- **Actors:** User.
- **Systems touched:** the Proto DataStore consent ledger, `:data:sms`, the market cache, `ENT-016`, `ENT-036`, `ENT-013`.
- **Outcome:** every consent's state, purpose and both timestamps are visible, and a revocation stops the dependent data path immediately rather than at the next launch.

#	Step	Actor	System	IDs
1	Every consent is listed with status, plain-language purpose, grant time and revoke time	System	Consents dashboard	`FR-059`, `BR-014`
2	Granting writes the record *before* the feature may run; a failed write fails closed	System	DataStore ledger	`UC-037-F1`, `P-01`
3	The in-app consent — not the OS permission — is the gate	System	Feature entry points	`UC-037-E1`
4	One tap revokes	User	Consents dashboard	`FR-059`
5	The dependent path stops at once	System	SMS repository, market worker	`UC-037`
6	SMS revocation deletes the parsed drafts; market revocation keeps the cached public close	System	`ENT-016`, `ENT-036`	`BR-028`, `ADR-0030`
7	The grant or revoke lands in the audit log	System	`ENT-013`	`FR-089`, `UC-052`

Why the two revocations differ. A parsed bank alert is a statement about the user; a gold price is a statement about gold. One is deleted, the other is kept, and the asymmetry is recorded rather than looking like an oversight.

FLOW-008 — Erase

- **Trigger:** the user chooses "erase all data" in Settings → Data.
- **Actors:** User.
- **Systems touched:** `:core:crypto` (the Keystore-wrapped database key), the erase repository, the attachment store, WorkManager.
- **Outcome:** the ciphertext is unrecoverable from the first irreversible step onward, and anything the sweep could not reach is named rather than assumed gone.

#	Step	Actor	System	IDs
1	The user chooses erase and reads what it destroys	User	Settings	`FR-058`
2	Biometric authentication	User	BiometricPrompt	`SEC-002`, `UC-036-E1`
3	A typed confirmation phrase	User	Erase dialog	`VAL-019`
4	**The wrapped database key is destroyed first**	System	Keystore, `:core:crypto`	`SEC-006`, `BR-015`
5	Database files, attachments and caches are removed	System	Erase repository	`FR-058`
6	Scheduled work is cancelled and the consent ledger is cleared	System	WorkManager, DataStore	`FR-058`
7	A report names anything unreachable — a file on removable storage, say	System	Erase screen	`UC-036-F1`

Why the order is not negotiable. If the sweep is interrupted after step 4, the data is already unreadable. Any other order leaves a window in which the files are gone but readable copies could survive (`ADR-0060`).

FLOW-009 — Ask a question

- **Trigger:** the user opens Chat and asks something in words.
- **Actors:** User, the chat engine, the on-device LLM, the guardrail.
- **Systems touched:** `:domain:engines:chat`, the tool registry, `:ml:llm`, `ai/chat/guardrail.md`, `ENT-035`.
- **Outcome:** an answer in plain language in which every figure was computed by a deterministic engine — or an explicit refusal.

#	Step	Actor	System	IDs
1	The question is planned into tool calls	System	Chat engine	FR-054
2	Only tools in the registry may run; nothing else is callable	System	Tool registry	FR-077, AI-ARC-004
3	Each tool returns an engine result with its provenance	System	Engines	BR-013
4	Figures are rendered to words by the allowlisted renderers	System	Verbaliser	BR-012, ADR-0048
5	The model composes a sentence from those rendered words only	On-device LLM	:ml:llm	P-03, BR-011
6	The guardrail checks every figure in the output against what the engines produced	System	Guardrail	FR-076, AI-ARC-004
7	The reply is shown with its tool calls as sources, and stored	System	Chat UI, ENT-035	FR-054, P-02

The two ways it ends without prose. No tool can answer — chat says so and names what it can answer instead (UC-033-E1). Or the model is unavailable or too slow, and the rendered figures are shown without a sentence wrapped round them (UC-033-F2).

FLOW-010 — Release

- **Trigger:** a promotion PR into stage or main.
- **Actors:** Maintainer.
- **Systems touched:** Gradle, GitHub Actions, an emulator or device, SonarQube.
- **Outcome:** a build that passed every gate, or a promotion that is blocked with the failing gate named.

#	Step	Gate	Command	IDs
1	Unit tests across every module, including the pure-Kotlin ones	Build	./gradlew unitTests	NFR-012, DoD
2	Coverage floors — 85% engines, 100% money math	Build	./gradlew koverVerify	NFR-012
3	ktlint, detekt, Android lint and the six custom detectors	Build	./gradlew lintDebug :lint	NFR-013, ARC-002, ARC-006
4	AI evaluation sets against frozen labelled data	Build	./gradlew aiEval	NFR-019, FR-060
5	Paparazzi screenshot diff in light, dark and 200% font	Build	./gradlew verifyPaparazziDebug	NFR-010, ADR-0082
6	Dependency vulnerability scan of what actually ships	Build	OSV gate	SEC-007
7	Offline smoke — every core feature in airplane mode	Device	./gradlew offlineSmoke	P-04, NFR-009
8	**Backup restore drill on real hardware**	Device	./gradlew restoreDrill	DRL-001, FR-088
9	SonarQube clean, with the file and ncloc denominator checked	Pre-commit	./gradlew sonarAnalysis	ADR-0081

Which of these actually block. Steps 1–6 fail the build. Steps 7–9 are agent-followed and documented as such in CLAUDE.md §8.1 and §10 — the repository says plainly that nothing in Gradle fails because somebody skipped them.

2. Use-case index

Every use case is declared here once, then detailed in §3.

UC	Name	Actor	Requirements	Flow
UC-001	Complete onboarding	User	FR-001, FR-003, FR-005	FLOW-001
UC-002	Seed the plan with quick setup	User	FR-002	FLOW-001
UC-003	Explore in demo mode	User	FR-004	—
UC-004	Read the dashboard	User	FR-006, FR-007, FR-009, FR-012, FR-083	FLOW-001
UC-005	Hide the amounts	User	FR-008, FR-086	—
UC-006	Glance at the widget	User	FR-011	—
UC-007	Record a transaction	User	FR-013, FR-014, FR-021	FLOW-001
UC-008	Move money between accounts	User	FR-015	—
UC-009	Split a bill	User	FR-016	—
UC-010	Categorise, and correct	User, classifier	FR-017, FR-060	—

UC	Name	Actor	Requirements	Flow
UC-011	Confirm a recurring series	User	FR-018	—
UC-012	Find a transaction	User	FR-019, FR-020	—
UC-013	Date something in the future	User, scheduler	FR-022, FR-094	—
UC-014	Scan a receipt	User, ML Kit	FR-023…FR-028, FR-024	FLOW-002
UC-015	Set and watch a budget	User	FR-029, FR-030, FR-031	—
UC-016	Be warned about a budget	System, user	FR-032	FLOW-005
UC-017	Review a closed month	User	FR-033	FLOW-005
UC-018	Create a goal	User	FR-034, FR-035, FR-072	—
UC-019	Discover that the goals do not fit	User	FR-036, FR-038	—
UC-020	Evidence a goal contribution	User	FR-037	—
UC-021	Keep the accounts	User	FR-039, FR-045	—
UC-022	Understand a credit card	User	FR-040	—
UC-023	See what a loan really costs	User	FR-041	—
UC-024	Value an asset that has a market price	User, market worker	FR-042, FR-073	—
UC-025	Watch net worth over time	User, snapshot worker	FR-043	—
UC-026	Reconcile with reality	User, integrity worker	FR-044, FR-095	—
UC-027	See a holding's real return	User	FR-046	—
UC-028	Check how the portfolio is spread	User	FR-047, FR-066	—
UC-029	Know when the vehicle is due	User	FR-048, FR-049, FR-050, FR-067	—
UC-030	Open the advisor	User	FR-051, FR-055	—
UC-031	Work the insight feed	User	FR-052, FR-075	—
UC-032	Ask whether a purchase fits	User	FR-053, FR-069	FLOW-003
UC-033	Ask a question in words	User	FR-054, FR-076, FR-077	FLOW-009
UC-034	Change a setting	User	FR-056	—
UC-035	Take the data out, and bring it back	User	FR-057	FLOW-006
UC-036	Erase everything	User	FR-058	FLOW-008
UC-037	Grant and revoke a consent	User	FR-059	FLOW-007
UC-038	Learn what recurs and what a rupee became	System	FR-061, FR-062	—
UC-039	See the next ninety days	System, user	FR-063, FR-064	FLOW-004
UC-040	Size the emergency fund	System, user	FR-065	—
UC-041	Keep an appliance serviced	User	FR-068	—
UC-042	Keep a wish out of the ledger	User	FR-070	—
UC-043	Read the health score	User	FR-071	—
UC-044	Decide whether to interrupt	System	FR-074	—
UC-045	Ask whether today is unusual	User	FR-078	—
UC-046	Send the surplus somewhere	User	FR-079	—
UC-047	Simulate prepaying a loan	User	FR-080	—
UC-048	Estimate the tax position	User	FR-081	—
UC-049	Check whether the cover is enough	User	FR-082	—
UC-050	Unlock the app	User	FR-084, FR-085	—
UC-051	Back up, and restore on a new phone	User, maintainer	FR-087, FR-088	FLOW-006
UC-052	Leave a record of what was allowed	System	FR-089	—
UC-053	Use the app in another language	User	FR-090	—
UC-054	Keep two books on one phone	User	FR-091, FR-092	—
UC-055	Import a statement	User	FR-093	—

3. Use cases — detail

UC-001 — Complete onboarding

Actor	User
Implements	FR-001, FR-003, FR-005
Business rules	BR-014, BR-017
Validation	VAL-006, VAL-008, VAL-017
Entities touched	ENT-001, ENT-002, ENT-003, ENT-013
Flow	FLOW-001
Preconditions	A fresh install; no profile row exists

Postconditions (success)	A profile, a security method and one account exist; the SMS decision is recorded whichever way it went
Frequency	Once per install

Main flow

1. The welcome screen states the privacy pledge: nothing leaves the device without consent (`P-01`).
2. The user names the profile and picks the currency and number format; the profile's time zone is captured here, because every later calendar decision resolves in it (`TIM-001`).
3. The user sets up the app lock — BiometricPrompt class 3 where the hardware allows, a PIN otherwise.
4. The SMS step explains exactly what would be read, shows a worked example, and offers skip with equal weight.
5. The user adds a first account with its opening balance, which is the only balance the app is ever told rather than deriving (`BR-007`).
6. The dashboard opens, within the three minutes `FR-005` allows.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-001-A1	step 4	The user skips quick setup	Budgets and the emergency-fund target stay unseeded until set by hand; no envelope is invented
UC-001-A2	step 3	The device has no biometric hardware or no enrolment	The PIN is the only method offered, and the screen says why rather than showing a disabled control
UC-001-A3	step 1	The user chooses demo mode instead	`UC-003` runs; no profile is created

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-001-E1	step 2	A blank profile name	The step is blocked with the field flagged	VAL-008
UC-001-E2	step 5	A missing or non-numeric opening balance	The step is blocked with the field flagged	VAL-006
UC-001-E3	step 3	The user denies biometric enrolment mid-flow	The step falls back to a PIN rather than failing the flow	VAL-017

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-001-F1	Keystore key generation fails	The crypto call returns an error before any row is written	Onboarding stops at the security step with an explicit error	Retry is a clean first run	None — no profile row was written
UC-001-F2	The process dies between steps	Next launch finds no completed profile	Onboarding restarts	Restart from step 1	None — each step is one transaction
UC-001-F3	StrongBox is absent on the device	The key request falls back at generation time	Nothing visible; the key is still hardware-backed where possible	None needed	A key wrapped by the available Keystore tier

UC-002 — Seed the plan with quick setup

Actor	User
Implements	FR-002
Business rules	BR-016, BR-018
Validation	VAL-011
Entities touched	ENT-009, ENT-022
Flow	FLOW-001
Preconditions	Onboarding reached the quick-setup step
Postconditions (success)	Budget rows and an emergency-fund target exist as real rows, not as remembered answers (ADR-0004)
Frequency	Once per profile

Main flow

1. The user enters monthly income.
2. The user enters rent or EMI.
3. The user enters typical monthly savings.
4. The engine derives an envelope per nature and an emergency-fund target from the named rule rows.
5. Every derived row is written in one transaction.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-002-A1	step 3	The user enters income only	The engine seeds what that supports and leaves the rest unset rather than guessing
UC-002-A2	step 5	A budget already exists for a category	The existing row wins; quick setup never overwrites a figure the user chose

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-002-E1	step 3	Savings exceed income	The combination is refused and the screen says which figure to change	VAL-011
UC-002-E2	step 1	Income of zero	Seeding is skipped; an envelope derived from zero would be meaningless	VAL-011

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-002-F1	The write fails part-way	Transaction abort	The user is returned to the step with their figures intact	Retry	None — the seed is one transaction (BR-016)
UC-002-F2	A threshold is hardcoded rather than read from `ai/`	A test asserts the typed mirror matches the data row	None	Build failure before release	None (ADR-0005, ADR-0017)

UC-003 — Explore in demo mode

Actor	User
Implements	FR-004

Business rules	BR-005, BR-006
Validation	—
Entities touched	ENT-002, and every profile-scoped table under the demo `profile_id`
Flow	—
Preconditions	None — demo mode needs no profile and no security setup
Postconditions (success)	Demo rows exist under their own `profile_id`; leaving hard-deletes every one of them
Frequency	Rare — an evaluation path, not a daily one

Main flow

1. The user picks demo mode from the welcome screen.
2. A sample profile with realistic Indian data is generated under a dedicated `profile_id`.
3. Every screen works against it, including the engines and the advisor.
4. Exiting demo mode hard-deletes every demo row — the one place the app does not soft-delete.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-003-A1	step 3	The user creates a real profile while demo data exists	The two never mix, because every query is scoped by `profile_id` (BR-006)
UC-003-A2	step 4	The user re-enters demo mode later	A fresh sample is generated; nothing from the earlier session survives

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-003-E1	step 3	The user tries to export from demo mode	Refused — an archive of fictional data would be indistinguishable from their own later	VAL-020

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-003-F1	The hard delete is interrupted	The demo profile is re-detected on next launch	A prompt to finish clearing the sample	The delete is retried	Demo rows under the demo `profile_id`, which can never be mistaken for real data (ADR-0006)

UC-004 — Read the dashboard

Actor	User
Implements	FR-006, FR-007, FR-009, FR-012, FR-083
Business rules	BR-011, BR-013
Validation	—
Entities touched	ENT-003, ENT-004, ENT-009, ENT-014, ENT-022, ENT-025 (all reads)
Flow	FLOW-001
Preconditions	The app is unlocked and one profile is selected
Postconditions (success)	None — this is a read. The screen's own state is not persisted beyond the card layout
Frequency	Several times a day; this is the app's front door

Main flow

1. The user opens the app, past the lock gate.
2. Net worth with its 30-day sparkline, from the snapshot series.
3. Safe-to-Spend today, with a "why" link to its inputs.
4. Month-to-date spend against budget.
5. Obligations due in the next 14 days.
6. The top three insights from the feed.
7. Goal progress chips.
8. Any figure taps through to the screen that explains it (`FR-007`).

Alternate flows

ID	Branches at	Condition	Behaviour
UC-004-A1	step 2	A new profile with no history	Every card states what it needs rather than showing a zero
UC-004-A2	step 1	The user pulls to refresh	Insights recompute; market data refreshes only when online and consented, and carries an "as of" label
UC-004-A3	step 7	The user reorders or hides cards in edit mode	The layout is stored per profile (`FR-010`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-004-E1	step 3	No income is declared, so Safe-to-Spend has no basis	The card names the missing input and links to it, rather than showing Rs. 0	—
UC-004-E2	step 5	No obligations in the window	The card says the next 14 days are clear, which is information, not an empty state	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-004-F1	An engine throws	A typed `Result` at the repository boundary	That one card renders an error state; the rest of the dashboard still draws	Pull to refresh	None
UC-004-F2	The database cannot be opened because the app is locked	The lock gate is checked before the database provider builds	The user is sent to the lock screen	Unlock	None (ADR-0003)
UC-004-F3	The dashboard exceeds the 500 ms budget on a mid-range device	A macrobenchmark against `NFR-001`	Slower first paint	Profiling; cards render independently so no card blocks another	None

UC-005 — Hide the amounts

Actor	User
Implements	FR-008, FR-086
Business rules	BR-014
Validation	—
Entities touched	None — the state lives in Proto DataStore
Flow	—
Preconditions	Any screen showing a monetary figure

Postconditions (success)	Every amount in the app and in the widget is masked, and the state persists for the session
Frequency	Situational — in public, on a shared screen

Main flow

1. The user taps the eye control.
2. Every amount becomes a fixed-width mask, so the width itself leaks nothing about the magnitude.
3. `FLAG_SECURE` is set, blocking screenshots and the recents thumbnail.
4. The widget masks too, because it renders from the same state.
5. The state persists for the session.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-005-A1	step 3	The user has deliberately enabled the screenshot setting	The `FLAG_SECURE` guard relaxes only while that setting is on
UC-005-A2	step 5	The app is backgrounded and returns	The mask survives; it is a session state, not a per-screen one

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-005-E1	step 2	A screen renders a figure without going through the masked formatter	The figure would leak; a screenshot test over every screen is what catches it (ADR-0082)	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-005-F1	The blur state cannot be read from DataStore	The read returns an error	The app fails **closed** — masked	Retry on next read	None; showing amounts by accident is the worse error (ADR-0022)

UC-006 — Glance at the widget

Actor	User
Implements	FR-011
Business rules	—
Validation	—
Entities touched	None at render time — Glance state only
Flow	—
Preconditions	The widget is placed on the home screen
Postconditions (success)	None — a read
Frequency	Passive; every time the user looks at their home screen

Main flow

1. The widget sits on the home screen.

2. A worker computes Safe-to-Spend and net worth and writes them into Glance state.
3. The widget renders from that state, with no database access on the render path.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-006-A1	step 3	The privacy blur is on	The widget shows masks (`UC-005`)
UC-006-A2	step 2	The worker has not run since the last change	The widget shows the last computed figures; staleness is bounded by the worker's cadence

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-006-E1	step 3	No figure has ever been computed	The widget shows its empty state rather than a zero	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-006-F1	The app is locked, so the encrypted database would throw	The render path never touches Room	The widget still renders from Glance state	None needed	None (ADR-0024)
UC-006-F2	The worker is deferred by Doze	The written state's timestamp is older than the cadence	Figures are slightly stale	The next worker run	None

UC-007 — Record a transaction

Actor	User
Implements	FR-013, FR-014, FR-021
Business rules	BR-001, BR-003, BR-013
Validation	VAL-001, VAL-002, VAL-003
Entities touched	ENT-004, ENT-008, ENT-005, ENT-006, ENT-007, ENT-015
Flow	FLOW-001
Preconditions	At least one account exists
Postconditions (success)	One transaction row, classified, with `source` recorded and its provenance attached
Frequency	The most frequent write in the app — several times a day

Main flow

1. The user taps the FAB from any screen.
2. The amount is entered on a numeric-first keypad, in `Long` paise throughout (`MNY-001`).
3. A suggested category is accepted.
4. Save writes one row with `source = manual`.
5. Classification runs on the saved row.
6. The insight orchestrator is nudged and debounces.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-007-A1	step 3	The user expands splits, tags, a note or an attachment	The extra rows are written with the parent in one transaction (`UC-009`, BR-016)
UC-007-A2	step 2	The user picks income or transfer instead of expense	Type is derived from the leg, never supplied (`UC-008`, BR-009)
UC-007-A3	step 2	The user dates the transaction ahead	It is excluded from actuals and included in the forecast (`UC-013`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-007-E1	step 2	A zero or empty amount	Save is refused with the field flagged	VAL-001
UC-007-E2	step 1	No account exists yet	The flow offers to create one rather than dead-ending	VAL-003
UC-007-E3	step 2	A date outside the accepted range	Save is refused and the range is stated	VAL-002

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-007-F1	The insert fails	The repository returns a typed error	The sheet stays open with the values intact and the error is shown	Retry from the open sheet	None
UC-007-F2	Classification throws after the row is saved	The classification call is separate from the insert	The transaction appears uncategorised	The user categorises, or a later pass does	A valid, uncategorised transaction — capture never depends on inference
UC-007-F3	A `Double` is introduced into the amount path	`CfoMoneyAsFloatingPoint`	None	Build failure before release	None (ADR-0001)

UC-008 — Move money between accounts

Actor	User
Implements	FR-015
Business rules	BR-009, BR-016
Validation	VAL-005
Entities touched	ENT-004 (two rows sharing a `transfer_id`), ENT-003
Flow	—
Preconditions	Two distinct accounts exist
Postconditions (success)	Exactly two transaction rows share one `transfer_id`; both balances move; neither leg can exist alone
Frequency	Weekly to monthly for most users

Main flow

1. The user picks transfer in the add sheet.
2. Source, destination and amount are entered.
3. Save writes two legs sharing one `transfer_id` inside one transaction.

4. Both account balances move, each derived from its own transactions.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-008-A1	step 3	The destination is a goal's funding account	Evidenced goal progress advances (`UC-020`)
UC-008-A2	step 2	The transfer is dated ahead	Both legs carry the same date, so neither side posts early (`UC-013-E1`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-008-E1	step 2	Source equals destination	Refused	VAL-005
UC-008-E2	step 2	One side is an archived account	Refused, with the archive offered for reactivation	VAL-003

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-008-F1	The second leg fails to write	Transaction abort	"Could not record the transfer" with the values intact	Retry	None — a one-legged transfer is unrepresentable (BR-016)
UC-008-F2	One leg is deleted by a later bug or an import	The nightly integrity job finds the orphan	The discrepancy is raised as an adjustment prompt	The delete path removes both by `transfer_id`; the job raises what slipped through	An orphan leg, named rather than silent (`FR-095`)

UC-009 — Split a bill

Actor	User
Implements	FR-016
Business rules	BR-001, BR-008
Validation	VAL-004
Entities touched	ENT-005, ENT-004, ENT-008
Flow	—
Preconditions	A transaction being created or edited
Postconditions (success)	N split lines whose amounts sum **exactly** to the parent, each with its own category
Frequency	Occasional — a shared bill, a mixed basket

Main flow

1. The user expands splits on a transaction.
2. Lines are added, each with its own category.
3. The remainder is distributed largest-remainder, not `HALF_EVEN` per line.
4. The lines sum exactly to the parent amount, with no drift.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-009-A1	step 2	The user splits by percentage	The engine converts to paise and still makes the lines sum exactly
UC-009-A2	step 2	The user removes a line after entering several	The remainder is redistributed; the sum identity still holds

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-009-E1	step 4	The lines do not sum to the parent	Save is refused with the difference named	VAL-004
UC-009-E2	step 2	A line with no category	Allowed — a line without a category is still a line; the parent's category does not silently fill it	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-009-F1	A rounding path produces a one-paisa drift	A property test asserts the sum identity over generated inputs	None	Build failure before release	None (ADR-0009)
UC-009-F2	The lines write but the parent update fails	Transaction abort	The split is not saved	Retry	None (BR-016)

UC-010 — Categorise, and correct

Actor	User, the classification engine
Implements	FR-017, FR-060
Business rules	BR-018, and `P-02`
Validation	VAL-009
Entities touched	ENT-004, ENT-008
Flow	FLOW-001
Preconditions	A saved transaction
Postconditions (success)	The transaction carries a category, a confidence and the evidence that produced it; a correction is remembered for that merchant
Frequency	Once per transaction, so as often as UC-007

Main flow

1. A transaction is saved.
2. Stage 1 maps the merchant to a category from the knowledge base.
3. A confidence and the evidence — which rule or mapping fired — are attached.
4. The user accepts or corrects.
5. A correction is remembered for that merchant and offered against similar past transactions.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-010-A1	step 2	The merchant is unknown	The lower tier fires on the description, with a lower confidence stated
UC-010-A2	step 5	The user accepts the offer to apply the correction backwards	Similar past transactions are recategorised in one undoable batch (`FR-020`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-010-E1	step 2	No tier matches	The transaction stays uncategorised and is shown as such — an invented category is worse than none	—
UC-010-E2	step 4	The user corrects to a category that does not exist	The category is created first, with its name validated	VAL-009

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-010-F1	The knowledge base and its typed mirror disagree	`ClassificationKbDriftTest`	None	Build failure before release	None — the two can never diverge in a shipped binary (ADR-0014)
UC-010-F2	Accuracy falls below the frozen set's 92% floor	`./gradlew aiEval`	None	The merge is blocked	None (`NFR-019`)

UC-011 — Confirm a recurring series

Actor	User
Implements	FR-018
Business rules	BR-018, and `P-07`
Validation	—
Entities touched	ENT-012, ENT-004
Flow	—
Preconditions	Two or more transactions matching on merchant, amount and cadence
Postconditions (success)	A `recurring_rule` row exists with its cadence and next due date — created only by confirmation, never automatically
Frequency	A handful of times in the first months, then rarely

Main flow

1. Two or more similar transactions match on merchant, amount and cadence.
2. A series is proposed, with the evidence that produced it.
3. The user confirms, edits or dismisses.
4. A `recurring_rule` row is created on confirmation.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-011-A1	step 3	The user edits the cadence before confirming	The edited cadence is stored, not the detected one
UC-011-A2	step 3	The user dismisses	Nothing is written, and the same series is not re-proposed from the same evidence

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-011-E1	step 2	The amount varies beyond tolerance	The proposal states the range rather than a single figure	—
UC-011-E2	step 2	The cadence is irregular	No series is proposed; a false series would mislead the forecast	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-011-F1	The detector proposes a duplicate of an existing rule	The rule's identity is checked before proposing	Nothing is shown	None needed	None
UC-011-F2	The recurring-rules manager that `FR-056` names does not exist	Traced in `GAP-010`	A confirmed rule cannot be edited from Settings	Edit through the proposal path	A rule the user cannot reach to change (`GAP-010`)

UC-012 — Find a transaction

Actor	User
Implements	FR-019, FR-020
Business rules	BR-005, BR-006
Validation	—
Entities touched	ENT-004, ENT-006, ENT-007, ENT-008, ENT-003 (all reads, plus soft-delete writes)
Flow	—
Preconditions	Transactions exist
Postconditions (success)	None for a search; a bulk edit writes categories, tags or soft deletes with an undo window
Frequency	Weekly, and whenever a figure is questioned

Main flow

1. Search by payee, note, amount or tag.
2. Filter by date, account, category, type or amount range.
3. Results group by day with per-day totals.
4. Paging loads more on scroll.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-012-A1	step 3	The user multi-selects	Bulk recategorise, retag or delete with an undo snackbar and a seven-day soft delete

ID	Branches at	Condition	Behaviour
UC-012-A2	step 2	The user arrives from a dashboard figure	The filters arrive pre-set to the figure's scope, so the list explains the number (`FR-007`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-012-E1	step 3	No results	The empty state names which filter excluded everything	—
UC-012-E2	step 2	An amount range with the bounds reversed	The bounds are swapped rather than returning nothing	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-012-F1	A paging query fails mid-scroll	The paging source reports an error	The list keeps what it has and shows a retry rather than clearing	Retry the page	None
UC-012-F2	A query omits its profile scope	`ProfileScopingTest`	None	Build failure before release	None (BR-006)

UC-013 — Date something in the future

Actor	User, the posting worker
Implements	FR-022, FR-094
Business rules	BR-010
Validation	VAL-002
Entities touched	ENT-004
Flow	—
Preconditions	A transaction being created
Postconditions (success)	The row exists, is excluded from actuals until its date, is included in the forecast, and counts as actual on its day whether or not a worker ran
Frequency	Monthly — rent, EMIs, known bills

Main flow

1. The user dates a transaction ahead.
2. It is excluded from actuals and included in the forecast.
3. On its day it counts as actual, decided by the date rather than by a posting event.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-013-A1	step 1	The user back-dates instead	The net-worth series recomputes the affected days rather than leaving a step (ADR-0012)
UC-013-A2	step 2	The future-dated row is edited before its date	The forecast recomputes; nothing has posted, so nothing has to be reversed

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-013-E1	step 1	A future-dated transfer	Both legs carry the same date, so neither side posts early	VAL-005
UC-013-E2	step 1	A date far beyond the forecast horizon	Accepted and stored, but excluded from the 90-day projection, which says so	VAL-002

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-013-F1	The posting worker never runs — the device was off for a week	Nothing to detect	Figures are correct anyway	None needed	None: correctness is decided by the date, and the worker only updates derived state (ADR-0010)
UC-013-F2	The device time zone changes while a row sits on the boundary	The month resolves through the profile time zone, not the device's	The row stays in the month the user meant	None needed	None (`TIM-001`)

UC-014 — Scan a receipt

Actor	User, ML Kit
Implements	FR-023, FR-024, FR-025, FR-026, FR-027, FR-028
Business rules	BR-016, and `P-01`
Validation	VAL-010
Entities touched	ENT-004, ENT-015
Flow	FLOW-002
Preconditions	Camera permission, or a gallery image
Postconditions (success)	One transaction with `source = ocr` and, unless the user removed it, one encrypted attachment linked by content hash
Frequency	Daily for users who keep paper receipts

Main flow

1. Capture with the camera, or import from the gallery.
2. On-device text recognition, with nothing leaving the device.
3. Total, date, merchant and best-effort tax and line items are extracted, each with a confidence.
4. The user reviews and corrects; low-confidence fields are flagged.
5. The duplicate guard checks amount within 1% and date within one day.
6. Save writes the transaction and the encrypted image as one unit.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-014-A1	step 1	The user imports from the gallery rather than the camera	The same pipeline runs; the source is still recorded as OCR
UC-014-A2	step 6	The user deletes the image after saving	The transaction is kept and the attachment row is soft-deleted (`FR-027`)
UC-014-A3	step 1	A multi-page receipt	Pages may be stitched before recognition (`FR-023`, MAY)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-014-E1	step 4	No amount or no date was found	Save is refused and the field is flagged	VAL-010
UC-014-E2	step 5	A duplicate is detected	A merge is offered instead of creating a second row	—
UC-014-E3	step 1	Camera permission is denied	Gallery import remains available; the flow does not dead-end	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-014-F1	Recognition fails on a crumpled receipt	Confidences at or near zero	Every field is editable, so the flow degrades to manual entry	Type the fields	A valid transaction with the image attached
UC-014-F2	The attachment write fails	Transaction abort	Nothing is saved	Retry	None — a receipt row never points at a blob that is not there
UC-014-F3	A blob survives with no row pointing at it	The orphan-blob case is unhandled	Disk use grows silently	Manual clear	An orphan encrypted blob — recorded as `RISK-009`, not fixed (ADR-0078)

UC-015 — Set and watch a budget

Actor	User
Implements	FR-029, FR-030, FR-031
Business rules	BR-018, BR-019
Validation	VAL-011
Entities touched	ENT-009, ENT-008, ENT-004
Flow	—
Preconditions	At least one category
Postconditions (success)	One budget row per category per month, with its nature, and optionally a rollover of the unused amount
Frequency	Monthly, and once at setup

Main flow

1. Pick a category and a month.
2. Accept the suggestion, or type a figure.
3. The screen shows spent, remaining, the daily pace against safe pace, and the projected end-of-month figure.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-015-A1	step 2	The user accepts all suggestions at once	One tap per category, written in one batch
UC-015-A2	step 1	A category group rather than a single category	The envelope covers the group, and the children roll into it (`FR-029`)
UC-015-A3	step 3	Rollover is enabled	The unused amount carries into the next month's envelope

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-015-E1	step 1	A budget already exists for that category and month	Refused by a unique index, with the existing one offered for editing	VAL-011
UC-015-E2	step 2	A negative or zero envelope	Refused	VAL-011

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-015-F1	Fewer than three months of history, so a median is not meaningful	The suggestion engine counts closed months first	The suggestion is withheld and says why	Set the figure by hand	None — nothing is extrapolated from one month
UC-015-F2	The projection assumes an even daily pace that the user does not follow	Visible in the projected figure versus the outcome	The projection is labelled a projection	None needed	None

UC-016 — Be warned about a budget

Actor	The alert worker, then the user
Implements	FR-032, FR-074
Business rules	BR-019
Validation	—
Entities touched	ENT-010, ENT-009, ENT-026
Flow	FLOW-005
Preconditions	A budget exists for the current month, with spend against it
Postconditions (success)	One alert per band per month, claimed in `budget_alert` so it cannot repeat
Frequency	At most twice per budget per month — 80% and 100%

Main flow

1. The alert worker runs.
2. Spend crosses 80% of the envelope.
3. The notification gate checks caps, quiet hours and prior claims.
4. One warning is delivered, and the band is claimed.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-016-A1	step 2	Spend crosses 100% later in the same month	A second, different alert fires once, under its own band claim
UC-016-A2	step 2	The projected overshoot exceeds 10% before the 20th	A pace alert fires, which is a third distinct type (`FR-032`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-016-E1	step 3	The band was already claimed this month	Nothing is delivered	—
UC-016-E2	step 3	The crossing happens inside quiet hours	A non-critical alert waits rather than firing (`UC-044-E1`)	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-016-F1	The worker is skipped by Doze for a day	The claim is keyed by band and month	The alert fires late rather than twice or never	The next worker run	A claim row recording when it actually fired
UC-016-F2	Two worker runs overlap	The claim is written in the same transaction as the delivery decision	One alert	None needed	One claim row

UC-017 — Review a closed month

Actor	User
Implements	FR-033
Business rules	BR-018
Validation	—
Entities touched	ENT-011, ENT-009, ENT-004
Flow	FLOW-005
Preconditions	A month has closed in the profile's time zone
Postconditions (success)	Exactly one review row for `(profile, month)`, and optionally next month's budget rows
Frequency	Monthly

Main flow

1. The month closes.
2. One review card is minted, keyed by profile and month.
3. Wins and misses per envelope are summarised.
4. Next month's numbers are proposed.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-017-A1	step 4	The user accepts the proposed numbers	Next month's budgets are written in one transaction
UC-017-A2	step 2	The user opens the app weeks after the month closed	The card is minted on first run after the boundary, not on the first of the month exactly

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-017-E1	step 3	The month had no spend	The card says so rather than reporting a perfect score	—
UC-017-E2	step 3	No budgets existed for the month	The card reports the spend and says there was nothing to measure it against	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-017-F1	The card is minted twice by two worker runs	The `(profile, month)` unique key	One card	None needed	One row — a second mint is impossible (ADR-0020)

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-017-F2	Six envelopes went badly and each would interrupt	One card covers the whole month by design	One interruption	None needed	One row

UC-018 — Create a goal

Actor	User
Implements	FR-034, FR-035, FR-072
Business rules	BR-018, and `P-02`
Validation	VAL-012
Entities touched	ENT-022, ENT-024, ENT-003
Flow	—
Preconditions	A profile exists; the emergency fund is a system goal and is always present
Postconditions (success)	A goal row with target, optional date, priority and funding accounts, plus a computed required monthly contribution
Frequency	A few times a year

Main flow

1. Pick a type — emergency fund, purchase, travel, home down-payment, education, retirement, debt payoff or custom.
2. Enter the target amount and, optionally, a target date.
3. Set the priority and the funding accounts.
4. The engine computes the required monthly contribution and the ETA.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-018-A1	step 2	An open-ended goal with no date	The engine reports an ETA from the current contribution rather than a required monthly
UC-018-A2	step 1	The goal comes from a purchase verdict	The price is carried across from the trace (`UC-032-A1`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-018-E1	step 2	A target date in the past	Refused	VAL-012
UC-018-E2	step 2	A target of zero	Refused	VAL-012
UC-018-E3	step 3	No funding account chosen	Allowed; progress can then only be declared, not evidenced (`UC-020-A1`)	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-018-F1	The user expects compounding on the target	The engine assumes **no** rate of return, because no rate exists anywhere in `ai/`	The screen states the assumption	None needed	None (ADR-0033)

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-018-F2	The required monthly exceeds any plausible surplus	Feasibility runs against the forecast	The infeasibility is reported with three levers (`UC-019`)	Change date, target or contribution	A goal the app says does not fit

UC-019 — Discover that the goals do not fit

Actor	User
Implements	FR-036, FR-038
Business rules	BR-023
Validation	—
Entities touched	ENT-022, ENT-023 (reads), the forecast engine's output
Flow	—
Preconditions	Two or more goals with required contributions, and a forecast surplus to test them against
Postconditions (success)	None stored — the split is derived; what changes is what the user decides
Frequency	Whenever a goal is added or re-prioritised

Main flow

1. Combined required contributions exceed the forecast surplus.
2. Feasibility reports the gap.
3. The waterfall ranks the goals and splits the surplus.
4. Three levers are offered per goal — extend the date, reduce the target, increase the contribution.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-019-A1	step 3	The user re-prioritises	The split recomputes from the new ranking
UC-019-A2	step 4	The user takes a lever on one goal	Only that goal changes; the others keep their plans

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-019-E1	step 1	The surplus is negative	The screen says the shortfall is in the budget, not in the goals, and links there	—
UC-019-E2	step 1	Only one goal exists	No conflict; feasibility still reports whether that one fits	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-019-F1	The forecast has too little history to produce a surplus	The forecast reports insufficient history	An observed P50 is substituted and labelled as such	More history	None (ADR-0035)
UC-019-F2	The goal split and the surplus waterfall could disagree about the same rupee	The ranking is the base and the goal split consumes it, in that order	One consistent figure	None needed	None (ADR-0038)

UC-020 — Evidence a goal contribution

Actor	User
Implements	FR-037
Business rules	BR-027
Validation	—
Entities touched	ENT-023, ENT-022, ENT-004
Flow	—
Preconditions	A goal, and a transaction in one of its funding accounts
Postconditions (success)	A contribution row linked to the transaction; evidenced and declared progress are stored and shown separately
Frequency	Monthly per funded goal

Main flow

1. A transaction lands in a funding account.
2. The user links it to a goal.
3. Evidenced progress rises.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-020-A1	step 2	The user declares progress with no transaction	It is stored separately and shown separately — the two are never merged
UC-020-A2	step 1	The transaction is a transfer into the funding account	The destination leg is what evidences the contribution (`UC-008-A1`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-020-E1	step 2	The linked transaction is later deleted	The contribution is removed with it and progress falls back	—
UC-020-E2	step 2	The transaction is already linked to another goal	Refused; one transaction evidences one goal	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-020-F1	Evidenced and declared progress disagree	They are never summed into one figure	Both are shown, so the disagreement is visible	The user reconciles	Two honest figures rather than one invented one (ADR-0036)

UC-021 — Keep the accounts

Actor	User
Implements	FR-039, FR-045
Business rules	BR-006, BR-007

Validation	VAL-007, VAL-008
Entities touched	ENT-003, ENT-017, ENT-018, ENT-031
Flow	—
Preconditions	A profile exists
Postconditions (success)	An account of one of the eleven types, with its opening balance; archiving keeps history and leaves active totals
Frequency	A handful at setup, then rarely

Main flow

1. The user creates an account of any of the eleven types — bank, cash, credit card, loan, investment, gold, crypto, property, vehicle, receivable, payable.
2. It appears under its section with a subtotal.
3. Editing and archiving are available from the account screen.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-021-A1	step 3	The user archives a closed account	History stays readable and the balance leaves active totals (`FR-045`)
UC-021-A2	step 1	The type is credit card or loan	A one-to-one terms row is created, keyed by `account_id` (`UC-022`, `UC-023`)
UC-021-A3	step 1	The type is vehicle	A vehicle profile may be attached, or may exist without an account (`UC-029`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-021-E1	step 1	A duplicate account name within one profile	Refused by a unique index	VAL-008
UC-021-E2	step 1	An unrecognised type	Refused at the UI, which offers only the eleven	VAL-007

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-021-F1	An unknown `type` string arrives from an archive written by a newer version	The column carries no CHECK constraint, so the row loads	The UI shows it as an unrecognised type rather than crashing	Import refuses an archive whose schema version it cannot read	A loadable row with a type the UI names as unknown (VAL-020)
UC-021-F2	A stored balance drifts from the ledger	The nightly integrity job	An adjustment prompt	`UC-026`	A derived balance that is still the truth (DB-001)

UC-022 — Understand a credit card

Actor	User
Implements	FR-040
Business rules	BR-002, BR-020

Validation	VAL-013
Entities touched	ENT-017, ENT-019, ENT-003, ENT-004
Flow	—
Preconditions	A credit-card account
Postconditions (success)	The cycle, the unbilled amount, utilisation and the due date are derivable; at most one reminder per statement cycle is claimed
Frequency	Monthly per card

Main flow

1. Limit, statement day and due day are entered.
2. The engine derives the current cycle, the unbilled amount, utilisation and the due date.
3. Utilisation feeds the health score (`UC-043`).

Alternate flows

ID	Branches at	Condition	Behaviour
UC-022-A1	step 2	A statement is due in this cycle	A reminder is raised, claimed once per cycle in `card_alert`
UC-022-A2	step 2	The card is paid in full before the due date	The reminder is satisfied and not repeated

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-022-E1	step 1	A statement day of 31 in a 30-day month	The cycle resolves to the month's last day rather than failing	VAL-013
UC-022-E2	step 1	A due day earlier in the month than the statement day	Accepted — the due date falls in the following month, which is how cards work	VAL-013

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-022-F1	A cycle straddles two calendar months, so a month-keyed reminder would fire twice	The claim is keyed by `cycle_start_iso_date`, not by month	One reminder	None needed	One claim row (ADR-0025)
UC-022-F2	Utilisation computed in floating point would drift	Amounts are `Long` paise and the ratio is basis points	Exact figures	None needed	None (BR-001, BR-002)

UC-023 — See what a loan really costs

Actor	User
Implements	FR-041
Business rules	BR-002, BR-026
Validation	VAL-014
Entities touched	ENT-018, ENT-003
Flow	—
Preconditions	A loan account with its five terms

Postconditions (success)	None stored — the schedule is derived on every read, so a stale copy cannot exist
Frequency	Monthly, and whenever a prepayment is considered

Main flow

1. Principal, rate in basis points, EMI and tenure are entered.
2. The engine derives the amortisation schedule on read.
3. The principal and interest split per instalment and the payoff date are shown.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-023-A1	step 3	The user runs the prepayment simulator	It uses this same engine's arithmetic, minting nothing of its own (`UC-047`)
UC-023-A2	step 1	The rate is revised mid-loan	The schedule re-derives; there is nothing stored to migrate

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-023-E1	step 2	An EMI below the monthly interest	The engine says the loan never closes at that EMI rather than looping	VAL-014
UC-023-E2	step 1	A tenure of zero or a negative principal	Refused	VAL-014

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-023-F1	A rate edit would make an earlier stored schedule wrong	There is no stored schedule	Correct figures	None needed	None — five columns, derived every time (ADR-0026)
UC-023-F2	Rounding in the per-instalment split would not sum to the principal	A property test asserts the identity	None	Build failure before release	None

UC-024 — Value an asset that has a market price

Actor	User, the market worker
Implements	FR-042, FR-073
Business rules	BR-028, and `P-04`
Validation	VAL-015
Entities touched	ENT-020, ENT-036, ENT-003
Flow	—
Preconditions	A holding with units; a price observation from the user or the cache
Postconditions (success)	Value, cost and gain are derivable, with the price's observation date carried so staleness can be labelled
Frequency	Daily when consented and online; otherwise whenever the user types a price

Main flow

1. A holding carries units and an observed unit price with its date.
2. Value, cost and gain are derived.
3. When online and consented, the price worker refreshes the daily close into the cache.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-024-A1	step 1	The user enters the price by hand	The observation date is theirs, and the staleness label uses it
UC-024-A2	step 3	The device is offline	The cached close is used and labelled "as of <date>" (`FR-012`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-024-E1	step 2	No price has ever been observed	The holding is excluded from the allocation denominator and reported as missing coverage, never counted as Rs. 0	—
UC-024-E2	step 1	Units or price of zero	Refused	VAL-015

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-024-F1	There is no configured host and no certificate pins, which is how the client ships	No socket is opened at all	Every priced screen runs off the cache or reports coverage	Type prices by hand	The cache as it stands (ADR-0030, `GAP-001`)
UC-024-F2	The user revokes MARKET_DATA	The consent gate	Fetching stops; the cached public price is deliberately kept	Re-grant	The cached close — a gold price is a fact about gold, not an inference about the user (BR-028)

UC-025 — Watch net worth over time

Actor	User, the snapshot worker
Implements	FR-043
Business rules	BR-004, BR-007
Validation	—
Entities touched	ENT-014, ENT-003, ENT-004
Flow	—
Preconditions	At least one account
Postconditions (success)	One immutable snapshot row per date, keyed by date so a second worker run updates rather than duplicates
Frequency	Nightly, automatically

Main flow

1. The nightly worker writes one snapshot per day.
2. Assets minus liabilities, from the derived balances.
3. The chart ranges over 1M / 6M / 1Y / All.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-025-A1	step 1	The user back-dates a transaction	The series recomputes the affected days rather than leaving a step in the chart (ADR-0012)
UC-025-A2	step 3	The history is shorter than the chosen range	The chart shows what exists and says the range is partial

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-025-E1	step 1	A day was missed because the device was off	The chart interpolates nothing and the gap is visible	—
UC-025-E2	step 2	Liabilities exceed assets	A negative net worth is shown as it is	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-025-F1	Two snapshots are written for one date by two worker runs	The date is the claim key	One row	None needed	One row — the second write is an update
UC-025-F2	A price-less holding would make the snapshot understate assets	Coverage is reported rather than assumed	The snapshot is drawn with the coverage it had	Supply prices	A snapshot that is honest about what it could value (`UC-024-E1`)

UC-026 — Reconcile with reality

Actor	User, the integrity worker
Implements	FR-044, FR-095
Business rules	BR-007
Validation	—
Entities touched	ENT-003, ENT-004
Flow	—
Preconditions	An account whose derived balance can be compared with a real one
Postconditions (success)	An adjustment transaction exists; no historical row was edited and no balance was silently mutated
Frequency	Monthly, or whenever the integrity job raises a discrepancy

Main flow

1. The user enters the real balance from their bank or their wallet.
2. The difference against the derived balance is computed.
3. An adjustment transaction is posted for the difference.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-026-A1	step 1	The nightly integrity job finds the discrepancy first	It prompts for the adjustment rather than making one
UC-026-A2	step 3	The user would rather find the missing transaction	The adjustment is optional; the discrepancy stays visible until resolved

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-026-E1	step 2	The difference is zero	Nothing is posted; the screen says the account already agrees	—
UC-026-E2	step 1	The account is archived	Reconciliation is refused; an archived account has no live balance to agree with	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-026-F1	A stored balance has drifted from the ledger	The integrity job compares the two	The adjustment restores agreement without editing history	Post the adjustment	A cached column corrected against the derived truth (ADR-0007, DB-001)
UC-026-F2	The adjustment itself is miscategorised and distorts spend analysis	Adjustments carry their own source and are visible as adjustments	Analysis can exclude them	Recategorise	An adjustment that is labelled as one

UC-027 — See a holding's real return

Actor	User
Implements	FR-046
Business rules	BR-001, BR-002, BR-013
Validation	VAL-015
Entities touched	ENT-021, ENT-020
Flow	—
Preconditions	At least one dated lot
Postconditions (success)	None stored — absolute gain and XIRR are derived, and the same lots always produce the same rate
Frequency	Monthly, or whenever the portfolio is reviewed

Main flow

1. Lots are recorded with their dates, which is what makes a money-weighted return computable at all.
2. Absolute gain is derived.
3. XIRR is computed by bisection over the daily growth factor.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-027-A1	step 3	A single-lot holding	XIRR still answers, and matches the simple annualised figure

ID	Branches at	Condition	Behaviour
UC-027-A2	step 1	A sell lot is recorded	It enters the cash-flow series with the opposite sign

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-027-E1	step 3	All cash flows have the same sign, so no rate exists	The engine reports that rather than returning a number	—
UC-027-E2	step 1	A lot dated in the future	Refused	VAL-015

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-027-F1	A naive solver diverges on a large early outflow	A fixed 128 bisections over a literal bracket, with no early exit	The same lots always produce the same rate	None needed	None — reproducible six months from now too (ADR-0028, `AI-ARC-006`)
UC-027-F2	A price-less holding has no current value to measure the return against	Coverage is reported	The return is withheld for that holding	Supply a price	None

UC-028 — Check how the portfolio is spread

Actor	User
Implements	FR-047, FR-066
Business rules	BR-021, and `P-02`
Validation	—
Entities touched	ENT-020, ENT-003, ENT-036
Flow	—
Preconditions	At least one holding in an investable account
Postconditions (success)	None stored — allocation and concentration are derived, and a breach cites the rule id and version that fired
Frequency	Monthly

Main flow

1. The investable accounts form the denominator — deliberately not net worth.
2. Allocation by asset class is computed.
3. Concentration is tested against the rulebook row.
4. A breach cites the rule id and its version in the evidence.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-028-A1	step 2	Coverage is partial	The figure is reported with the share of the portfolio that carries a price
UC-028-A2	step 3	No rule fires	The allocation is still shown; silence from the rulebook is a result

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-028-E1	step 1	The portfolio is empty	The screen asks for a holding rather than drawing an empty donut	—
UC-028-E2	step 1	One holding is the whole portfolio	Concentration is 100%, reported plainly, not softened	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-028-F1	Including net worth in the denominator would put nearly every user permanently past the 70% line	The denominator is the investable accounts only, by decision	The rule stays meaningful	None needed	None (ADR-0029)
UC-028-F2	An unpriced holding counted as Rs. 0 would understate concentration	It is excluded and reported as missing coverage	Honest partial coverage	Supply a price	None (`UC-024-E1`)

UC-029 — Know when the vehicle is due

Actor	User
Implements	FR-048, FR-049, FR-050, FR-067
Business rules	BR-018
Validation	VAL-016
Entities touched	ENT-031, ENT-032, ENT-033, ENT-034
Flow	—
Preconditions	A vehicle profile
Postconditions (success)	A predicted next-service date with a cost range, and renewals on a timeline
Frequency	Odometer entries monthly; services a few times a year

Main flow

1. The vehicle profile, odometer readings and service history are recorded.
2. Theil–Sen regression over the personal usage index predicts the next service date.
3. A cost range comes from the knowledge base.
4. Renewals — insurance, PUC — appear on a timeline.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-029-A1	step 2	A new vehicle with one odometer reading	The class default interval is used, and labelled a default rather than a personal trend
UC-029-A2	step 4	A renewal is near	It joins the obligations the dashboard shows for the next 14 days (`FR-006`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-029-E1	step 1	An odometer reading lower than an earlier one	Refused	VAL-016
UC-029-E2	step 2	Readings too sparse for a regression	The class default is used and said to be a default	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-029-F1	No local service price exists	The knowledge base has no row for the class	A range from the knowledge base, or no quote at all	Record a real service cost	None — it never invents a price (ADR-0052)
UC-029-F2	The fuel log that `FR-049` depends on does not exist	Traced in `GAP-009`	Mileage anomalies are not detected at all	Not available in v1	A stated gap, not a silent one

UC-030 — Open the advisor

Actor	User
Implements	FR-051, FR-055
Business rules	BR-025, and `P-02`
Validation	—
Entities touched	ENT-025 (read), plus every engine the signals read
Flow	—
Preconditions	The app is unlocked
Postconditions (success)	None — a read. The disclaimer is shown on every path out of this hub
Frequency	Weekly

Main flow

1. The health score with its band and the reason it moved.
2. Signal chips drill down into each contribution.
3. The insight feed.
4. The tool grid — purchase advisor, simulators, chat.
5. The disclaimer in the footer: informational, not regulated investment advice.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-030-A1	step 1	A sparse profile	The score is reweighted over the signals that exist, and says which it could not read
UC-030-A2	step 4	The user opens a simulator	It shares the engines rather than re-implementing their arithmetic (`UC-047-F1`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-030-E1	step 3	No insight is currently worth showing	The feed says so rather than padding itself	—

ID	Branches at	Condition	Behaviour	Validation
UC-030-E2	step 1	Only one signal is available	The score says so and the band is withheld (`UC-043-E1`)	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-030-F1	A signal engine fails	The signal returns an error rather than a zero	The score reweights over the rest and names the missing signal	Retry	None — a failed signal is never scored zero (ADR-0045)
UC-030-F2	An output reaches the user without the disclaimer	Asserted by test on the advisor surfaces	None	Build failure before release	None (BR-025)

UC-031 — Work the insight feed

Actor	User
Implements	FR-052, FR-075
Business rules	BR-013, and `P-02`
Validation	—
Entities touched	ENT-025
Flow	FLOW-001
Preconditions	At least one engine has published a result
Postconditions (success)	Each card's state — acted, snoozed, dismissed — is stored against its fingerprint
Frequency	Daily

Main flow

1. Engines publish results, each with provenance.
2. The orchestrator deduplicates by fingerprint and ranks.
3. Cards show the finding, its evidence, a confidence and at most one action.
4. The user acts, snoozes or dismisses.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-031-A1	step 4	A dismissed insight's condition changes materially	It returns under a new fingerprint, rather than being suppressed forever
UC-031-A2	step 3	Several cards are equally ranked	The order is still deterministic, because ranking is a pure function of the results

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-031-E1	step 2	Two engines produce the same finding	The duplicate is collapsed, not shown twice	—
UC-031-E2	step 3	An engine offers two actions	Only one reaches the card; a card with two actions is a decision the user has not been helped with	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-031-F1	The ranking is not reproducible across runs	A golden test pins the order	A stable feed	Build failure before release	None (ADR-0046)
UC-031-F2	A card is published with no evidence	`BR-013` requires provenance on every result	The card could not be built	Build failure before release	None

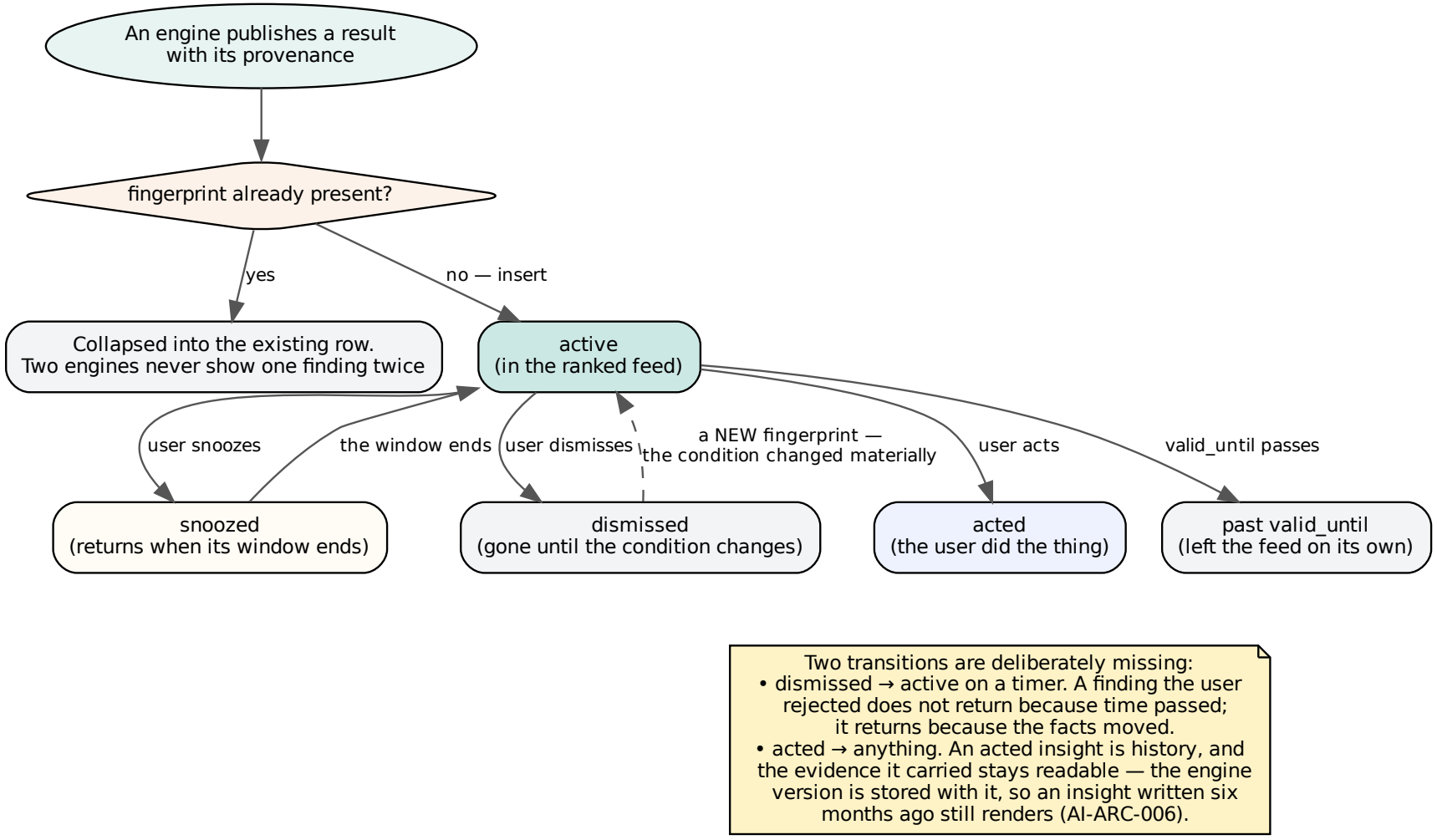


Figure 5 — The lifecycle of an insight (ENT-025). The two transitions that are deliberately absent are named in the note.

Source: diagrams/state-insight.dot.

UC-032 — Ask whether a purchase fits

Actor	User
Implements	FR-053, FR-069
Business rules	BR-024, `P-02`, `P-07`
Validation	VAL-021
Entities touched	ENT-027, ENT-028, ENT-029
Flow	FLOW-003
Preconditions	The app is unlocked; the gates read whatever engines have data
Postconditions (success)	A verdict, and a trace with one row per gate carrying the figures that gate used
Frequency	A few times a month

Main flow

1. Item, price, payment method and urgency are entered.
2. The gates run in a fixed order.
3. Each gate reports pass, warn or fail.
4. One verdict follows — Comfortable, Stretch or Not now.
5. The impact strip shows the before and after.
6. The trace is stored.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-032-A1	step 4	The verdict is "Not now"	The user creates a goal from it instead, carrying the price across (`UC-018-A2`)
UC-032-A2	step 1	The question comes from a wishlist item	The interview answers are reused rather than asked again (`UC-042`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-032-E1	step 1	A price of zero or an empty item	Refused	VAL-021
UC-032-E2	step 1	A price far beyond net worth	Still assessed; the verdict is the answer, not a refusal to answer	VAL-021

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-032-F1	A gate's input is missing — no forecast yet	The gate reports "cannot assess"	The verdict states which gate abstained rather than passing silently	More history	A trace recording the abstention (ADR-0049)
UC-032-F2	The engines move, making an old verdict unexplainable	The trace stores the figures, not a recomputation	An old verdict still reads correctly	None needed	The trace as it was (`AI-ARC-006`)

UC-033 — Ask a question in words

Actor	User
Implements	FR-054, FR-076, FR-077
Business rules	BR-011, BR-012
Validation	VAL-022
Entities touched	ENT-035
Flow	FLOW-009
Preconditions	The app is unlocked; the on-device model is present
Postconditions (success)	A stored message pair, with the tool calls that produced every figure in the answer
Frequency	Occasional, and rising with trust

Main flow

1. The question is planned into tool calls.
2. Only registry tools run.
3. Engine figures are rendered to words by allowlisted renderers.
4. The model composes a sentence from those words.
5. The guardrail checks every figure in the output.
6. The reply is shown with its sources.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-033-A1	step 2	The question needs several tools	Each call is listed under the reply
UC-033-A2	step 6	The user asks a follow-up	The prior turn is context; the figures are fetched again rather than remembered

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-033-E1	step 1	No tool can answer	Chat says so and offers what it can answer instead — it does not guess	VAL-022
UC-033-E2	step 1	An empty or abusive question	Refused at validation	VAL-022

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-033-F1	The model emits a figure no engine produced	The numeric guardrail, before display	The sentence is refused	Re-ask	None — and the refusal rate is measured against a frozen set with an absolute threshold (ADR-0048, ADR-0054)
UC-033-F2	The on-device model is unavailable or too slow	A timeout on the generation call	The rendered figures are shown without prose, which is still an answer	Retry later	A stored answer without a sentence

UC-034 — Change a setting

Actor	User
Implements	FR-056
Business rules	BR-014
Validation	—
Entities touched	None — Proto DataStore
Flow	—
Preconditions	The app is unlocked
Postconditions (success)	The preference is written to Proto DataStore and takes effect immediately
Frequency	Rare after setup

Main flow

1. Settings lists its groups — profile and household, categories, notifications and quiet hours, security, data, consents, about.
2. The user changes a preference.
3. It is written to Proto DataStore and takes effect immediately, with no restart.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-034-A1	step 2	The user changes the language	The app re-renders in it (`UC-053`)
UC-034-A2	step 2	The user changes the auto-lock timeout	It applies to the next idle period, not retroactively (`UC-050`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-034-E1	step 1	The user looks for the recurring-rules manager that `FR-056` names	It is not built	— (`GAP-010`)
UC-034-E2	step 1	The user looks for household settings	Household mode is off by a constant in v1 (`UC-054-E1`)	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-034-F1	A DataStore write fails	The write returns an error	The toggle reverts visibly rather than showing a state that was not saved	Retry	None
UC-034-F2	SharedPreferences is used instead of DataStore	Banned by project rule and caught in review	None	Build or review failure	None

UC-035 — Take the data out, and bring it back

Actor	User
Implements	FR-057
Business rules	BR-005, BR-016
Validation	VAL-020

Entities touched	Every profile-scoped entity; `ENT-013` is deliberately excluded
Flow	FLOW-006
Preconditions	A profile with data; storage the user can write to
Postconditions (success)	A JSON archive declaring its schema version, which round-trips losslessly into a fresh install
Frequency	Occasional — a new phone, or an audit of one's own data

Main flow

1. Export writes a JSON archive of the serialised entities.
2. The archive declares its schema version.
3. Import on another install replaces the profile's data transactionally.
4. Every field returns, because every field name is pinned by test.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-035-A1	step 1	The user exports for inspection rather than transfer	The archive is plain JSON and readable without the app
UC-035-A2	step 1	The user wants the encrypted form instead	That is backup, not export (`UC-051`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-035-E1	step 3	The archive was written by a newer version	The import is refused rather than partially applied	VAL-020
UC-035-E2	step 1	The user expects CSVs alongside the JSON, as `FR-057` says	Only JSON exists	— (`GAP-011`)

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-035-F1	A property is renamed in code, silently changing the archive format	`ArchiveFormatTest` pins every field name	None	Build failure before release	None (ADR-0023)
UC-035-F2	Import is interrupted	Transaction abort	The profile is either the old one or the new one	Retry	None — never a merge of both (BR-016)

UC-036 — Erase everything

Actor	User
Implements	FR-058
Business rules	BR-015
Validation	VAL-019
Entities touched	All of them, and the Keystore-wrapped key before any of them
Flow	FLOW-008
Preconditions	The app is unlocked; biometric is available

Postconditions (success)	The key is destroyed, the files are gone, scheduled work is cancelled, and anything unreachable is named
Frequency	Once, if ever

Main flow

1. The user chooses erase and reads what it destroys.
2. Biometric authentication.
3. A typed confirmation.
4. The wrapped key is destroyed — first, deliberately.
5. Files are removed and work is cancelled.
6. A report names anything it could not reach.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-036-A1	step 3	The user cancels at the typed confirmation	Nothing is touched
UC-036-A2	step 6	The user wants a copy first	Export or backup is offered before the destructive step, not after

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-036-E1	step 2	Biometric fails	The erase does not proceed to the key	VAL-019
UC-036-E2	step 3	The typed phrase does not match	Refused	VAL-019

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-036-F1	The file sweep is interrupted after the key is gone	Next launch finds ciphertext with no key	The app starts as a fresh install	Nothing to recover	Unreadable ciphertext — which is why the key goes first (ADR-0060, `SEC-006`)
UC-036-F2	A file on removable storage cannot be reached	The sweep reports what it could not delete	The report names it	Delete it by hand	A named, unreachable file

UC-037 — Grant and revoke a consent

Actor	User
Implements	FR-059
Business rules	BR-014, BR-028, and `P-01`
Validation	—
Entities touched	ENT-016, ENT-036, ENT-013; the ledger itself is DataStore
Flow	FLOW-007
Preconditions	The app is unlocked
Postconditions (success)	A consent record with both timestamps; the dependent data path is running or stopped to match

Frequency	A few times, then rarely

Main flow

1. Every consent is listed with its status, its plain-language purpose, and when it was granted or revoked.
2. One tap revokes.
3. The dependent path stops at once.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-037-A1	step 3	The consent is SMS	The parsed drafts are deleted
UC-037-A2	step 3	The consent is market data	The cached public price is kept (BR-028)
UC-037-A3	step 2	The user re-grants later	A new grant timestamp is recorded; the earlier revoke is not overwritten

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-037-E1	step 1	The OS permission is granted but the in-app consent is not	The in-app consent is the gate, and the feature stays off	—
UC-037-E2	step 1	The OS permission was revoked outside the app	The consent is shown as unusable, with the reason	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-037-F1	A consent record cannot be written	The write returns an error	The grant fails **closed** — the feature stays off	Retry	None; a feature running without a record is the failure that matters
UC-037-F2	A feature reads data before checking the consent	Asserted at the feature's entry point	None	Build or review failure	None (`P-01`)

UC-038 — Learn what recurs and what a rupee became

Actor	The classification engine
Implements	FR-061, FR-062
Business rules	BR-018
Validation	—
Entities touched	ENT-004, ENT-008, ENT-003
Flow	FLOW-001
Preconditions	Enough history for a stream to have a variance at all
Postconditions (success)	Each stream carries FIXED, SEMI_FIXED or VARIABLE, and each transaction a nature — NEED, WANT, INVESTMENT, ASSET, LIABILITY
Frequency	On every transaction, and on every classification pass

Main flow

1. Stage 2 scores each expense stream's variance.
2. FIXED, SEMI_FIXED or VARIABLE is assigned.
3. Nature is assigned from the account type.
4. Both feed Safe-to-Spend and the emergency-fund target.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-038-A1	step 3	The user overrides a nature	The override is stored and the inference is not; a user statement outranks a heuristic
UC-038-A2	step 2	A stream's variance changes over months	The classification moves with it, and the forecast follows

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-038-E1	step 2	Too little history for a stream	It is left unclassified rather than guessed	—
UC-038-E2	step 3	The account type does not imply a nature	The transaction is left without one rather than defaulted to WANT	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-038-F1	Nature inferred from account type understates true discretionary spend	A known limitation, stated in the ADR	Safe-to-Spend may read generously	The user overrides natures	A documented limitation rather than a hidden one (ADR-0016)

UC-039 — See the next ninety days

Actor	The forecast worker, then the user
Implements	FR-063, FR-064
Business rules	BR-013, and `P-08`
Validation	—
Entities touched	ENT-004, ENT-012, ENT-003 (reads)
Flow	FLOW-004
Preconditions	At least a month of history, and confirmed schedules to project
Postconditions (success)	P10 / P50 / P90 paths and a list of crunch days, each attributed to its causes
Frequency	Nightly

Main flow

1. Confirmed schedules and fixed streams are projected.
2. Seasonality adjusts the monthly factor.
3. P10, P50 and P90 paths are produced.
4. Crunch days are listed with their causes.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-039-A1	step 4	The user opens a crunch day	The obligations that caused it are listed, with suggested moves
UC-039-A2	step 1	A future-dated transaction exists	It is included in the projection and excluded from actuals (`UC-013`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-039-E1	step 1	Under a month of history	The forecast declines to project 90 days and says what it needs	—
UC-039-E2	step 4	No crunch day in the window	That is reported as the result, not as an empty list	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-039-F1	Forecast accuracy is not measured against actuals	Traced in `GAP-003` and ADR-0043	The forecast runs; no accuracy claim is made	Deferred to a later milestone	An unevidenced accuracy claim, named as such
UC-039-F2	Two runs produce different paths from the same data	Seeded determinism tests	Identical paths	Build failure before release	None (`P-08`)

UC-040 — Size the emergency fund

Actor	The engine, then the user
Implements	FR-065
Business rules	BR-022
Validation	—
Entities touched	ENT-004, ENT-009, ENT-003, ENT-022
Flow	—
Preconditions	At least one closed month, or a quick-setup envelope to stand in
Postconditions (success)	A target, a runway in months, and a coaching line — each stating what it was computed from
Frequency	Monthly

Main flow

1. Essentials are the median of closed months' NEED spend.
2. A multiplier comes from `RULE-EMF-MULT`.
3. A target, a runway in months and a coaching line are produced.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-040-A1	step 1	No closed month yet	The quick-setup envelope stands in, and is labelled as the substitute it is

ID	Branches at	Condition	Behaviour
UC-040-A2	step 3	The target is already met	The coach says so and the surplus moves to the next stage (`UC-046`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-040-E1	step 3	Liquid balances are zero	The runway is zero months and the coach names the first step	—
UC-040-E2	step 1	NEED spend is zero across every closed month	The target is withheld; a multiplier on zero is not a target	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-040-F1	Three of SRS §10.1's five multiplier terms have no field anywhere in the app	Their absence is asserted by a test rather than defaulted	The output names what it did not consider	Add the fields in a later version	An honest partial computation (ADR-0034)

UC-041 — Keep an appliance serviced

Actor	User
Implements	FR-068
Business rules	BR-018
Validation	—
Entities touched	ENT-037, ENT-038, ENT-039
Flow	—
Preconditions	An appliance with a class and a purchase date
Postconditions (success)	A next-service date, consumable due dates and a warranty state — each derived from a dated event, not from an assumption
Frequency	A few times a year per appliance

Main flow

1. The appliance, its class and its purchase date are recorded.
2. Next service, consumables and warranty state are derived.
3. A running cost is estimated where a basis exists.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-041-A1	step 2	A consumable is replaced	The next due date moves from the replacement, not from the purchase
UC-041-A2	step 2	The warranty has expired	The state says so, and a service cost is no longer assumed covered

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-041-E1	step 1	An unknown appliance class	The generic interval is used and labelled generic	—
UC-041-E2	step 1	A purchase date in the future	Refused	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-041-F1	No running-cost basis exists for the class	The knowledge base has no row	The figure is withheld rather than estimated from nothing	Record real costs	None (ADR-0070)

UC-042 — Keep a wish out of the ledger

Actor	User
Implements	FR-070
Business rules	`P-07`
Validation	—
Entities touched	ENT-029, ENT-030, ENT-027
Flow	—
Preconditions	The app is unlocked
Postconditions (success)	A wish with its interview answers and a ladder position — and no transaction, because a wish is not a purchase
Frequency	Occasional

Main flow

1. A wish is added to the buy list.
2. The interview adapts its questions to the answers already given.
3. A ladder orders the wishes.
4. Nothing is removed automatically.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-042-A1	step 4	A wish becomes a purchase	The user converts it, and the trace links the two (`ENT-029.last_trace_id`)
UC-042-A2	step 3	A wish's price changes	The ladder reorders; the wish stays

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-042-E1	step 2	The interview is abandoned part-way	The answers given are kept and the wish stays incomplete rather than being discarded	—
UC-042-E2	step 1	A duplicate wish	Allowed; two people in a household may want the same thing	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-042-F1	A ladder rebuild would drop a wish whose score fell	Removal is never automatic, by decision	The wish stays	The user removes it	The wish (ADR-0050)

UC-043 — Read the health score

Actor	User
Implements	FR-071
Business rules	BR-013, and `P-02`
Validation	—
Entities touched	Reads across ENT-003, ENT-004, ENT-009, ENT-017, ENT-020, ENT-022
Flow	—
Preconditions	At least one signal has an input
Postconditions (success)	None stored — the score is always computed live from current data
Frequency	Weekly

Main flow

1. Six live signals are computed.
2. Weights are normalised over the signals that are actually present.
3. A 0–1000 score and a band are produced.
4. Each signal's contribution and the strongest lever are shown.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-043-A1	step 4	The user opens a signal	The inputs behind it are shown (`P-02`)
UC-043-A2	step 3	The score moved since last month	The reason it moved is named, not just the delta

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-043-E1	step 2	Only one signal is available	The score says so and the band is withheld rather than implying a complete assessment	—
UC-043-E2	step 2	No signal is available	No score is produced	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-043-F1	There is no snapshots table, so a historical score is not re-derivable from storage	Traced in `GAP-013`	The score is always live, which is stated rather than implied	Deferred	No history to audit (ADR-0045)
UC-043-F2	A failing signal scored zero would read as a bad score rather than a missing one	Signals return errors, not zeros	The weight redistributes and the gap is named	Retry	None (`UC-030-F1`)

UC-044 — Decide whether to interrupt

Actor	The notification gate
Implements	FR-074
Business rules	BR-019, BR-020
Validation	—
Entities touched	ENT-026, ENT-010, ENT-019
Flow	FLOW-004
Preconditions	A candidate notification from any engine
Postconditions (success)	A delivery recorded under its claim key, or a drop with the reason
Frequency	Several candidates a day; far fewer deliveries

Main flow

1. A candidate notification arrives.
2. The gate checks type caps, quiet hours and prior claims.
3. It is delivered or dropped, and either way the decision is recorded.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-044-A1	step 2	A critical crunch alert	It outranks the cap and is delivered
UC-044-A2	step 3	The same candidate recurs tomorrow	The claim suppresses it; the condition has not changed

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-044-E1	step 2	Inside quiet hours	Non-critical candidates wait rather than queueing up to fire at once	—
UC-044-E2	step 2	Notifications are disabled at the OS level	Nothing is delivered; the decision is still recorded	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-044-F1	Several engines fire on one day and would each notify	One gate sees them all	The day has a ceiling	None needed	A log of what was dropped (ADR-0047)
UC-044-F2	The digest that would batch a day's findings is not built	Traced in `GAP-014`	Several separate notifications instead of one summary	Deferred	A stated gap

UC-045 — Ask whether today is unusual

Actor	User
Implements	FR-078
Business rules	BR-028, and `P-02`
Validation	—
Entities touched	ENT-036
Flow	—
Preconditions	Cached daily closes, which require a configured host and a granted consent

Postconditions (success)	A verdict with a measured hit rate, and an explicit list of what could not be measured
Frequency	Occasional

Main flow

1. Cached daily closes are read.
2. Each signal scores what it can measure.
3. A verdict is produced with its measured hit rate.
4. What cannot be measured is stated.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-045-A1	step 3	An all-in verdict would overstate confidence	A tranche policy is suggested instead
UC-045-A2	step 1	The cache is stale but non-empty	The verdict carries the "as of" date

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-045-E1	step 2	Too little history for a backtest	The signal reports "not measurable yet" rather than an unbacked score	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-045-F1	No price history at all, because the market-data proxy does not exist	The cache is empty and no host is configured	The screen explains that it needs market data and a consent, and offers nothing in the meantime	Host the proxy, or descope it (`OQ-003`)	None (`GAP-001`, ADR-0055)

UC-046 — Send the surplus somewhere

Actor	User
Implements	FR-079
Business rules	BR-018, BR-023
Validation	—
Entities touched	Reads across ENT-004, ENT-009, ENT-022, ENT-018
Flow	—
Preconditions	A computable monthly surplus, and the order-of-operations data file
Postconditions (success)	None stored — each stage reports what it consumed, and the remainder is named
Frequency	Monthly

Main flow

1. The monthly surplus is computed.
2. It flows down the eight stages of the order of operations.
3. Each stage reports what it consumed.
4. The remainder is named.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-046-A1	step 2	A goal's required monthly competes with a stage	The ranking is the base and the goal split consumes it, in that order (ADR-0038)
UC-046-A2	step 4	The remainder is substantial	The last stage names it rather than absorbing it silently

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-046-E1	step 1	The surplus is zero or negative	The waterfall stops at stage one and says what has to change	—
UC-046-E2	step 2	A stage has no applicable target — no loan, say	It is skipped and reported as skipped	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-046-F1	The order-of-operations data file is missing or malformed	The engine validates it before running	The engine refuses to run	Fix the data row	None — hardcoding the order would be exactly the defect `CLAUDE.md` §6 forbids

UC-047 — Simulate prepaying a loan

Actor	User
Implements	FR-080
Business rules	BR-026, and `P-08`
Validation	VAL-014
Entities touched	ENT-018 (read)
Flow	—
Preconditions	A loan account
Postconditions (success)	None stored — a comparison that states its assumptions
Frequency	Occasional

Main flow

1. The user enters a lump sum and an assumed rate of return.
2. Both branches — prepay, or invest — are computed from the loan engine's own arithmetic.
3. The comparison states its assumptions, including that the assumed return is the user's.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-047-A1	step 1	Several debts rather than one	The debt-payoff simulator orders them instead of comparing one against investing
UC-047-A2	step 1	No assumed return is given	The prepay branch is shown alone, with the interest saved

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-047-E1	step 1	No loan exists	The simulator asks for one rather than modelling a hypothetical	VAL-014
UC-047-E2	step 1	A lump sum larger than the outstanding principal	The loan closes and the surplus is reported as surplus	VAL-014

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-047-F1	A second implementation of amortisation would drift from the loan screen's	The simulators share the engine and mint nothing of their own	Consistent figures	Build or review failure	None (ADR-0051)

UC-048 — Estimate the tax position

Actor	User
Implements	FR-081
Business rules	BR-018, BR-025
Validation	—
Entities touched	Reads across ENT-004, ENT-020, ENT-021
Flow	—
Preconditions	Income and deduction inputs, from data or from the user
Postconditions (success)	An estimate under both regimes, naming every component it did not estimate
Frequency	Annually, and around a large capital event

Main flow

1. Income and deductions are read.
2. Both regimes are computed.
3. Capital gains follow the post-23-July-2024 rules.
4. The output names what it did not estimate.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-048-A1	step 1	The user supplies a figure the app does not hold	The estimate updates from it
UC-048-A2	step 2	One regime is clearly better	That is stated, with the margin, not just the winner

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-048-E1	step 1	A required input is missing	That component is excluded and listed as excluded	—

ID	Branches at	Condition	Behaviour	Validation
UC-048-E2	step 3	A holding has no lots, so a gain cannot be computed	It is excluded and named	VAL-015

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-048-F1	The tax knowledge base had no slab table when the engine landed	The engine states what it left out	An estimate with named exclusions	Populate the data rows	A partial estimate rather than last year's slabs presented as this year's (ADR-0072)

UC-049 — Check whether the cover is enough

Actor	User
Implements	FR-082
Business rules	BR-025, and `P-07`
Validation	—
Entities touched	Reads across ENT-004, ENT-003, ENT-022
Flow	—
Preconditions	Declared cover, dependants and income — all user-supplied
Postconditions (success)	A term gap and a health gap, with the larger of two readings reported
Frequency	Annually

Main flow

1. Declared cover, dependants and income are read.
2. Term and health gaps are computed.
3. The gap reported is the larger of two readings, because understating a gap is the costlier error.
4. An endowment policy sold as insurance is flagged for what it is.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-049-A1	step 2	No dependants	The term requirement falls and the health gap dominates
UC-049-A2	step 4	The policy mix is sound	That is said, rather than leaving silence to be read as approval

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-049-E1	step 1	No policy is recorded	The gap is the whole requirement, and the screen says it is based on nothing declared	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-049-F1	The engine could recommend surrendering a policy, which is regulated advice	It reports the gap and the cost of the mismatch, and never recommends a surrender	No regulated advice	None needed	None (ADR-0071, BR-025)

UC-050 — Unlock the app

Actor	User
Implements	FR-084, FR-085
Business rules	BR-015, and `P-01`
Validation	VAL-017
Entities touched	ENT-013; the key itself is in the Keystore
Flow	—
Preconditions	A cold start, or an idle timeout
Postconditions (success)	A session flag is set, and only then may the encrypted database provider build
Frequency	Several times a day

Main flow

1. A cold start, or the idle timeout expires.
2. BiometricPrompt class 3.
3. On success a session flag is set and the database provider may build.
4. The app opens on the screen the user left.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-050-A1	step 2	No biometric hardware or enrolment	The PIN is used
UC-050-A2	step 1	The user disabled the lock in settings	The gate passes through, and the state is read from one source of truth

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-050-E1	step 2	Five wrong PINs	A 30-second lockout that doubles	VAL-017
UC-050-E2	step 2	The user cancels the prompt	The app stays locked; there is no path past the gate	VAL-017

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-050-F1	A disabled lock and an unlock racing each other crashed an earlier build	Reproduced as issue 2.8 and now covered by a test	No crash	The gate is a single source of truth for lock state	None
UC-050-F2	The Keystore key is invalidated by a biometric re-enrolment	The unwrap fails with a specific error	The key rotates without losing the database	Automatic rotation	The database, intact (ADR-0057)

UC-051 — Back up, and restore on a new phone

Actor	User; the maintainer, when it runs as a gate
Implements	FR-087, FR-088
Business rules	BR-015, BR-016
Validation	VAL-018
Entities touched	Every profile-scoped entity, through the archive
Flow	FLOW-006
Preconditions	A profile with data; storage the user can write to
Postconditions (success)	An AES-256-GCM archive that restores on a fresh device, or an explicit statement that a lost passphrase means lost data
Frequency	On a schedule the user chooses; the drill runs before every promotion

Main flow

1. The user sets a passphrase and acknowledges irrecoverability.
2. Argon2id derives a key from it.
3. AES-256-GCM encrypts the archive to user-chosen storage.
4. On a new device, restore decrypts and replaces.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-051-A1	step 4	The maintainer runs the drill on an emulator as a release gate	The same path, exercised as `./gradlew restoreDrill`
UC-051-A2	step 3	The user changes the passphrase	A new archive is written; old archives still need the old passphrase

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-051-E1	step 4	A wrong passphrase	The restore fails closed with no partial write	VAL-018
UC-051-E2	step 4	Restoring into a different profile	Refused	— (ADR-0040)
UC-051-E3	step 1	A weak passphrase	Refused, with the requirement stated	VAL-018

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-051-F1	The passphrase is lost	Nothing can detect it	The data is unrecoverable, which the recovery screen states **in advance**	None — by design	An archive nobody can open
UC-051-F2	A restore regression ships	The drill runs on real hardware before every promotion	A blocked release	Fix before promoting	None (ADR-0041, `DRL-001`)

UC-052 — Leave a record of what was allowed

Actor	The system
Implements	FR-089
Business rules	BR-005
Validation	—
Entities touched	ENT-013
Flow	FLOW-007
Preconditions	A security-relevant event — a consent change, an unlock failure, an erase, a key rotation
Postconditions (success)	An append-only row, readable in the app and excluded from the export
Frequency	Whenever such an event occurs

Main flow

1. A security-relevant event occurs.
2. A row lands in `audit_log`, append-only.
3. It is readable in the app and excluded from the export.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-052-A1	step 3	The user reads the record in settings	They see what the app did and when
UC-052-A2	step 2	An erase runs	The log goes with everything else; it is not preserved across an erase

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-052-E1	step 2	An event arrives with no profile context	The table carries no `profile_id` by design, so it is still recorded	—

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-052-F1	Including the log in the export would leak events across profiles, since it cannot be scoped	Export-excluded by decision	The archive omits it	None needed	A log that stays on the device (ADR-0023)

UC-053 — Use the app in another language

Actor	User
Implements	FR-090
Business rules	NFR-011

Validation	—
Entities touched	None
Flow	—
Preconditions	The app is installed
Postconditions (success)	Every string renders in the chosen language, and numbers keep Indian grouping
Frequency	Once, usually

Main flow

1. The user picks English, Hindi, Kannada or Tamil.
2. Every string re-renders.
3. Numbers keep Indian digit grouping — Rs. 1,23,456.78.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-053-A1	step 1	The device locale already matches	No choice is needed
UC-053-A2	step 3	The language changes but the currency does not	Grouping stays Indian, because it follows the format, not the language

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-053-E1	step 2	A string is missing a translation	The build fails before release rather than falling back silently	— (ADR-0056)

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-053-F1	A hardcoded string bypasses the resource system	`CfoHardcodedUiString` in `:feature:*`	None	Build failure before release	None
UC-053-F2	A translation is longer than the layout allows	Paparazzi screenshot tests in each locale	A clipped or wrapped label	Fix the layout	None

UC-054 — Keep two books on one phone

Actor	User
Implements	FR-091, FR-092
Business rules	BR-006
Validation	—
Entities touched	ENT-002, ENT-001, and every profile-scoped table
Flow	—
Preconditions	One profile exists
Postconditions (success)	A second profile whose rows never appear in the first's queries
Frequency	Rare — a household member, or a business book

Main flow

1. A second profile is created.
2. Every query is scoped by `profile_id`.
3. Household aggregates compose per-profile reads rather than widening a query.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-054-A1	step 2	The user switches profiles	No data crosses, and the lock state is not bypassed
UC-054-A2	step 3	Only one profile exists	The household row still exists above it, so the aggregation path needs no special case

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-054-E1	step 3	Household mode is off by a constant in v1	The aggregate view is not reachable	— (ADR-0069)

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-054-F1	A query that forgot its profile scope would leak one member's rows into another's screen	`ProfileScopingTest`	None	Build failure before release	None (BR-006)

UC-055 — Import a statement

Actor	User
Implements	FR-093
Business rules	BR-005, BR-016
Validation	VAL-020
Entities touched	ENT-040, ENT-004
Flow	—
Preconditions	A file the user chose, and an account to import into
Postconditions (success)	One batch row carrying statement-grade provenance, and rows each pointing at it, so a later wipe can be verified rather than trusted
Frequency	Monthly, for users who import

Main flow

1. A file is chosen.
2. A batch row records the source, the time and the provenance.
3. Rows are created inside one transaction, each pointing at the batch.

Alternate flows

ID	Branches at	Condition	Behaviour
UC-055-A1	step 3	The user wipes a batch	Every row it created is removed, and the removal is verified rather than assumed (ADR-0078)

ID	Branches at	Condition	Behaviour
UC-055-A2	step 3	A row duplicates an existing transaction	The duplicate guard offers a merge (`UC-014-E2`)

Exception flows

ID	Branches at	Condition	Behaviour	Validation
UC-055-E1	step 3	A malformed row	The batch reports which lines it refused instead of failing whole	VAL-020
UC-055-E2	step 1	An unsupported file format	Refused, with the supported formats named	VAL-020

Failure flows

ID	Failure	Detection	User-visible behaviour	Recovery	Data left behind
UC-055-F1	An interrupted import leaves rows with no batch	The batch is written first, inside the same transaction	Nothing partial	Retry	None — orphan rows cannot exist
UC-055-F2	A wipe believes it removed rows it did not	The removal is counted and verified	The count is reported	Retry the wipe	A verified result rather than a trusted one (ADR-0078)

---

4. Coverage of the use-case set

Measure	Count
Use cases	55
With at least one alternate path	55
With at least one exception path	55
With at least one failure path	55
Alternate paths in total	117
Exception paths in total	107
Failure paths in total	100
Requirements referenced by at least one use case	95 of 95 `FR`

The use cases with no server to fall back on. Six describe a path whose failure cannot be retried against anything: `UC-024` (no market host exists), `UC-033` (the model is on the device or nowhere), `UC-035` and `UC-051` (the archive is the user's to keep), `UC-036` (erase is irreversible by design), `UC-045` (no price history at all). Each one's failure table names what survives, which is the only remedy available when there is nothing to retry against (`P-04`).

Where a failure is a build failure. Thirteen of the 100 failure paths end in "build failure before release" rather than in a user-visible state. That is the intended shape: a lint rule, a property test or a golden file is cheaper than a recovery screen, and `07-test-strategy-and-cases.md` names the test that enforces each one.

Part 04 — Data Dictionary and Database Design

← Master Blueprint · Related: 02 Requirements · 03 Flows & Use Cases · 05 Architecture

> ***This document is read off the schema, not written beside it.*** *The attribute tables in §3 were generated from* `core/database/src/main/kotlin/com/aicfo/core/database/entity/Entities.kt` *at schema* ***version 33****, and their Notes column is abridged from that file's own KDoc. A column that is in the code is in this document; one that is not, is not.*

1. Conventions

Confirmed — SRS §20, verified against the schema.

Convention	Rule	Enforced by
Table names	`snake_case`, singular except where the SRS named them plural (`transactions`, `tags`, `attachments`, `import_batches`)	Review
Primary key	A `TEXT` UUID in `id`, except `credit_card` and `loan`, whose PK **is** `account_id` because each is strictly 1:1 with its account	Room `@PrimaryKey`
Money	`INTEGER` minor units (paise) in a column named `*_minor`, with the currency on the owning row	BR-001; `CfoMoneyAsFloatingPoint` fails the build
Rates	`INTEGER` basis points in a column named `*_bps`	BR-002
Instants	`INTEGER` UTC epoch millis in a column named `*_utc_millis`	BR-003; `CfoWallClockInDomain` fails the build in `:domain:*`
Dates	ISO `yyyy-MM-dd` `TEXT` in a column named `*_iso_date` — never a midnight timestamp, because midnight depends on a zone that can change under a stored value	BR-004
Soft delete	`deleted_at_utc_millis` is set instead of deleting; every query filters it	BR-005; DB-002
Profile scoping	Every profile-scoped row carries `profile_id`, and no query spans profiles	BR-006; `ProfileScopingTest`
Audit columns	`created_at_utc_millis` and `updated_at_utc_millis` on every user-data table	Review
Foreign keys	Declared as a column plus an index; Room FK constraints are not used, so an archive import cannot be refused by a constraint it cannot satisfy in order	Index declarations in §8
Enum columns	Stored as the enum's `storedValue` string with no SQL `CHECK`, so a newer archive's unknown value loads and renders as unrecognised rather than crashing	`:core:model` enums; `UC-021-F1`
Serialisation	Every profile-scoped entity is `@Serializable`, because the export archive's shape **is** the schema; `audit_log` is deliberately excluded	ADR-0023; `ArchiveFormatTest`

Two tables are not profile-scoped, each for a stated reason: `household` sits above the profile and holds no money (ADR-0069), and `audit_log` has no profile context to scope by — which is also why the export cannot carry it (ADR-0023). Every other table, `market_close` included, carries `profile_id`. `market_close` is nonetheless the one table a consent revoke deliberately leaves in place, because a closing price is a public fact about an instrument rather than an inference about the user (ADR-0030, BR-028).

2. Entity index

40 tables at schema version 33. Lifecycle values: **soft** = `deleted_at` is set; **hard** = removed outright, with the reason; **claim** = one row per claim key, written once and never duplicated; **append** = never updated.

ENT	Table	Purpose	Owning module	Lifecycle
ENT-001	`household`	The row above the profile, so household aggregation composes per-profile reads rather than widening a query	`:core:database`	Never deleted; one per installation in v1
ENT-002	`profile`	A person using the app; the root every other row hangs off, and the owner of the time zone all calendar logic resolves in	`:core:database`	soft
ENT-003	`account`	One of the eleven account types; the unit net worth is composed from	`:data:repository`	soft, and separately archivable (FR-045)
ENT-004	`transactions`	The ledger. Every figure in the app is ultimately derived from these rows	`:data:repository`	soft
ENT-005	`transaction_splits`	The lines a transaction was split into. Moves no money itself — the parent holds the amount	`:data:repository`	soft with the parent
ENT-006	`tags`	A user-defined label, unique per profile	`:data:repository`	soft
ENT-007	`transaction_tags`	The many-to-many join between a transaction and its tags	`:data:repository`	hard — a join row carries nothing to recover
ENT-008	`category`	The spend taxonomy, self-referencing through `parent_id` for subcategories	`:data:repository`	soft
ENT-009	`budget`	One envelope per category per month, with its nature	`:data:repository`	soft
ENT-010	`budget_alert`	The claim that a band — 80% or 100% — has already been announced this month	`:data:repository`	claim, keyed by budget, band and month
ENT-011	`budget_review`	The claim that a closed month has been reviewed; one card for the whole month, not one per finding	`:data:repository`	claim, keyed by profile and month
ENT-012	`recurring_rule`	A confirmed recurring series, its cadence and its next due date	`:data:repository`	soft
ENT-013	`audit_log`	What the app was allowed to do, and when. Not profile-scoped and export-excluded	`:data:repository`	append
ENT-014	`net_worth_snapshot`	One immutable record per date of assets, liabilities and net worth	`:data:repository`	claim, keyed by date; recomputed on a back-dated edit
ENT-015	`attachments`	Metadata for a file on the encrypted filesystem, referenced by content hash. The database never holds the blob	`:data:repository`	soft; the blob is deleted separately (`RISK-009`)
ENT-016	`sms_draft`	A proposed transaction parsed from a bank alert, awaiting the user's confirmation	`:data:sms`	hard on consent revoke — an inference about the user must not outlive the permission
ENT-017	`credit_card`	A card's terms: limit, statement day, due day. PK is `account_id`	`:data:repository`	soft with its account
ENT-018	`loan`	A loan's five terms. The amortisation schedule is derived from them, never stored	`:data:repository`	soft with its account
ENT-019	`card_alert`	The claim that a card reminder has fired for a statement cycle	`:data:repository`	claim, keyed by `cycle_start_iso_date`

ENT	Table	Purpose	Owning module	Lifecycle
ENT-020	`investment_holding`	An instrument held inside an investment account, with its own asset class	`:data:repository`	soft
ENT-021	`investment_lot`	A dated buy or sell, which is what makes a money-weighted return computable	`:data:repository`	soft
ENT-022	`goal`	A target amount, an optional date, a priority and a status	`:data:repository`	soft
ENT-023	`goal_contribution`	A contribution, evidenced by a transaction or declared by the user — the two are never merged	`:data:repository`	soft
ENT-024	`goal_funding_account`	Which accounts fund a goal	`:data:repository`	hard — a join row
ENT-025	`insight`	A published engine finding with its evidence, confidence and provenance, deduplicated by `fingerprint`	`:data:repository`	status: active → dismissed / snoozed / acted
ENT-026	`notification_log`	What was delivered, when, under which claim key	`:data:repository`	claim
ENT-027	`purchase_trace`	The kept record of a purchase verdict: the inputs, the figures and the outcome	`:data:repository`	soft
ENT-028	`purchase_trace_gate`	One row per gate in that verdict, with its pass, warn or fail and the figures it used	`:data:repository`	soft with its trace
ENT-029	`wishlist_item`	A wish, not a purchase, with its ladder position	`:data:repository`	soft; never removed automatically
ENT-030	`interview_answer`	What the adaptive purchase interview was told	`:data:repository`	soft
ENT-031	`vehicle`	A vehicle profile and its class	`:data:repository`	soft
ENT-032	`vehicle_odometer`	A dated odometer reading; the series the usage trend is regressed over	`:data:repository`	soft
ENT-033	`vehicle_service`	A service event and what it cost	`:data:repository`	soft
ENT-034	`vehicle_renewal`	Insurance, PUC and other dated renewals	`:data:repository`	soft
ENT-035	`chat_message`	A question, the answer given, and the tool calls behind it	`:data:repository`	soft
ENT-036	`market_close`	A cached daily close for a price key. A public fact, not user data	`:core:network` cache	Retained through a consent revoke, by design
ENT-037	`appliance`	An appliance, its class and its warranty	`:data:repository`	soft
ENT-038	`appliance_service`	A service event for an appliance	`:data:repository`	soft
ENT-039	`appliance_consumable`	A consumable, its interval and its last replacement	`:data:repository`	soft
ENT-040	`import_batches`	One row per import, carrying statement-grade provenance, so a later wipe can be verified rather than trusted	`:data:repository`	soft; a wipe hard-deletes the rows it created

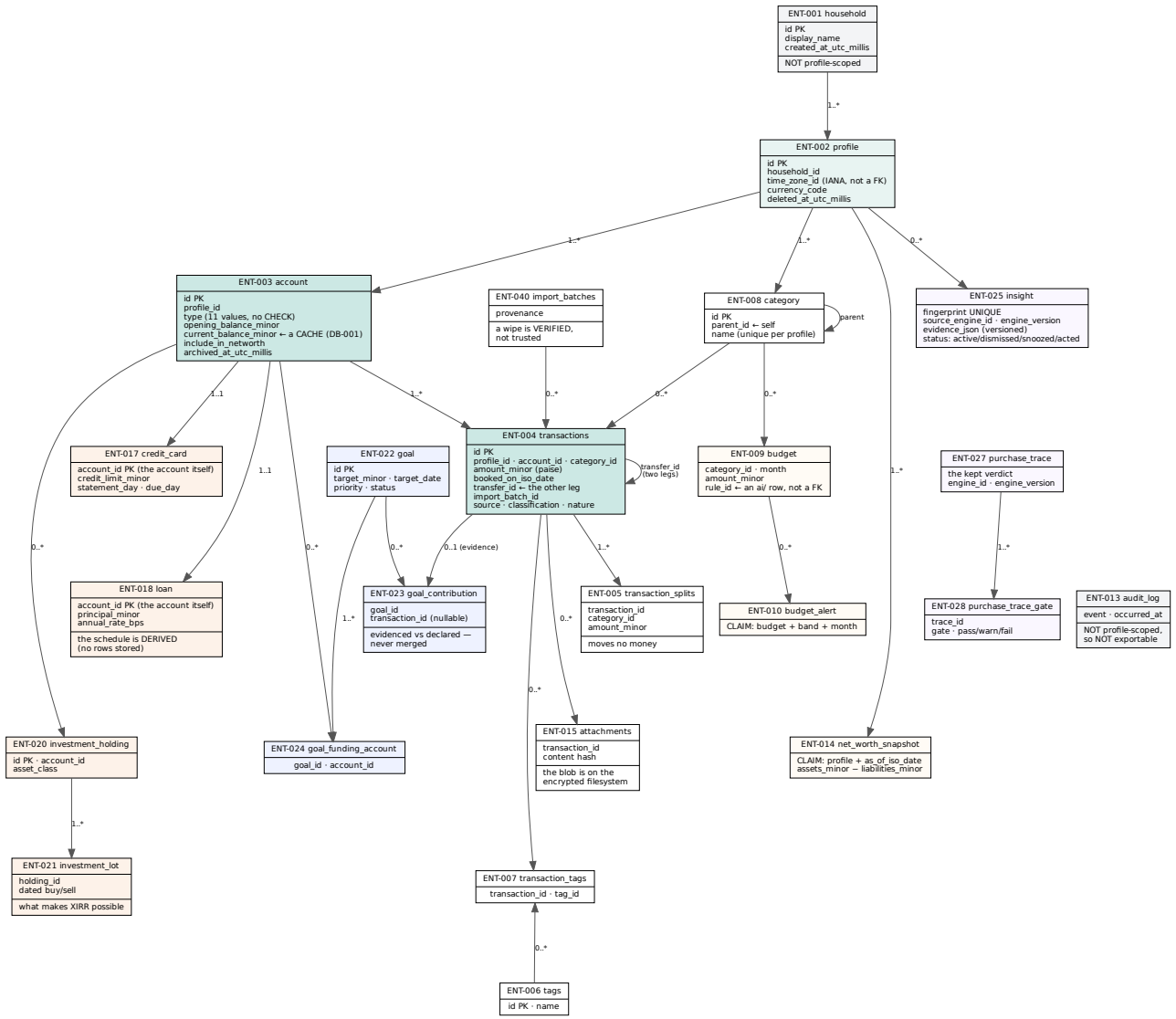


Figure 3 — The schema's spine. The full 435-column dictionary is §3; this is the part a reader needs to follow how a figure is derived.

Source: diagrams/er-diagram.dot.

3. Attribute tables

Generated from the schema. `Req?` reads the Kotlin nullability: `no` means the column is nullable. **Sensitivity** is `personal` for free text the user typed that describes them or their life, `financial` for an amount in paise, and `--` otherwise; it is the column that decides what may never reach a log line (BR / NFR-018) and what the export carries.

ENT-001 — `household` (HouseholdEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`display_name`	`String`	yes	—	personal	What the user calls this household. Never a person's name by default.
`created_at_utc_millis`	`Long`	yes	—	—	UTC epoch millis (TIM-001). For the row the migration backfills, the oldest profile's date.

ENT-002 — `profile` (ProfileEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`display_name`	`String`	yes	—	personal	
`time_zone_id`	`String`	yes	—	—	IANA zone id, e.g. `Asia/Kolkata`. The profile time zone TIM-001 refers to.
`currency_code`	`String`	yes	—	—	ISO-4217, e.g. `INR`. A `Money` is single-currency; this says which.
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	
`household_id`	`String`	yes	HouseholdEntity.DEFAULT_ID	—	The household this profile belongs to (issue 13.1, schema 30; ADR-0069 §1). Not null: every profile is in exactly one household. The migration adds th

ENT-003 — `account` (AccountEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`name`	`String`	yes	—	personal	
`type`	`String`	yes	—	—	An `AccountType.storedValue` (`:core:model`) — the §20.2 vocabulary, e.g. `credit_card`. A plain string with no CHECK constraint, so **a new type is n
`opening_balance_minor`	`Long`	yes	—	financial	**paise** MNY-001: paise, never a floating-point rupee value.
`current_balance_minor`	`Long`	yes	—	financial	**paise** MNY-001: paise. **A cache, not the truth (DB-001).** The current balance "is derivable from opening balance + transactions" and is "never mutated ad h
`currency_code`	`String`	yes	—	—	

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`institution`	`String?`	no	null	personal	The bank, issuer or custodian. Free text and optional — cash has none (issue 2.5).
`include_in_networth`	`Boolean`	yes	true	—	Whether this account counts towards net worth (issue 2.6; §20.2, FR-ACC-005). **Distinct from [archivedAtUtcMillis], and both from [deletedAtUtcMillis
`archived_at_utc_millis`	`Long?`	no	null	—	FR-ACC-007: when the user retired this account, or null while it is active. **Distinct from [deletedAtUtcMillis] on purpose.** A deleted account is a
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	

Indices: `(profile_id)`; `(profile_id, deleted_at_utc_millis)`

ENT-004 — `transactions` (TransactionEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`account_id`	`String`	yes	—	—	
`amount_minor`	`Long`	yes	—	financial	**paise** MNY-001: paise. Negative is an outflow, positive an inflow.
`currency_code`	`String`	yes	—	—	
`occurred_at_utc_millis`	`Long`	yes	—	—	TIM-001: the exact instant, UTC epoch millis.
`booked_on_iso_date`	`String`	yes	—	—	TIM-002: the profile-zone day this belongs to, ISO `yyyy-MM-dd`. Never a midnight timestamp.
`category_id`	`String?`	no	null	—	
`merchant`	`String?`	no	null	personal	
`note`	`String?`	no	null	personal	
`source`	`String`	yes	—	—	`manual` / `ocr` / `sms` / `import` / `reconciliation`. Provenance, so the UI can show where a row came from (P-02). `reconciliation` (issue 2.7, FR-A
`type`	`String`	yes	'expense'	—	`expense` / `income` / `transfer_out` / `transfer_in` / `adjustment` (issue 3.2; §20.2). **Direction is stored twice — here and as [amountMinor]'s sig
`transfer_id`	`String?`	no	null	—	Links the two legs of a transfer (issue 3.2; FR-TXN-003). Null on every other row. Two rows share one id: the outflow from the source account and the
`posted_at_utc_millis`	`Long?`	no	null	—	When this row's booked day arrived and the app recorded it as posted (issue 3.4; FR-TXN-010). Null while the row is still future-dated. **This column
`nature`	`String?`	no	null	—	The user's nature override, or `null` (issue 4.3; §8.3, `CategoryNature.storedValue`). **`null` does not mean "unknown".** §8.3 makes nature "auto-ass
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	
`tax_minor`	`Long?`	no	null	financial	**paise** The tax on this bill — in practice GST — in paise (MNY-001). Schema 32; §33, ADR-0077. Why: §33's forward-compatibility table promised "GST fields cap

Column	Kotlin type	Req?	Default	Sensitivity	Notes
import_batch_id	String?	no	null	—	The import run this row arrived in — §33's "statement-grade provenance". Schema 33; ADR-0078. Why: §33 promised import_batches from v1 and ADR-0074

Indices: (profile_id, booked_on_iso_date); (account_id); (category_id); (transfer_id)

ENT-005 — transaction_splits (TransactionSplitEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
transaction_id	String	yes	—	—	
amount_minor	Long	yes	—	financial	**paise** MNY-001: paise, signed **the same way as the parent**. A line of an expense is negative like its parent, so SUM(lines) = parent.amount_minor is a pl
category_id	String?	no	null	—	The category this line is attributed to. Null until issue 4.1 gives a real profile any categories at all — the same state transactions.category_id i
note	String?	no	null	personal	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (transaction_id); (profile_id); (category_id)

ENT-006 — tags (TagEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
name	String	yes	—	personal	The label as the user typed it, trimmed. **Case is preserved but uniqueness is not case-sensitive** — the repository lower-cases before looking a tag
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: unique (profile_id, name)

ENT-007 — transaction_tags (TransactionTagEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
transaction_id	String	yes	—	—	
tag_id	String	yes	—	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (transaction_id); (tag_id); (profile_id)

ENT-008 — `category` (CategoryEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`name`	`String`	yes	—	personal	
`parent_id`	`String?`	no	null	—	
`nature`	`String`	yes	—	—	`need` / `want` / `invest` / `asset` / `liability`. Set by issue 4.3's classifier.
`is_system`	`Boolean`	yes	—	—	True for the seeded defaults, so a user-created category can be told apart from ours.
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	

Indices: `(profile_id)`; `(parent_id)`

ENT-009 — `budget` (BudgetEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`category_id`	`String?`	no	null	—	Set when this budget is for one category (issue 4.4); null for a nature-level envelope.
`nature`	`String?`	no	null	—	`need` / `want` / `invest`. Set when [categoryId] is null — the group this envelope covers.
`period_start_iso_date`	`String`	yes	—	—	TIM-002: the first day of the budget month, ISO `yyyy-MM-dd`. Never a midnight timestamp.
`amount_minor`	`Long`	yes	—	financial	**paise** MNY-001: paise.
`rollover_enabled`	`Boolean`	yes	false	—	FR-BUD-001's optional rollover of an unused amount. Off by default.
`source`	`String`	yes	—	—	`quick_setup` / `manual` / `suggested`. Where the figure came from (P-02).
`rule_id`	`String?`	no	null	—	The rulebook row that produced this amount, or null when the user typed it (AI-ARC-006).
`rule_version`	`String?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	

Indices: `(profile_id, period_start_iso_date)`; `(category_id)`

ENT-010 — `budget_alert` (BudgetAlertEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`budget_id`	`String`	yes	—	—	The `budget` row this alert is about.
`category_id`	`String?`	no	null	—	Denormalised from the budget so the banner can group without a join. Null for an envelope.
`month_start_iso_date`	`String`	yes	—	—	TIM-002: the first day of the budget month, ISO `yyyy-MM-dd`. Part of the unique key.
`band`	`String`	yes	—	—	`WARN` / `EXCEEDED` — `BudgetAlertBand.name`. Part of the unique key, so both can fire.

Column	Kotlin type	Req?	Default	Sensitivity	Notes
rule_id	String	yes	—	—	The rulebook row that set this band, and its version (AI-ARC-006, P-02).
rule_version	String	yes	—	—	
notified_at_utc_millis	Long	yes	—	—	TIM-001: when the user was actually told, UTC epoch millis.

Indices: unique (profile_id, budget_id, month_start_iso_date, band); (profile_id, month_start_iso_date)

ENT-011 — budget_review (BudgetReviewEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
month_start_iso_date	String	yes	—	—	TIM-002: the first day of the **reviewed** (closed) month, ISO yyyy-MM-dd.
rule_id	String	yes	—	—	The rulebook row that produced this review, and its version (AI-ARC-006, P-02).
rule_version	String	yes	—	—	
total_budgeted_minor	Long	yes	—	financial	**paise** Audit-only snapshot of the reviewed month's totals (MNY-001), not re-read by the app.
total_actual_minor	Long	yes	—	financial	**paise**
reviewed_at_utc_millis	Long	yes	—	—	TIM-001: when the user dismissed or acted on the review, UTC epoch millis.

Indices: unique (profile_id, month_start_iso_date)

ENT-012 — recurring_rule (RecurringRuleEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
account_id	String?	no	null	—	Which account it moves through. Null until issue 2.5 gives the user an account to attach.
category_id	String?	no	null	—	Attached by issue 4.1/4.3 once categories exist.
name	String?	no	null	personal	A user- or merchant-supplied name (issue 3.7). Null for quick-setup rows — see [seedKind].
seed_kind	String?	no	null	—	income / rent_emi / savings for a quick-setup row; null otherwise. A code, not copy.
amount_minor	Long	yes	—	financial	**paise** MNY-001: paise, signed. Negative is an outflow, positive an inflow — as in transactions.
cadence	String	yes	—	—	monthly. A string, so a new cadence is not a migration.
next_due_iso_date	String	yes	—	—	TIM-002: the next occurrence, ISO yyyy-MM-dd.
source	String	yes	—	—	quick_setup / detected / manual. Provenance, so the UI can show where it came from.
is_confirmed	Boolean	yes	false	—	FR-TXN-006: false until the user confirms. An unconfirmed rule creates nothing.

Column	Kotlin type	Req?	Default	Sensitivity	Notes
dismissed_at_utc_millis	Long?	no	null	—	FR-TXN-006: when the user said "not recurring", UTC epoch millis (TIM-001); null otherwise. **Deliberately not [deletedAtUtcMillis].** A tombstone mea
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id, next_due_iso_date); (account_id)

ENT-013 — audit_log (AuditLogEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	Long	yes	0L	—	**PK**
event	String	yes	—	—	An AuditEvent constant name. A code from a closed set, never a sentence.
occurred_at_utc_millis	Long	yes	—	—	TIM-001: the exact instant, UTC epoch millis, from the injected Clock.
method	String?	no	null	—	An AuditMethod constant name, or null where the event had no factor.

Indices: (occurred_at_utc_millis)

ENT-014 — net_worth_snapshot (NetWorthSnapshotEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
as_of_iso_date	String	yes	—	—	TIM-002: the profile-zone day this figure describes, ISO yyyy-MM-dd. Never a timestamp.
assets_minor	Long	yes	—	financial	**paise** MNY-001: paise. Signed — an overdrawn asset account legitimately reduces this.
liabilities_minor	Long	yes	—	financial	**paise** MNY-001: paise, a positive magnitude — what the user owes, as the engine reports it.
net_worth_minor	Long	yes	—	financial	**paise** MNY-001: paise. assets_minor - liabilities_minor, stored so history cannot drift.
engine_id	String	yes	—	—	The engine that produced this row (AI-ARC-003).
engine_version	String	yes	—	—	That engine's version at the moment it ran (AI-ARC-006).
computed_at_utc_millis	Long	yes	—	—	TIM-001: when it was computed, which is not the same as the day it describes.
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id, as_of_iso_date)

ENT-015 — attachments (AttachmentEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`transaction_id`	`String`	yes	—	—	The transaction this belongs to. One transaction may hold several attachments.
`kind`	`String`	yes	—	—	`receipt` today. A code from a closed set, never copy — the UI maps it to a word, so a translation never reaches the database (§21.6). /
`file_name`	`String`	yes	—	—	The ciphertext blob's file name inside app-private storage — **not a path**. A stored absolute path would break the moment Android moved the app's dat
`mime_type`	`String`	yes	—	—	e.g. `image/jpeg`. What the *plaintext* was, so a future export knows what it is handing over.
`byte_size`	`Long`	yes	—	—	The plaintext size in bytes, so a settings screen can say what receipts are costing.
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	FR-OCR-005's "delete image while keeping the transaction": this is stamped **and** the blob is erased. The tombstone stays so a later sync can see the

Indices: `(transaction_id)`; `(profile_id)`

ENT-016 — `sms_draft` (SmsDraftEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`sms_id`	`Long`	yes	—	—	`Telephony.Sms._ID` of the message this was read from. Kept so the same alert is never proposed twice, and so a draft can be traced back to its source
`sender`	`String`	yes	—	personal	The DLT header the alert came from, e.g. `VM-HDFCBK`. Shown so the user can recognise it.
`amount_minor`	`Long`	yes	—	financial	**paise** MNY-001: paise, as a positive magnitude. The sign comes from [direction].
`direction`	`String`	yes	—	—	`debit` or `credit`. A code from a closed set, never copy (§21.6).
`booked_on`	`String`	yes	—	—	TIM-002: ISO `yyyy-MM-dd`, the day the message arrived in the profile zone.
`counterparty`	`String?`	no	null	—	The payee or payer as the alert printed it. Null when it named none — best-effort.
`account_tail`	`String?`	no	null	—	The masked account or card digits the alert quoted, e.g. `4521`. Null when it quoted none.
`confidence_bps`	`Int`	yes	—	—	MNY-002: basis points. Below the rulebook's floor the review screen flags the draft.
`engine_version`	`String`	yes	—	—	The parser version and the rulebook version that produced this reading (AI-ARC-006). Stored on the row rather than assumed from the current build, bec
`rule_version`	`String`	yes	—	—	
`status`	`String`	yes	—	—	`pending`, `accepted` or `dismissed`. A code from a closed set, never copy.
`transaction_id`	`String?`	no	null	—	The transaction this draft became, once accepted. Null while pending or dismissed.
`created_at_utc_millis`	`Long`	yes	—	—	

Column	Kotlin type	Req?	Default	Sensitivity	Notes
updated_at_utc_millis	Long	yes	—	—	

Indices: unique (profile_id, sms_id); (profile_id, status)

ENT-017 — credit_card (CreditCardEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
account_id	String	yes	—	—	**PK** The account row this describes. One card per account, which is why it is the key.
profile_id	String	yes	—	—	
credit_limit_minor	Long	yes	—	financial	**paise** MNY-001: paise. Positive — utilisation divides by it, and the model refuses zero.
statement_day	Int	yes	—	—	1..31, clamped to the month's length when resolved into a date.
due_day	Int	yes	—	—	1..31. May precede [statementDay] — the payment then falls in the following month.
last_statement_minor	Long?	no	null	financial	**paise** MNY-001: paise on the most recent statement. Null until one is recorded — not zero (P-03).
last_statement_iso_date	String?	no	null	—	TIM-002: when that statement was cut, ISO yyyy-MM-dd.
minimum_due_minor	Long?	no	null	financial	**paise** MNY-001: paise. The minimum payment on that statement, when the user has entered it.
apr_bps	Int?	no	null	—	MNY-002: the purchase APR in integer basis points. Never a Double.
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id); (profile_id, deleted_at_utc_millis)

ENT-018 — loan (LoanEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
account_id	String	yes	—	—	**PK** The account row this describes. One loan per account, which is why it is the key.
profile_id	String	yes	—	—	
principal_minor	Long	yes	—	financial	**paise** MNY-001: the sanctioned amount in paise. Positive — the model refuses zero and below.
annual_rate_bps	Int	yes	—	—	MNY-002: the annual rate in integer basis points. Never a Double. 0 is a real 0% EMI.
tenure_months	Int	yes	—	—	The number of instalments, 1..600 (fifty years — longer than any Indian retail loan).
first_emi_iso_date	String	yes	—	—	TIM-002: when the first instalment falls, ISO yyyy-MM-dd. Its day is the loan's EMI day.
emi_override_minor	Long?	no	null	financial	**paise** MNY-001: paise. The instalment the lender actually charges, when the user has entered it. Null means "derive it" — not zero, which would be an instalm
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id); (profile_id, deleted_at_utc_millis)

ENT-019 — `card_alert` (CardAlertEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`account_id`	`String`	yes	—	—	The `credit_card` / `account` row this alert is about. Part of the unique key.
`cycle_start_iso_date`	`String`	yes	—	—	TIM-002: the statement date opening the cycle, ISO `yyyy-MM-dd`. Part of the unique key.
`kind`	`String`	yes	—	—	`DUE_SOON` / `UTILISATION` — `CardAlertKind.name`. Part of the key, so both can fire.
`rule_id`	`String`	yes	—	—	The rulebook row that fired this alert, and its version (AI-ARC-006, P-02).
`rule_version`	`String`	yes	—	—	
`notified_at_utc_millis`	`Long`	yes	—	—	TIM-001: when the user was actually told, UTC epoch millis.

Indices: `unique (profile_id, account_id, cycle_start_iso_date, kind)`; `(profile_id, cycle_start_iso_date)`

ENT-020 — `investment_holding` (InvestmentHoldingEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`account_id`	`String`	yes	—	—	The `account` row this sits in. Indexed, not the key: an account holds many of these.
`name`	`String`	yes	—	personal	The user's own label, e.g. "Parag Parikh Flexi Cap". Never blank — the model refuses it.
`asset_class`	`String`	yes	—	—	`AssetClass.storedValue`. Issue 6.4's allocation dimension; `RULE-GOLD-CAP` names two.
`unit_price_minor`	`Long?`	no	null	financial	**paise** MNY-001: paise **per unit** as last observed. Null means never priced — not zero, which would read as a worthless holding rather than an unvalued one
`priced_on_iso_date`	`String?`	no	null	—	TIM-002: the day [unitPriceMinor] was observed, ISO `yyyy-MM-dd`. Both-or-neither with the price: it is XIRR's terminal flow date, and without it the
`price_key`	`String?`	no	null	—	The instrument identifier a market-data proxy resolves (issue 6.5; §16 EXT-003, §22.2), e.g. `gold:inr.gram.24k`. Null means this holding is priced by
`price_fetched_at_utc_millis`	`Long?`	no	null	—	TIM-001: when a *fetched* price was last written, UTC epoch millis. Null when the current price was typed by the user rather than fetched (issue 6.5).
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	

Indices: `(profile_id)`; `(profile_id, deleted_at_utc_millis)`; `(account_id)`

ENT-021 — `investment_lot` (InvestmentLotEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
holding_id	String	yes	—	—	The investment_holding row this belongs to. Indexed: a holding has many lots.
kind	String	yes	—	—	buy / sell / income — LotKind.storedValue. The only source of the cash direction.
transacted_on_iso_date	String	yes	—	—	TIM-002: the day the money moved, ISO yyyy-MM-dd. XIRR weights by this.
quantity_nano	Long	yes	—	—	Units moved x 10^9, as a magnitude. Zero for an income lot: a dividend moves no units.
amount_minor	Long	yes	—	financial	**paise** MNY-001: cash moved in paise, as a magnitude, charges included. Always positive.
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id); (profile_id, deleted_at_utc_millis); (holding_id)

ENT-022 — goal (GoalEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
name	String	yes	—	personal	The user's own label, e.g. "Emergency fund" or "Kerala trip". Never blank — the model refuses it.
target_minor	Long	yes	—	financial	**paise** MNY-001: what the goal needs in total, in paise. Zero is legitimate and means "no target set yet" — an ordinary state for a goal the user has just cre
target_date_iso	String	yes	—	—	TIM-002: the day the money is needed, ISO yyyy-MM-dd. A date, not a timestamp — "the 1st of April" is the same day in every time zone, and storing i
saved_minor	Long	yes	0L	financial	**paise** MNY-001: what is set aside so far, in paise. May exceed [targetMinor] — that is over-funded.
planned_monthly_minor	Long	yes	0L	financial	**paise** MNY-001: what the user intends to contribute each month, in paise. Zero means "no plan yet", and the engine then reports no ETA rather than a date inf
sort_order	Int	yes	0	—	Where the user dragged this goal in the contribution waterfall (issue 7.3; §15). The order the surplus is poured in, and therefore the only input to t
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (profile_id); (profile_id, deleted_at_utc_millis)

ENT-023 — goal_contribution (GoalContributionEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
goal_id	String	yes	—	—	

Column	Kotlin type	Req?	Default	Sensitivity	Notes
transaction_id	String	yes	—	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (goal_id); (transaction_id); (profile_id); unique (goal_id, transaction_id)

ENT-024 — goal_funding_account (GoalFundingAccountEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
goal_id	String	yes	—	—	
account_id	String	yes	—	—	
linked_from_iso_date	String	yes	—	—	TIM-002: the first day whose movements count, ISO yyyy-MM-dd. A date, not a timestamp, for the reason [GoalEntity.targetDateIso] is one — the bounda
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	

Indices: (goal_id); (account_id); (profile_id); unique (goal_id, account_id)

ENT-025 — insight (InsightEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
fingerprint	String	yes	—	—	
type	String	yes	—	—	
severity	String	yes	—	—	
subject	String?	no	null	—	
subject_label	String?	no	null	personal	
period	String	yes	—	—	
amount_minor	Long?	no	null	financial	**paise**
secondary_minor	Long?	no	null	financial	**paise**
date_iso	String?	no	null	—	
quantity	Int?	no	null	—	
confidence_bps	Int?	no	null	—	
citations	String	yes	—	—	The cited rules as ID vVERSION, comma-separated — the card's "why", rendered as it is stored.
source_engine_id	String	yes	—	—	
source_engine_version	String	yes	—	—	
status	String	yes	—	—	active, dismissed, snoozed or acted (§20.2).
suppressed_until_iso_date	String?	no	null	—	ISO date the suppression ends, for a dismissed or snoozed row (TIM-002).
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); unique (profile_id, fingerprint)

ENT-026 — `notification_log` (NotificationLogEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`key`	`String`	yes	—	—	What makes this message this message — an insight's fingerprint, an alert's id.
`kind`	`String`	yes	—	—	The §17.1 taxonomy row, as `NotificationKind.name`.
`outcome`	`String`	yes	—	—	`deliver`, `wait_for_quiet_hours`, `fold_into_digest` or `already_sent`.
`decided_at_utc_millis`	`Long`	yes	—	—	
`sent_at_utc_millis`	`Long?`	no	null	—	When it was sent — set only for `deliver`, and what the caps count.
`deliver_after_utc_millis`	`Long?`	no	null	—	When a held message may go (NTF-002).
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `unique (profile_id, key)`

ENT-027 — `purchase_trace` (PurchaseTraceEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`item`	`String`	yes	—	—	The user's own words for the thing. Their data, never the app's copy.
`price_minor`	`Long`	yes	—	financial	**paise**
`method`	`String`	yes	—	—	`CASH`, `CARD` or `EMI`.
`urgency`	`String`	yes	—	—	`ROUTINE`, `SOON` or `URGENT` — what the user said when asked.
`monthly_emi_minor`	`Long?`	no	null	financial	**paise**
`category_id`	`String?`	no	null	—	
`verdict`	`String`	yes	—	—	`COMFORTABLE`, `STRETCH` or `NOT_NOW`.
`hard_fail`	`Boolean`	yes	—	—	Whether a gate failed outright — what urgency was not allowed to soften.
`liquid_before_minor`	`Long`	yes	—	financial	**paise**
`liquid_after_minor`	`Long`	yes	—	financial	**paise**
`runway_before_tenths`	`Int`	yes	—	—	Months of essentials covered, in tenths (MNY-001 keeps floats away from money).
`runway_after_tenths`	`Int`	yes	—	—	
`goal_delay_days`	`Int`	yes	—	—	
`comfortable_price_minor`	`Long?`	no	null	financial	**paise**
`comfortable_from_iso_date`	`String?`	no	null	—	ISO date the price would stop crossing the emergency floor (TIM-002).
`cool_off_suggested`	`Boolean`	yes	—	—	
`engine_id`	`String`	yes	—	—	
`engine_version`	`String`	yes	—	—	
`citations`	`String`	yes	—	—	The card's own cited rules as `ID vVERSION`, comma-separated. Not derivable from the gate rows: RULE-COOL-OFF decides the cooling-off suggestion and i
`decided_on_iso_date`	`String`	yes	—	—	The day the advice was given, in the profile's zone (TIM-002).
`decided_at_utc_millis`	`Long`	yes	—	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(profile_id, decided_at_utc_millis)`

ENT-028 — `purchase_trace_gate` (PurchaseTraceGateEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`trace_id`	`String`	yes	—	—	
`gate`	`String`	yes	—	—	The `GateId` name — `AFFORDABILITY`, `CASH_FLOW`, and so on.
`outcome`	`String`	yes	—	—	`PASS`, `WARN` or `FAIL`.
`ordinal`	`Int`	yes	—	—	§13.1's order, kept so the card reads the way it was decided.
`citations`	`String`	yes	—	—	The gate's cited rules as `ID vVERSION`, comma-separated (P-02, AI-ARC-006).
`figure_key`	`String?`	no	null	—	The figure's stable key, or `null` for a gate that judged nothing.
`amount_minor`	`Long?`	no	null	financial	**paise**
`count_value`	`Int?`	no	null	—	
`bps_value`	`Int?`	no	null	—	
`text_value`	`String?`	no	null	—	
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(trace_id)`

ENT-029 — `wishlist_item` (WishlistItemEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`name`	`String`	yes	—	personal	The user's own words for the thing.
`est_price_minor`	`Long`	yes	—	financial	**paise**
`category_id`	`String?`	no	null	—	
`method`	`String`	yes	—	—	`CASH`, `CARD` or `EMI` — an instalment makes any wish heavy (§13.3.1).
`monthly_emi_minor`	`Long?`	no	null	financial	**paise**
`urgency`	`String`	yes	—	—	`ROUTINE`, `SOON` or `URGENT`.
`want_score`	`Int`	yes	—	—	§13.3.2's WantScore, as it stood after the last answer.
`status`	`String`	yes	—	—	`watching`, `parked`, `removed` or `bought` (§13.3).
`target_price_minor`	`Long?`	no	null	financial	**paise** An optional price to watch for.
`last_interviewed_at_utc_millis`	`Long?`	no	null	—	When the interview last ran, so the thirty-day window has something to count from.
`last_trace_id`	`String?`	no	null	—	The last Purchase Advisor card for this wish, when one has been asked for (issue 10.1).
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(profile_id, status)`

ENT-030 — `interview_answer` (InterviewAnswerEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**

Column	Kotlin type	Req?	Default	Sensitivity	Notes
profile_id	String	yes	—	—	
item_id	String	yes	—	—	
question	String	yes	—	personal	The InterviewQuestion name.
answer_key	String	yes	—	personal	The answer's stable key — need, uses_high, owns_similar — which is also the delta's name.
points	Int	yes	—	—	What the answer was worth when it was given, kept so an old score stays explicable (AI-ARC-006).
amount_minor	Long?	no	null	financial	**paise** An amount the answer carried, where it had one (total cost of ownership).
answered_at_utc_millis	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (item_id); unique (profile_id, item_id, question)

ENT-031 — vehicle (VehicleEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
label	String	yes	—	personal	
vehicle_class	String	yes	—	—	The knowledge base's own class string, so a row and a KB row cannot drift apart.
account_id	String?	no	null	—	
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id)

ENT-032 — vehicle_odometer (VehicleOdometerEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
vehicle_id	String	yes	—	—	
read_iso_date	String	yes	—	—	
km	Long	yes	—	—	
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (profile_id, vehicle_id); unique (profile_id, vehicle_id, read_iso_date)

ENT-033 — vehicle_service (VehicleServiceEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
vehicle_id	String	yes	—	—	
serviced_iso_date	String	yes	—	—	

Column	Kotlin type	Req?	Default	Sensitivity	Notes
odometer_km	Long	yes	—	—	
cost_minor	Long	yes	—	financial	**paise**
note	String?	no	null	personal	
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (profile_id, vehicle_id)

ENT-034 — vehicle_renewal (VehicleRenewalEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
vehicle_id	String	yes	—	—	
item	String	yes	—	—	
last_done_iso_date	String	yes	—	—	
last_cost_minor	Long?	no	null	financial	**paise**
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); unique (profile_id, vehicle_id, item)

ENT-035 — chat_message (ChatMessageEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
question	String	yes	—	personal	
answer	String	yes	—	personal	
intent	String?	no	null	—	
refusal	String?	no	null	—	
model_id	String	yes	—	—	
citations	String	yes	""	—	
asked_at_utc_millis	Long	yes	—	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (profile_id, asked_at_utc_millis)

ENT-036 — market_close (MarketCloseEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
id	String	yes	—	—	**PK**
profile_id	String	yes	—	—	
price_key	String	yes	—	—	
close_iso_date	String	yes	—	—	
close_minor	Long	yes	—	financial	**paise**
source	String	yes	"quote"	—	
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (profile_id, price_key); unique (profile_id, price_key, close_iso_date)

ENT-037 — `appliance` (ApplianceEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`label`	`String`	yes	—	personal	
`appliance_class`	`String`	yes	—	—	
`purchased_on_iso_date`	`String`	yes	—	—	
`rated_watts`	`Int?`	no	null	—	Null means the knowledge base's figure for the class, not zero.
`minutes_per_day`	`Int?`	no	null	—	Minutes, never hours: a fraction of an hour would put a decimal into money arithmetic.
`tariff_paise_per_kwh`	`Int?`	no	null	—	What this household pays for a unit, in paise (MNY-001). Null means the KB's default.
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(profile_id, deleted_at_utc_millis)`

ENT-038 — `appliance_service` (ApplianceServiceEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`appliance_id`	`String`	yes	—	—	
`serviced_iso_date`	`String`	yes	—	—	
`cost_minor`	`Long?`	no	null	financial	**paise**
`note`	`String?`	no	null	personal	
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(profile_id, appliance_id)`

ENT-039 — `appliance_consumable` (ApplianceConsumableEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**
`profile_id`	`String`	yes	—	—	
`appliance_id`	`String`	yes	—	—	
`item`	`String`	yes	—	—	
`replaced_iso_date`	`String`	yes	—	—	
`cost_minor`	`Long?`	no	null	financial	**paise**
`deleted_at_utc_millis`	`Long?`	no	null	—	
`created_at_utc_millis`	`Long`	yes	—	—	
`updated_at_utc_millis`	`Long`	yes	—	—	

Indices: `(profile_id)`; `(profile_id, appliance_id, item)`

ENT-040 — `import_batches` (ImportBatchEntity)

Column	Kotlin type	Req?	Default	Sensitivity	Notes
`id`	`String`	yes	—	—	**PK**

Column	Kotlin type	Req?	Default	Sensitivity	Notes
profile_id	String	yes	—	—	
source	String	yes	—	—	aa / import — a TransactionSource.storedValue, the same vocabulary the rows carry.
started_at_utc_millis	Long	yes	—	—	TIM-001: when the app ran this import, from the injected clock.
fetched_at_utc_millis	Long?	no	null	—	TIM-001: when the data was true at the source. Null for a file, which has no fetch instant. Separate from [startedAtUtcMillis] because they answer dif
window_start_iso_date	String?	no	null	—	TIM-002: the first day the statement covered, ISO yyyy-MM-dd. Null when unstated.
window_end_iso_date	String?	no	null	—	TIM-002: the last day the statement covered. Null when unstated.
complete	Boolean	yes	1	—	False when the source truncated the window — FetchedStatements.complete. NOT NULL with a default, which the [taxMinor] column deliberately refused
line_count	Int	yes	0	—	How many rows the source offered.
accepted_count	Int	yes	0	—	How many became transactions. Stored rather than counted from transactions, and the difference is the point: a row the user later deletes would make
deleted_at_utc_millis	Long?	no	null	—	
created_at_utc_millis	Long	yes	—	—	
updated_at_utc_millis	Long	yes	—	—	

Indices: (profile_id); (profile_id, started_at_utc_millis)

<!– 40 entities, 435 columns –>

4. Relationships

Cardinality is read from the schema. **On delete** describes what the repository does, because Room foreign-key constraints are deliberately not declared: an archive import replaces a whole profile, and a constraint would refuse rows that arrive before their parents (ADR-0023). The invariant is enforced in the repository and asserted by tests rather than by the engine.

From	To	Cardinality	Column	On delete	Optional?
ENT-002 profile	ENT-001 household	many-to-one	household_id	Not applicable — the household is never deleted	No; the migration defaults it
ENT-003 account	ENT-002 profile	many-to-one	profile_id	Soft-delete cascade through the profile	No
ENT-004 transactions	ENT-003 account	many-to-one	account_id	Soft-deleting an account soft-deletes its transactions	No
ENT-004 transactions	ENT-008 category	many-to-one	category_id	Category delete leaves the transaction uncategorised rather than deleting it	Yes — an uncategorised transaction is legal
ENT-004 transactions	ENT-004 transactions	the two legs of one transfer	transfer_id	Deleting one leg deletes both (BR-009)	Yes — only transfers carry it
ENT-004 transactions	ENT-040 import_batches	many-to-one	import_batch_id	Wiping a batch hard-deletes the rows it created, verified afterwards	Yes
ENT-005 transaction_splits	ENT-004 transactions	many-to-one	transaction_id	Cascade with the parent	No

From	To	Cardinality	Column	On delete	Optional?
ENT-005 `transaction_splits`	ENT-008 `category`	many-to-one	`category_id`	As for the parent	No — a line without a category has no purpose
ENT-007 `transaction_tags`	ENT-004, ENT-006	join	`transaction_id`, `tag_id`	Hard-deleted with either side	No
ENT-008 `category`	ENT-008 `category`	self, one level	`parent_id`	A parent delete orphans nothing: children are re-parented or soft-deleted together	Yes — a top-level category has none
ENT-009 `budget`	ENT-008 `category`	many-to-one	`category_id`	Soft-delete together	No
ENT-010 `budget_alert`	ENT-009 `budget`	many-to-one	`budget_id`	With the budget	No
ENT-012 `recurring_rule`	ENT-003, ENT-008	many-to-one	`account_id`, `category_id`	With either	No
ENT-015 `attachments`	ENT-004 `transactions`	many-to-one	`transaction_id`	The row is soft-deleted; the blob is removed separately, which is where `RISK-009` lives	No
ENT-016 `sms_draft`	ENT-004 `transactions`	zero-or-one	`transaction_id`	Set when the draft is confirmed into a transaction	Yes — an unconfirmed draft has none
ENT-017 `credit_card`	ENT-003 `account`	**one-to-one**	`account_id` **is** the PK	With the account	No
ENT-018 `loan`	ENT-003 `account`	**one-to-one**	`account_id` **is** the PK	With the account	No
ENT-019 `card_alert`	ENT-003 `account`	many-to-one	`account_id`	With the account	No
ENT-020 `investment_holding`	ENT-003 `account`	many-to-one	`account_id`	With the account	No
ENT-021 `investment_lot`	ENT-020 `investment_holding`	many-to-one	`holding_id`	With the holding	No
ENT-023 `goal_contribution`	ENT-022 `goal`	many-to-one	`goal_id`	With the goal	No
ENT-023 `goal_contribution`	ENT-004 `transactions`	zero-or-one	`transaction_id`	Deleting the transaction removes the evidenced contribution; a declared one has no link to lose	Yes — this is exactly what separates evidenced from declared (BR-027)
ENT-024 `goal_funding_account`	ENT-022, ENT-003	join	`goal_id`, `account_id`	Hard-deleted with either side	No
ENT-028 `purchase_trace_gate`	ENT-027 `purchase_trace`	many-to-one	`trace_id`	With the trace	No
ENT-029 `wishlist_item`	ENT-027 `purchase_trace`	zero-or-one	`last_trace_id`	The trace outlives the wish	Yes
ENT-030 `interview_answer`	ENT-029 `wishlist_item`	many-to-one	`item_id`	With the item	No
ENT-031 `vehicle`	ENT-003 `account`	zero-or-one	`account_id`	A vehicle may exist without a vehicle-type account	Yes
ENT-032/033/034	ENT-031 `vehicle`	many-to-one	`vehicle_id`	With the vehicle	No
ENT-038/039	ENT-037 `appliance`	many-to-one	`appliance_id`	With the appliance	No

Columns that look like foreign keys and are not. `profile.time_zone_id` is an IANA zone identifier. `budget.rule_id`, `budget_alert.rule_id`, `budget_review.rule_id` and `card_alert.rule_id` hold a **rulebook row id** from `ai/rules/`, not a table key — that is how an insight cites the threshold that fired (BR-018). `net_worth_snapshot.engine_id`, `insight.source_engine_id`, `purchase_trace.engine_id` and `chat_message.model_id` are provenance (AI-ARC-003). `sms_draft.sms_id` is the inbox message id, which is why the draft can be de-duplicated without storing the message body.

5. Keys

Decision	Choice	Why
Primary-key strategy	A client-generated `TEXT` UUID	There is no server to allocate ids, and an archive restored on another device must not collide with rows created there. An auto-increment integer would collide on exactly the path the product promises (FLOW-006)
Natural keys	None are used as primary keys	Every candidate — an account name, a category name, a date — is user-editable, and an editable primary key makes a rename a data migration
Uniqueness that matters	`(profile_id, name)` on `tags`, `category`, `account`; `fingerprint` on `insight`; the claim keys in §7	These are the real-world constraints, enforced as unique indexes rather than as application checks
1:1 extension tables	`credit_card` and `loan` use `account_id` as the primary key	The relationship is strictly one-to-one, and a separate surrogate id would permit two cards on one account — a state no code could handle
Composite keys	Not used	Room supports them, but every join in this schema is a single column, and a composite key would complicate the archive's shape for nothing

6. Constraints, and the rule each enforces

Constraint	Where it lives	Rule
Amounts are integers	Column type plus a lint detector	BR-001
Rates are integers	Column naming plus review	BR-002
Split lines sum to the parent	The engine, plus a property test over generated inputs	BR-008
A transfer has exactly two legs	The repository, inside one transaction	BR-009, BR-016
A balance equals opening plus transactions	A correlated subquery on read, plus the nightly integrity job	BR-007, FR-095
One budget per category per month	Unique index	VAL-011
One alert per band per month	Claim row `budget_alert`	BR-019
One card reminder per statement cycle	Claim row keyed by `cycle_start_iso_date`	BR-020
One review per closed month	Claim row keyed by `(profile, month)`	ADR-0020
One snapshot per date	Claim row keyed by date	ENT-014
One insight per finding	Unique `fingerprint`	FR-075
No query spans profiles	`ProfileScopingTest` reads every DAO query	BR-006
Every row is recoverable from a mistaken delete	`deleted_at_utc_millis`	BR-005
Every multi-row write is atomic	Room `@Transaction`	BR-016

7. Normalisation, and every deliberate exception

The schema is third normal form, with four denormalisations, each argued:

Denormalisation	Why it is there	What keeps it honest
`account.current_balance_minor`	A derived balance needs a correlated subquery; a cached column keeps a list of 30 accounts cheap	It is documented in the entity as "a cache, not the truth", the derivation is authoritative on read, and the nightly integrity job compares them (DB-001, ADR-0007)
`transactions.classification`, `nature` and their confidences	Classification is expensive and its result is read on every list row	Each carries a confidence, and a user override is stored while an inference is re-derivable (ADR-0016)

Denormalisation	Why it is there	What keeps it honest
`transaction_splits.category_id` duplicating the parent's category	A split line's category is the point of splitting	The parent's category is the fallback for an unsplit transaction only; category spend is a `UNION ALL` of the two, so a payment is counted exactly once (ADR-0018)
`insight.evidence_json` and the other `*_json` blobs	Evidence shape differs per engine, and a normalised evidence table would be a column per engine or a key-value sprawl	Each blob is versioned `{"v":1,...}` so an old insight still renders after an upgrade (DB-005)

What is deliberately *not* stored, because a stored copy of a derivable figure is a second source of truth that drifts: account balances (ADR-0007), loan amortisation schedules (ADR-0026), holding values, costs and gains (ADR-0027), and health-score history (ADR-0045).

8. Indexes

81 indexes are declared across the 40 tables. Each exists for a named query:

Index	The query it serves
`transactions(profile_id, booked_on_iso_date)`	The transaction list, every month total, and the forecast's window reads — the single hottest path in the app
`transactions(account_id)`	The balance derivation per account
`transactions(category_id)`	Category spend, budget pace, and the classification feedback loop
`transactions(transfer_id)`	Finding the other leg of a transfer, on both read and delete
`transaction_splits(transaction_id)`	Expanding a split; also the `UNION ALL` half of category spend
`account(profile_id, deleted_at_utc_millis)`	The accounts list and the net-worth composition, which must skip deleted rows
`budget(profile_id, month)` unique with `category_id`	The one-budget-per-category-per-month constraint, and the budget screen's month switch
`insight(fingerprint)` unique	The orchestrator's deduplication
`net_worth_snapshot(profile_id, as_of_iso_date)`	The chart's range queries, and the per-date claim
`investment_lot(holding_id)`	XIRR's cash-flow series per holding
`goal_contribution(goal_id)`	Goal progress
`vehicle_odometer(vehicle_id)`	The usage trend the service prediction regresses over
`market_close(price_key, close_iso_date)`	The signal engine's history reads and the valuation's latest-close lookup
`attachments(transaction_id)`	Rendering a transaction's receipts
`import_batches(profile_id, started_at_utc_millis)`	The import history, and the wipe verification

Every declared index is listed under its entity in §3.

9. Transaction boundaries

Operation	Boundary	Why
Record a transfer	One transaction, both legs	A one-legged transfer would misstate two balances (BR-009)
Save a split	One transaction: parent and every line	Lines that do not sum are invalid, and a partial write is exactly that (BR-008)
Quick-setup seed	One transaction: budgets and recurring rules together	A half-seeded plan is worse than an unseeded one (UC-002-F1)

Operation	Boundary	Why
Import a batch	One transaction: the batch row first, then its rows	An orphan row with no batch could never be wiped (UC-055-F1)
Archive import	One transaction: replace the profile's data	Replace is the only import a lossless round-trip can be asserted against (ADR-0023)
Erase	Not a database transaction — the key is destroyed first	After the key is gone, the ciphertext is unreadable whatever happens next (ADR-0060)
Daily snapshot	One upsert per date	The date is the claim key, so a re-run updates rather than duplicates

10. Concurrency

Concern	Approach
Writers	One SQLCipher connection in WAL mode. Room serialises writes; readers are not blocked by the writer
UI reads	`Flow`-returning DAO queries, collected on the main dispatcher and computed on an injected IO dispatcher. `runBlocking` is banned outside tests
Background work	Nine WorkManager jobs, each idempotent through a claim key, so two runs of the same job on one day cannot produce two rows
The lock	While the app is locked, the Hilt provider for the database **refuses to build**, so no coroutine can hold an open connection across a lock (ADR-0003)
Clock	Every time read goes through the injected `Clock`, so two components in one computation cannot disagree about "now" (BR-003)
Engines	Pure functions over immutable inputs. There is no shared mutable state to contend for (`P-08`)

11. Retention and archival

Data	Retained	Purged by
Transactions, accounts, budgets, goals and every other profile-scoped row	Indefinitely — this is the user's own ledger, and a finance app that forgets is useless	Soft delete, then erase-all
Soft-deleted rows	Indefinitely today. The SRS specifies a 30-day trash compactor; it is not built (`GAP-006`)	erase-all
Receipt blobs	With their transaction	Deleting the image, or erase-all. An orphaned blob is a known defect (`RISK-009`, ADR-0078)
SMS drafts	Only while the SMS consent stands	Revoking the consent deletes them immediately (BR-028)
`market_close`	Through a consent revoke, by design	erase-all
`audit_log`	Indefinitely, and never exported	erase-all
Backup blobs	Wherever the user put them — the app does not manage their lifetime	The user, in their own storage
Archived accounts	Fully, including history	erase-all

Rights served locally (DPDP Act 2023, SEC-008): export is the access right, erase-all is the erasure right, and the consents dashboard is purpose limitation. None of them needs a server, which is the point.

12. Migrations

Schema version **33**, with 32 hand-written migrations from version 1. Every one ships with a test against a real pre-change database fixture, and **destructive migrations are forbidden** (BR-017, DB-003).

Rule	How it holds
Never drop or rewrite user data	A column is added, never repurposed. A table is added, never replaced
Every migration has a test	`:core:database` has a migration-test harness (issue 1.7); a version bump without a test fails review
A column added by `ALTER TABLE` is declared **last** in the entity	`ALTER TABLE ... ADD COLUMN` appends, so a mid-list declaration would make Room's exported schema disagree with a migrated database's real column order. `profile.household_id` carries this comment in the source
The SQL default and the Kotlin default must agree	Both are declared, so Room's exported schema and the hand-written migration *agree* rather than merely coinciding
An archive from a newer version is refused	`VAL-020` — refusing is honest; guessing is not

---

13. The engine choice

Room over SQLCipher (AES-256), in WAL mode, with the key wrapped in the Android Keystore. Pinned by SRS §21.3 and implemented in `:core:database`.

- **Context.** Every figure the product computes is derived from local rows, on a device that can be lost or stolen. The data must be queryable with joins and aggregates, and unreadable to anyone holding the storage.
- **Options considered.** Room over plain SQLite — rejected, because extracted storage would be plaintext (SEC-001). A document store or flat files — rejected, because the balance derivation, category spend and forecast windows are all relational aggregates. Realm or ObjectBox — rejected, because Room's migration story is explicit and testable, which `BR-017` depends on.
- **Decision.** Room + SQLCipher, key in Keystore/StrongBox, rotatable (ADR-0057).
- **Consequences.** Positive: encryption at rest is transparent to every query, and crypto-shredding becomes possible — destroy the key and the ciphertext is gone (ADR-0060). Negative: the database throws while the app is locked, which is why the widget cannot read it (ADR-0024); and SQLCipher adds to the APK, which NFR-005 budgets.

Related decisions sit in 05 §6 and in `docs/adr/`.

---

14. What the SRS specified and the schema does not have

The SRS §20.1 inventories roughly 75 tables. The schema holds 40. That is not a shortfall to hide — it is the consequence of a rule the project kept: a table is added by the feature issue that needs it, because "under DB-003 a wrong column guessed today costs a migration tomorrow, while a table added later is a trivial non-destructive one" (`Entities.kt`).

SRS table	Status	Where the need went
`loan_amortization_rows`	Deliberately absent	The schedule is derived from five columns on every read (ADR-0026)
`account_balances_history`	Deliberately absent	`net_worth_snapshot` holds the daily series; a per-account history would duplicate the ledger
`asset_valuations`, `price_points`	Partly present as `market_close`	One cached close per price key per date is what the engines actually read
`health_scores`, `health_score_components`	Deliberately absent	The score is computed live and reweighted over present signals (ADR-0045) — which is also `GAP-013`
`forecasts`, `forecast_points`, `forecast_accuracy`	Absent	The forecast is computed on demand; accuracy tracking is deferred (ADR-0043, `GAP-003`)
`consents`, `user_settings`, `feature_flags`	Present, but in Proto DataStore rather than in the database	Settings and the consent ledger are preferences, not relational data (SRS §21.3; SharedPreferences is banned)

SRS table	Status	Where the need went
`household_members`	Replaced	ADR-0069 models a household as a row *above* the profile, and rejected the join table
`merchants`, `merchant_aliases`, `classification_rules`	Absent as tables	They live as versioned rows in `ai/`, mirrored into typed Kotlin and guarded by a drift test (ADR-0014)
`rules_knowledge_base`, `signal_configs`, `backtest_results`	Absent as tables	Same — `ai/rules/` is the store, and an insight cites the row id and version
`chat_sessions`, `chat_tool_calls`	Folded into `chat_message`	One row per exchange carries its own tool calls; sessions had no user-visible meaning
`gold_items`, `properties`, `subscriptions`, `scheduled_items`, `fuel_logs`, `service_knowledge_base`, `renewal_items`	Absent or renamed	Gold and property are account types; `vehicle_renewal` generalised `renewal_items`; the **fuel log is a real gap** (`GAP-009`)
`jobs_log`, `backups_log`, `schema_migrations_log`, `export_batches`, `kv_cache`, `sms_parse_rules`, `fx_rates`, `macro_indicators`, `fuel_prices`, `calendar_events`	Absent	Each belongs to a feature that is not built, and each would be a non-destructive addition when it is (`GAP-001`, `GAP-015`)

The honest summary: **35 of the SRS's tables were not needed in the shape the SRS imagined**, mostly because derived figures were not stored and thresholds moved out of the database into versioned data files. **Two are genuine gaps** — the fuel log (`GAP-009`) and the external-data tables that the unbuilt backend would fill (`GAP-001`).

---

15. The eight table families, and the invariant each one holds

The attribute tables in §3 say what every column is. This section says why the forty tables are grouped the way they are, because a reader who understands the eight families can predict where a new table belongs — and that is the property that keeps a schema from turning into forty unrelated tables.

Each family is described by the invariant it exists to hold, what reads it, and the one way it could go wrong.

15.1 Identity — `ENT-001`, `ENT-002` **The invariant.** Every row of user data belongs to exactly one profile, and no query spans profiles (`BR-006`).

The household row sits *above* the profile rather than beside it, so household aggregation composes per-profile reads instead of widening a query (ADR-0069). That is the whole reason `household` exists in v1 while household mode is off: the foundation is built so the feature does not require a schema change later.

Read by every repository, on every query. **The way it goes wrong:** a query that forgets its `profile_id`, which `ProfileScopingTest` fails the build on.

15.2 The ledger — `ENT-003`...`ENT-008`, `ENT-040` **The invariant.** Every monetary figure in the app derives from `transactions`. Nothing else is a source of truth about money.

`account.current_balance_minor` exists and is **not read**: the balance a screen shows is `opening_balance + SUM(live transactions)`, derived on every read (ADR-0007). The cached column is the left-hand side of the nightly integrity comparison, nothing more. A transfer is two legs sharing a `transfer_id` with no parent row, because a parent would hold no fact the legs do not (ADR-0008). Splits are a child table that moves no money — the parent holds the amount (ADR-0009).

Read by every engine, every screen that shows a figure, and the forecast. **The way it goes wrong:** a one-legged transfer or a split that does not sum, both unrepresentable because the writes are transactional (`BR-016`) and both asserted by property tests.

15.3 The plan and its claims — ENT-009...ENT-012, ENT-019, ENT-022...ENT-024 **The invariant.** A claim is keyed by the thing it claims, so a worker that runs twice cannot act twice.

This is the family where the keys carry the design. `budget_alert` is keyed by band and month, so the 80% warning fires once even if the worker is deferred for a day. `budget_review` is keyed by `(profile, month)` and deliberately **not** by category, so a month that went badly in six envelopes interrupts once rather than six times (ADR-0020). `goal_contribution` keeps evidenced and declared progress in separate rows that are never summed (ADR-0036).

Read by the budget, review, goal and waterfall engines, and the monthly ritual. **The way it goes wrong:** a claim keyed by the calendar month where the real cycle is not a month — which is exactly the bug `card_alert` avoids by keying on `cycle_start_iso_date` (ENT-019, ADR-0025).

15.4 Instruments — ENT-017, ENT-018, ENT-020, ENT-021 **The invariant.** Store the terms; derive everything else.

A loan stores five columns and the amortisation schedule is derived on every read, so a stale schedule cannot exist (ADR-0026) — the SRS's own `loan_amortization_rows` table was refused for exactly that reason. A holding stores units and an observed price with its date; a lot stores a dated buy or sell, which is what makes a money-weighted return computable at all (FR-046).

Read by the loan, investment, card, simulator and health-score engines. **The way it goes wrong:** a derived figure cached somewhere and then read after the terms change. Nothing in this family caches, which is the point.

15.5 Evidence — ENT-013, ENT-025...ENT-028, ENT-030, ENT-035 **The invariant.** A figure the user was shown must still be explainable months later, from what the engine was given at the time.

A chat message stores the answer given **and the tool calls behind it**, which is the same discipline applied to prose. An insight carries its provenance and is deduplicated by fingerprint. A purchase trace keeps the figures each gate used, rather than a recipe for recomputing them — because the engines move, and a verdict is only defensible if its inputs are kept with it (AI-ARC-006). `audit_log` is append-only, carries no `profile_id` and is excluded from the export, which is a deliberate asymmetry rather than an oversight (ADR-0023).

Read by the insight feed, the advisor, the notification gate, and the user reading their own record. **The way it goes wrong:** an evidence row that stores a reference instead of a value, which turns an audit trail into a recomputation.

15.6 Attachments and capture — ENT-015, ENT-016 **The invariant.** The database holds metadata; the encrypted filesystem holds bytes.

An attachment row references a blob by content hash. The two are written in one transaction, so a row never points at a blob that is not there. The reverse — a blob with no row — is a known open defect (RISK-009, ADR-0078), and it is a storage leak rather than a confidentiality one, because the blob is unreadable without the key an erase destroys. `sms_draft` is the only table that is hard-deleted on a consent revocation, and that asymmetry is argued in BR-028.

Read by the receipt flow, the transaction detail screen, the SMS inbox. **The way it goes wrong:** the orphan blob above. It is in the register rather than in a comment.

15.7 The derived series — ENT-014 **The invariant.** A series row is keyed by the period it describes, so a second write is an update and never a duplicate.

`net_worth_snapshot` is one immutable row per date. A missed day leaves a visible gap and nothing is interpolated; a back-dated transaction repairs the affected days rather than leaving a step in the chart (ADR-0012). There is deliberately **no** health-score series: the score is computed live from current data, which is stated rather than implied (ADR-0045, GAP-013).

Read by the net-worth chart, the dashboard sparkline, the widget. **The way it goes wrong:** two rows for one date, which the date key makes impossible.

15.8 Owned things — `ENT-029`, `ENT-031`…`ENT-035`, `ENT-037`…`ENT-039` **The invariant.** A prediction about a thing is derived from dated observations of that thing, never from a class default presented as a personal figure.

Vehicles and appliances share a shape: the thing, its dated readings or services, and its dated renewals. `chat_message` sits in this numeric range by accident of when it was added and belongs to §15.5's family, which is noted here so the ranges above read correctly. The next-service prediction is a Theil–Sen regression over the personal usage index where enough readings exist, and the class default otherwise — **labelled as a default** (ADR-0052). A wishlist item is explicitly not a transaction, and nothing removes one automatically (ADR-0050).

Read by the vehicle and appliance engines, the obligations strip, the buy list. **The way it goes wrong:** a class default shown as a personal prediction, which the labelling rule exists to prevent.

15.9 The public fact — `ENT-036` One table is not user data at all. `market_close` caches a daily close for a price key: a public fact about the world, not an inference about the person. That is why it survives a MARKET_DATA revocation while `sms_draft` does not (`BR-028`), and why it and `audit_log` are the only two tables in the schema that are not profile-scoped.

15.10 The partition, checked The nine families are a partition, not a loose grouping: every one of the forty entities appears in exactly one.

Family	Entities	Count
15.1 Identity	ENT-001, ENT-002	2
15.2 The ledger	ENT-003…ENT-008, ENT-040	7
15.3 The plan and its claims	ENT-009…ENT-012, ENT-019, ENT-022…ENT-024	8
15.4 Instruments	ENT-017, ENT-018, ENT-020, ENT-021	4
15.5 Evidence	ENT-013, ENT-025…ENT-028, ENT-030, ENT-035	7
15.6 Attachments and capture	ENT-015, ENT-016	2
15.7 The derived series	ENT-014	1
15.8 Owned things	ENT-029, ENT-031…ENT-034, ENT-037…ENT-039	8
15.9 The public fact	ENT-036	1
Total		**40**

The smallest family has one member, and that is the point of listing it separately: `net_worth_snapshot` is the only table in the schema that stores a derived figure as a dated series, and `market_close` is the only one that is not about the user at all.

Part 05 — Application Architecture

← Master Blueprint · Related: 04 Data · 06 API · 09 Risks & Decisions

1. Architecture style, and why

A feature-based, one-way modular monolith on the device, with a layered AI pipeline inside it. 60 Gradle modules, no server in the product's path.

Three forces picked this shape, and each one rules out the obvious alternative:

1. **Privacy first (`P-01`) removes the server.** If no financial data may leave the device, there is nowhere else to put the logic. A client-server split would have to either send the data or duplicate every engine — so the engines live on the phone.
2. **Offline first (`P-04`) makes the network a decoration.** Every core feature must work in airplane mode, and the build and the tests must pass with the backend absent. That forbids an architecture in which anything important is a remote call.
3. **"Numbers from math, words from AI" (`P-03`) forces the layering.** If a language model may never compute, then computation and language have to be separable components with a checkable boundary between them. That boundary is the guardrail.

The module graph runs `feature` → `domain` → `data/core` and never backwards (ARC-001). Feature modules never depend on each other; cross-feature navigation goes through typed routes in a single nav graph assembled in `:app`. `:core:model`, `:core:common` and every `:domain:*` module are **pure Kotlin with no Android imports** (ARC-002), enforced by a convention-plugin TestKit guard that runs in CI — which is what keeps the engines unit-testable without a device, and is the only reason a KMP port was even attemptable (ADR-0075).

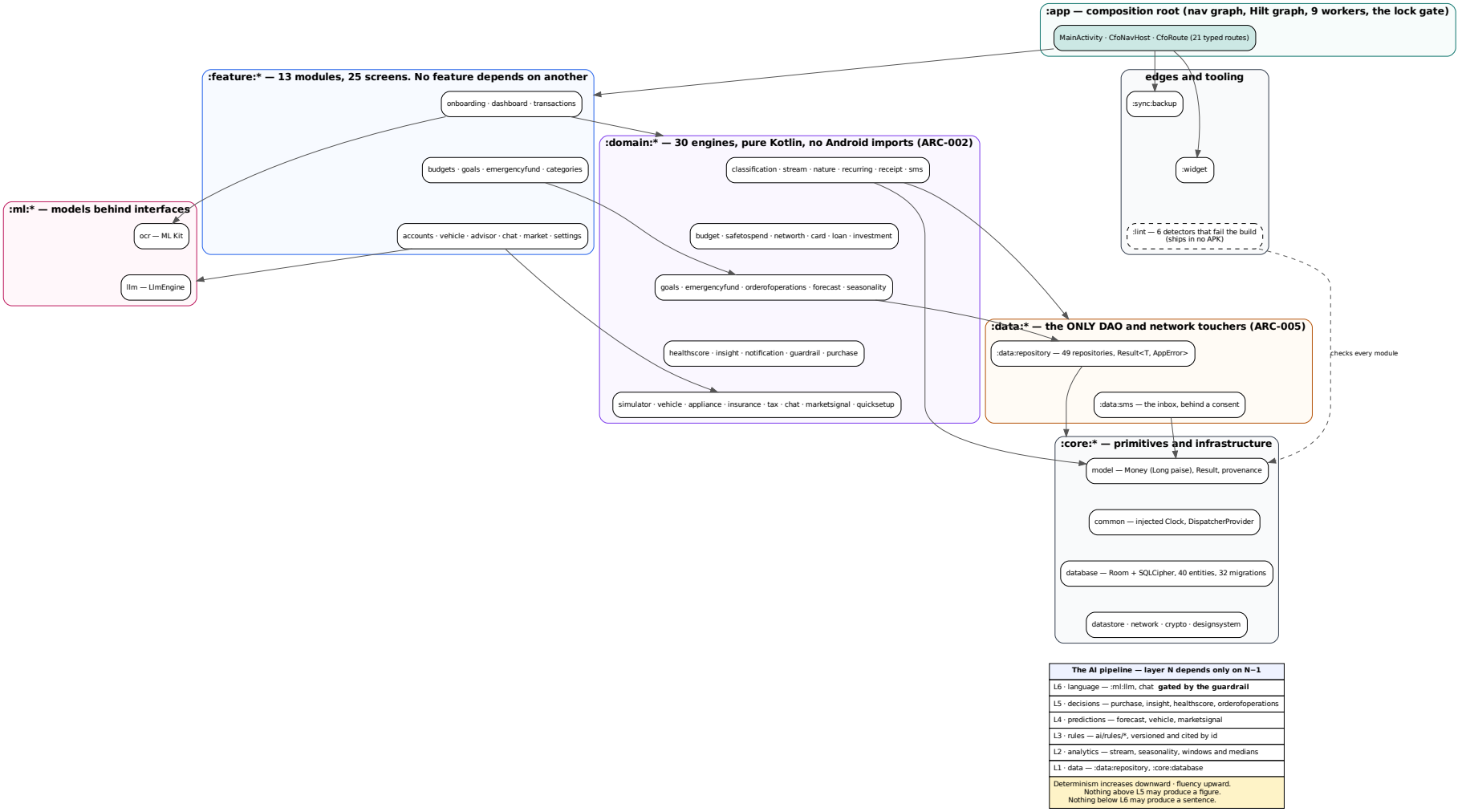


Figure 4 — The module graph and the AI pipeline. Arrows run one way only; the panel on the right is the six-layer pipeline, in which nothing above L5 may produce a figure.

Source: diagrams/architecture.dot.

1.1 The AI pipeline Layer N depends only on the layers below it (SRS §7; `ai/architecture/ai-architecture.md`):

Layer	What it does	Where it lives
L1 — data	The ledger and its derived reads	`:data:repository`, `:core:database`
L2 — analytics	Windows, medians, variances, streams, seasonality	`:domain:engines:stream`, `:seasonality`
L3 — rules	Versioned thresholds, applied and cited	`ai/rules/*`, read by every engine
L4 — predictions	Forecast, service prediction, market signals	`:domain:engines:forecast`, `:vehicle`, `:marketsignal`
L5 — decisions	Verdicts, rankings, scores, waterfalls	`:domain:engines:purchase`, `:insight`, `:healthscore`, `:orderofoperations`
L6 — language	Turning finished figures into sentences	`:ml:llm`, `:domain:engines:chat`, gated by `:domain:engines:guardrail`

Determinism increases downward; fluency increases upward. **Nothing above L5 may produce a figure, and nothing below L6 may produce a sentence.**

2. Components

60 Gradle modules. One responsibility each; the dependency column names what it is allowed to see.

2.1 Application shell

CMP	Module	Responsibility	Owns	Depends on
CMP-001	`:app`	Single-activity Compose host: the nav graph, the Hilt graph, the nine workers, the notification channels, the app lock gate, the SMS consent watcher	No entities	Everything — it is the composition root and nothing depends on it

`:app` also holds the one module-level exception worth naming: the app **lock gate** (`AppLockGate`) is the single source of truth for lock state, and the Hilt provider for the database refuses to build while it says locked (ADR-0003). That is a wiring responsibility, so it belongs here rather than in a feature.

2.2 Core — portable primitives and Android infrastructure

CMP	Module	Responsibility	Owns	Depends on
CMP-002	`:core:model`	`Money` (Long paise), `Result<T, AppError>`, `EngineProvenance`, the account/category/asset-class enums, `MoneyFormatter` with Indian grouping	ENT-free; the types every layer shares	Nothing. Pure Kotlin, 100% line coverage
CMP-003	`:core:common`	The injected `Clock`, `DispatcherProvider`, and `SystemClock` — the one sanctioned wall-clock read in the whole tree	—	`:core:model`
CMP-004	`:core:database`	Room + SQLCipher: the 40 entities, 34 DAOs, 32 migrations, the key management and rekeying	ENT-001…ENT-040	`:core:model`, `:core:crypto`
CMP-005	`:core:datastore`	Proto DataStore: settings and the per-feature consent ledger with grant and revoke timestamps	The consent ledger	`:core:model`
CMP-006	`:core:network`	The Retrofit/OkHttp edge. Ships **unconfigured** — a blank base URL, so no HTTP stack is ever constructed	—	`:core:model`
CMP-007	`:core:crypto`	Tink and Keystore: the database key, StrongBox placement, rotation, Argon2id derivation for backups	—	`:core:model`

CMP	Module	Responsibility	Owns	Depends on
CMP-008	`:core:designsystem`	Material 3 theme tokens and the Compose canvas charts. No heavyweight chart library	—	`:core:model`

2.3 Domain — the engines Every engine is pure Kotlin, exposes exactly one public interface plus its result types, has an `internal` implementation injected by constructor (ARC-003), carries provenance (AI-ARC-003), and owns a frozen golden-file fixture.

CMP	Module	Public interface	What it decides	SRS
CMP-009	`:domain:engines: quicksetup`	`QuickSetupEngine`	Income, rent and savings → budget envelopes and an emergency-fund target	FR-002
CMP-010	`:domain:engines: classification`	`ClassificationEngine`	Merchant → category, with confidence and evidence	§8.1
CMP-011	`:domain:engines:stream`	`StreamEngine`	Each expense stream → FIXED / SEMI_FIXED / VARIABLE	§8.2
CMP-012	`:domain:engines:nature`	`NatureEngine`	What a rupee became: NEED / WANT / INVESTMENT / ASSET / LIABILITY	§8.3
CMP-013	`:domain:engines:recurring`	`RecurringEngine`	Which repeated transactions are a series worth proposing	FR-018
CMP-014	`:domain:engines:receipt`	`ReceiptEngine`	Total, date, merchant and GST off recognised text	FR-025
CMP-015	`:domain:engines:sms`	`SmsEngine`	Amount, direction and payee off an opt-in bank alert	§18, §23
CMP-016	`:domain:engines:budget`	`BudgetEngine`	Suggested budgets, and spent / pace / projection	§5.5
CMP-017	`:domain:engines: safetospend`	`SafeToSpendEngine`	What is left to spend this month, and every line withheld	§5.2
CMP-018	`:domain:engines:networth`	`NetWorthEngine`	Assets − liabilities, and the daily snapshot	FR-043
CMP-019	`:domain:engines:card`	`CardEngine`	A card's billing cycle, utilisation and reminder dates	FR-040
CMP-020	`:domain:engines:loan`	`LoanEngine`	An EMI's principal/interest split, and the schedule	FR-041
CMP-021	`:domain:engines: investment`	`InvestmentEngine`	A holding's value, gain and money-weighted return	§11
CMP-022	`:domain:engines:goals`	`GoalEngine`, `GoalWaterfallEngine`	Required monthly, ETA, horizon; and who gets the surplus first	§15
CMP-023	`:domain:engines: emergencyfund`	`EmergencyFundEngine`	Essentials × multiplier → target, runway, coaching line	§10.1
CMP-024	`:domain:engines: orderofoperations`	`OrderOfOperationsEngine`	The surplus down the eight-stage waterfall	§36
CMP-025	`:domain:engines:forecast`	`ForecastEngine`	The 90-day liquid-balance forecast, P10/P50/P90, and crunch days	§9
CMP-026	`:domain:engines: seasonality`	`SeasonalityEngine`	A seasonal index per category and month	§9.3
CMP-027	`:domain:engines: healthscore`	`HealthScoreEngine`	Weighted signals → a 0–1000 score, a band, contributions, the lever	§14
CMP-028	`:domain:engines:insight`	`InsightEngine`	Engine results → a deduplicated, ranked feed	§7.2
CMP-029	`:domain:engines: notification`	`NotificationPolicyEngine`	What may interrupt the user, and when	§17.2
CMP-030	`:domain:engines:guardrail`	`GuardrailEngine`	Whether a figure may be said at all	AI-ARC-004
CMP-031	`:domain:engines:purchase`	`PurchaseAdvisorEngine`, `PurchaseInterviewEngine`	Can I afford this, and the trace that says why	§13
CMP-032	`:domain:engines:simulator`	`PrepayVsInvestSimulator`, `DebtPayoffSimulator`	Prepay versus invest, and which debt first	§36, §40.2
CMP-033	`:domain:engines:vehicle`	`VehicleEngine`	When the next service falls due, and what it will cost	§12

CMP	Module	Public interface	What it decides	SRS
CMP-034	`:domain:engines:appliance`	`ApplianceEngine`	An appliance's service, consumables, warranty and running cost	§12
CMP-035	`:domain:engines:insurance`	`ProtectionEngine`	Term and health cover gaps, and the endowment detector	§39.1
CMP-036	`:domain:engines:tax`	`TaxEngine`	The two regimes, capital gains, and what it will not estimate	§38
CMP-037	`:domain:engines:chat`	`ChatEngine`, `LlmEngine`	Intent → registry tools → a guardrailed reply	§19
CMP-038	`:domain:engines:marketsignal`	`MarketSignalEngine`	Cached closes → an opportunity score and its measured hit rate	§30
CMP-039	`:domain:usecase`	Use-case types shared by features	—	`:core:model`

2.4 Data — the only DAO and network touchers (ARC-005)

CMP	Module	Responsibility	Owns	Depends on
CMP-040	`:data:repository`	49 repository and support types: the only classes that touch DAOs. Returns `Result<T, AppError>` and domain models — never Room or Retrofit types	Every entity's read and write path	`:core:database`, `:core:model`, `:domain:*`
CMP-041	`:data:sms`	The phone's inbox, behind `READ_SMS` and an explicit consent	ENT-016	`:core:model`, `:core:datastore`

2.5 ML — on-device models behind interfaces

CMP	Module	Responsibility	Depends on
CMP-042	`:ml:ocr`	ML Kit Text Recognition v2 behind an interface, so the recogniser is replaceable — which is what makes a fully free build a contained change	`:core:model`
CMP-043	`:ml:llm`	The on-device LLM behind `LlmEngine`, with Apache-2.0 weights (ADR-0079). Swappable; a cloud implementation would be opt-in and is not built	`:core:model`

2.6 Features — Compose screens and ViewModels Each feature owns its screens and ViewModels, exposes one immutable `UiState` per screen as a `StateFlow`, and takes events up through a sealed interface (ARC-004). **No feature depends on another.**

CMP	Module	Screens	Requirements
CMP-044	`:feature:onboarding`	`OnboardingScreen`	FR-001…FR-005
CMP-045	`:feature:dashboard`	`DashboardScreen`, `OrderOfOperationsScreen`	FR-006…FR-012, FR-079, FR-083
CMP-046	`:feature:transactions`	`TransactionsScreen`, `AddTransactionScreen`, `ReceiptReviewScreen`, `SmsDraftsScreen`	FR-013…FR-028
CMP-047	`:feature:categories`	`CategoriesScreen`	FR-056, FR-060
CMP-048	`:feature:budgets`	`BudgetsScreen`	FR-029…FR-033
CMP-049	`:feature:goals`	`GoalsScreen`, `GoalDetailScreen`	FR-034…FR-038, FR-072
CMP-050	`:feature:emergencyfund`	`EmergencyFundScreen`	FR-065
CMP-051	`:feature:accounts`	`AccountsScreen`, `AccountEditorScreen`, `HoldingsScreen`, `AllocationScreen`	FR-039…FR-047
CMP-052	`:feature:vehicle`	`VehiclesScreen`	FR-048…FR-050, FR-067, FR-068

CMP	Module	Screens	Requirements
CMP-053	`:feature:advisor`	`AdvisorScreen`, `SimulatorsScreen`	FR-051…FR-053, FR-069…FR-071, FR-080…FR-082
CMP-054	`:feature:chat`	`ChatScreen`	FR-054, FR-076, FR-077
CMP-055	`:feature:market`	`OpportunityScreen`	FR-078
CMP-056	`:feature:settings`	`SettingsScreen`, `ConsentsScreen`, `EraseScreen`	FR-056…FR-059, FR-087…FR-090

2.7 Platform edges and build tooling

CMP	Module	Responsibility
CMP-057	`:sync:backup`	End-to-end-encrypted backup and restore: Argon2id → AES-256-GCM, to user-chosen storage through SAF
CMP-058	`:widget`	The Glance home-screen widget. Renders from Glance preference state **only**, never from Room, because the database throws while the app is locked (ADR-0024)
CMP-059	`:lint`	Six custom Android Lint detectors that make the binding rules fail the build. Build tooling, in no APK, and deliberately outside the §21.2 graph (ADR-0001)
CMP-060	`:spike:kmp`	A time-boxed feasibility spike: can the money maths run on Kotlin/Native? Answer: not yet, and `Money` is the blocker (ADR-0075). Never shipped

2.8 How a request travels Three shapes cover the whole app, and `FLOW.md` holds the traced call chains:

- **A screen.** `Screen` → `ViewModel` → `UseCase/Repository` → `Engine` → `Repository` → `DAO` → `SQLCipher`, with state coming back as one immutable `UiState` on a `StateFlow`.
- **A background worker.** `WorkManager` → `Worker` → `Repository` → `Engine` → `Repository` → `DAO`, each worker idempotent through a claim key.
- **An engine.** Pure function: typed input → typed result + provenance. No I/O, no clock read, no randomness except through an injected seedable source (`P-08`).

3. Where the layers are enforced

A rule that is only written down is a rule that drifts. Each of these fails a build:

Rule	Enforcement	Mechanism
ARC-002 — engines have no Android imports	CI	A TestKit test over the convention plugins
ARC-006 — no `GlobalScope`	`:lint`	`CfoGlobalScope`
MNY-001 — no floating-point money	`:lint`	`CfoMoneyAsFloatingPoint`, by **name** heuristic (ADR-0001)
TIM-001 — no wall-clock read in domain	`:lint`	`CfoWallClockInDomain`
NFR-018 — no PII or amounts in logs	`:lint`	`CfoPiiInLogs`
NFR-011 — no hardcoded UI strings	`:lint`	`CfoHardcodedUiString`, with `@Preview` data exempt
SEC-003 — no hand-rolled crypto	`:lint`	`CfoHandRolledCrypto`, with two argued exemptions (ADR-0063)
BR-006 — no query spans profiles	`unitTests`	`ProfileScopingTest` reads every DAO query
BR-018 — a typed mirror may not drift from its `ai/` row	`unitTests`	36 drift-test classes / 252 cases, with the data files declared as task inputs so the gate cannot be skipped
NFR-013 — coverage floors	`koverVerify`	A real `minBound`: 85%, and 100% on `:core:model`
NFR-020 — every screen has a current render	`unitTests`	`verifyPaparazziDebug`, 115 committed renders

Rule	Enforcement	Mechanism
NFR-009 — it works offline	`offlineSmoke`	An end-to-end test on an emulator in airplane mode (ADR-0067)
NFR-017 — a backup restores	`restoreDrill`	On a device, before every promotion (ADR-0041)

4. Integrations

INT	System	Direction	Protocol	Auth	Failure behaviour	Owner
INT-001	Market-data proxy (SRS §22)	Outbound, optional	HTTPS/REST, TLS 1.3 with certificate pinning	An anonymous device token	**Ships inert.** A blank base URL means no HTTP stack is constructed; a configured host cannot be expressed without pins. Every priced screen runs off cache or reports coverage (ADR-0030)	Would be the maintainer's; **not built** (`GAP-001`)
INT-002	ML Kit Text Recognition v2	On-device library	In-process	—	A failed parse degrades to manual entry; every field is editable	Google, linked under the AGPL linking exception
INT-003	On-device LLM (`LlmEngine`, Apache-2.0 weights)	On-device library	In-process	—	Unavailable or slow → the reply falls back to rendered figures without prose	ADR-0079
INT-004	Android Keystore / StrongBox	Platform	In-process	Device credential	Key generation failure stops onboarding at the security step; no profile is written	Platform
INT-005	BiometricPrompt (class 3)	Platform	In-process	Biometric	No hardware or no enrolment → PIN. Five failures → a doubling lockout	Platform
INT-006	WorkManager	Platform	In-process	—	A skipped run makes an alert late, never duplicated: every job is claim-keyed	Platform
INT-007	Storage Access Framework	Platform	In-process	User-granted document URI	A revoked URI fails the backup with a message, and the previous backup is untouched	Platform
INT-008	The SMS inbox (`READ_SMS`)	Inbound, opt-in	Content provider	Runtime permission **plus** an in-app consent	Revoking either stops reading and deletes the drafts	Platform; policy risk recorded in ADR-0013
INT-009	CameraX	Platform	In-process	Camera permission	Denied → gallery import remains	Platform
INT-010	Account Aggregator	Inbound, future	Would be HTTPS/REST	AA consent artefact	Not built. `transactions.source` reserves `aa`, and `import_batches` carries statement-grade provenance so the integration extends the schema (ADR-0074)	`OQ-001`
INT-011	Play Integrity	Platform, release builds	In-process	—	A failed check is reported, not enforced as a kill switch	Platform

No integration is on a core path. That is `P-04` stated as an architectural property rather than an intention: remove every row of this table and the app still records transactions, computes a forecast, and answers a purchase question.

5. Security architecture

5.1 Trust boundaries

1. **The device is the boundary** (`P-01`). Inside it, the app trusts itself; nothing crosses outward without an explicit, per-feature, revocable consent.
2. **The encrypted store is a boundary inside the device.** SQLCipher AES-256 under a key wrapped by a hardware-backed Keystore key. Extracted storage is ciphertext.
3. **The lock is a boundary in time.** While locked, the database provider will not build, so no code path reaches data without an unlock (ADR-0003).
4. **The guardrail is a boundary on language.** Figures cross into sentences only through a check (AI-ARC-004).

5.2 Controls

Concern	Control	Requirement
Authentication	There is no account to authenticate to. Local authentication is biometric class 3 with a PIN fallback	FR-084, SEC-002
Authorisation	One user, one role, per-profile data scoping	BR-006
Data at rest	SQLCipher AES-256; attachments on the encrypted filesystem, referenced by content hash	SEC-001, DB-006
Data in transit	TLS 1.3 with certificate pinning, no cleartext traffic (manifest-enforced). Today there is no configured host, so there is no traffic	SEC-001
Key management	256-bit key generated on first run, wrapped in Keystore/StrongBox, never written unwrapped, rotatable without data loss	SEC-001, ADR-0057
Secret management	There are no app secrets to manage — no API keys ship, because no host is configured	ADR-0030
Screen capture	`FLAG_SECURE` on every screen, with a user-controlled override	FR-086, ADR-0058
Shoulder surfing	A fixed-width amount mask that refuses to leak even the order of magnitude	FR-008, ADR-0022
Deletion	Crypto-shredding: the key dies first, then the files, then a report of what could not be reached	FR-058, SEC-006
Supply chain	OSV scan over the **resolved** coordinates that actually ship, after an earlier version queried versions nobody had	NFR-015, SEC-007
Release hardening	R8 minification, debuggable and backup off, and the log strip verified by reading the shipped DEX	NFR-016, ADR-0062
Cryptography discipline	Tink and platform APIs only; a lint rule makes hand-rolled crypto fail the build	SEC-003, ADR-0063
Compliance	DPDP Act 2023 alignment held as a checked document, with the consent record export-only	SEC-008, ADR-0061

5.3 Threat model

Threat	Mitigation
Lost or stolen **unlocked** phone	App lock with an idle timeout, the privacy blur, amount-free notifications
Lost or stolen **locked** phone, storage extracted	SQLCipher AES-256 with the key in hardware-backed Keystore
A malicious app, or a screen recorder	`FLAG_SECURE` everywhere; nothing financial in the recents thumbnail
Server breach	There is nothing to breach: no user data is held anywhere but the device, and a backup blob is client-side encrypted
Network attacker	TLS 1.3, certificate pinning, no cleartext — and no configured host at all today
A data-hungry vendor, meaning us	Analytics default off, aggregate only, one-tap revoke, and a full export
An AI that invents a figure	The guardrail, measured against a frozen set with an absolute threshold (ADR-0054)

6. Technology choices

Every row is pinned by SRS §21.3 and recorded in `DECISIONS.md`. A dependency added, removed or swapped needs a row there naming what it was chosen over — no exceptions, test-only libraries included.

Concern	Choice	Rule, and where the argument lives
Language	Kotlin only, JDK 17 toolchain (built and tested on JDK 21)	No Java sources
UI	Jetpack Compose + Material 3	No XML layouts except widget and notification remote views
Navigation	Navigation-Compose with typed routes	One nav graph, assembled in `:app` (ARC-001)

Concern	Choice	Rule, and where the argument lives
DI	Hilt	Constructor injection only; no service locators (ARC-003)
Persistence	Room + SQLCipher	04 §13
Preferences	Proto DataStore	SharedPreferences is banned
Background	WorkManager	Every periodic job; expedited only for backup-on-demand
Network	Retrofit + OkHttp + kotlinx.serialization	Certificate pinning to our own backend — and it ships unconfigured (ADR-0030)
OCR	CameraX + ML Kit Text Recognition v2	On-device only (FR-024)
On-device LLM	A Gemma-class model behind `LlmEngine`, Apache-2.0 weights	Swappable (ADR-0079); judged by the app's own renderings (ADR-0080)
Charts	Compose canvas charts in `:core:designsystem`	No heavyweight chart library
Crypto	Google Tink, BouncyCastle for Argon2id	SEC-003; the library row is in `DECISIONS.md` (ADR-0039)
Testing	JUnit, Turbine, MockK, Robolectric, Compose UI tests, Paparazzi	07 Tests
Static analysis	ktlint, detekt, Android lint, six custom detectors, SonarQube	ADR-0001, ADR-0081
CI/CD	GitHub Actions on JDK 21: build, test, lint, AI evals, screenshot diff, OSV, release hardening, restore drill	`main` is always releasable

7. Deployment architecture

Aspect	Reality
Environments	`dev` (integration) → `stage` (live testing) → `main` (releases). `main` and `stage` are protected; changes land only by reviewed, CI-passing PR
Runtime	One Android APK. There is no server-side runtime to deploy, operate, or pay for
Scaling unit	The user's phone. The product scales by installation, not by capacity
State	Entirely on the device: SQLCipher database, Proto DataStore, encrypted attachment files, Glance widget state
Release	`restoreDrill` on a device plus the offline smoke before any promotion into `stage` or `main`; a red drill blocks the release (DRL-001)
Rollback	A previous APK version. Because migrations are non-destructive and additive, a downgrade is survivable for the schema but is not a supported user path — the supported recovery is restore-from-backup
Versioning	SemVer in `VERSION`, with `versionCode`, `CHANGELOG.md` and the release metadata checked against each other by a task that has already caught two drifts in the history (ADR-0068)

8. Observability

The constraint that shapes this section: **the app may not phone home, and may not log an amount.** So observability is local, and it is mostly about the *integrity* of what the app computed rather than about traffic.

Signal	Approach
Logs	A structured logger. Amounts and personal data are banned, enforced by `CfoPiiInLogs`, and release builds have their log calls stripped — verified by reading the shipped DEX
Audit	Security-relevant events to `audit_log`, readable in the app and deliberately export-excluded
Metrics	Computed on-device. Nothing is uploaded unless analytics consent is on, which defaults off
Traces	The product's own answer to tracing is the **decision trace**: `purchase_trace` and `purchase_trace_gate` keep why a verdict was reached, and `insight.evidence_json` keeps what a finding was based on
Job health	`notification_log` and the claim tables make it visible whether a worker ran and what it claimed
Integrity	The nightly balance-integrity job is the closest thing to an alert: a drift between the derived and cached balance raises a prompt

Signal	Approach
Crash reporting	Not built. Crash-free rate (NFR-004) will come from Play vitals after launch (`OQ-005`)

9. Reliability

Failure mode	Behaviour	Recovery
Process death mid-write	Room + WAL: the write either landed or did not	None needed
The device is off for days	Workers are claim-keyed, so alerts are late rather than duplicated, and correctness never depends on a worker having run (ADR-0010)	Next run
An engine throws	A typed `Result` at the repository boundary; one card errors and the rest of the screen draws	Retry on the next read
The database cannot be opened while locked	The gate routes to the lock screen	Unlock
A key is invalidated by biometric re-enrolment	The key rotates without losing the database	Automatic (ADR-0057)
Storage lost — the phone is gone	**This is the real risk.** There is no server copy by design	The E2EE backup, which is why `restoreDrill` is a release gate (FR-088)
A lost backup passphrase	Unrecoverable, stated in advance on the recovery screen	None — and the UI says so rather than discovering it later
RPO	Since the last backup the user took. The app does not manage a schedule for them	—
RTO	One restore on a fresh device: minutes, proven by the drill on real hardware	—

10. Architecture decisions

The project's decision records are **four-digit** and live in `docs/adr/` — 80 of them, indexed in `DECISIONS.md`. Each one is the single source for its own argument, and this document does not restate one by hand: a hand-written second copy drifts from the first, and the copy is the one that loses.

What follows instead is **generated from the ADR files themselves** by `scripts/gen_adr_digest.py`. For each record it reads the status and date, the SRS clauses it binds, the problem it opens with, the claims its Decision section makes, the alternatives it turned down, and what it cost or what followed — all in the ADR's own words. Regenerate it and the digest moves with the records; it cannot say something the repository does not.

- **§10.1** is the 33 records that shape *this* document — the module graph, the storage decisions, the AI pipeline, the security boundary and the gates.
- **§10.2** is the complete index of all 80, so a reader can see the whole decision history and go to the record that interests them.

10.1 The decisions that shape this architecture

ADR-0001 — A `:lint` module outside the §21.2 graph, and name-based money detection

Status	accepted, 2026-07-25
Binds	§21.2 (module graph), §21.3 (custom lint in the pinned stack), §21.4 (MNY-001, TIM-001), §21.6 (strings, style), ARC-006, P-01
Record	`docs/adr/0001-custom-lint-module-and-money-heuristic.md`

The problem. `CLAUDE.md` §3 and §5 state four bans in the present tense — `Double` on money, wall-clock reads in domain code, `GlobalScope`, and hardcoded user-visible strings — plus a ban on logging PII or amounts.

What was decided.

- Put the detectors in a new top-level `:lint` module.
- Detect money by name, not by type.
- Scope by package as well as path.
- `:core:common` is exempt from TIM-001.

What was turned down.

- Put the detectors in `build-logic`.
- Detect money by type only (flag every `Double`).
- Detect money by annotation (`@MonetaryAmount`).
- Use a third-party rule set.

What followed.

- Positive: five rules that were honour-system now fail `./gradlew lint`.
- Negative / cost: the name heuristic will miss money in a badly named variable, and can flag a `Double` that merely sounds monetary.
- Negative / cost: `lintVersion` in the catalog must move in lockstep with `agp` (lint version = AGP version + 23) or the detectors compile against a different API than the one running them.

ADR-0003 — The app lock gates the session, not the Keystore key

Status	accepted, 2026-07-26
Binds	§23.1, §23.2 (SEC-001, SEC-002, SEC-003), §21.2, §21.6, P-01, TIM-001
Record	`docs/adr/0003-app-lock-gate-and-deferred-user-auth-key.md`

The problem. Issue 2.2 builds SEC-002: *"Biometric unlock via BiometricPrompt (class 3) with PIN fallback; 5 failed PIN attempts → 30 s lockout doubling."* The issue's own description adds that the lock "gates access to the encrypted store on cold start and after an idle timeout".

What was decided.

- The lock gates a session flag, and the Keystore key stays as issue 1.6 built it.
- SEC-001's user-authentication clause is deliberately left open.
- This is an accepted, open deviation from SEC-001, not a completed requirement.
- The lock screen lives in `:app`, not a new `:feature:*` module.
- The gate is a composable **wrapper**, not a `CfoRoute`.
- The PIN is verified against a Keystore-bound Tink MAC.

What was turned down.

- Bind the Tink master key to user authentication now — Rejected for the data-loss reason in decision 1: a user who removes their device lock screen would lose everything, with no server copy and .
- A UI-only lock, with no database guard — Rejected: "gates access to the encrypted store" would then be aspirational, and this project has already been bitten once by a gate that cou.
- Give the audit log its own unencrypted store, so no exemption is needed — Rejected: a security log outside the encrypted database is a security log an attacker can read and edit, which is worse than the narrow, typ.

ADR-0007 — Account balances are derived on read, and `wallet` is not an account type

Status	accepted, 2026-07-28

Binds	§5.7 (FR-ACC-001, FR-ACC-006, FR-ACC-007), §20.2 (`accounts` DDL), §20.3 (DB-001, DB-002), MNY-001, P-03
Record	`docs/adr/0007-account-balances-derived-not-stored.md`

The problem. Issue 2.5 builds the first CRUD a user has over accounts.

What was decided.

- Every balance the app reads is derived: `opening_balance_minor + SUM(live transactions)`.
- `AccountDraft` cannot express a balance.
- The account type vocabulary is `AccountType` in `:core:model`, with FR-ACC-001's eleven values and §20.2's exact stored strings.

What was turned down.

- Maintain `current_balance_minor` on every transaction write — Rejected for now.
- Drop `current_balance_minor` entirely — Rejected: DB-003 forbids destructive migrations, and the column is the thing FR-ACC-006's reconciliation compares against.
- Keep `wallet` as a synonym for `cash` — Rejected: a type that is in no requirement is a type no engine will ever handle, and "which of these two do I pick?" is a question no user s.

What it costs.

- A read now costs a subquery per account.
- The two figures can disagree.

ADR-0008 — A transfer is two linked legs, not a parent row; `type` is derived, never supplied

Status	accepted, 2026-08-02
Binds	§5.3 (FR-TXN-003, FR-TXN-001, FR-TXN-009), §20.1 (table inventory), §20.2 (`transactions` DDL), §20.3 (DB-001, DB-002, DB-003, DB-004), MNY-001, P-03
Record	`docs/adr/0008-transfers-as-linked-legs.md`

The problem. Two things in the SRS's own schema bear on how to build that, and the codebase matched neither before this issue.

What was decided.

- A transfer is two rows in `transactions` sharing a `transfer_id`. There is no `transfers` table.
- `transactions.type` is adopted from §20.2 — all five values.
- No caller ever supplies a `type`.
- The 5 → 6 migration backfills `type` for existing rows.

What was turned down.

- `transfer_id` alone, with no `type` column.
- A `transfers` parent table, per §20.1 — Rejected as above: no facts to hold.
- One row with two account columns.

What followed.

- "Money the user actually spent or received" is `type IN ('expense','income')` — a filter that cannot be got wrong the way a `transfer_id IS NULL AND amount < 0` convention can.
- Each account's balance stays a plain `SUM` over its own rows, so DB-001's derivation and ADR-0007's "the balance is derived, the column is a cache" survive untouched.
- Deleting either leg is one `UPDATE ... WHERE transfer_id = ?`, so there is no window in which one leg is gone and the other is not — FR-TXN-003's second clause is structural rather than sequenced.

ADR-0009 — Split lines are a child table that moves no money; the remainder rule is largest-remainder, not HALF_EVEN

Status	accepted, 2026-08-02
Binds	§5.3 (FR-TXN-004, FR-TXN-001), §20.1 (table inventory), §20.2 (`transactions` DDL), §20.3 (DB-001, DB-002, DB-003, DB-004), MNY-001, P-03
Record	`docs/adr/0009-splits-as-a-child-table.md`

The problem. Issue 3.2 had just answered a structurally similar question for transfers and answered it the *other* way — ADR-0008 chose two linked sibling rows in `transactions` over a `transfers` parent table.

What was decided.

- Split lines live in their own table, `transaction_splits`, keyed by `transaction_id`.
- The parent is an ordinary transaction and the lines are weightless.
- Lines are signed like their parent.
- The exact-sum rule is enforced by refusal, never by adjustment.
- "Split evenly" is `Money.split`, and largest-remainder is the correct rule.
- `profile_id` is denormalised onto every line.

What was turned down.

- N child rows in `transactions`, linked by a `split_id`.
- Lines replacing the parent.
- HALF_EVEN on each line.
- Storing a percentage per line.

What followed.

- No balance code was written or changed for this feature.
- Deleting a parent takes its lines in the same database transaction (DB-004), so no line can outlive the amount it attributes.
- Budgets (4.4) and Need/Want/Invest (4.3) inherit the right shape: they will sum split *lines* where a transaction is split and the parent where it is not, rather than being stuck with one category per purchase.

ADR-0010 — A future-dated transaction is excluded from actuals by its date, not by its posting stamp

Status	accepted, 2026-08-03
Binds	§5.3 (FR-TXN-010, FR-TXN-001), §5.2 (FR-HOME-001), §20.2 (`transactions` DDL), §20.3 (DB-001, DB-003), §21.4 (TIM-001, TIM-002), P-04, P-08
Record	`docs/adr/0010-future-dated-posting.md`

The problem. FR-TXN-010 is a MUST, and it is one sentence.

What was decided.

- `transactions.posted_at_utc_millis` (nullable) is added at schema v8, and `ScheduledTransactionWorker` stamps it daily.
- The column does not gate a single balance, and nothing may make it.
- Both account-balance queries gain the same bound.
- "Is this scheduled?" is `bookedOn > today` everywhere — money and UI alike.
- What the stamp is actually for:.
- The 7 → 8 migration backfills every existing row.

What was turned down.

- Derive posted-ness from the date and add no column or worker at all.
- Gate balances on `posted_at IS NOT NULL`.
- A separate `scheduled_transactions` table.
- Widen `observeRecent` to include future rows and filter in the UI.

What followed.

- Posting is correct in every zone and on every DST boundary, and is correct on a device that has been switched off for a week, because it is a property of the date rather than of a job having run.
- The accounts screen and net worth can no longer disagree about the same money.
- `postDueTransactions()` returning `0` is a success, so the overwhelmingly common case — a user who has scheduled nothing — never makes WorkManager back off.

ADR-0014 — The classification KB is mirrored in Kotlin, and its merchant rules ship without a consumer

Status	accepted, with a named trigger to revisit, 2026-08-08
Binds	—
Record	`docs/adr/0014-classification-kb-seed-mirror-and-unconsumed-merchant-rules.md`

The problem. Issue 4.1 is "categories editor + merchant-rule KB".

What was decided.

- `CategorySeed` is a typed Kotlin mirror of `category_defaults`,.
- The merchant rules ship as data with no runtime consumer,.
- Two vocabularies for one nature, translated in one place.

What was turned down.

- Build the loader now — Rejected on scope, for the reason ADR-0005 gave and this issue does not change: it is a new module, an asset pipeline and a parser, designed.
- Ship the merchant matcher in 4.1 too.
- Leave the merchant rows without ids until 4.2 needs them — Rejected: see decision 2.
- Seed the categories from Kotlin without referencing `ai/` at all — Rejected — it is the version of this decision that rots.

ADR-0016 — Nature fires on account *type*, only the override is stored, and true spend ships understated

Status	accepted, with named triggers to revisit, 2026-08-10
Binds	—
Record	`docs/adr/0016-nature-classification-by-account-type.md`

The problem. §8.3.1 is a six-step decision order, and **three of its steps name data this codebase does not have**.

What was decided.

- Steps 1–3 fire on the account's *type*.
- Step 1 checks both sides of the movement; steps 2–3 check both sides of a transfer.
- What this gives up, written down:.
- Only the user's override is stored; everything else is derived on read.
- `trueSpend` ships understated, and says so on screen.
- Step 6 is a flag, not a prompt.

What was turned down.

- Wait for the tables §8.3.1 names — Rejected: it would leave nature unimplemented across all of Epic 4, and every engine in Epics 5 and 6 is specified in terms of it.
- Store the resolved nature — Rejected — see decision 2.
- Include LIABILITY in true spend — Rejected: it counts debt principal as consumption, which is the exact confusion §8.3 introduces the LIABILITY nature to remove.
- Infer an asset purchase from the category rather than the account — Rejected: "Shopping" is where a gold purchase and a pair of shoes both land, and treating the category as an asset signal would exclude ordi.

ADR-0020 — `budget_review`'s claim is keyed by (profile, month), not (profile, month, category)

Status	accepted, 2026-08-15
Binds	—
Record	`docs/adr/0020-budget-review-keyed-by-month-not-category.md`

The problem. `RULE-BUD-REVIEW`'s `review_once_per_month` param, like `RULE-BUD-ALERT`'s `notify_once_per_band_per_month` before it, is documentation rather than enforcement — the rule row records the intent, and a schema constraint has to be the mechanism, exactly as ADR-0019 and `RulebookDriftTest`'s citation of `budget_alert`'s unique index already establish for the alert path.

What was decided.

- `budget_review` is keyed `UNIQUE(profile_id, month_start_iso_date)` — no `band`, no `category_id`, no `budget_id`.

What was turned down.

- Key by `(profile_id, month_start_iso_date, category_id)`, one row per material finding — Rejected: it answers a question this issue was not asked — "which findings has the user seen?" — and it would let the card empty out one row.
- No claim table at all — recompute the review every time and never hide it — Rejected outright. `RULE-BUD-REVIEW.review_once_per_month` is an explicit rule param, and the rulebook's whole discipline (§6, 'RulebookDrif.
- Reuse `budget_alert` and give it a nullable `band` — Rejected: it would make one table answer two different questions — "was this band's notification sent?" and "was this month's review shown?".

ADR-0022 — The privacy blur masks text rather than blurring pixels, and sets FLAG_SECURE while it is on

Status	accepted, 2026-08-16
Binds	—
Record	`docs/adr/0022-privacy-blur-masks-text-and-sets-flag-secure.md`

The problem. FR-PRIV-* asks for "a one-tap privacy blur that hides all amounts on screen (and in the widget/screenshots) for shoulder-surfing and screen-sharing safety".

What was decided.

- The blur replaces the text.
- 3 sets `FLAG_SECURE` while the blur is on.
- Editable amount fields are exempt.

What was turned down.

- `Modifier.blur` over the content.
- Masking inside `MoneyFormatter`.
- Replacing amounts in the ViewModel.

- Pulling all of 11.2 forward.

What followed.

- Positive: works on every supported API level, defeats screenshots and screen-sharing, and leaves the app fully navigable while blurred — labels, dates, categories and the rule citations stay readable, so the user can still see *what* the screen is telling them, only not how much.
- Positive: testable as an assertion rather than an eyeball.
- Negative / cost: it is not automatic.

ADR-0023 — The export archive is the schema, replaces on import, and carries no receipts or audit log

Status	accepted, 2026-08-16
Binds	§5.10, §34; P-01, P-07, DB-003; issue 5.4, and Epic 8 which this deliberately is not
Record	`docs/adr/0023-archive-replaces-and-carries-no-blobs.md`

The problem. §5.10 asks for "a full local JSON export/import archive" — the user-owned copy of their own data.

What was decided.

- The Room entities *are* the archive format.
- Receipts do not travel.
- `audit_log` does not travel.
- One schema version is accepted.

What was turned down.

- DTOs with hand-written mappers.
- Merge on import (upsert by id).
- Import only into an empty profile.
- Base64 the receipt images inline.

What followed.

- Positive (format): a column added to a table is in the archive the moment it is in the table.
- Negative / cost (format): a Kotlin property rename now changes the on-disk format, because kotlinx serialises by property name.
- Negative / cost (format): `:core:database` gains kotlinx.serialization in `main`, where it was previously test-only.

ADR-0024 — The home-screen widget renders from Glance state, not the database — and is not screenshot-tested

Status	accepted, 2026-08-17
Binds	§5.2, §35; P-01, P-03, P-04, SEC-002, ARC-001; issues 5.5 and 5.3
Record	`docs/adr/0024-the-widget-renders-from-glance-state-not-the-database.md`

The problem. §35 asks for a home-screen widget showing Safe-to-Spend and net worth, honouring the privacy blur and updating from cached engine output.

What was decided.

- Glance's own preference state *is* the cache.
- `:app` is the only writer, through two deliberately separate calls.
- A `null` figure removes its key rather than writing zero.
- `maskOf` moved from `:core:designsystem` down to `MoneyFormatter.mask` in `:core:model`.

- The widget is not screenshot-tested, and the tracker says so rather than shipping a baseline that checks nothing.

What was turned down.

- Read repositories in `provideGlance` via a Hilt entry point.
- A new `widget_snapshot` Room table.
- New fields on `CfoSettingsProto`.
- Read the blur live in `provideGlance`.

What followed.

- Positive (state as cache): the value the widget reads is the value it was redrawn for.
- Positive (state as cache): the render path cannot throw.
- Negative / cost (state as cache): two amounts now sit in an unencrypted app-private file, outside SQLCipher.

ADR-0026 — The amortisation schedule is derived, never stored

Status	accepted, 2026-08-20
Binds	—
Record	`docs/adr/0026-amortisation-schedule-is-derived-not-stored.md`

The problem. Issue 6.2 gives a loan its terms — principal, rate, tenure, first EMI date — and the schedule those terms imply: for each of up to 600 instalments, how much is interest and how much repays the loan.

What was decided.

- Store the terms. Derive the schedule on every read. No `loan_amortization_rows` table.
- The engine is exact, not approximate.
- The common read does not build a schedule at all.

What was turned down.

- Store `loan_amortization_rows`, as ADR-0016 named — Rejected above: it is a cache of a pure function, and a cache with no invalidation is a bug with a schema.
- Store only the current instalment's split on the `loan` row.
- Store the derived EMI but not the rows — Rejected as unnecessary: the EMI is one `BigDecimal` expression, and the user-entered override — the one figure the app genuinely cannot com.

What followed.

- A loan's terms are always self-consistent.
- ADR-0016's gap is half closed, and the remaining half is named.
- Issue 10.3's prepay-vs-invest simulator gets what it needs for free.

ADR-0028 — XIRR is solved by bisection over the daily growth factor, not Newton-Raphson over a rate

Status	accepted, 2026-08-24
Binds	—
Record	`docs/adr/0028-xirr-by-bisection-over-the-daily-growth-factor.md`

The problem. §11.2 names XIRR — money-weighted return per holding — and it is the figure that makes an investment screen worth building.

What was decided.

- Reparametrise, then bisect a fixed bracket a fixed number of times, with no early exit.

- Three refusals, not a fallback number:.

What was turned down.

- Newton-Raphson over `r` with `Double`.
- Newton-Raphson over `r` with `BigDecimal`.
- Bisection over `r` directly, with `BigDecimal` exp/ln for the fractional exponent.
- A shipped numerics library.

What it costs.

- It is slower than Newton by a constant factor.
- Multiple sign changes admit multiple roots.

ADR-0029 — The portfolio is the investable accounts, and unpriced holdings are excluded from it

Status	accepted, 2026-08-28
Binds	—
Record	`docs/adr/0029-the-portfolio-is-the-investable-accounts.md`

The problem. FR-INV-002 (MUST) says "Portfolio view MUST show allocation by asset class, diversification analysis, and concentration warnings".

What was decided.

- The portfolio is the investment, gold and crypto accounts.
- Allocation is computed over the priced positions only.

What was turned down.

- Every asset account, matching net worth.
- Holdings only, the issue's literal wording.
- Refuse to report until every holding is priced.
- Suppress the flags below a coverage floor.

What followed.

- `RULE-CONC-15-70`'s 70% single-class line stays meaningful.
- `RULE-GOLD-CAP` fires for the users most likely to trip it.
- A user's house is not in this screen.

ADR-0030 — The market-data client ships unconfigured, and revoking consent keeps the cached price

Status	accepted · 2026-08-29 · issue 6.5 (gold/crypto valuation via market api), —
Binds	—
Record	`docs/adr/0030-the-market-data-client-ships-unconfigured.md`

The problem. Issue 6.3 gave a holding a per-unit price and the day it was observed.

What was decided.

- The client ships unconfigured, and that is a supported state.
- A configured host cannot be expressed without certificate pins.
- Revoking MARKET_DATA stops fetching and keeps the cached price.
- So there is no `MarketDataConsentWatcher`,.
- Staleness and refresh cadence are two rulebook numbers, not one.
- The price columns have a second, narrower writer.

What followed.

- AC-1 of issue 6.5 — "prices come through our backend proxy" — cannot be demonstrated.
- TLS and certificate pinning are untested, and are the only part of `:core:network` that is.
- A real airplane-mode test is structurally unmeetable here.

ADR-0034 — The emergency fund mints two rows, defers three of §10.1's terms, and counts only cash

Status	accepted, 2026-09-02
Binds	§10.1 (AI-EMF), §15, §29 (AI-ARC-006), §8.3, MNY-001, MNY-002, TIM-001, P-02, P-03, P-07
Record	`docs/adr/0034-emergency-fund-mints-two-rows-and-defers-three-terms.md`

The problem. SRS §10.1 specifies `AI-EMF` precisely, and it is worth quoting because four separate decisions below all come from reading it against what this codebase actually holds.

What was decided.

- Two rulebook rows, and only the params the engine applies.
- Two rather than one.
- Neither extends `RULE-RUNWAY-M`.
- Three of §10.1's five multiplier terms are deferred, and their absence is asserted.
- A consequence worth stating: with the shipped params the clamp never fires.
- Liquid funds are `BANK` and `CASH` with a positive balance.

What was turned down.

- Mint one row instead of two — Rejected on ADR-0019's precedent and on the merits: the coach bands would then share a version with the sizing params, so retuning the urgen.
- Mint the three deferred params now, applied as zero — Rejected: a param the engine reads and always finds zero is indistinguishable from one that works, and the drift test could not tell the dif.
- Add the three profile fields in this issue — Rejected as scope: three settings fields, a screen to capture them, and a privacy review for each, bolted onto an engine issue. §10.1's own .
- Add the liquidity tier column in this issue — Rejected for the same reason — a schema 20 → 21 migration, an account-editor field and a screenshot re-record, inside an issue whose accepta.

ADR-0036 — Goal progress is evidenced *and* declared, and the two never merge

Status	accepted, 2026-09-06
Binds	—
Record	`docs/adr/0036-progress-is-evidenced-and-declared-and-the-two-never-merge.md`

The problem. Issue 7.1 shipped `goal.saved_minor` as a number the user types, and said so in three places — the entity's own KDoc, ADR-0033 (*"Until 7.4, a user who saves towards a goal must tell the app"*) and the 0.7.1 changelog.

What was decided.

- Two join tables, and neither holds an amount.
- The two link kinds sum differently, on purpose.
- Progress becomes two numbers that never merge.
- The pair is floored together.
- Reversal is a soft delete, because the criterion asks for two things at once.
- `RULE-PAY-FIRST` is consumed, and mirrors nothing.

What followed.

- A goal's progress is now the sum of movements in the user's own ledger, and the part that is not is labelled as such and clearable in one tap.
- The editor now loads the declared half.

ADR-0038 — AI-FOO is the base; the goal waterfall splits what it leaves

Status	accepted, 2026-09-18
Binds	§36 (AI-FOO), §15.1 (FR-GOAL-003, FR-GOAL-005), §10.1, AI-ARC-003, P-02, P-03; supersedes the divergence recorded in ADR-0037, builds on ADR-0035
Record	`docs/adr/0038-the-ranking-is-the-base-and-the-goal-split-consumes-it.md`

The problem. §36 says AI-FOO "supersedes the simple waterfall in §15".

What was decided.

- The surplus derivation moves out of both into `SurplusRepository`.
- AI-FOO is the base.
- The goal waterfall consumes the ranking.
- The plan carries two echoes so the smaller figure explains itself.

What was turned down.

- Make AI-FOO match 7.3 instead.
- Let the goals screen call the ranking directly from its ViewModel — Rejected: the split would.
- Keep both and label them.
- Have AI-FOO read the waterfall's remainder.

What followed.

- Positive: the two screens cannot disagree; the goals' share is now a single figure computed once.
- Positive: §36's supersession is real rather than asserted.
- Negative / cost: `GoalWaterfall.monthlySurplus` changed meaning — it is what the goals may have, not the month's surplus.

ADR-0043 — The forecast schedules confirmed rules and FIXED streams, forecasts one consolidated balance, and defers on-device accuracy tracking

Status	accepted, 2026-09-19
Binds	§9.1 (AI-FCT-001..004), §9.2, §9.3, §17/§18, §21.5, P-01, P-02, P-03, P-08; issue 9.2. Builds on ADR-0007 (derived balances), ADR-0010 (future-dated rows), ADR-0017 (typed mirrors), ADR-0034 (liquid accounts), ADR-0042 (streams)
Record	`docs/adr/0043-forecast-schedules-confirmed-and-fixed-and-defers-accuracy-tracking.md`

The problem. §9.2 gives the method: opening balance plus scheduled items, less predicted variable spend, with bands from seeded resampling.

What was decided.

- Scheduled = confirmed recurring rules + FIXED streams + future-dated rows.
- Everyday spend is liquid outflow with the scheduled rows removed.
- The method is §9.2's own words, with the gaps closed simply.
- The numbers are rulebook rows.
- Accuracy is gated offline on a frozen synthetic set.

What was turned down.

- Schedule all FIXED streams, obligations included — Rejected: it counts a confirmed rent twice.

- Deduct card purchases from the liquid balance — Rejected: the money leaves the bank on the bill.
- Measure crunch on P10 — Rejected: one day in ten would read as a crisis.
- Build AI-FCT-004's snapshot table now.

What followed.

- The dashboard gains "The next 90 days": the lowest point with its range, crunch days, the three components, the next scheduled items with their source, the history note and the rules.
- `StreamMetrics` gains `modalDayOfMonth` (additive; AI-CLS.stream stays 1.0).

ADR-0045 — The health score scores six signals the app can measure, re-weights what it cannot, and defers the weekly cadence

Status	accepted, 2026-09-19
Binds	§14 (AI-FHS), FR-AI-001, P-02, P-03, P-08, MNY-002, AI-ARC-003/006, CLAUDE.md §6; issue 9.4. Builds on ADR-0007 (derived figures from their owners), ADR-0017 (typed mirrors), ADR-0034 (no insurance or dependant fields), ADR-0042 (streams)
Record	`docs/adr/0045-health-score-six-signals-reweighted-live-no-snapshots.md`

The problem. The issue asks for a deterministic, weighted composite with the weights as rule rows, every component shown, provenance, and a golden file.

What was decided.

- v1.0 scores the six signals the app measures, each on a straight line between stated anchors.
- Missing is never zero.
- The score is exact, and its parts add up.
- The engine is L3 and pure; the repository composes owners.

What was turned down.

- Score a missing pillar as 0.
- Invent curves for the unstated signals.
- Read the database directly.

What followed.

- A young profile sees fewer pillars, and says so: the demo reads "Based on 2 of 5 pillars" until a repeat is confirmed, then three.
- The dashboard gains a "Financial health" card: the total and its band; each pillar's points, contribution and share; "—" with the weight it gave away; each signal against its full-marks anchor; the lever; "Based on N of 5 pillars"; and the rules.
- A profile with data in one pillar gets a score from that pillar alone, and the card says it rests on 1 of 5 pillars.

ADR-0046 — The orchestrator ranks what the engines already published, persists the feed, and leaves notifying to 9.6

Status	accepted, 2026-09-20
Binds	§7.1 (AI-ARC-001/003/005/006), §7.2, §20.2 (`insights`), DB-003, DB-005, FR-AI-001/002, FR-HOME-001, P-02, P-03, P-08, CLAUDE.md §6; issue 9.5. Builds on ADR-0006 (the demo leaves nothing behind), ADR-0017 (typed mirrors), ADR-0043/0044/0045 (the engines it reads)
Record	`docs/adr/0046-orchestrator-ranks-published-results-and-persists-the-feed.md`

The problem. §7.2 defines the Insight Orchestrator.

What was decided.

- The orchestrator ranks published results; it computes nothing.

- Six insight types in v1.0.
- The order is a rule, and the tie-breaks are total.
- The feed is persisted, with typed columns rather than a JSON blob.
- Triggers: the daily job and opening the dashboard.

What was turned down.

- Compute the feed on every read, without a table.
- One row per recomputation, deduplicated at read time.
- Let the orchestrator read the DAOs directly.
- `evidence_json` as §20.2 sketches it.

What followed.

- The dashboard gains "What needs attention": up to three cards, each with its finding, one recommended action, the engine and rules behind it, and two ways to put it away.
- The demo wipe and the archive wipe both clear `insight`, and `countRowsFor` counts it, so the residue ADR-0006 forbids cannot accumulate.
- rules-kb moves to 1.18.0, and eight mirrors restate it.

ADR-0048 — One guardrail engine: an allowlist of renderings, rounding that stops at the rupee, and the minimal subset retired

Status	accepted, 2026-09-23
Binds	AI-ARC-004, AI-ARC-003, P-02, P-03, P-08, §19.1, §19.3 (CHT-001), MNY-001/002, TIM-002, CLAUDE.md §6; `ai/chat/guardrail.md` (GRD-001 … GRD-006); issue 9.7. Builds on ADR-0017 (typed mirrors), ADR-0019 (which wrote down the subset's limits)
Record	`docs/adr/0048-one-guardrail-engine-rendering-allowlist-rounding-stops-at-the-rupee.md`

The problem. AI-ARC-004: LLM output must pass a rules-engine guardrail before display; numeric claims are verified against engine results, and unverifiable numbers are stripped and the response regenerated or refused.

What was decided.

- One engine, `:domain:engines:guardrail` (AI-GRD 1.0, L3), and the subset is retired.
- Verification is an allowlist of renderings, not a tolerance.
- Rounding stops at the rupee.
- Grouping is not normalised; whitespace and case are.
- The ladder is data, and the engine hands back spans rather than a sentence.
- Names are declared, and that hole is stated.

What followed.

- No figure reaches a user that an engine did not produce, and the rule holds for the notifications that exist today as well as for the chat that does not.
- The gate's settings are a data edit.
- A false refusal is now the likelier failure than a false pass, which is the correct direction: the property tests pin both — anything the app itself formatted always survives, and anything no engine produced never does.

ADR-0049 — The Purchase Advisor: seven gates, a worst-of verdict urgency may only soften so far, and a trace kept in typed columns

Status	accepted, 2026-09-25
Binds	§13 (§13.1 pipeline, §13.2 output contract), FR-AI-003, §19.2 (`purchase_check`), AI-ARC-001/003/006, P-02, P-03, P-07, P-08, MNY-001/002, TIM-002, DB-003, DB-005, CLAUDE.md §6; issue 10.1. Builds on ADR-0017 (typed mirrors), ADR-0046 (an L5 engine imports no other engine), ADR-0006 (the demo leaves nothing behind)

Record	`docs/adr/0049-purchase-advisor-gates-verdict-and-the-kept-trace.md`

The problem. §13.1 gives the pipeline: affordability, cash-flow impact, obligation ratio, goal impact, budget fit, opportunity cost, timing — and then "VERDICT = worst severity across gates, softened by urgency flag".

What was decided.

- One pure engine, `:domain:engines:purchase` (AI-PA 1.0, L5).
- Every gate states its figures as typed values with stable keys, never as sentences.
- Affordability warns where §13 says it warns, and fails only where nothing else can.
- The cash-flow gate reports the crunch days it *adds*, and says how many were already there.
- A goal delay is the price divided by everything going to goals each month.
- Urgency softens one step, never onto a hard fail, and never to a lie.

What followed.

- The flagship question has a deterministic answer with its working attached, and the working survives: a card read in December reproduces what AI-PA 1.0 decided in September.
- Every threshold that decides a verdict is a rulebook row — the lender's lines, the assumed return, the softening policy — and a drift test fails the build when a mirror lags.
- The engine is proven at each gate, against an independent Python oracle on five households, and by properties that caught two real bugs before release.

ADR-0053 — The assistant plans with data, reads through tools, and the model only ever gets words to write

Status	accepted, 2026-09-27
Binds	§19 (§19.1 pipeline, §19.2 tool registry, §19.3 behaviour rules CHT-001..005, §19.4 context pack), §7.1 (AI-ARC-004, AI-ARC-007), §6 (data, not code), P-01, P-03, P-04, P-07, P-08, MNY-001, §21.6, ADR-0017 (mirrors and drift tests), ADR-0048 (AI-GRD, which this layer exists behind); issue 10.5.
Record	`docs/adr/0053-chat-plans-tools-and-the-model-only-gets-words.md`

The problem. §19 asks for a chat assistant that is "a narrow financial copilot over the user's own, on-device financial data" — not a general chatbot.

What was decided.

- The pipeline is plan → execute → verbalise → verify, and the engine owns the first and last.
- Routing is keyword matching over the registry, and the routing table is data.
- The out-of-scope check runs *before* the intent match.
- A turn may call at most two tools.
- The model the app ships with is a deterministic template verbaliser, and that is the offline path rather than a stand-in.
- The sentences are string resources, so the assistant can be translated.

What followed.

- The assistant can answer seven kinds of question about the household's own data, and every figure it states was computed by an engine that publishes the same figure elsewhere in the app.
- A fabricated figure cannot reach the user: six deliberate mutations of the engine were each watched go red, including removing the guardrail call and collapsing the figure kinds.
- What the assistant can ever see is one type in one file — the executor — which is the surface a reviewer reads to answer "what does chat have access to?".

ADR-0057 — The database key lives in the strongest place the phone has, and rotates without losing the database

Status	accepted, 2026-09-28
Binds	§23 (security), SEC-003 (Tink/Keystore only, no hand-rolled crypto), DB-003, P-01, P-04; issue 11.1. Supersedes the StrongBox note written into `KeystoreAeadFactory` at issue 1.6.
Record	`docs/adr/0057-the-database-key-lives-in-the-strongest-place-the-phone-has-and-rotates-without-losing-the-database.md`

The problem. Issue 1.6 built the key path: a random SQLCipher passphrase, wrapped by a Tink AEAD whose master key lives in the Android Keystore, with only ciphertext on disk.

What was decided.

- The master key is generated by the platform, StrongBox first, before Tink looks for it.
- This is not hand-rolled crypto, and the 1.6 note that said it was has been corrected.
- An existing alias is never replaced.
- A refusal is not a failure.
- Rotation is a two-slot protocol, and the file is re-keyed in the middle of it.
- The promotion is a rename.

What followed.

- On a fresh install with StrongBox, the key that wraps everything else lives in a separate chip; everywhere else it is TEE-backed exactly as before; and no existing installation's key is touched.
- The database key can be rotated, and the file is re-keyed with it — proved on a device, with every row still readable afterwards and the old key no longer opening the file.
- No sequence of crashes during a rotation leaves the database unopenable.

ADR-0060 — An erase destroys keys first, and says what it cannot reach

Status	accepted, 2026-10-01
Binds	§23, §34, SEC-003, P-01, P-07
Record	`docs/adr/0060-an-erase-destroys-keys-first-and-says-what-it-cannot-reach.md`

The problem. §34 and SEC-003 ask for an erase-all that is *cryptographically* unrecoverable, requires explicit confirmation and authentication, records an audit event with no PII, and is proven by an instrumented test that no readable data remains.

What was decided.

- Destroy the keys, then delete the files — in that order.
- The deletion is verified, not assumed.
- Each module declares its own secrets; the eraser knows no names.
- Two gates: a typed word in the user's own language, then authentication.
- What the erase cannot reach, said on the screen.
- Two things only the device run found.

What followed.

- The data on the device is unrecoverable by anyone once `destroyKeys` reports `Ok` — verified, not assumed.
- Adding a key chain without declaring it in its module's `SecretInventory` leaves it undestroyed.
- The erase is not reachable by accident: a typed word, a PIN, and a page the user navigated to.

ADR-0063 — SEC-003 is enforced by a lint rule, with two argued exemptions

Status	accepted, 2026-10-02
Binds	SEC-003
Record	`docs/adr/0063-sec-003-is-enforced-by-a-lint-rule-with-two-argued-exemptions.md`

The problem. SEC-003 says all cryptography goes through Tink or the Android Keystore — no hand-rolled crypto, review-blocking.

What was decided.

- A lint rule, because this failure mode is silent.
- Resolved types, not identifier names.
- `KeyGenerator` is allowed only alongside a `KeyGenParameterSpec`.
- File-scoped rather than call-scoped.
- Test sources are exempt.
- The audit's inventory is checked too.

What followed.

- SEC-003 is enforced at compile time in every module, proven by a seeded `Cipher.getInstance` failing a real `lintDebug` — not only the test harness.
- The rule cannot be bypassed by an import alias.
- Adding a crypto-touching file without listing it in the audit fails the build.

ADR-0069 — A household is a row above the profile, and aggregation composes scoped reads

Status	accepted, 2026-10-03
Binds	§27, §33
Record	`docs/adr/0069-household-is-a-row-above-the-profile-and-aggregation-composes-scoped-reads.md`

The problem. Household mode is an Epic 13 *design-for* item: v1 ships the foundation, not the feature.

What was decided.

- A household is one row above the profile, not a column inside it.
- Strict scoping is a test over the DAO, not a convention in a doc.
- Aggregation composes scoped reads; cross-profile SQL stays banned.
- The flag holds back the views, not the foundation.
- The fourteen id-keyed queries are the residual risk, and this is the precondition.
- Before `HouseholdMode.isEnabled` is allowed to become `true`, every id-keyed query must also constrain `profile_id`.

What was turned down.

- A `household_member` join table.
- One cross-profile SQL view per aggregate.
- Scoping enforced by a custom lint rule rather than a test.
- Adding `profile_id` to all 14 id-keyed queries now.

What followed.

- Adding a query that crosses profiles fails `:core:database`'s tests, with the three categories and the marker convention named in the failure message.
- Adding a database version remains the five-step checklist in `Migrations.kt`; schema 30 follows it.
- The 14 id-keyed queries are recorded here as a precondition rather than a TODO in code, so turning the flag on starts by reading this ADR.

ADR-0075 — "Android-free" is not "KMP-portable", and `Money` is the blocker

Status	accepted, 2026-10-10
Binds	§27, §33, ARC-002
Record	`docs/adr/0075-android-free-is-not-kmp-portable-and-money-is-the-blocker.md`

The problem. §33's forward-compatibility table closes with.

What was decided.

- The two claims are not the same, and the difference is 40 files.
- The spike ports `Money`, not an engine — because every engine sits behind it.
- `linuxX64`, not an iOS target, and that is stated rather than glossed.
- What losing `BigDecimal` costs: a bounded range.
- The audit is pinned, so it cannot rot.

What was turned down.

- Nominate one of the six clean engines and call it shared — Rejected, §2: it would satisfy the.
- Target an iOS simulator.
- Keep `BigDecimal` via a multiplatform big-decimal library.

What followed.

- `:spike:kmp` exists and nothing depends on it.
- The Kotlin Multiplatform plugin is applied without a version and without a catalog alias: it is already on the build classpath via `build-logic`'s `kotlin-gradle-plugin`, and asking for it by alias fails with "already on the classpath with an unknown version".
- Building `linuxX64` downloads the Kotlin/Native toolchain (LLVM, libffi, lldb) into `~/.konan` — about a gigabyte, once.

ADR-0078 — `import_batches` exists, and the wipe is checked rather than trusted

Status	accepted, 2026-10-10
Binds	§20.1, §33, §16
Record	`docs/adr/0078-import-batches-exists-and-the-wipe-is-checked-not-trusted.md`

The problem. §20.1 lists `import_batches` among the system tables.

What was decided.

- One table, and "statement-grade" is the demanding word.
- It holds no account number, no bank name and no file name.
- The refusal is the foreign key.
- Both halves of that scoping matter.
- Building it uncovered a larger defect: sixteen missing deletes.
- The fix is the check; the sixteen deletes are its first output.

What was turned down.

- A `DEFAULT ''` on the new column — Rejected, §3: a dangling pointer in every historical row.
- Real foreign keys for this one table — Rejected: issue 1.6 chose application-level integrity.
- Derive `accepted_count` from `transactions` — Rejected, §1: evidence that moves is not evidence.
- Put the tax/provenance on `attachments` instead — Rejected for the reason ADR-0077 gives: a.

What followed.

- Schema 33; `MIGRATION_32_33` is one `CREATE TABLE` and one `ALTER TABLE ... ADD COLUMN`, so DB-003 holds.
- The archive gains `importBatches` (keys 41 → 42) and restores them before the transactions that name them.
- The restore drill's fixture seeds a truncated batch, so the round trip proves `complete = false` survives rather than merely that the column compiles.

ADR-0081 — Static analysis gets a server, and the first run was vacuous

Status	accepted, 2026-10-15
Binds	§21.6, §21.5
Record	`docs/adr/0081-static-analysis-gets-a-server-and-the-first-run-was-vacuous.md`

The problem. The repository already fails a build on four analysers: ktlint, detekt, Android Lint, and the five custom detectors in `:lint`.

What was decided.

- SonarQube reports; it does not gate.
- The first run covered a third of the repository and nothing said so.
- Nine rules are off, per file, with the argument next to the path.
- 3b · The 49 `fun interface` conversions, and the two that came back.
- One issue is accepted on the server, deliberately visible.
- It gates the commit, not the build.

What followed.

- The dashboard covers the whole repository, so a clean result means something.
- `sonarAnalysis` is a pre-commit gate by convention (§5), not a build gate.
- `./gradlew build` on a machine with no server is unchanged.

ADR-0082 — Every screen gets a render, and the app module needed its own JVM

Status	accepted, 2026-10-17
Binds	§21.5
Record	`docs/adr/0082-every-screen-gets-a-render-and-the-app-module-needed-its-own-jvm.md`

The problem. ADR-0066 took screenshot coverage to 26 baselines across five modules — the design-system gallery, the dashboard, and the three flows `CLAUDE.md` §4 calls critical.

What was decided.

- Every screen gets a render, in the three configurations.
- Fixtures come from the real engines wherever a figure appears.
- The renders are also the screens document.
- Paparazzi stays per-module, and now applies to ten more.
- `:app` forks a JVM per test class.

What followed.

- 115 committed baselines across 15 modules; a visual diff in any of them blocks a merge.
- Every screen in the app is visible to a reviewer without a device, from the repository.
- `docs/screens/SCREENS.md` is current by construction, and says so with the command that counts the renders.

10.2 The complete decision index All 80 records, oldest first. The ones digested above are the subset §10.1 names.

ADR	Date	Decision	Status
ADR-0001	2026-07-25	A `:lint` module outside the §21.2 graph, and name-based money detection	accepted
ADR-0002	2026-07-25	Onboarding's four steps are not FR-ONB-001's four steps	accepted
ADR-0003	2026-07-26	The app lock gates the session, not the Keystore key	accepted
ADR-0004	2026-07-27	Quick setup defines `budget` and `recurring_rule` at schema v3	accepted
ADR-0005	2026-07-27	Quick-setup thresholds are typed Kotlin defaults, not `ai/` rows (for now)	accepted, with a named trigger to revisit
ADR-0006	2026-07-28	Demo mode is a second profile in the same database, wiped by hard delete	accepted
ADR-0007	2026-07-28	Account balances are derived on read, and `wallet` is not an account type	accepted
ADR-0008	2026-08-02	A transfer is two linked legs, not a parent row; `type` is derived, never supplied	accepted
ADR-0009	2026-08-02	Split lines are a child table that moves no money; the remainder rule is largest-remainder, not HALF_EVEN	accepted
ADR-0010	2026-08-03	A future-dated transaction is excluded from actuals by its date, not by its posting stamp	accepted
ADR-0011	2026-08-06	A scanned receipt may be back-dated; the daily net-worth series does not follow it	superseded by adr-0012 (2026-08-06)
ADR-0012	2026-08-06	Any transaction may be back-dated, because the net-worth series can repair itself	accepted
ADR-0013	2026-08-07	READ_SMS is a Play-policy risk we take with our eyes open	accepted
ADR-0014	2026-08-08	The classification KB is mirrored in Kotlin, and its merchant rules ship without a consumer	accepted, with a named trigger to revisit
ADR-0015	2026-08-10	Stage 1 ships two of its three tiers, and the knowledge base is still a Kotlin mirror	accepted, with named triggers to revisit
ADR-0016	2026-08-10	Nature fires on account *type*, only the override is stored, and true spend ships understated	accepted, with named triggers to revisit
ADR-0017	2026-08-11	Budget thresholds stay a typed mirror; the `ai/` loader is deferred again	accepted, with the trigger list narrowed
ADR-0018	2026-08-11	Spending is summed from split lines, which moves figures already shipped	accepted
ADR-0019	2026-08-13	FR-BUD-004's alert bands mint a new rule row rather than bump a shipped one	accepted
ADR-0020	2026-08-15	`budget_review`'s claim is keyed by (profile, month), not (profile, month, category)	accepted

ADR	Date	Decision	Status
ADR-0021	2026-08-16	Safe-to-Spend withholds a buffer §5.2 does not name, and stands in for goal contributions until issue 7.1	accepted
ADR-0022	2026-08-16	The privacy blur masks text rather than blurring pixels, and sets FLAG_SECURE while it is on	accepted
ADR-0023	2026-08-16	The export archive is the schema, replaces on import, and carries no receipts or audit log	accepted
ADR-0024	2026-08-17	The home-screen widget renders from Glance state, not the database — and is not screenshot-tested	accepted
ADR-0025	2026-08-17	Card alerts mint `RULE-CC-DUE`, and claim once per **statement cycle**, not per month	accepted
ADR-0026	2026-08-20	The amortisation schedule is derived, never stored	accepted
ADR-0027	2026-08-24	Asset class is a column on the holding, and the unit price is the one observation we store	accepted
ADR-0028	2026-08-24	XIRR is solved by bisection over the daily growth factor, not Newton-Raphson over a rate	accepted
ADR-0029	2026-08-28	The portfolio is the investable accounts, and unpriced holdings are excluded from it	accepted
ADR-0030	—	The market-data client ships unconfigured, and revoking consent keeps the cached price	accepted · 2026-08-29 · issue 6.5 (gold/crypto valuation via market api)
ADR-0033	—	Goals mint no rulebook row, assume no growth, and take the *greater* of two savings terms	accepted · 2026-08-30 · issue 7.1 (goals engine, ai-goal)
ADR-0034	2026-09-02	The emergency fund mints two rows, defers three of §10.1's terms, and counts only cash	accepted
ADR-0035	2026-09-03	The waterfall substitutes an observed P50, mirrors nothing, and stores only the order	accepted
ADR-0036	2026-09-06	Goal progress is evidenced *and* declared, and the two never merge	accepted
ADR-0037	2026-09-17	The order of operations reads the FOO file, mints nothing, and composes the goal waterfall rather than replacing it	accepted
ADR-0038	2026-09-18	AI-FOO is the base; the goal waterfall splits what it leaves	accepted
ADR-0039	2026-09-18	Argon2id comes from BouncyCastle; the backup consent gates the file-picker backup	accepted
ADR-0040	2026-09-18	Restore goes through the archive import, asks no consent, and refuses another profile's data	accepted
ADR-0041	2026-09-19	The restore drill is a release gate that reads the schema itself; DRL-001's in-app drill waits	accepted
ADR-0042	2026-09-19	Stage 2 classifies by category, scores exactly, and ships without a pin store	accepted

ADR	Date	Decision	Status
ADR-0043	2026-09-19	The forecast schedules confirmed rules and FIXED streams, forecasts one consolidated balance, and defers on-device accuracy tracking	accepted
ADR-0044	2026-09-19	Seasonality reads the user's own months first, divides the lookback's season out of the forecast, and owns the calendar mirror	accepted
ADR-0045	2026-09-19	The health score scores six signals the app can measure, re-weights what it cannot, and defers the weekly cadence	accepted
ADR-0046	2026-09-20	The orchestrator ranks what the engines already published, persists the feed, and leaves notifying to 9.6	accepted
ADR-0047	2026-09-20	One notification gate: §17.2's caps and quiet hours as rules, claim after the gate, the digest deferred	accepted
ADR-0048	2026-09-23	One guardrail engine: an allowlist of renderings, rounding that stops at the rupee, and the minimal subset retired	accepted
ADR-0049	2026-09-25	The Purchase Advisor: seven gates, a worst-of verdict urgency may only soften so far, and a trace kept in typed columns	accepted
ADR-0050	2026-09-26	The buy list: a ladder in basis points of income, minted score deltas no single answer can decide, and nothing ever removed by the app	accepted
ADR-0051	2026-09-26	The simulators reuse the loan engine's arithmetic, mint no rule, and treat a rounding residue as rounding	accepted
ADR-0052	2026-09-26	AI-VEH predicts with a median slope, prices with the household's own bills, and invents nothing	accepted
ADR-0053	2026-09-27	The assistant plans with data, reads through tools, and the model only ever gets words to write	accepted
ADR-0054	2026-09-27	The guardrail is measured against a frozen set, and one of its two thresholds is absolute	accepted
ADR-0055	2026-09-27	AI-MKT scores what it can measure, says what it cannot, and builds its own history	accepted
ADR-0056	2026-09-28	The app speaks Hindi, Kannada and Tamil, and the build proves it	accepted
ADR-0057	2026-09-28	The database key lives in the strongest place the phone has, and rotates without losing the database	accepted
ADR-0058	2026-09-28	Every screen is capture-guarded, because every screen is financial	accepted
ADR-0059	2026-10-01	The consents dashboard shows the record, not just the switch	accepted
ADR-0060	2026-10-01	An erase destroys keys first, and says what it cannot reach	accepted
ADR-0061	2026-10-01	DPDP alignment is a checked document, and the consent record is export-only	accepted

ADR	Date	Decision	Status
ADR-0062	2026-10-02	The release is minified, and the log strip is read out of the shipped DEX	accepted
ADR-0063	2026-10-02	SEC-003 is enforced by a lint rule, with two argued exemptions	accepted
ADR-0064	2026-10-02	The golden harness refuses to be vacuous, and never rewrites its own fixture	accepted
ADR-0065	2026-10-02	Evaluation datasets are versioned, and every run reports what it measured	accepted
ADR-0066	2026-10-02	Screenshot coverage reaches the critical screens, and recording is never automatic	accepted
ADR-0067	2026-10-02	Airplane mode is turned on by a test, and a check that could not fail was removed	accepted
ADR-0068	2026-10-03	The release train checks four files, and found two drifts already in the history	accepted
ADR-0069	2026-10-03	A household is a row above the profile, and aggregation composes scoped reads	accepted
ADR-0070	2026-10-03	Appliances are a sibling of AI-VEH, and the running cost is the new number	accepted
ADR-0071	2026-10-03	The cover gap is the larger of two readings, and the engine never says "surrender"	accepted
ADR-0072	2026-10-09	The tax KB had no slabs, and an estimate must say what it left out	accepted
ADR-0073	2026-10-09	The business book is a second profile, and two forward-compatibility promises were broken	accepted
ADR-0074	2026-10-09	AA is designed, and the reserved `source` value had to ship now	accepted
ADR-0075	2026-10-10	"Android-free" is not "KMP-portable", and `Money` is the blocker	accepted
ADR-0076	2026-10-10	The engine registry is reconciled, and now checked	accepted
ADR-0077	2026-10-10	The GST figure is stored, and `null` is not zero	accepted
ADR-0078	2026-10-10	`import_batches` exists, and the wipe is checked rather than trusted	accepted
ADR-0079	2026-10-10	The on-device model is real, and its weights are Apache-2.0	accepted
ADR-0080	2026-10-10	The verbaliser is judged by the app's own renderings, and measured by a frozen set	accepted
ADR-0081	2026-10-15	Static analysis gets a server, and the first run was vacuous	accepted
ADR-0082	2026-10-17	Every screen gets a render, and the app module needed its own JVM	accepted

11. Where this document is thin, and why

Three things a reader might expect in an architecture document are deliberately not here.

Expected	Why it is not here
Server capacity planning, scaling, load balancing	There is no server. §7's deployment section is one paragraph because a Play Store release is the whole deployment, and `06-api-specification.md` §8 records what the app does instead of calling one
A service decomposition diagram	The app is a single process with 60 Gradle modules. The module graph *is* the decomposition, and it is enforced by `enforceNoAndroidPlugins()` and the `:lint` detectors rather than by a diagram
Runtime observability dashboards	Nothing phones home (`P-01`). §8's observability is on-device and user-visible by necessity, which is a weaker position than a server's and is stated as such rather than dressed up

Part 06 — API Specification

← Master Blueprint · Related: 05 Architecture · 09 Risks & Open Questions · 10 Traceability

> ***Read this first.*** *This document is* ***not*** *marked NOT APPLICABLE, and it is also not a description of running software. The app has a specified backend (SRS §22) and a client written against it, and* ***the backend is not built****. One endpoint is consumed by shipped code; eleven are specified and unimplemented on both sides. That is* `GAP-001`*, and it is the largest single gap in this blueprint. The honest framing: this is a* ***client-side contract****. It exists so that the one network path in the app is written once, correctly, against a specification — and so that the day a proxy is stood up, nothing in the app has to be redesigned to use it.*

---

1. Why there is an API at all, and why it is this small

`P-01` says no financial data leaves the device. `P-04` says every core feature works offline. Between them they leave exactly one legitimate reason for a server: **facts about the world that the device cannot compute** — a gold price, a mutual-fund NAV, the repo rate, a fuel price, an FX rate, a tax table.

So the API design rule is not "what would be convenient to put on a server" but **"what does the device have no way of knowing?"** Everything else stays local. That is why there are no user-data endpoints, no account endpoints, no transaction endpoints, and no login.

The two consequences are worth stating plainly:

- **The server is never an authority on the user.** It holds no row about anyone. A breach of it exposes gold prices.
- **The client treats every endpoint as optional.** Five-second timeouts, exponential backoff, a circuit breaker, and silent degradation to cache with a staleness label. Not as resilience engineering — as the normal case.

2. Style, and the environments

Aspect	Decision
Style	REST over HTTPS, JSON bodies, a versioned path prefix `/v1`
Evolution	Additive only within a version. A field may be added; none may be removed or retyped
Transport	TLS 1.3, certificate pinning to our own host, no cleartext (manifest-enforced)
Base URL	**Blank, in every shipped build.** A configured host cannot even be expressed without certificate pins, which is enforced in `NetworkConfig` (ADR-0030)
Environments	None exist. There is no dev, stage or production host to name — recording a placeholder URL here would read as a control while protecting nothing
Content type	`application/json; charset=utf-8`
Dates	ISO-8601. Date-only fields are `yyyy-MM-dd`; instants are RFC-3339 UTC
Money in payloads	Minor units as a JSON integer, with an explicit `currency`. No decimal numbers for money, ever (BR-001)

3. Authentication and authorisation

Aspect	Decision
Registration	An anonymous device key pair. No email, no password, no account (SRS §22.1)
Credential	A signed device token obtained from `API-012`
Authorisation	There is nothing to authorise. Every endpoint serves public facts, or stores an opaque blob the server cannot read
Rate limiting	Per device token, server-side. The client batches: one market refresh per app-open, otherwise a daily WorkManager job
What the token is **not**	It is not an identity. It cannot be linked to a person, and losing it costs the user nothing but a re-registration

4. Conventions

4.1 The error envelope Every non-2xx response carries the same body:

```
{
  "code": "RATE_LIMITED",
  "message": "Too many requests for this device token.",
  "retryable": true
}
```

code	HTTP	retryable	Client behaviour
BAD_REQUEST	400	false	A bug in the client. Log (without PII), do not retry
UNAUTHENTICATED	401	false	Re-register once, then give up and use cache
NOT_FOUND	404	false	Treat the instrument as unpriced — report coverage, never substitute zero
RATE_LIMITED	429	true	Exponential backoff; the daily job picks it up
UPSTREAM_UNAVAILABLE	502	true	Serve cache with its staleness label
SERVER_ERROR	500	true	Serve cache; open the circuit breaker after repeated failures
VERSION_CONFLICT	409	false	Backup blob only: a newer blob exists. Surface the conflict, never overwrite silently
PAYLOAD_TOO_LARGE	413	false	Backup blob only: over 50 MB. Report it to the user

The rule that matters more than the envelope: no error path may leave a screen blank or a figure invented. Every failure resolves to cached data with a label, or to an explicit "not available" (P-02, P-04).

4.2 Pagination, filtering, sorting Not used, and that is a decision rather than an omission. Every endpoint is either a **batch lookup by explicit ids** (?ids=, ?schemes=, ?pairs=) or a **bounded series** (?days=). There is no collection to page through, because the server holds no collection about the user. Batch size is capped server-side at 100 ids per request; the client splits.

4.3 Caching

Mechanism	Use
ETag / If-None-Match	Every GET. A 304 is the expected steady state for a daily-updating price
Client TTL	Per feed, stored with the cached row alongside its observation date
Staleness label	Every screen rendering external data shows "as of <date>" from the cached observation date, not from the fetch time

4.4 Idempotency POST and PUT carry Idempotency-Key (a client UUID). The server must treat a repeat as the original. This matters for exactly one endpoint in practice — a backup blob PUT retried after a timeout must not create a second version.

4.5 Versioning and deprecation

Rule	Detail
Path versioning	/v1. A breaking change is /v2, never a changed /v1
Deprecation	A Deprecation header and a minimum six-month overlap — but note the client already treats every endpoint as optional, so the realistic deprecation path is "the client falls back to cache", which is a degradation the app is built for
Client tolerance	Unknown JSON fields are ignored by kotlinx.serialization configuration, so an additive server change cannot break an older client

5. Operations

API	Method	Path	Purpose	Auth	Requirements	As built
API-001	GET	`/v1/market/prices?ids=...`	Batch quotes: gold, silver, crypto, indices	Device token	FR-042, FR-073	**Client only** — `MarketDataApi.getPrices`, and it ships unconfigured
API-002	GET	`/v1/market/nav?schemes=...`	Mutual-fund NAVs, AMFI-sourced, end of day	Device token	FR-042, FR-046	No
API-003	GET	`/v1/market/macro`	CPI series, repo rate, small-savings and FD rate ranges	Device token	FR-073, FR-081	No
API-004	GET	`/v1/market/fuel?city=...`	Daily petrol and diesel price for a coarse city id	Device token	FR-050	No
API-005	GET	`/v1/market/history?ids=...&days=...`	Daily OHLC series for the signal engine and its backtests	Device token	FR-078	No
API-006	GET	`/v1/market/fx?pairs=...`	Daily FX reference rates for multi-currency consolidation	Device token	FR-073	No
API-007	GET	`/v1/knowledge/tax?fy=...`	Versioned Indian tax rule tables	Device token	FR-081	No — which is why the tax engine states what it left out (ADR-0072)
API-008	GET	`/v1/knowledge/calendar?year=...`	Festival and season events for the seasonality engine	Device token	FR-064	No
API-009	GET	`/v1/knowledge/maintenance?class=...`	Vehicle and appliance maintenance knowledge-base updates	Device token	FR-067, FR-068	No — the KB ships in `ai/` instead
API-010	PUT/GET/DELETE	`/v1/backup/blob`	An end-to-end-encrypted backup blob, $\leq$ 50 MB, server-blind	Device token	FR-087	No — local SAF backup ships instead
API-011	POST	`/v1/llm/chat`	An opt-in cloud LLM relay taking a structured context pack, never raw transactions	Device token	FR-077	No — and deliberately so: the on-device model shipped first (ADR-0079)
API-012	POST	`/v1/device/register`	Anonymous device registration returning a signed token	None	API-001... API-011	No

6. Operation detail

Every operation is specified to the same depth, whether or not it is built, because an endpoint the app will one day call has to be designable from this document alone. The **As built** column in §5 is the honest status; this section is the contract.

Three rules hold across all twelve and are not repeated per operation:

- **No request carries user data.** Not an amount, not a merchant, not a transaction id, not a precise location. The parameters are public identifiers — a price key, a scheme code, a financial year, a coarse city (`P-01`).
- **Every money figure is an integer in minor units** and every date is an ISO `yyyy-MM-dd` string. A fractional `*_minor` field is rejected by the client's deserialiser rather than rounded (`BR-001`, `MNY-001`, `TIM-002`).
- **Every fact carries the date it was observed or published**, because an engine result has to be reproducible from the inputs it had (`AI-ARC-006`, `BR-013`).

API-001 — Batch quotes `GET /v1/market/prices?ids=GOLD_24K_INR,BTC_INR&as_of=2026-10-23`

Request

Parameter	Type	Required	Meaning
`ids`	comma-separated list, $\leq$ 100	yes	Price keys. A key is a public identifier, never a holding id
`as_of`	`yyyy-MM-dd`	no	Defaults to the latest available close

Response 200

```
{
  "as_of": "2026-10-23",
  "prices": [
    { "id": "GOLD_24K_INR", "unit": "GRAM", "price_minor": 982340, "currency": "INR", "observed_on": "2026-10-23" },
    { "id": "BTC_INR", "unit": "UNIT", "price_minor": 812345600, "currency": "INR", "observed_on": "2026-10-23" }
  ],
  "unavailable": ["SILVER_INR"]
}
```

Field	Type	Meaning
`as_of`	`yyyy-MM-dd`	The close date this response speaks for
`prices[].id`	string	Echoes the requested key, so a partial response is attributable
`prices[].unit`	enum	`GRAM`, `UNIT`, `INDEX_POINT` — what one `price_minor` buys
`prices[].price_minor`	integer	Minor units. Never a decimal (BR-001, VAL-015)
`prices[].observed_on`	`yyyy-MM-dd`	What the staleness label shows. **Not** the fetch time
`unavailable[]`	array of id	Ids the upstream had no price for. The client reports these as missing coverage and **never** substitutes zero (ADR-0029)

Status codes 200 · 304 (ETag match) · 400 (`BAD_REQUEST`, malformed or over-long `ids`) · 401 · 429 · 502 · 500.

Client behaviour on every failure — serve `market_close` from cache with its `close_iso_date` as the label; if there is no cached row, report the holding as unpriced. A price is never interpolated, extrapolated or defaulted.

Validation — VAL-015 (a price must carry the date it was observed). A response whose `price_minor` is fractional is rejected by the client's deserialiser rather than rounded.

API-002 — Mutual-fund NAVs `GET /v1/market/nav?schemes=118989,120503&as_of=2026-10-23`

Request

Parameter	Type	Required	Meaning
`schemes`	comma-separated AMFI scheme codes, ≤ 100	yes	Public scheme codes. The user's units are never sent
`as_of`	`yyyy-MM-dd`	no	Defaults to the latest published NAV date

Response 200

```
{
  "as_of": "2026-10-23",
  "navs": [
    { "scheme": "118989", "nav_minor": 8742, "currency": "INR", "nav_date": "2026-10-23" },
    { "scheme": "120503", "nav_minor": 134560, "currency": "INR", "nav_date": "2026-10-22" }
  ],
  "unavailable": ["999999"]
}
```

Field	Type	Meaning
`navs[].nav_minor`	integer	Paise per unit. Four-decimal NAVs are carried as paise and the precision loss is accepted deliberately — see below
`navs[].nav_date`	`yyyy-MM-dd`	The date AMFI published it. Rows in one response may differ
`unavailable[]`	array	Codes with no published NAV. Reported as missing coverage

Status codes 200 · 304 · 400 · 401 · 429 · 502 · 500.

Client behaviour on every failure — as API-001: the cached close, labelled with its own date, or unpriced. A holding whose NAV is unavailable is excluded from the allocation denominator rather than valued at zero (`UC-024-E1`).

Validation — VAL-015. **End of day only**: an intraday NAV does not exist, and the client must not imply one by relabelling a stale figure as current.

Why the return is not in this response. The endpoint supplies a *price*. XIRR and absolute gain are computed on the device from the user's own lots, because a return is a figure about the user and `P-03` puts every such figure in a deterministic engine (`FR-046`, ADR-0028).

The precision decision. NAVs are published to four decimals and `nav_minor` holds two. The unit price is therefore rounded at the paise, and the resulting valuation error is bounded by half a paisa per unit — material only at holdings far larger than this app targets. Carrying a third unit scale to avoid it was rejected as a worse trade than a bounded, stated error (`MNY-001` keeps one scale everywhere).

API-003 — Macro series `GET /v1/market/macro`

Request — no parameters. The response is small and fully cacheable; a parameterised series would invite the client to ask for a window it then interprets, and the engines want the whole published series.

Response 200

```
{
  "repo_rate_bps": 650,
  "repo_published_on": "2026-10-01",
  "cpi": [
    { "period": "2026-09", "index_points_bps": 1874300, "published_on": "2026-10-12" }
```

```
  ],
  "savings_rates": [
    { "instrument": "PPF", "rate_bps": 710, "effective_from": "2026-10-01" },
    { "instrument": "FD_1Y", "rate_bps_low": 650, "rate_bps_high": 745, "effective_from": "2026-10-01" }
  ]
}
```

Field	Type	Meaning
`repo_rate_bps`	integer bps	Feeds floating-rate loan logic (`BR-002` — rates are bps, never a float)
`cpi[].index_points_bps`	integer bps	The index, scaled, so no decimal ever appears
`savings_rates[]`	array	A single rate, or a low–high range where the market quotes one
`*_published_on` / `effective_from`	`yyyy-MM-dd`	Mandatory. A macro figure with no date is unusable for a reproducible result

Status codes 200 · 304 · 401 · 429 · 502 · 500.

Client behaviour on every failure — the loan screens use the rate stored on the loan itself, which is the user's own contracted figure and needs no network at all. Only the *comparison* against the current market rate is lost, and the screen says the comparison is unavailable rather than comparing against a stale rate silently.

Validation — a rate arriving as a decimal percentage rather than bps is rejected, not converted: silently multiplying by 100 is how a 6.5% rate becomes 650% in a forecast.

API-004 — Fuel price `GET /v1/market/fuel?city=BLR`

Request

Parameter	Type	Required	Meaning
`city`	enum, coarse city id	yes	**Deliberately coarse.** A precise location would be personal data leaving the device, which `P-01` forbids

Response 200

```
{
  "city": "BLR",
  "fuels": [
    { "fuel": "PETROL", "price_minor": 10287, "unit": "LITRE", "observed_on": "2026-10-23" },
    { "fuel": "DIESEL", "price_minor": 9134, "unit": "LITRE", "observed_on": "2026-10-23" }
  ]
}
```

Status codes 200 · 304 · 400 (unknown city) · 401 · 429 · 502 · 500.

Client behaviour on every failure — the vehicle running-cost figure uses the user's own last recorded fuel entry. Where there is none, the fuel component is omitted from the monthly cost and named as omitted (`FR-050`, `UC-029-F1`).

What this endpoint cannot fix. `FR-049`'s mileage-anomaly insight needs a fuel *log*, which the app does not have (`GAP-009`). A price feed alone does not deliver that requirement, and this document does not pretend otherwise.

API-005 — Price history `GET /v1/market/history?ids=NIFTY50&days=1825`

Request

Parameter	Type	Required	Meaning
`ids`	comma-separated keys, ≤ 10	yes	Fewer than API-001 allows, because each id returns a series
`days`	integer, ≤ 1825	no	Defaults to 365. Five years is the cap
`since`	`yyyy-MM-dd`	no	For the incremental fetch after the first full one

Response 200

```
{
  "series": [
    {
      "id": "NIFTY50",
      "unit": "INDEX_POINT",
      "candles": [
        { "d": "2026-10-23", "o": 2456780, "h": 2461200, "l": 2448900, "c": 2459100 }
      ]
    }
  ],
  "truncated": false
}
```

Field	Type	Meaning
`candles[].d`	`yyyy-MM-dd`	Trading day. Non-trading days are absent, not zero-filled
`candles[].o/h/l/c`	integer	Minor units or scaled index points. Short names because a five-year series is 1,250 objects
`truncated`	boolean	True when the window was clipped to the cap, so the client knows the series is not the whole request

Status codes 200 · 304 · 400 (`days` over the cap, or over 10 ids) · 401 · 429 · 502 · 500.

Client behaviour on every failure — this is the one endpoint whose absence **removes** a feature rather than degrading it. With no history there is no backtest, and the signal engine reports "not measurable yet" instead of scoring (ADR-0055, `UC-045-F1`).

Why the fetch is incremental. A full five-year series per instrument is the single largest payload the client would ever request. It is fetched once when an instrument is first watched, then extended by `since`, so the steady state is a handful of candles a day (`NFR-006` — battery and data are a cost the user pays).

API-006 — FX rates `GET /v1/market/fx?pairs=USDINR,AEDINR`

Request

Parameter	Type	Required	Meaning
`pairs`	comma-separated pairs, ≤ 20	yes	Reference pairs. The amount being converted is never sent

Response 200

```
{
  "as_of": "2026-10-23",
  "rates": [
    { "pair": "USDINR", "rate_bps": 886400, "observed_on": "2026-10-23" }
  ]
}
```

Field	Type	Meaning
`rates[].rate_bps`	integer bps	88.64 INR per USD as 886400 bps. Integers only (`MNY-002`)
`rates[].observed_on`	`yyyy-MM-dd`	Shown beside every converted figure

Status codes 200 · 304 · 400 · 401 · 429 · 502 · 500.

Client behaviour on every failure — a foreign-currency balance is shown in its own currency, unconverted, with a note that the consolidated total excludes it. A conversion at a stale rate presented as current is the error this avoids.

For display only. No engine computes on a converted figure. A converted amount is always shown with its rate and that rate's date, because an amount the user cannot re-derive is an amount they cannot check (`P-02`).

API-007 — Tax knowledge `GET /v1/knowledge/tax?fy=2025-26`

Request

Parameter	Type	Required	Meaning
`fy`	`yyyy-yy`	yes	The Indian financial year the tables are wanted for

Response 200

```
{
  "fy": "2025-26",
  "version": "tax-kb-1.4.0",
  "effective_from": "2025-04-01",
  "regimes": [
    {
      "regime": "NEW",
      "slabs": [ { "upto_minor": 30000000, "rate_bps": 0 }, { "upto_minor": 70000000, "rate_bps": 500 } ],
      "standard_deduction_minor": 7500000,
      "rebate": { "limit_minor": 120000000, "max_minor": 6000000 }
    }
  ],
  "capital_gains": [
    { "asset": "EQUITY_LISTED", "horizon_months": 12, "short_rate_bps": 2000, "long_rate_bps": 1250,
      "long_exemption_minor": 12500000, "rules_effective_from": "2024-07-23" }
  ]
}
```

Field	Type	Meaning
`version`	string	**Cited in every engine result that used it** (`AI-ARC-006`)
`slabs[].upto_minor`	integer	The upper bound of the band in paise; the last band omits it
`capital_gains[].rules_effective_from`	`yyyy-MM-dd`	The post-23-July-2024 regime is a different rule set, not a different rate

Status codes 200 · 304 · 400 (unknown `fy`) · 401 · 429 · 502 · 500.

Client behaviour on every failure — and this is the shipped state. The tax engine runs on whatever the local `ai/` knowledge base holds and **states exactly what it could not estimate**, rather than filling the gap with last year's slabs (ADR-0072, `UC-048-F1`). The estimate is partial and says so.

Why a version and not just a date. Two engines run months apart must be able to agree on which table each used. A date says when the response was fetched; the version says what was in it.

API-008 — Calendar `GET /v1/knowledge/calendar?year=2026&region=IN`

Request

Parameter	Type	Required	Meaning
`year`	integer	yes	Calendar year
`region`	enum	no	Defaults to `IN`. Regional festival sets differ within India

Response 200

```
{
  "year": 2026,
  "region": "IN",
  "version": "calendar-kb-1.1.0",
  "events": [
    { "id": "DIWALI", "start": "2026-11-08", "end": "2026-11-12", "kind": "FESTIVAL", "spend_prior": "HIGH" },
    { "id": "SCHOOL_FEES_Q1", "start": "2026-04-01", "end": "2026-04-30", "kind": "SEASON", "spend_prior": "MEDIUM" }
  ]
}
```

Field	Type	Meaning
`events[].spend_prior`	enum	A *shared* prior — `HIGH`, `MEDIUM`, `LOW` — never a figure. The amount is always the user's own history
`version`	string	Cited in the seasonality engine's evidence

Status codes 200 · 304 · 400 · 401 · 429 · 502 · 500.

Client behaviour on every failure — the seasonality engine uses the user's own history first and only then this prior (ADR-0044), so the endpoint's absence **lowers precision rather than breaking the feature**. A first-year user loses the festival nudge and nothing else.

Why a prior and not a number. A shared figure would be a prediction about this user drawn from other people, which `P-02` cannot explain and `P-03` will not let an engine treat as an input to money arithmetic.

API-009 — Maintenance knowledge `GET /v1/knowledge/maintenance?class=HATCHBACK_PETROL`

Request

Parameter	Type	Required	Meaning
`class`	enum	yes	A vehicle or appliance class. Never a registration, a VIN or a serial number

Response 200

```
{
  "class": "HATCHBACK_PETROL",
  "version": "maint-kb-1.3.0",
  "service_interval": { "months": 12, "km": 10000 },
  "cost_ranges": [
    { "item": "PERIODIC_SERVICE", "low_minor": 350000, "high_minor": 700000, "currency": "INR" }
  ],
  "consumables": [ { "item": "AIR_FILTER", "interval_months": 12 } ]
}
```

Field	Type	Meaning
`cost_ranges[]`	array	A **range**, never a point estimate. The engine quotes the range or declines (ADR-0052)
`service_interval`	object	The class default, used only until the user's own odometer trend exists (`UC-029-A1`)

Status codes 200 · 304 · 400 · 401 · 429 · 502 · 500.

Client behaviour on every failure — this knowledge ships inside `ai/` as checked data, so its absence costs **freshness only**. Nothing in the vehicle or appliance screens depends on the network.

API-010 — Backup blob `PUT /v1/backup/blob` · `GET /v1/backup/blob` · `DELETE /v1/backup/blob`

The server is blind. The body is ciphertext — AES-256-GCM under a key derived from the user's passphrase by Argon2id. The manifest (schema version, created-at, device) is *inside* the ciphertext; the server sees only a size and a version header.

Request — `PUT`

Header	Required	Purpose
`Content-Type: application/octet-stream`	yes	Opaque bytes. No JSON wrapper, because a wrapper invites a field the server could read
`X-Blob-Version`	yes	Monotonic. A `PUT` with a stale version gets 409 `VERSION_CONFLICT`
`Idempotency-Key`	yes	A retried `PUT` must not create a second version
`Content-Length`	yes	Rejected over 50 MB with 413 before a byte is stored

Response 201

```
{ "version": 7, "size_bytes": 1048576, "stored_at": "2026-10-23T11:04:22Z" }
```

Response — `GET` the ciphertext, with `X-Blob-Version` and `Content-Length`. No `Last-Modified`-based cache: the client either has the version or wants the bytes.

Status codes 200 · 201 · 204 (DELETE) · 401 · 404 (no blob for this device) · 409 · 413 · 429 · 500.

Error cases that matter

Case	Behaviour
409 `VERSION_CONFLICT`	**Never** resolved by overwriting. The user is shown the conflict and chooses, because the loser of a silent overwrite is an entire profile
404 on a fresh device	A *total* failure for that user — which is exactly why the local SAF backup is what shipped and why the restore drill is a release gate (`FR-088`, `DRL-001`)
413	The archive is over the cap; the client says which, rather than truncating

What the server cannot do, by construction. It cannot read the archive, enumerate a user's entities, restore on the user's behalf, or recover a lost passphrase. Those are not policies; they follow from the key never leaving the device (`SEC-005`).

API-011 — LLM relay `POST /v1/llm/chat`

Request — the body is a **structured context pack**: rendered figures and labels, never raw transactions (SRS §19).

```
{
  "intent": "EXPLAIN_SAFE_TO_SPEND",
  "locale": "en-IN",
  "facts": [
    { "key": "safe_to_spend_today", "rendered": "Rs.1,240" },
    { "key": "month_remaining_days", "rendered": "9 days" }
  ],
  "question_class": "WHY_FIGURE"
}
```

Field	Type	Meaning
`facts[].rendered`	string	**Already a string.** The relay never receives a number it could recompute, and never a merchant, payee or account name
`intent` / `question_class`	enum	Drawn from the tool registry, so the request shape is a closed set rather than free text
`locale`	BCP-47	So the sentence comes back in the user's language (`FR-090`)

Response 200

```
{ "sentence_spans": ["Your safe-to-spend is ", "Rs.1,240", " with ", "9 days", " left this month."] }
```

Spans, not a sentence: the guardrail checks each figure span against the fact that produced it, and a span it cannot match is refused before display (ADR-0048).

Status codes 200 · 400 · 401 · 429 · 502 · 500.

Client behaviour on every failure — the on-device model answers instead; if that is unavailable too, the rendered figures are shown without prose (`UC-033-F2`). An answer is never withheld because a sentence could not be written around it.

Three rules survive whether or not this endpoint ever exists, because they are enforced on the device:

1. The model receives **words**, not rows (ADR-0053).
2. Every reply passes the guardrail before display (`FR-076`).
3. Using it at all requires an explicit, revocable consent that defaults off (`P-01`).

Operational commitments if it is ever deployed — no body logging, no conversation retention, and an Indian region for data-residency expectations. These belong in this document because they are part of the contract, not of the implementation.

API-012 — Device registration `POST /v1/device/register`

Request

```
{ "public_key": "base64url...", "app_version": "1.0.0", "platform": "android" }
```

Field	Type	Meaning
`public_key`	base64url	Generated on the device. **No email, no password, no recoverable identity**
`app_version`	semver	So a deprecated client can be told to upgrade (§4.5)

Response 201

```
{ "token": "eyJ...", "expires_at": "2027-10-23T00:00:00Z" }
```

Status codes 201 · 400 · 429 · 500. Never 401 — this is the one unauthenticated operation, because it is what issues the credential.

Client behaviour on every failure — every other endpoint is optional, so a failed registration costs exactly the optional features and nothing else. The app does not block, warn repeatedly or retry aggressively on a path the user did not ask for.

What losing the token costs. A re-registration, and nothing else. There is no account to recover because there is no account: the token identifies a device's right to read public facts, not a person.

---

7. Specific error cases, operation by operation

§4.1 gives the envelope and the eight codes. This table gives the cases that are specific to one operation — the ones a client author would otherwise have to guess.

Operation	Case	Code	HTTP	What the client does
API-001	More than 100 ids	`BAD_REQUEST`	400	A client bug; the batcher caps at 100 before sending
API-001	A known id with no price for `as_of`	—	200	It arrives in `unavailable[]`, not as an error. Reported as missing coverage (ADR-0029)
API-001	An id the proxy has never heard of	—	200	Also `unavailable[]`. An unknown key is not an error, because the client's key set may be newer than the proxy's
API-002	A scheme code that is not a valid AMFI code	`BAD_REQUEST`	400	Validated client-side first; a 400 here means the validation drifted
API-002	A NAV not yet published for today	—	200	The previous `nav_date` is returned and labelled. The client never shows today's date on yesterday's NAV
API-003	The CPI series is empty because the publisher is late	—	200	An empty array is a valid answer. The screen says the series is unavailable, not that inflation is zero
API-004	An unknown city id	`NOT_FOUND`	404	Fall back to the user's own last fuel entry (`UC-029-F1`)
API-005	`days` above 1825	`BAD_REQUEST`	400	The client caps it; a 400 means the cap drifted
API-005	More than 10 ids	`BAD_REQUEST`	400	The series endpoint is deliberately narrower than API-001
API-005	The window was clipped to the cap	—	200	`truncated: true`. The backtest states the window it actually had
API-006	An unsupported pair	—	200	Absent from `rates[]`; that balance stays unconverted and is excluded from the consolidated total
API-007	A financial year the tables do not cover	`NOT_FOUND`	404	The engine states what it could not estimate (ADR-0072) — it does not reach for an adjacent year
API-008	A region with no event set	—	200	An empty `events[]`. Seasonality falls back to the user's own history, which it prefers anyway (ADR-0044)
API-009	An unknown vehicle or appliance class	`NOT_FOUND`	404	The generic interval is used and labelled generic (`UC-029-A1`, `UC-041-E1`)
API-010 `PUT`	A stale `X-Blob-Version`	`VERSION_CONFLICT`	409	**Never** overwritten. The conflict is shown to the user

Operation	Case	Code	HTTP	What the client does
API-010 `PUT`	Over 50 MB	`PAYLOAD_TOO_LARGE`	413	Reported with the size, before any byte is stored
API-010 `PUT`	The same `Idempotency-Key` retried	—	200	The existing version is returned. A retry must not create a second version
API-010 `GET`	No blob for this device	`NOT_FOUND`	404	On a fresh device this is a total failure for that user — which is why the local backup is what shipped (`FR-088`)
API-010 `DELETE`	Already deleted	—	204	Idempotent. A second delete is a success, not a 404
API-011	A fact value that is not already a string	`BAD_REQUEST`	400	A client bug, and a serious one: it means a raw number reached the relay (`P-03`)
API-011	An intent outside the registry	`BAD_REQUEST`	400	The registry is the closed set; a free-text intent is unrepresentable
API-011	The model returns a span the guardrail cannot match	—	200	The **client** refuses the sentence before display. The transport succeeded; the content did not (`FR-076`)
API-012	A malformed public key	`BAD_REQUEST`	400	Regenerate the key pair once, then stop
API-012	Registration rate-limited	`RATE_LIMITED`	429	Back off. Every other endpoint is optional, so there is nothing to block on
Any	A `*_minor` field arriving fractional	—	200	**Rejected by the client's deserialiser**, not rounded. A fractional paisa is a contract violation, and silently rounding one is how money bugs start (`MNY-001`)
Any	A fact with no `observed_on` or `published_on`	—	200	Rejected by the client. A dated fact is what makes an engine result reproducible (`AI-ARC-006`)

The two rows at the bottom are the important ones. They are not HTTP failures: the server answered, and the client refused the body. That is deliberate. The device is the only place this contract can be enforced, because there is no server yet to enforce it at the other end — so the client is written as though the proxy were hostile.

8. OpenAPI skeleton

Complete for all twelve operations, so the proxy could be built from this document alone. `servers` is empty because no shipped build configures a host (ADR-0030); that is a fact about the client, not a hole in the specification.

```
openapi: 3.1.0
info:
  title: AI Personal CFO -- public-facts proxy
  version: 1.0.0
  description: >
    Facts about the world that a phone cannot compute. Holds no user data.
    Every endpoint is optional to the client; the app is fully functional offline.
    No request carries an amount, a merchant, a transaction id or a precise location.
servers: []   # no host is configured in any shipped build -- see ADR-0030
security:
  - deviceToken: []

components:
  securitySchemes:
    deviceToken:
      type: http
      scheme: bearer
      description: An anonymous device token from POST /v1/device/register.

  parameters:
    AsOf:
      name: as_of
      in: query
      required: false
      description: Defaults to the latest available close.
      schema: { type: string, format: date }

  schemas:
    Error:
      type: object
      required: [code, message, retryable]
      properties:
        code:
          type: string
          enum: [BAD_REQUEST, UNAUTHENTICATED, NOT_FOUND, RATE_LIMITED,
                 UPSTREAM_UNAVAILABLE, SERVER_ERROR, VERSION_CONFLICT, PAYLOAD_TOO_LARGE]
        message:   { type: string }
        retryable: { type: boolean }
```

```
Price:                                # API-001
  type: object
  required: [id, unit, price_minor, currency, observed_on]
  properties:
    id:          { type: string }
    unit:        { type: string, enum: [GRAM, UNIT, SHARE, INDEX_POINT] }
    price_minor: { type: integer, format: int64, description: Minor units. Never fractional. }
    currency:    { type: string, minLength: 3, maxLength: 3 }
    observed_on: { type: string, format: date }

Nav:                                  # API-002
  type: object
  required: [scheme, nav_minor, currency, nav_date]
  properties:
    scheme:    { type: string }
    nav_minor: { type: integer, format: int64 }
    currency:  { type: string, minLength: 3, maxLength: 3 }
    nav_date:  { type: string, format: date }

CpiPoint:                             # API-003
  type: object
  required: [period, index_points_bps, published_on]
  properties:
    period:           { type: string, pattern: '^\d{4}-\d{2}$' }
    index_points_bps: { type: integer }
    published_on:     { type: string, format: date }

SavingsRate:                          # API-003
  type: object
  required: [instrument, effective_from]
  properties:
    instrument:     { type: string }
    rate_bps:       { type: integer }
    rate_bps_low:   { type: integer }
    rate_bps_high:  { type: integer }
    effective_from: { type: string, format: date }

FuelPrice:                            # API-004
  type: object
  required: [fuel, price_minor, unit, observed_on]
  properties:
    fuel:        { type: string, enum: [PETROL, DIESEL, CNG] }
    price_minor: { type: integer, format: int64 }
    unit:        { type: string, enum: [LITRE, KG] }
    observed_on: { type: string, format: date }

Candle:                               # API-005
  type: object
  required: [d, o, h, l, c]
  properties:
    d: { type: string, format: date }
    o: { type: integer, format: int64 }
    h: { type: integer, format: int64 }
    l: { type: integer, format: int64 }
    c: { type: integer, format: int64 }

Series:                               # API-005
  type: object
  required: [id, unit, candles]
  properties:
    id:      { type: string }
    unit:    { type: string, enum: [GRAM, UNIT, SHARE, INDEX_POINT] }
    candles: { type: array, items: { $ref: '#/components/schemas/Candle' } }

FxRate:                               # API-006
  type: object
  required: [pair, rate_bps, observed_on]
  properties:
    pair:        { type: string, pattern: '^[A-Z]{6}$' }
    rate_bps:    { type: integer, description: Integer basis points. MNY-002. }
    observed_on: { type: string, format: date }

TaxSlab:                              # API-007
  type: object
  required: [rate_bps]
  properties:
    upto_minor: { type: integer, format: int64, description: Absent on the last band. }
    rate_bps:   { type: integer }

TaxRegime:                            # API-007
  type: object
  required: [regime, slabs]
  properties:
    regime: { type: string, enum: [OLD, NEW] }
    slabs:  { type: array, items: { $ref: '#/components/schemas/TaxSlab' } }
    standard_deduction_minor: { type: integer, format: int64 }
    rebate:
      type: object
```

```
        properties:
          limit_minor: { type: integer, format: int64 }
          max_minor:   { type: integer, format: int64 }

    CapitalGainsRule:                     # API-007
      type: object
      required: [asset, horizon_months, short_rate_bps, long_rate_bps, rules_effective_from]
      properties:
        asset:               { type: string }
        horizon_months:      { type: integer }
        short_rate_bps:      { type: integer }
        long_rate_bps:       { type: integer }
        long_exemption_minor: { type: integer, format: int64 }
        rules_effective_from: { type: string, format: date }

    CalendarEvent:                        # API-008
      type: object
      required: [id, start, end, kind, spend_prior]
      properties:
        id:          { type: string }
        start:       { type: string, format: date }
        end:         { type: string, format: date }
        kind:        { type: string, enum: [FESTIVAL, SEASON, STATUTORY] }
        spend_prior: { type: string, enum: [HIGH, MEDIUM, LOW], description: 'A prior, never a figure.' }

    CostRange:                            # API-009
      type: object
      required: [item, low_minor, high_minor, currency]
      properties:
        item:       { type: string }
        low_minor:  { type: integer, format: int64 }
        high_minor: { type: integer, format: int64 }
        currency:   { type: string, minLength: 3, maxLength: 3 }

    BlobMeta:                             # API-010
      type: object
      required: [version, size_bytes, stored_at]
      properties:
        version:    { type: integer }
        size_bytes: { type: integer, format: int64 }
        stored_at:  { type: string, format: date-time }

    ContextFact:                          # API-011
      type: object
      required: [key, rendered]
      properties:
        key:      { type: string }
        rendered: { type: string, description: Already a string. The relay never receives a number. }

  responses:
    BadRequest:      { description: 400, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    Unauthenticated: { description: 401, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    NotFound:        { description: 404, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    RateLimited:     { description: 429, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    Upstream:        { description: 502, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    ServerError:     { description: 500, content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } } }
    NotModified:     { description: '304 -- ETag match; serve cache.' }

paths:
  /v1/market/prices:
    get:
      operationId: getPrices              # API-001 -- the one operation shipped code calls
      summary: Batch quotes for public price keys.
      parameters:
        - { name: ids, in: query, required: true, schema: { type: string }, description: 'Comma-separated, max 100.' }
        - $ref: '#/components/parameters/AsOf'
      responses:
        '200':
          description: Quotes, plus the ids that had none.
          content:
            application/json:
              schema:
                type: object
                required: [as_of, prices]
                properties:
                  as_of:       { type: string, format: date }
                  prices:      { type: array, items: { $ref: '#/components/schemas/Price' } }
                  unavailable: { type: array, items: { type: string } }
        '304': { $ref: '#/components/responses/NotModified' }
        '400': { $ref: '#/components/responses/BadRequest' }
        '401': { $ref: '#/components/responses/Unauthenticated' }
        '429': { $ref: '#/components/responses/RateLimited' }
        '502': { $ref: '#/components/responses/Upstream' }
        '500': { $ref: '#/components/responses/ServerError' }

  /v1/market/nav:
    get:
      operationId: getNavs                # API-002
```

```
    summary: End-of-day mutual-fund NAVs. No intraday NAV exists.
    parameters:
      - name: schemes
        in: query
        required: true
        schema: { type: string }
        description: 'Comma-separated AMFI scheme codes, max 100.'
      - $ref: '#/components/parameters/AsOf'
    responses:
      '200':
        description: NAVs, plus the codes that had none.
        content:
          application/json:
            schema:
              type: object
              required: [as_of, navs]
              properties:
                as_of:       { type: string, format: date }
                navs:        { type: array, items: { $ref: '#/components/schemas/Nav' } }
                unavailable: { type: array, items: { type: string } }
      '304': { $ref: '#/components/responses/NotModified' }
      '400': { $ref: '#/components/responses/BadRequest' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }
      '502': { $ref: '#/components/responses/Upstream' }

/v1/market/macro:
  get:
    operationId: getMacro               # API-003
    summary: Repo rate, CPI series and small-savings ranges, each dated.
    responses:
      '200':
        description: Every value carries the date it was published.
        content:
          application/json:
            schema:
              type: object
              required: [repo_rate_bps, repo_published_on]
              properties:
                repo_rate_bps:     { type: integer }
                repo_published_on: { type: string, format: date }
                cpi:               { type: array, items: { $ref: '#/components/schemas/CpiPoint' } }
                savings_rates:     { type: array, items: { $ref: '#/components/schemas/SavingsRate' } }
      '304': { $ref: '#/components/responses/NotModified' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }
      '502': { $ref: '#/components/responses/Upstream' }

/v1/market/fuel:
  get:
    operationId: getFuelPrices          # API-004
    summary: Petrol and diesel for a coarse city id. Coarse by privacy requirement.
    parameters:
      - { name: city, in: query, required: true, schema: { type: string } }
    responses:
      '200':
        description: Dated fuel prices for that city.
        content:
          application/json:
            schema:
              type: object
              required: [city, fuels]
              properties:
                city:  { type: string }
                fuels: { type: array, items: { $ref: '#/components/schemas/FuelPrice' } }
      '304': { $ref: '#/components/responses/NotModified' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '404': { $ref: '#/components/responses/NotFound' }
      '429': { $ref: '#/components/responses/RateLimited' }
      '502': { $ref: '#/components/responses/Upstream' }

/v1/market/history:
  get:
    operationId: getPriceHistory        # API-005
    summary: Daily OHLC series for the signal engine and its backtests.
    parameters:
      - { name: ids,   in: query, required: true,  schema: { type: string }, description: 'Comma-separated, max 10.' }
      - { name: days,  in: query, required: false, schema: { type: integer, maximum: 1825, default: 365 } }
      - { name: since, in: query, required: false, schema: { type: string, format: date } }
    responses:
      '200':
        description: One series per id. Non-trading days are absent, never zero-filled.
        content:
          application/json:
            schema:
              type: object
              required: [series]
              properties:
```

```
                series:    { type: array, items: { $ref: '#/components/schemas/Series' } }
                truncated: { type: boolean }
      '304': { $ref: '#/components/responses/NotModified' }
      '400': { $ref: '#/components/responses/BadRequest' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }
      '502': { $ref: '#/components/responses/Upstream' }

/v1/market/fx:
  get:
    operationId: getFxRates              # API-006
    summary: Daily reference rates, for display only.
    parameters:
      - { name: pairs, in: query, required: true, schema: { type: string }, description: 'Comma-separated, max 20.' }
    responses:
      '200':
        description: Dated reference rates. An unsupported pair is simply absent.
        content:
          application/json:
            schema:
              type: object
              required: [as_of, rates]
              properties:
                as_of: { type: string, format: date }
                rates: { type: array, items: { $ref: '#/components/schemas/FxRate' } }
      '304': { $ref: '#/components/responses/NotModified' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }
      '502': { $ref: '#/components/responses/Upstream' }

/v1/knowledge/tax:
  get:
    operationId: getTaxKnowledge         # API-007
    summary: Versioned Indian tax tables. The version is cited in every engine result.
    parameters:
      - { name: fy, in: query, required: true, schema: { type: string, pattern: '^\d{4}-\d{2}$' } }
    responses:
      '200':
        description: Both regimes plus the capital-gains rules in force.
        content:
          application/json:
            schema:
              type: object
              required: [fy, version, effective_from, regimes]
              properties:
                fy:             { type: string }
                version:        { type: string }
                effective_from: { type: string, format: date }
                regimes:        { type: array, items: { $ref: '#/components/schemas/TaxRegime' } }
                capital_gains:  { type: array, items: { $ref: '#/components/schemas/CapitalGainsRule' } }
      '304': { $ref: '#/components/responses/NotModified' }
      '400': { $ref: '#/components/responses/BadRequest' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '404': { $ref: '#/components/responses/NotFound' }
      '429': { $ref: '#/components/responses/RateLimited' }

/v1/knowledge/calendar:
  get:
    operationId: getCalendar             # API-008
    summary: Festival and season events as shared priors, never as figures.
    parameters:
      - { name: year,   in: query, required: true,  schema: { type: integer } }
      - { name: region, in: query, required: false, schema: { type: string, default: IN } }
    responses:
      '200':
        description: An empty events array is a valid answer.
        content:
          application/json:
            schema:
              type: object
              required: [year, region, version, events]
              properties:
                year:    { type: integer }
                region:  { type: string }
                version: { type: string }
                events:  { type: array, items: { $ref: '#/components/schemas/CalendarEvent' } }
      '304': { $ref: '#/components/responses/NotModified' }
      '400': { $ref: '#/components/responses/BadRequest' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }

/v1/knowledge/maintenance:
  get:
    operationId: getMaintenanceKnowledge # API-009
    summary: Service intervals and cost ranges by class. Ranges, never point estimates.
    parameters:
      - { name: class, in: query, required: true, schema: { type: string } }
    responses:
```

```
      '200':
        description: Intervals, cost ranges and consumables for that class.
        content:
          application/json:
            schema:
              type: object
              required: [class, version, service_interval]
              properties:
                class:   { type: string }
                version: { type: string }
                service_interval:
                  type: object
                  properties:
                    months: { type: integer }
                    km:     { type: integer }
                cost_ranges: { type: array, items: { $ref: '#/components/schemas/CostRange' } }
                consumables:
                  type: array
                  items:
                    type: object
                    required: [item, interval_months]
                    properties:
                      item:            { type: string }
                      interval_months: { type: integer }
      '304': { $ref: '#/components/responses/NotModified' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '404': { $ref: '#/components/responses/NotFound' }
      '429': { $ref: '#/components/responses/RateLimited' }

/v1/backup/blob:
  put:
    operationId: putBackupBlob           # API-010
    summary: Store an end-to-end-encrypted archive. The server cannot read it.
    parameters:
      - { name: X-Blob-Version,  in: header, required: true, schema: { type: integer } }
      - { name: Idempotency-Key, in: header, required: true, schema: { type: string } }
    requestBody:
      required: true
      content:
        application/octet-stream:
          schema: { type: string, format: binary, maxLength: 52428800 }
    responses:
      '200':
        description: The same Idempotency-Key was already stored; the existing version is returned.
        content:
          application/json:
            schema: { $ref: '#/components/schemas/BlobMeta' }
      '201':
        description: Stored.
        content:
          application/json:
            schema: { $ref: '#/components/schemas/BlobMeta' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '409':
        description: 409 -- a newer blob exists. Never resolved by overwriting.
        content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }
      '413':
        description: 413 -- over 50 MB.
        content: { application/json: { schema: { $ref: '#/components/schemas/Error' } } }
      '429': { $ref: '#/components/responses/RateLimited' }
  get:
    operationId: getBackupBlob           # API-010
    summary: Fetch the ciphertext. A 404 on a fresh device is a total loss for that user.
    responses:
      '200':
        description: The ciphertext, with X-Blob-Version.
        content:
          application/octet-stream:
            schema: { type: string, format: binary }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '404': { $ref: '#/components/responses/NotFound' }
      '429': { $ref: '#/components/responses/RateLimited' }
  delete:
    operationId: deleteBackupBlob        # API-010
    summary: Remove the stored archive. Idempotent.
    responses:
      '204': { description: 'Deleted, or already absent.' }
      '401': { $ref: '#/components/responses/Unauthenticated' }
      '429': { $ref: '#/components/responses/RateLimited' }

/v1/llm/chat:
  post:
    operationId: relayChat               # API-011
    summary: Turn rendered figures into a sentence. Receives words, never rows.
    requestBody:
      required: true
      content:
        application/json:
```

```
            schema:
              type: object
              required: [intent, locale, facts, question_class]
              properties:
                intent:         { type: string }
                locale:         { type: string }
                question_class: { type: string }
                facts:          { type: array, items: { $ref: '#/components/schemas/ContextFact' } }
      responses:
        '200':
          description: Spans, so the guardrail can check each figure against the fact behind it.
          content:
            application/json:
              schema:
                type: object
                required: [sentence_spans]
                properties:
                  sentence_spans: { type: array, items: { type: string } }
        '400': { $ref: '#/components/responses/BadRequest' }
        '401': { $ref: '#/components/responses/Unauthenticated' }
        '429': { $ref: '#/components/responses/RateLimited' }
        '502': { $ref: '#/components/responses/Upstream' }

  /v1/device/register:
    post:
      operationId: registerDevice          # API-012
      summary: Anonymous device registration. No email, no password, no recoverable identity.
      security: []                         # the one unauthenticated operation
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              required: [public_key, app_version, platform]
              properties:
                public_key:  { type: string }
                app_version: { type: string }
                platform:    { type: string, enum: [android] }
      responses:
        '201':
          description: A bearer token for the public-facts endpoints.
          content:
            application/json:
              schema:
                type: object
                required: [token, expires_at]
                properties:
                  token:      { type: string }
                  expires_at: { type: string, format: date-time }
        '400': { $ref: '#/components/responses/BadRequest' }
        '429': { $ref: '#/components/responses/RateLimited' }
```

What is deliberately absent from this document. No pagination parameters beyond API-005's window, because no endpoint returns an unbounded user-scoped collection — there is no user-scoped collection to return. No search. No write operation except the opaque backup blob and the registration. An endpoint that accepted a transaction would be a different product.

---

9. What the app does instead, today

Specified endpoint	What ships in its place
API-001, API-002, API-006	`market_close` is read from cache; an unpriced holding is reported as missing coverage, never as Rs. 0 (ADR-0029)
API-003, API-007	Tax and macro parameters are versioned rows in `ai/knowledge/`, and the engine names what it could not estimate (ADR-0072)
API-005	No history, so the signal engine reports "not measurable yet" rather than an unbacked score (ADR-0055)
API-008, API-009	The calendar and maintenance knowledge ship as checked data in `ai/`, guarded by drift tests
API-010	Local backup to user-chosen storage through SAF, with the same Argon2id → AES-256-GCM envelope (ADR-0039)
API-011	The on-device model, which is strictly better for `P-01` and shipped first (ADR-0079)
API-012	Nothing to register with, so nothing registers

The design holds. Every one of these substitutions is a *degradation the app was built for*, not a workaround

bolted on: the cache-with-staleness path, the coverage-not-zero rule, and the on-device model are all first-class behaviours with their own tests. The gap is real — but the architecture is not waiting on it.

Part 07 — Test Strategy and Cases

← Master Blueprint · Related: 02 Requirements · 03 Flows & Use Cases · 10 Traceability

> ***The constraint that shaped this strategy.*** *There is one developer and no QA team (`CON-008`). A checklist a human runs is a gate that eventually does not run, so every promise in this document is a task that fails a build. The project has already shipped one gate that nothing ever ran — `koverVerify` with zero rules, which passed at 0% coverage — and that experience is why each row below names the mechanism rather than the intention.* ***Current size:*** *432 Kotlin files in test source sets — 386 test classes plus 46 fakes and fixtures — carrying* ***3,986 test cases*** *(3,923 on the JVM, 63 instrumented), 115 committed screenshot renders, at `v0.13.10`.*

1. Test levels, and what each one proves

Level	Proves	Tools	Where it runs
Engine unit	A pure function maps fixed input to fixed output, for every branch	JUnit, pure Kotlin	`:core:*`, `:domain:*` — no device, no Android
Golden file	The engine's answer has not changed since it was frozen, byte for byte	A golden harness that **refuses to be vacuous and never rewrites its own fixture** (ADR-0064)	`:domain:engines:*`
Property	A mathematical identity holds over generated inputs — splits sum, forecasts are monotonic, a balance equals its ledger	JUnit with generated cases	`:core:model`, `:domain:*`
Repository	A query returns what the schema holds, and a migration does not lose data	Room in-memory, plus real pre-change fixtures	`:core:database`, `:data:*`
ViewModel	The full `UiState` sequence including loading and error	Turbine on `StateFlow`	`:feature:*`
Compose UI	A critical flow works through the real composables	Compose UI test, Robolectric	`:feature:*`, `:app`
Screenshot	Every screen still looks like its committed render, in light, dark and at 200% font	Paparazzi, 115 baselines	`:feature:*`, `:app` (its own JVM — ADR-0082)
Drift	A typed Kotlin mirror still agrees with the `ai/` data row it claims to copy	36 drift-test classes, 252 cases, with the data files declared as Gradle task **inputs** so the gate cannot go stale	Across the tree
AI evaluation	Accuracy against a frozen labelled dataset clears an absolute floor	Eval harness with versioned datasets (ADR-0065)	`:domain:engines:*`, `:ml:*`
Instrumented end-to-end	The app works on a real device — including in airplane mode	Emulator in CI, KVM	`:app`
Release drill	A backup actually restores on a fresh device	`restoreDrill` (ADR-0041)	Emulator in CI
Lint detector	The rule that fails a build actually fails it	14 tests over the six custom detectors	`:lint`

1.1 One command `./gradlew unitTests` — never `testDebugUnitTest`, which is an Android variant task and **silently skips** every pure-Kotlin module, `:lint`, and the Paparazzi verification. `unitTests` depends on `testDebugUnitTest`, `test`, `verifyPaparazziDebug` and `jvmTest` across every subproject, so a module cannot be missed by having the "wrong" kind of build script.

2. Entry and exit criteria

Gate	Entry	Exit
A feature is ready to start	The requirement has an id, an `AC`, and a `TC` planned	—
A feature is done	Code compiles, `unitTests` green	Every DoD item in 08 §6
A PR may open	`/pre-merge` run locally	CI green: build, test, lint, evals, screenshot diff, OSV, release hardening
A promotion to `stage` or `main`	CI green on `dev`	**Plus** the restore drill and the offline end-to-end smoke on a device. A red drill blocks the release (DRL-001)
A commit	—	SonarQube clean, with the denominator checked (≈582 files / ≈51,400 ncloc; a run reporting ~293 files has dropped every Android module and means nothing)

3. Environments and test data

Aspect	Approach
Unit and golden	No environment. Pure Kotlin on the JVM, JDK 21 (Robolectric requires it)
Database	Room in-memory, plus **real** pre-change database fixtures for every migration test
Instrumented	An emulator in CI with KVM enabled; airplane mode is turned **on by a test**, and a check that could not fail was removed when it was found (ADR-0067)
Test data	Generated fixtures and the demo dataset. **No real user data exists anywhere** — there is no server, no analytics store, and no shared database, so the usual PII-in-test-data problem has nothing to draw from
PII in fixtures	Fixture names and merchants are invented. The privacy risk in this project is not test data; it is a log line, and `CfoPiiInLogs` is what addresses it
AI evaluation sets	Frozen, labelled and **versioned**; every run reports which dataset version it measured (ADR-0065), so an accuracy number is never quoted without saying what it was measured on
Golden fixtures	Committed per engine and per engine version. The harness never rewrites a fixture to make a test pass — recording is deliberate and separate (ADR-0064)

3.1 Where the tests actually are Counted from the tree, not quoted: `@Test`, `@ParameterizedTest` and `@RepeatedTest` annotations in every non-`build` test source set.

Module group	Test files	Test cases	What dominates
`domain/engines/*`	129	1,257	Golden fixtures, property tests and seeded-determinism tests. The engines are pure Kotlin, so this is the cheapest place to test and the project leans on that
`data/repository`	57	791	Room in-memory, profile scoping, soft-delete semantics, transaction boundaries
`feature/transactions`	15	294	The most-used screen, and the one with splits and transfers behind it
`core/model`	23	227	`Money`, the value classes, the enums, and the classification KB mirror
`app`	28	159	Navigation, the lock gate, and the module's own screenshot JVM (ADR-0082)
`feature/accounts`	14	149	Eleven account types, cards, loans, reconciliation
`core/common`	13	111	The engine registry, `SystemClock`, the SEC-003 audit
`feature/dashboard`	11	108	Card-by-card state, including every empty and error state
`core/database`	11	100	32 migrations and the schema invariants
`feature/settings`	10	97	Consents, erase, backup, export
`ml/llm`	10	78	The verbaliser, the model manifest, the prompt
`core/crypto`	4	66	Keys, wrapping, rotation, Argon2id

Module group	Test files	Test cases	What dominates
Remaining 19 groups	61	449	Features, lint, network, widget, the KMP spike
Total	**386** `*Test.kt` **files**	**3,986**	3,923 on the JVM, 63 instrumented

Two things in that table are worth a sentence.

The engine modules hold a third of every test in the project. That is the intended shape: an engine is pure Kotlin with no Android dependency (`ARC-002`), so it can be tested exhaustively for the price of a JVM test, and `P-08` means its output is fixed for a fixed input. The expensive tests — instrumented, screenshot, device — are reserved for what only a device can prove.

63 instrumented cases against 3,923 on the JVM. That ratio is deliberate and it has a cost: the migration round-trip, the erase sweep and the restore drill are the paths that need real hardware, and they are the ones that run least often. `MigrationSafetyTest` exists precisely to move as much of that proof as possible onto the JVM, where it runs every time (`08 §10.1`).

The counts above exclude 46 fakes and fixture files in the same source sets, which carry no `@Test` of their own: 432 Kotlin files in test sources, of which 386 are test classes. Python tooling adds 145 further test methods under `scripts/`, run by `scriptTests`.

---

4. Automation approach

Principle	Consequence
Write the failing test first	The DoD's test items are not retrofitted
A gate must be able to fail	Proved by temporarily breaking it — the coverage bound was raised to an impossible 101% and watched go red
A gate must be reachable	`:lint`'s own 14 tests were reachable from nothing until `unitTests` existed; a detector could have broken while every gate stayed green
Data a test reads is a task input	Otherwise a threshold edit leaves every drift test `UP-TO-DATE` and the build green — true of 29 of them when ADR-0061 found it, and of all 36 today
A render is a test	115 Paparazzi baselines mean a UI change that nobody looked at fails the build (ADR-0082)
Never change behaviour to satisfy a test written during a refactor	Assert the real behaviour, fix the comment that claimed otherwise, and report the finding

4.1 The gates, and what each one has caught A gate's value is not that it exists; it is that it has failed on something real. This project has twice found a gate that could not fail, which is why the column on the right is in this document at all.

Gate	Command	Has it caught something?
Unit and golden tests	`unitTests`	Continuously. The two property tests on split rounding and amortisation each caught a real one-unit drift before release
Coverage floors	`koverVerify`	**The gate itself was the finding.** It was vacuous until it was proved able to fail by raising the bound to an impossible 101% and watching it go red
Drift tests	`unitTests`	**Twice.** The KB-versus-mirror divergence they exist for, and then the discovery that 29 of them could not fail because their data files were not declared Gradle inputs (ADR-0061)
Custom lint — six detectors	`:lint`	Yes: a `Double` on a monetary field, a wall-clock read in domain code, and hardcoded strings in feature modules. `:lint`'s own 14 tests were themselves unreachable until `unitTests` existed

Gate	Command	Has it caught something?
Architecture guard	`unitTests`	Yes — `enforceNoAndroidPlugins()` has failed the build since issue 1.1 on a pure-Kotlin module applying an Android plugin
Screenshot renders	`verifyPaparazziDebug`	Yes, including the finding that the `:app` module needed its own JVM to render at all (ADR-0082)
AI evaluation	`aiEval`	Yes — a categorisation regression against the frozen set, caught before merge
Migration safety	`unitTests`	Yes: a missing exported schema fixture, which is the assertion that stops every other check in the file passing vacuously
OSV dependency scan	CI	**Yes, and the gate was wrong:** it was querying versions nobody had. Fixed in `7dfeda3`
Offline smoke	`offlineSmoke`	Yes — and a check inside it that could not fail was found and removed (ADR-0067)
Restore drill	`restoreDrill`	It is the release gate; a red drill blocks the promotion, with no override
Static analysis	`sonarAnalysis`	**Yes, and the first run was vacuous** — it had silently dropped every Android module, reporting ~293 files against a true ~582 (ADR-0081)

Four of those twelve rows record a gate that was broken rather than a defect it caught. That is the uncomfortable finding of this project's testing history, and it is the reason §4's second principle is "a gate must be able to fail" rather than "a gate must exist". A gate nobody has seen go red is a hypothesis.

---

5. Test cases

Type — U unit · G golden · P property · R repository/migration · V ViewModel · C Compose UI · S screenshot · D drift · E AI evaluation · I instrumented · L lint. **Edge** names the negative, boundary or failure case the row also covers.

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-001	Onboarding completes and leaves a usable app	C	FR-001, AC-001	Run the four steps; a profile, a security method and one account exist, and the dashboard opens	Keystore failure leaves no profile row	MUST	`OnboardingFlowTest`
TC-002	Quick setup writes real rows	U, R	FR-002, AC-002	Enter income, rent, savings; budget and recurring-rule rows exist afterwards	Savings exceeding income is refused	MUST	`QuickSetupPropertyTest`
TC-003	The SMS step is skippable	C	FR-003, AC-003	Skip it; onboarding completes, no permission is requested, the app is fully usable	Granting then revoking deletes drafts	MUST	`OnboardingConsentTest`
TC-004	Demo mode is isolated and removable	R	FR-004, AC-004	Enter demo mode, then leave; every demo row is gone and no real row is touched	An interrupted wipe is re-detected	SHOULD	`DemoModeRepositoryTest`
TC-005	The default onboarding path has no blocking I/O	C	FR-005, AC-005	Walk the default path; every step is reachable without waiting on a network or a disk scan	The three-minute figure is **not** asserted — nothing measures it (`GAP-004`)	SHOULD	`OnboardingFlowTest`
TC-006	The dashboard shows all seven elements	S, V	FR-006, AC-006	Load a populated profile; net worth, Safe-to-Spend, month-to-date, 14-day obligations, three insights and goal chips are present	An empty profile states what it needs instead of showing zeros	MUST	`DashboardViewModelTest`, `DashboardScreenshotTest`
TC-007	Every dashboard figure routes somewhere	C	FR-007, AC-007	Tap each figure; the explaining screen opens by a typed route	A route with a missing argument fails to compile, not at runtime	MUST	`CfoNavHostTest`
TC-008	The privacy blur masks everything, everywhere	S	FR-008, AC-008	Toggle the blur; every amount on every screen is a fixed-width mask and `FLAG_SECURE` is set	It fails **closed** when the stored state cannot be read	MUST	`PrivacyBlurTest`, every `*ScreenshotTest`
TC-009	The dashboard renders with no network	I	FR-009, AC-009	Airplane mode; the dashboard draws fully from local data	A cached market figure keeps its staleness label	MUST	`OfflineSmokeTest`
TC-010	There is no dashboard edit mode	—	FR-010, AC-010	Not built; recorded rather than tested (`OQ-007`)	—	SHOULD	Not applicable
TC-011	The widget renders while the app is locked	U, S	FR-011, AC-011	Lock the app; the widget still shows its last figures from Glance state	It never touches Room, which would throw	SHOULD	`CfoWidgetContentTest`
TC-012	Pull-to-refresh degrades rather than blanks	V	FR-012, AC-012	Refresh offline; insights recompute and the market figure keeps "as of <date>"	Stale cache is labelled, never silently refreshed	MUST	`DashboardViewModelTest`
TC-013	A transaction round-trips every field	R	FR-013, AC-013	Save with every field set, reload; all of them return	Null-hostile domain types reject a missing required field	MUST	`TransactionRepositoryTest`
TC-014	A common expense takes three taps	C	FR-014, AC-014	Amount, suggested category, save	A rejected suggestion still saves in three	MUST	`AddTransactionFabTest`, `AddTransactionFlowTest`
TC-015	A transfer is one record with two legs	R, P	FR-015, AC-015	Save a transfer, delete one leg; both vanish and both balances return	Source equal to destination is refused	MUST	`TransferTest`, `AccountBalancePropertyTest`
TC-016	Split lines always sum to the parent	P	FR-016, AC-016	Generate splits over random amounts and line counts; the sum identity holds every time	A three-way split of an indivisible amount loses no paise	MUST	`MoneySplitPropertyTest`
TC-017	Classification runs and a correction sticks	U, E	FR-017, AC-017, FR-060	Save at a known merchant; a category, a confidence and evidence appear; correct it and the correction is reused	An unknown merchant falls to a lower tier with a lower confidence	MUST	`ClassificationEvalTest`, `CategorySuggestionTest`
TC-018	Recurring detection proposes, never creates	P	FR-018, AC-018	Two matching transactions; a proposal appears and no rule exists until confirmed	A varying amount reports a range	MUST	`RecurringPropertyTest`
TC-019	Search, filter, group and page	V	FR-019, AC-019	Over 10,000 rows: search, combine filters, scroll; day groups carry totals	An empty result names the excluding filter	MUST	`TransactionsViewModelTest`
TC-020	Bulk edit is undoable and soft	R	FR-020, AC-020	Multi-select and delete; rows leave every view, undo restores them, nothing is hard-deleted	The seven-day compactor does not exist (`GAP-006`)	SHOULD	`BulkEditTest`
TC-021	Every transaction records its source	R	FR-021, AC-021	Create one from each path; `source` is correct in each	The reserved `aa` value is readable and unused	MUST	`AccountAggregatorReadinessTest`
TC-022	A future-dated row is out of actuals and in the forecast	U, R	FR-022, AC-022	Date a transaction ahead; month-to-date excludes it and the forecast includes it	On its day it counts even if the worker never ran	MUST	`AddTransactionScheduleTest`
TC-023	Camera and gallery both reach the parser	C	FR-023, AC-023	Capture, then import; both produce text for extraction	A denied camera permission leaves gallery import working	MUST	`ReceiptReviewScreenTest`

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-024	OCR opens no socket	I	FR-024, AC-024	Scan in airplane mode; text is recognised	—	MUST	`OfflineSmokeTest`
TC-025	Extraction yields total, date, merchant and GST with confidences	U, E	FR-025, AC-025	Run the frozen receipt set; the total clears the $\geq$ 95% floor	A stored GST of zero is distinguished from absent (ADR-0077)	MUST	`ReceiptEvalTest`
TC-026	Review refuses to save without an amount and a date	C	FR-026, AC-026, VAL-010	Clear the amount; save is blocked and the field is flagged	A low-confidence field is visually flagged but still saveable	MUST	`ReceiptReviewScreenTest`
TC-027	The image is encrypted, linked, and separately deletable	R	FR-027, AC-027	Save with an image, then delete the image; the transaction survives	An orphaned blob is a known defect (`RISK-009`)	MUST	`AttachmentRepositoryTest`
TC-028	The duplicate guard offers a merge	U	FR-028, AC-028	A Rs. 500 manual row today, a Rs. 502 receipt yesterday; a merge is offered	$\pm$1% and $\pm$1 day are the boundaries, tested on both sides	MUST	`ReceiptDuplicateTest`
TC-029	One budget per category per month	R	FR-029, AC-029, VAL-011	Create, then create again; the second is refused	Rollover is not built (`GAP-008`)	MUST	`BudgetRepositoryTest`
TC-030	Suggestions are a median with seasonality	G	FR-030, AC-030	Run against three months of fixture history; the frozen figures reproduce	Under three months, the suggestion is withheld with a reason	MUST	`BudgetGoldenTest`
TC-031	Spent, remaining, pace and projection	G, V	FR-031, AC-031	Mid-month fixture; all four figures match the frozen expectations	Day 1 and the last day are both tested	MUST	`BudgetGoldenTest`, `BudgetsViewModelTest`
TC-032	Alerts fire once per band per month	U, R	FR-032, AC-032, BR-019	Cross 80%, then 100%; exactly one alert each, claimed	A worker skipped by Doze makes the alert late, not double	MUST	`BudgetAlertWorkerTest`
TC-033	One review per closed month	R	FR-033, AC-033	Close a month and run twice; one card exists	A month with no spend says so	SHOULD	`BudgetReviewTest`
TC-034	The emergency fund is always present	R	FR-034, AC-034	A new profile; the system goal exists and the other seven types are creatable	It cannot be deleted	MUST	`GoalRepositoryTest`
TC-035	Required monthly, with no assumed return	G	FR-035, AC-035	Target and date; the frozen figure reproduces and no rate is applied	An open-ended goal reports an ETA instead	MUST	`GoalGoldenTest`
TC-036	Feasibility names the gap and three levers	G	FR-036, AC-036	A goal beyond the surplus; gap and levers match the frozen output	A negative surplus points at the budget, not the goal	MUST	`GoalWaterfallGoldenTest`
TC-037	Evidenced progress follows the transaction	R	FR-037, AC-037, BR-027	Link a transaction, then delete it; progress rises then falls	Declared progress is stored and shown separately, never summed	MUST	`GoalContributionRepository Test`
TC-038	The waterfall splits a short surplus by priority	P	FR-038, AC-038	Generate competing goals; the split never exceeds the surplus and the shortfall is named	Re-prioritising recomputes deterministically	SHOULD	`GoalWaterfallPropertyTest`
TC-039	All eleven account types	R, C	FR-039, AC-039	Create one of each; each lands in its section	An unknown type from a newer archive renders as unrecognised	MUST	`AccountRepositoryTest`, `AccountsFlowTest`
TC-040	A card cycle across a month boundary	G, P	FR-040, AC-040	Statement day 25; cycle, unbilled, utilisation and due date are right	Day 31 in a 30-day month resolves to the last day	MUST	`BillingCycleTest`, `CardGoldenTest`
TC-041	An amortisation schedule is stable and exact	G, P	FR-041, AC-041	Read twice; identical to the paise, and the payoff date matches the last row	An EMI below the monthly interest reports that the loan never closes	MUST	`LoanGoldenTest`, `LoanEnginePropertyTest`
TC-042	A valuation uses the observed price and its date	U	FR-042, AC-042	Price a gold holding from cache; the label is the observation date	No price $\rightarrow$ coverage reported, **never** Rs. 0	MUST	`MarketPriceRepositoryTest`
TC-043	Net worth equals assets minus liabilities, daily	P	FR-043, AC-043	Generate ledgers; the identity holds, and a back-dated entry repairs the series	Two runs on one date update rather than duplicate	MUST	`NetWorthPropertyTest`
TC-044	Reconciliation posts an adjustment	R	FR-044, AC-044	Enter a differing real balance; an adjustment transaction appears and no stored balance is edited	A zero difference posts nothing	MUST	`AccountReconciliationTest`
TC-045	An archived account keeps history and leaves totals	R	FR-045, AC-045	Archive it; history reads, active totals drop it	`archived_at` and `deleted_at` are independent	MUST	`AccountRepositoryTest`

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-046	XIRR is money-weighted and reproducible	G, P	FR-046, AC-046	Frozen lots; the rate reproduces exactly, including across versions	Same-sign cash flows report that no rate exists	MUST	`InvestmentGoldenTest`, `InvestmentEnginePropertyTest`
TC-047	Allocation cites the rule that fired	G, P	FR-047, AC-047, BR-021	An over-weight portfolio; the breach names the rule id and version	Partial coverage is reported as coverage, not as zero	MUST	`AllocationGoldenTest`, `AllocationPropertyTest`
TC-048	A vehicle records profile, odometer, service and renewals	R	FR-048, AC-048	Record each; all persist	A lower odometer reading is refused (VAL-016)	MUST	`VehicleRepositoryTest`
TC-049	Mileage anomaly detection	—	FR-049, AC-049	Not testable: the fuel log it depends on is not built (`GAP-009`)	—	MUST	Not applicable
TC-050	Monthly cost of ownership, minus fuel	G	FR-050, AC-050	Service and renewals present; the figure matches the frozen output and does not imply a fuel component	—	SHOULD	`VehicleGoldenTest`
TC-051	The advisor hub lists all five entries	S	FR-051, AC-051	Open it; score, feed, advisor, simulators and chat are present	A sparse profile still renders	MUST	`AdvisorScreenshotTest`, `AdvisorScreenTest`
TC-052	Every insight card carries its four parts	G	FR-052, AC-052	Render each insight type; finding, evidence, confidence and at most one action	Two actions would fail the contract test	MUST	`InsightGoldenTest`
TC-053	One verdict, with a stored gate-by-gate trace	G	FR-053, AC-053	Run the advisor; the verdict and every gate row reproduce from the fixture	A gate with a missing input reports "cannot assess"	MUST	`PurchaseGoldenTest`
TC-054	Chat refuses rather than invents	E, G	FR-054, AC-054	Ask what no tool can answer; it declines	Measured against the frozen guardrail set	MUST	`ChatGoldenTest`, `GuardrailEvalTest`
TC-055	The disclaimer is on every advisor screen	S	FR-055, AC-055	Render each; the footer is present	Dark mode and 200% font both keep it visible	MUST	`AdvisorScreenshotTest`
TC-056	Settings lists every group it has	S	FR-056, AC-056	Open settings; each group renders	The recurring-rules manager is absent (`GAP-010`)	MUST	`SettingsScreenshotTest`
TC-057	The archive round-trips losslessly, field by field	R	FR-057, AC-057	Export, import into a fresh database, compare; every field returns	A renamed Kotlin property **fails the build**, because the archive format is a contract with files on users' phones	MUST	`ArchiveFormatTest`, `ArchiveRepositoryTest`
TC-058	Erase destroys the key first	U, R	FR-058, AC-058, BR-015	Run erase; the key goes, then files, then a report of what was unreachable	Biometric failure stops before the key	MUST	`EraseRepositoryTest`, `AndroidSecureEraserTest`
TC-059	A revoke stops the dependent path at once	R	FR-059, AC-059, BR-028	Revoke SMS and market data; drafts are deleted, the cached price is kept	Both grant and revoke timestamps are on the record	MUST	`ConsentsViewModelTest`, `ArchiveConsentRecordTest`
TC-060	Stage 1 classification clears its accuracy floor	E, D	FR-060, AC-060	Run the frozen labelled set; accuracy $\geq 92\%$	The KB and its typed mirror may not disagree	MUST	`ClassificationEvalTest`, `ClassificationKbDriftTest`
TC-061	Stream classification is stable	G, D	FR-061, AC-061	Frozen history; FIXED / SEMI_FIXED / VARIABLE reproduce	Too little history leaves it unclassified	MUST	`StreamGoldenTest`, `StreamKbDriftTest`
TC-062	Nature classification, and only the override is stored	G, D	FR-062, AC-062	Classify from account type; an override persists and an inference does not	The understatement is asserted, not hidden (ADR-0016)	MUST	`NatureGoldenTest`, `NatureKbDriftTest`
TC-063	The forecast's bands and crunch days reproduce	G, P	FR-063, AC-063	Frozen ledger; P10/P50/P90 and every crunch day match	Monotonic identities hold over generated inputs	MUST	`ForecastGoldenTest`, `ForecastPropertyTest`
TC-064	Seasonality uses own history first	G, P	FR-064, AC-064	Two years of one category; the index reproduces with the lookback divided out	A shared prior is only a fallback	MUST	`SeasonalityGoldenTest`, `SeasonalityPropertyTest`
TC-065	The emergency fund uses a median, not a mean	G, P	FR-065, AC-065, BR-022	Six months with one outlier; the target follows the median	The three absent multiplier terms are **asserted absent**	MUST	`EmergencyFundGoldenTest`, `EmergencyFundPropertyTest`
TC-066	Investment intelligence cites its rulebook	G, D	FR-066, AC-066	Run on a fixture portfolio; findings cite rule ids and versions	A drift test guards the mirror	MUST	`AllocationGoldenTest`, `MarketKbDriftTest`
TC-067	Vehicle prediction never invents a price	G, P	FR-067, AC-067	Frozen odometer trend; date and range reproduce	With no KB price it declines to quote	MUST	`VehicleGoldenTest`, `VehiclePropertyTest`

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-068	Appliance service, consumables, warranty, running cost	G, D	FR-068, AC-068	Frozen appliance; all four reproduce	An unknown class uses a labelled generic interval	SHOULD	`ApplianceGoldenTest`, `ApplianceKbDriftTest`
TC-069	The purchase gates run in order	G, P	FR-069, AC-069	Frozen inputs; the gate sequence and verdict reproduce	An abstaining gate is named in the verdict	MUST	`PurchaseGoldenTest`, `PurchasePropertyTest`
TC-070	The buy-list ladder removes nothing	G, P	FR-070, AC-070	Score wishes, re-run; order changes, nothing disappears	An abandoned interview keeps its answers	SHOULD	`InterviewGoldenTest`, `InterviewPropertyTest`
TC-071	The score reweights over present signals	G, P	FR-071, AC-071	A sparse profile; the score reweights and names what it could not read	One signal alone withholds the band	MUST	`HealthGoldenTest`, `HealthPropertyTest`
TC-072	Goal maths: required monthly, ETA, horizon	G, P	FR-072, AC-072	Frozen targets; all three reproduce	A past target date is refused (VAL-012)	MUST	`GoalGoldenTest`, `GoalPropertyTest`
TC-073	External data degrades to a labelled cache	U	FR-073, AC-073	No host configured; every priced screen uses cache with its date	No socket is opened at all	MUST	`MarketPriceRepositoryTest`
TC-074	The notification gate holds caps and quiet hours	G, P	FR-074, AC-074	Several candidates inside quiet hours; none is delivered	A critical crunch alert outranks the cap	MUST	`NotificationGoldenTest`, `NotificationPropertyTest`
TC-075	The feed deduplicates and ranks reproducibly	G, P	FR-075, AC-075	Duplicate findings; one card, and a stable order	A dismissed insight returns only with a new fingerprint	MUST	`InsightGoldenTest`, `InsightPropertyTest`
TC-076	The guardrail refuses an unverified figure	E, G, P	FR-076, AC-076, BR-011, BR-012	Feed sentences containing figures no engine produced; each is refused, measured against a frozen set with an absolute threshold	Rounding stops at the rupee; a rendering outside the allowlist is refused	MUST	`GuardrailEvalTest`, `GuardrailGoldenTest`, `GuardrailPropertyTest`
TC-077	Chat passes words, never rows	G, D, E	FR-077, AC-077	Plan a question; only registry tools run and the model receives rendered words	The registry and its mirror may not drift	MUST	`ChatGoldenTest`, `ChatRegistryDriftTest`, `VerbaliserEvalTest`
TC-078	Signals say what they cannot measure	G, P	FR-078, AC-078	Short history; "not measurable yet" rather than a score	A measured hit rate accompanies every verdict	MUST	`MarketGoldenTest`, `MarketPropertyTest`
TC-079	The eight stages account for the whole surplus	G, P	FR-079, AC-079, BR-023	Generate surpluses; the stages' consumption sums to the input	A missing FOO data file refuses to run rather than hardcoding	MUST	`OrderOfOperationsGoldenTest`, `OrderOfOperationsPropertyTest`
TC-080	Simulators share the loan engine's arithmetic	G, P	FR-080, AC-080	Compare branches; both match the loan screen to the paise	With no loan, the simulator asks for one	MUST	`SimulatorGoldenTest`, `SimulatorPropertyTest`
TC-081	The tax estimate names what it left out	G, D	FR-081, AC-081	Run both regimes; the exclusions list is part of the output	A missing input excludes that component explicitly	MUST	`TaxGoldenTest`, `TaxKbDriftTest`
TC-082	Cover gaps take the larger reading, and never say surrender	G, D	FR-082, AC-082	Frozen declarations; the gap is the larger of two readings	No output recommends surrendering a policy	MUST	`ProtectionGoldenTest`, `ProtectionRulebookDriftTest`
TC-083	Safe-to-Spend labels every withheld line	G, P	FR-083, AC-083	Frozen month; the figure and every withheld line reproduce, buffer included	No declared income states the missing input instead of Rs. 0	MUST	`SafeToSpendGoldenTest`, `SafeToSpendPropertyTest`
TC-084	The lock holds, and the lockout doubles	U, V	FR-084, AC-084	Five wrong PINs; a 30 s lockout that doubles, and no data path opens	The disable/unlock race that once crashed the app is covered (issue 2.8)	MUST	`AppLockViewModelTest`, `AppLockStoreTest`
TC-085	The key is hardware-wrapped and rotatable	U, R	FR-085, AC-085	Generate, rotate; the database survives and the key is never written unwrapped	Biometric re-enrolment rotates rather than orphaning the data	MUST	`KeystoreMasterKeyTest`, `DatabaseRekeyerTest`
TC-086	Every screen is capture-guarded	S, U	FR-086, AC-086	Render each screen; the secure flag is set	A new screen without the guard fails the sweep	MUST	`SecureWindowTest`, every `*ScreenshotTest`
TC-087	A backup restores, and fails closed on a wrong passphrase	R	FR-087, AC-087	Back up, restore; every row returns. Wrong passphrase → nothing written	Restoring into another profile is refused	MUST	`BackupRestoreTest`, `BackupCipherTest`
TC-088	The drill blocks a release when restore breaks	I	FR-088, AC-088	Run `restoreDrill`; a broken restore goes red	It runs on a device, not on the JVM	MUST	`BackupRestoreDrillTest`
TC-089	Security events are logged and not exported	R	FR-089, AC-089	Trigger events; rows land in `audit_log` and the archive does not carry them	The table has no `profile_id` to scope by	MUST	`AuditLogRepositoryTest`, `ArchiveFormatTest`

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-090	Four languages, proved	U, S	FR-090, AC-090	Render in each locale; every string is translated	A missing translation fails the build	SHOULD	`AppLanguageTest`
TC-091	Household aggregation leaks nothing	R	FR-091, AC-091, BR-006	Two profiles; an aggregate composes scoped reads	A query without its profile scope fails the build	SHOULD	`ProfileScopingTest`, `HouseholdTest`
TC-092	The business book is a second profile	R, D	FR-092, AC-092	Use it; no transaction carries a business flag	A drift test holds the shape	SHOULD	`BusinessModeDriftTest`
TC-093	A wiped import batch is verified, not trusted	R	FR-093, AC-093	Import, then wipe; every created row is gone and the check proves it	An interrupted import leaves no orphans	MUST	`ImportBatchRepositoryTest`
TC-094	Posting does not depend on the worker	U, R	FR-094, AC-094, BR-010	Skip the worker for a week; the row still counts on its date	A future-dated transfer posts neither leg early	MUST	`ScheduledTransactionWorkerTest`
TC-095	The integrity job detects a drifted balance	R	FR-095, AC-095, BR-007	Tamper with the cached balance; the job raises an adjustment prompt	It proposes, never applies silently	MUST	`BalanceIntegrityWorkerTest`
TC-096	Cold-start timing	—	NFR-001, AC-096	Not automated (`GAP-004`); measured by hand when it is measured at all	—	MUST	Not applicable
TC-097	Save latency	—	NFR-002, AC-097	Not automated (`GAP-004`). The commit path is local and transactional, which is tested; the 100 ms bound is not	—	MUST	Not applicable
TC-098	Recompute over 50k transactions	—	NFR-003, AC-098	Not automated (`GAP-004`). Work is off the main thread by construction	—	MUST	Not applicable
TC-099	Crash-free sessions	—	NFR-004, AC-099	Not measurable before launch (`OQ-005`)	—	MUST	Not applicable
TC-100	APK size is under budget	I	NFR-005, AC-100	Assemble release; the artefact is measured in CI	—	MUST	CI release job
TC-101	No unstructured background work	L	NFR-006, AC-101	Introduce `GlobalScope`; the build fails	Every job is WorkManager and claim-keyed	MUST	`CfoLintDetectorsTest`
TC-102	No data loss on process death	R	NFR-007, AC-102	Kill mid-write; the database reopens consistent	WAL plus transactional writes	MUST	`MigrationSafetyTest`, repository tests
TC-103	It runs on API 26	I	NFR-008, AC-103	Instrumented run at `minSdk`; core flows work	—	MUST	CI instrumented job
TC-104	Every core flow works offline, on a device	I	NFR-009, AC-104	Airplane mode enabled by the test; the end-to-end smoke passes	A check that could not fail was removed when found (ADR-0067)	MUST	`OfflineSmokeTest`
TC-105	Dark mode and 200% font do not break a layout	S	NFR-010, AC-105	Render all 115 baselines in both themes and at 200%	A full TalkBack pass remains manual (`GAP-012`)	MUST	All `*ScreenshotTest`
TC-106	A hardcoded UI string fails the build	L	NFR-011, AC-106	Add one in `:feature:*`; lint fails	`@Preview` sample data is exempt	MUST	`CfoLintDetectorsTest`
TC-107	A floating-point money declaration fails the build	L	NFR-012, AC-107	Declare `val totalAmount: Double`; lint fails	Detection is by **name** (ADR-0001), with its word list on record	MUST	`CfoLintDetectorsTest`
TC-108	Coverage floors actually fail	U	NFR-013, AC-108	Delete a money test; `koverVerify` goes red below 100% on `:core:model`	Proved by raising the bound to an impossible 101%	MUST	`koverVerify` configuration
TC-109	A new static-analysis warning fails the build	L	NFR-014, AC-109	Introduce one; `unitTests` fails	`:lint`'s own 14 tests are inside `unitTests`	MUST	`CfoLintDetectorsTest`, `CfoHandRolledCryptoDetectorTest`
TC-110	The dependency scan reads shipped versions	U	NFR-015, AC-110	Run the OSV scan; it queries resolved coordinates	An earlier version queried versions nobody had — fixed	MUST	`test_osv_scan.py`
TC-111	The release build is hardened, read from the DEX	I	NFR-016, AC-111	Assemble release; the log strip is verified by reading the shipped DEX	Config alone is not accepted as evidence	MUST	`verify_release_log_stripping.py`
TC-112	The restore drill gates the release	I	NFR-017, AC-112	Run it on an emulator; red blocks promotion	—	MUST	`BackupRestoreDrillTest`
TC-113	A log line naming money fails the build	L	NFR-018, AC-113	Log an amount; lint fails	Personal data is covered by the same detector	MUST	`CfoLintDetectorsTest`

TC	Title	Type	Covers	Steps and expected result	Edge	Priority	Evidence in the tree
TC-114	An old insight still renders	R	NFR-019, AC-114	Read a row written by an earlier engine version; it renders and its version explains it	`evidence_json` is versioned `{"v":1,...}`	MUST	`InsightRepositoryTest`
TC-115	A changed screen fails until re-recorded	S	NFR-020, AC-115	Change a composable; `verifyPaparazziDebug` fails	Recording is never automatic (ADR-0066)	MUST	All `*ScreenshotTest`
TC-116	No query spans profiles	R	BR-006, FR-091	Read every DAO query; each is profile-scoped	The test reads the queries rather than trusting review	MUST	`ProfileScopingTest`
TC-117	A typed mirror may not drift from its data row	D	BR-018, FR-060	Edit a threshold in `ai/`; the drift test fails	The data files are declared Gradle **inputs**, so the gate cannot be skipped	MUST	`RulebookDriftTest` and 28 siblings
TC-118	A migration loses nothing	R	BR-017, FR-093	Migrate every version from a real fixture; data survives each step	A destructive migration fails review and the test	MUST	`MigrationSafetyTest`
TC-119	Engines import no Android	U	BR-013, NFR-019	Run the convention-plugin guard; an Android import in `:domain:*` fails	This is also what made the KMP spike possible	MUST	`Arc002GuardTest`, `Arc002Test`
TC-120	The engine registry matches the engines that exist	D	BR-013, FR-075	Compare the registry with the modules; a new engine must be registered	Reconciled once and now checked (ADR-0076)	MUST	`EngineRegistryDriftTest`
TC-121	The golden harness cannot be vacuous	G	BR-013, FR-063	Empty a fixture; the harness fails rather than passing trivially, and never rewrites it	This is the audit G-01 lesson applied to goldens	MUST	`GoldenFixtureTest`, `GoldenSnapshotTest`
TC-122	An eval run reports what it measured	E	NFR-019, FR-060	Run any evaluation; the dataset version is in the report	A floor quoted without its dataset is not evidence	MUST	`EvalDatasetTest`, `EvalReportTest`
TC-123	The DPDP matrix stays aligned with the app	D	FR-059, NFR-015	Change a consent; the compliance drift test fails	Compliance is a checked document, not a claim	MUST	`DpdpComplianceDriftTest`
TC-124	Hand-rolled crypto fails the build	L	BR-015, FR-085	Add a cipher construction outside Tink; lint fails	Two exemptions exist and are argued (ADR-0063)	MUST	`Sec003AuditDriftTest`, `CfoHandRolledCrypto DetectorTest`

6. Non-functional test approach, per category

Category	Approach	Status
Performance	No automated benchmark exists. The architecture keeps work off the main thread and reads from local storage, which is tested; the three numeric bounds in NFR-001…003 are **not measured**	`GAP-004` — the most significant testing gap in this blueprint
Security	Six lint detectors, a crypto-audit drift test, an OSV scan over resolved coordinates, a DEX-level log-strip check, and a DPDP matrix drift test	Covered
Privacy	`CfoPiiInLogs`, the consent-record tests, and the archive tests that assert what the export does **not** carry	Covered
Accessibility	115 renders in light, dark and 200% font; semantic tokens rather than raw colours	Partial — TalkBack is a manual pass (`GAP-012`)
Reliability	Migration safety across all 32 steps, the restore drill on a device, and claim-keyed workers	Covered
Offline	An end-to-end smoke with airplane mode turned on by the test	Covered
Determinism	Golden files per engine version, property tests for identities, and a harness that refuses to rewrite its own fixtures	Covered
AI accuracy	Frozen, versioned datasets with absolute floors: categorisation $\geq 92\%$, receipts $\geq 95\%$, and the guardrail's own threshold	Covered, except forecast MAPE (`GAP-003`)

7. Requirements with no test

Requirement	Why	Record
FR-010 — dashboard edit mode	Not built	`OQ-007`
FR-049 — mileage anomaly	Depends on the unbuilt fuel log	`GAP-009`
NFR-001, NFR-002, NFR-003	No performance benchmark exists	`GAP-004`
NFR-004 — crash-free rate	Not observable before launch	`OQ-005`
Forecast accuracy (MAPE, part of FR-063's claim)	Accuracy tracking is deferred; the forecast itself is tested	`GAP-003`

Every other `FR` and `NFR` has at least one `TC` above. The count is recomputed in 10 §4.

8. Tests with no requirement

Test	Justification
`Arc002GuardTest`, `Arc002Test`	Architecture enforcement. Traced to BR-013 and NFR-019 rather than to a feature
`EngineRegistryDriftTest`	Internal consistency of the engine registry (ADR-0076). Infrastructure, no user-facing requirement expected
`GoldenFixtureTest`, `GoldenSnapshotTest`	The test harness testing itself, which is exactly the lesson of a gate that could not fail
`EvalDatasetTest`, `EvalReportTest`	The evaluation harness's own correctness
`test_osv_scan.py`, `test_release_metadata.py`, `verify_release_metadata.py`, `test_doc_builders.py`	Python tooling tests. Build and release infrastructure
`AccountAggregatorApiTest`, `AccountAggregatorReadinessTest`	Guard the **reserved** `aa` source value and the unconfigured client, so an unbuilt integration cannot rot (ADR-0074)
`test_install_llm_model.py`, `ModelManifestDriftTest`	Model provisioning and its manifest (ADR-0079)

None of these is an orphan to be removed. Each is infrastructure whose parent is a binding rule rather than a feature, which is recorded here instead of being left to look like an accident.

Part 08 — Development and Delivery Plan

← Master Blueprint · Related: 02 Requirements · 07 Tests · 09 Risks & Open Questions

> ***This is a retrospective plan as much as a forward one.*** *The product is at* `v0.13.10` *with Epics 1–13 delivered, so the epics and stories below are read off* `docs/issues/` *(84 issue files, each with a tracker) and* `CHANGELOG.md` *rather than estimated in advance. The forward-looking half is §5: the 25 tasks that are actually open, each tied to the gap or question that opened it.*

1. Delivery approach

Aspect	How it works here
Shape	One epic at a time, one issue at a time, each issue closed only against the Definition of Done
Branching	GitFlow-lite: `feature/<id-dashes>-<slug>` → `dev` → `stage` → `main`. `main` and `stage` are protected
Commits	Conventional commits, and **every commit references its requirement id**. A plausible-looking id that does not exist is the failure mode this rule has to survive, which is why the ids in 02 map to the SRS's
Issue documents	Every issue has a specification file **and** a tracker file in `docs/issues/`, checked by `scripts/check_issue_docs.py`
Decisions	A deviation from the SRS needs an ADR; a dependency change needs a `DECISIONS.md` row naming what it was chosen over; a changed call path needs `FLOW.md` updated in the same commit
Sessions	One file per working session in `docs/sessions/`, holding the reasoning that the index files only point at
Feature flags	Incomplete work ships behind a constant — household mode is off this way today (ADR-0069)

2. Milestones

The SRS's six phases (§26), with what actually closed each one.

MS	Phase	Scope	Exit criteria	Status
MS-001	Phase 0 — Foundation	The module graph, `Money`, the injected `Clock`, `Result`, custom lint, the encrypted database, the migration harness, the design system, DataStore, the app shell	Add-transaction in three taps; an airplane-mode pass; the migration harness live	**Delivered** — EPIC-001, EPIC-002
MS-002	Phase 1 — Core finance	Categories, auto-categorisation, recurring detection, budgets and alerts, dashboard v1, search and filters, export/import, the widget	Classification clears its floor; budget alerts fire once per band; the export round-trips	**Delivered** — EPIC-003, EPIC-004, EPIC-005
MS-003	Phase 2 — Wealth and goals	All account types, loans and amortisation, investments and XIRR, valuation, goals, net-worth snapshots, E2EE backup	Net worth correct against fixtures; goal maths frozen in golden tests; restore on a fresh device	**Delivered** — EPIC-006, EPIC-007, EPIC-008
MS-004	Phase 3 — AI core	Stream and nature classification, forecasting and crunch alerts, seasonality, the emergency-fund coach, the health score, the orchestrator, notification policy, the guardrail	Forecast runs; OCR total ≥ 95%; the health score is explainable	**Delivered** — EPIC-009. One exit criterion is **not** met: forecast MAPE is not tracked (`GAP-003`)
MS-005	Phase 4 — Advisor	The purchase advisor, simulators, vehicle prediction, investment intelligence, the chat assistant on an on-device model, four languages	Verdict traces stored; the chat guardrail eval green; localisation proved	**Delivered** — EPIC-010, EPIC-011, EPIC-012
MS-006	Phase 5 — Expansion	Household mode, appliances, the protection suite, the tax engine, business mode, Account Aggregator, iOS through KMP	Per-feature addenda	**Partly delivered** — EPIC-013. Five of seven shipped; AA is designed and unbuilt (`OQ-001`), iOS is blocked (`OQ-002`)
MS-007	Public beta readiness	The open items in §5 that gate a public release: the proxy decision, the performance benchmark, Play submission for `READ_SMS`, crash reporting	A measured NFR-001…003; a Play submission decision on `READ_SMS`; `GAP-001` either closed or formally descoped	**Not started**

3. Epics

EPIC	Name	Issues	Requirements	Status
EPIC-001	Foundations — the graph, the money, the clock, the gates	1.1–1.10	BR-001… BR-004, NFR-012… NFR-014, FR-085	Delivered
EPIC-002	Onboarding, the app lock, accounts, net worth	2.1–2.9	FR-001… FR-005, FR-039… FR-045, FR-084	Delivered
EPIC-003	Transactions and capture	3.1–3.9	FR-013… FR-028	Delivered
EPIC-004	Categorisation and budgets	4.1–4.7	FR-029… FR-033, FR-060… FR-062	Delivered
EPIC-005	The dashboard, privacy, export, the widget	5.1–5.5	FR-006… FR-012, FR-057, FR-083	Delivered
EPIC-006	Wealth — cards, loans, investments, valuation	6.1–6.7	FR-040… FR-043, FR-046, FR-047, FR-066	Delivered, except 6.7 (the backend proxy) — `GAP-001`
EPIC-007	Goals, the emergency fund, the waterfall	7.1–7.5	FR-034… FR-038, FR-065, FR-072, FR-079	Delivered
EPIC-008	Backup, restore, and the drill	8.1–8.3	FR-087, FR-088, NFR-017	Delivered
EPIC-009	The AI core	9.1–9.7	FR-061… FR-064, FR-071, FR-074… FR-076	Delivered
EPIC-010	The advisor, the simulators, chat, languages	10.1–10.8	FR-053, FR-054, FR-067… FR-070, FR-077, FR-078, FR-080, FR-090	Delivered
EPIC-011	Privacy and security hardening	11.1–11.7	FR-058, FR-059, FR-085, FR-086, FR-089, NFR-015, NFR-016	Delivered
EPIC-012	Release quality — harnesses, evals, renders, the train	12.1–12.5	NFR-009, NFR-013, NFR-019, NFR-020	Delivered
EPIC-013	Expansion — household, appliances, protection, tax, business, AA, iOS	13.1–13.7	FR-068, FR-081, FR-082, FR-091, FR-092, FR-093	Partly delivered

4. User stories

One per issue file, in issue order. The **AC** column points at the acceptance criteria in 02 §5.

US	Issue	Epic	Story	AC
US-001	1.1	EPIC-001	As the maintainer, I want the §21.2 module graph buildable with a version catalog and CI, so that the dependency direction is a fact rather than an intention	AC-109
US-002	1.2	EPIC-001	As the maintainer, I want `Money` as `Long` paise with `HALF_EVEN` and exact splits, so that no amount can ever drift	AC-016, AC-107
US-003	1.3	EPIC-001	As the maintainer, I want an injected `Clock` and `DispatcherProvider`, so that calendar logic is testable and no engine reads a wall clock	AC-114
US-004	1.4	EPIC-001	As the maintainer, I want `Result<T, AppError>`, so that no exception crosses a layer boundary	AC-009
US-005	1.5	EPIC-001	As the maintainer, I want custom lint for money, `GlobalScope`, the clock and strings, so that the binding rules fail a build	AC-101, AC-106, AC-107, AC-113
US-006	1.6	EPIC-001	As the user, I want my data in an encrypted database whose key lives in the Keystore, so that losing the phone does not lose the data	AC-085
US-007	1.7	EPIC-001	As the maintainer, I want a migration-test harness, so that a schema change cannot silently lose a row	AC-102
US-008	1.8	EPIC-001	As the user, I want one coherent design system with theme tokens and charts, so that every screen reads the same way	AC-105
US-009	1.9	EPIC-001	As the user, I want settings and consents in Proto DataStore, so that a consent is a record rather than a boolean	AC-059
US-010	1.10	EPIC-001	As the user, I want one app shell with typed navigation, so that every figure can route to its explanation	AC-007
US-011	2.1	EPIC-002	As a new user, I want a four-step onboarding, so that the app is usable within minutes	AC-001, AC-003
US-012	2.2	EPIC-002	As the user, I want a biometric and PIN app lock, so that an unlocked phone is not an open ledger	AC-084
US-013	2.3	EPIC-002	As a new user, I want quick setup to seed my plan from three numbers, so that I do not start at an empty screen	AC-002
US-014	2.4	EPIC-002	As a prospective user, I want a demo mode, so that I can see the app work before trusting it with my money	AC-004

US	Issue	Epic	Story	AC
US-015	2.5	EPIC-002	As the user, I want all eleven account types, so that my whole balance sheet fits	AC-039
US-016	2.6	EPIC-002	As the user, I want net worth as assets minus liabilities, snapshotted daily, so that I can see the trend	AC-043
US-017	2.7	EPIC-002	As the user, I want to reconcile an account against reality, so that a difference is recorded rather than hidden	AC-044
US-018	2.8	EPIC-002	As the user, I want the lock not to crash when I disable it mid-unlock, so that a security control is not a liability	AC-084
US-019	2.9	EPIC-002	As the maintainer, I want repository errors to stay inside `Result`, so that a failed read renders an error state rather than a crash	AC-009
US-020	3.1	EPIC-003	As the user, I want to add a transaction in three taps, so that recording is not a chore I abandon	AC-014
US-021	3.2	EPIC-003	As the user, I want a transfer to be one record, so that two accounts cannot drift apart	AC-015
US-022	3.3	EPIC-003	As the user, I want to split a bill across categories, so that one payment can be two kinds of spending	AC-016
US-023	3.4	EPIC-003	As the user, I want to date something in the future, so that a known bill is in the forecast without being in this month's spend	AC-022
US-024	3.5	EPIC-003	As the user, I want to see where each entry came from, so that I can trust an automated one	AC-021
US-025	3.6	EPIC-003	As the user, I want search, filters and bulk edit, so that a year of history stays usable	AC-019, AC-020
US-026	3.7	EPIC-003	As the user, I want recurring series proposed, so that I confirm rather than retype	AC-018
US-027	3.8	EPIC-003	As the user, I want a receipt read by the phone, so that a paper bill becomes an entry without typing	AC-023, AC-025, AC-026
US-028	3.9	EPIC-003	As the user, I want bank alerts parsed on-device, with my explicit permission, so that UPI spending is captured without a server	AC-024, AC-059
US-029	4.1	EPIC-004	As the user, I want to edit my category taxonomy and merchant rules, so that the app learns my vocabulary	AC-017
US-030	4.2	EPIC-004	As the user, I want transactions categorised automatically, so that "where did it go" has an answer I did not have to type	AC-060
US-031	4.3	EPIC-004	As the user, I want each rupee labelled need, want, investment, asset or liability, so that the answer is better than a pie chart	AC-062
US-032	4.4	EPIC-004	As the user, I want budgets with suggestions, so that the first budget is not a blank form	AC-029, AC-030
US-033	4.5	EPIC-004	As the user, I want alerts at 80% and 100%, once each, so that I am warned without being nagged	AC-032
US-034	4.6	EPIC-004	As the user, I want a monthly review, so that a closed month teaches the next one	AC-033
US-035	4.7	EPIC-004	As the maintainer, I want the budget repository free of unchecked casts, so that an engine change cannot crash a screen	AC-009
US-036	5.1	EPIC-005	As the user, I want one dashboard answering how I am doing, what I can spend and what needs attention	AC-006
US-037	5.2	EPIC-005	As the user, I want a Safe-to-Spend figure with every withheld line shown, so that I can check it	AC-083
US-038	5.3	EPIC-005	As the user, I want one tap to hide every amount, so that the app is usable in public	AC-008
US-039	5.4	EPIC-005	As the user, I want a complete export that imports back losslessly, so that my data is actually mine	AC-057
US-040	5.5	EPIC-005	As the user, I want a home-screen widget, so that the one figure I check daily needs no app launch	AC-011
US-041	6.1	EPIC-006	As the user, I want my card's real cycle, utilisation and due date, so that I stop guessing the statement	AC-040
US-042	6.2	EPIC-006	As the user, I want my EMI split into principal and interest, so that I can see what the loan actually costs	AC-041
US-043	6.3	EPIC-006	As the user, I want holdings with dated lots and XIRR, so that my return is not understated by a cost-to-value ratio	AC-046
US-044	6.4	EPIC-006	As the user, I want allocation and concentration against a published rulebook, so that a warning cites a rule rather than an opinion	AC-047
US-045	6.5	EPIC-006	As the user, I want gold and crypto valued from market data when I allow it, so that net worth is current	AC-042
US-046	6.6	EPIC-006	As the user, I want a net-worth history that survives a back-dated entry, so that the chart is not a lie	AC-043
US-047	6.7	EPIC-006	As the maintainer, I want the §22 market proxy, so that prices come from one pinned host rather than from an exchange directly	AC-073
US-048	7.1	EPIC-007	As the user, I want a goal's required monthly contribution, so that a target becomes a plan	AC-035

US	Issue	Epic	Story	AC
US-049	7.2	EPIC-007	As the user, I want an emergency-fund target and runway, so that "am I safe" has a number	AC-065
US-050	7.3	EPIC-007	As the user, I want to know when my goals do not fit, so that I choose rather than silently fail	AC-036, AC-038
US-051	7.4	EPIC-007	As the user, I want progress evidenced by real transactions, so that it is not self-reported	AC-037
US-052	7.5	EPIC-007	As the user, I want the next rupee's destination decided in one order, so that priorities are not re-argued monthly	AC-079
US-053	8.1	EPIC-008	As the user, I want an end-to-end-encrypted backup, so that losing the phone does not lose the ledger	AC-087
US-054	8.2	EPIC-008	As the user, I want restore to work on a fresh device, so that the backup is worth having	AC-087
US-055	8.3	EPIC-008	As the maintainer, I want a restore drill gating every release, so that "it restores" is tested rather than believed	AC-088, AC-112
US-056	9.1	EPIC-009	As the user, I want to know which spending is fixed and which is discretionary, so that advice fits my actual obligations	AC-061
US-057	9.2	EPIC-009	As the user, I want a 90-day forecast with crunch days named, so that a shortfall is a warning rather than a surprise	AC-063
US-058	9.3	EPIC-009	As the user, I want seasonality from my own history, so that a festival month is not read as overspending	AC-064
US-059	9.4	EPIC-009	As the user, I want one score with its contributions and strongest lever, so that "am I doing well" is actionable	AC-071
US-060	9.5	EPIC-009	As the user, I want one ranked feed rather than many engines shouting, so that attention goes to what matters	AC-075
US-061	9.6	EPIC-009	As the user, I want a policy deciding what may interrupt me, so that I do not mute the app	AC-074
US-062	9.7	EPIC-009	As the user, I want a guarantee that no figure reaches me unverified, so that I can trust what I read	AC-076
US-063	10.1	EPIC-010	As the user, I want a purchase verdict with its whole trace, so that "not now" is checkable	AC-053
US-064	10.2	EPIC-010	As the user, I want a buy list that interviews me, so that wanting something is separated from buying it	AC-070
US-065	10.3	EPIC-010	As the user, I want prepay-versus-invest and debt-order simulators, so that a big decision is arithmetic rather than a feeling	AC-080
US-066	10.4	EPIC-010	As the user, I want my vehicle's next service predicted from my own usage, so that a bill is not a surprise	AC-067
US-067	10.5	EPIC-010	As the user, I want to ask about my money in words, so that I do not have to know which screen holds the answer	AC-077
US-068	10.6	EPIC-010	As the maintainer, I want the guardrail measured against a frozen set, so that its promise is evidenced	AC-076
US-069	10.7	EPIC-010	As the user, I want to know whether today is unusual, with a measured hit rate, so that a signal is not a claim	AC-078
US-070	10.8	EPIC-010	As the user, I want the app in my language, so that money advice is not in a second language	AC-090
US-071	11.1	EPIC-011	As the user, I want the database key in the strongest place the phone has, so that extracted storage is worthless	AC-085
US-072	11.2	EPIC-011	As the user, I want screenshots blocked on every screen, so that another app cannot read my balance	AC-086
US-073	11.3	EPIC-011	As the user, I want a consents dashboard with one-tap revoke, so that permission is mine to withdraw	AC-059
US-074	11.4	EPIC-011	As the user, I want erase to destroy the key first, so that deletion is real rather than a filesystem gesture	AC-058
US-075	11.5	EPIC-011	As the user, I want my DPDP rights served locally, so that exercising them needs nobody's cooperation	AC-057, AC-058
US-076	11.6	EPIC-011	As the maintainer, I want dependencies scanned and the release minified, so that a known vulnerability does not ship	AC-110, AC-111
US-077	11.7	EPIC-011	As the maintainer, I want hand-rolled crypto to fail the build, so that SEC-003 is enforced rather than remembered	AC-111
US-078	12.1	EPIC-012	As the maintainer, I want a golden-file and property harness, so that an engine's answer cannot change unnoticed	AC-114
US-079	12.2	EPIC-012	As the maintainer, I want frozen AI evaluation datasets with floors, so that accuracy is measured rather than asserted	AC-060
US-080	12.3	EPIC-012	As the maintainer, I want every screen under a screenshot test, so that a UI regression is a red build	AC-115
US-081	12.4	EPIC-012	As the maintainer, I want an airplane-mode end-to-end smoke on a device, so that "works offline" is proven	AC-104
US-082	12.5	EPIC-012	As the maintainer, I want the release train checked, so that version, code and changelog cannot drift	AC-100

US	Issue	Epic	Story	AC
US-083	13.1	EPIC-013	As a household, we want one app for two people without a shared diary, so that aggregation does not mean exposure	AC-091
US-084	13.2	EPIC-013	As the user, I want appliances treated like vehicles, so that a service and a running cost are visible	AC-068
US-085	13.3	EPIC-013	As the user, I want my cover gaps named, so that under-insurance is visible before it matters	AC-082
US-086	13.4	EPIC-013	As the user, I want both tax regimes compared honestly, so that I can see what the estimate left out	AC-081
US-087	13.5	EPIC-013	As a freelancer, I want a separate business book, so that business and personal do not mix	AC-092
US-088	13.6	EPIC-013	As the user, I want Account Aggregator import, so that capture stops depending on SMS	AC-021
US-089	13.7	EPIC-013	As the maintainer, I want the engines portable to iOS, so that the product is not Android-only forever	AC-107

5. Open tasks

Every row is tied to the gap, risk or question that opened it. **An unestimated task is fine; an unestimated task with no stated reason is not** — so the estimate column either holds a size or says why it cannot.

TASK	Title	Epic	Covers	Depends on	Estimate	Opened by
TASK-001	Decide the market-data proxy: build it, or formally descope it	EPIC-006	FR-042, FR-073, API-001	A hosting decision and a certificate-pinning story	**Unestimated** — the size depends entirely on the hosting decision, which is `OQ-003`	`GAP-001`
TASK-002	Account Aggregator integration	EPIC-013	FR-093, AC-021	TASK-001's host, AA sandbox access	**Unestimated** — gated on an external onboarding process nobody has started	`OQ-001`
TASK-003	Unblock the KMP port: make `Money` portable	EPIC-013	NFR-012, AC-107	A decision on whether exactness or portability yields	**Unestimated** — this is a design question, not a task	`OQ-002`
TASK-004	Delete orphaned receipt blobs	EPIC-003	FR-027, AC-027	—	Small — a sweep plus a test	`RISK-009`
TASK-005	A performance benchmark harness for NFR-001…003	EPIC-012	NFR-001, NFR-002, NFR-003	A reference device or a macrobenchmark setup	Medium	`GAP-004`
TASK-006	Forecast accuracy tracking (MAPE against actuals)	EPIC-009	FR-063, AC-063	A month of actuals per user; a storage shape	Medium	`GAP-003`
TASK-007	The fuel log, and the mileage-anomaly insight on top of it	EPIC-010	FR-048, FR-049, FR-050	—	Medium — one table, one screen, one engine rule	`GAP-009`
TASK-008	The seven-day trash compactor	EPIC-003	FR-020, AC-020	—	Small	`GAP-006`
TASK-009	Budget rollover of the unused amount	EPIC-004	FR-029, AC-029	—	Small	`GAP-008`
TASK-010	CSV export beside the JSON archive	EPIC-005	FR-057, AC-057	—	Small	`GAP-011`
TASK-011	A recurring-rules manager screen	EPIC-003	FR-056, AC-056	—	Small	`GAP-010`
TASK-012	"Apply to similar past transactions" after a correction	EPIC-004	FR-017, AC-017	—	Medium — the matching rule needs a frozen fixture	`GAP-005`
TASK-013	Receipt line items	EPIC-003	FR-025, AC-025	—	Medium — accuracy on Indian receipt layouts is the risk	`GAP-007`
TASK-014	A dashboard edit mode	EPIC-005	FR-010, AC-010	—	Small	`OQ-007`
TASK-015	A TalkBack audit pass across every screen	EPIC-012	NFR-010, AC-105	—	Medium — manual by nature	`GAP-012`
TASK-016	Health-score history, if it is wanted at all	EPIC-009	FR-071, AC-071	A decision: ADR-0045 chose live computation deliberately	Small, **but** it needs the ADR revisited first	`GAP-013`
TASK-017	The notification digest	EPIC-009	FR-074, AC-074	—	Small	`GAP-014`
TASK-018	Reconcile or supersede the SRS's promised `docs/traceability.csv`	EPIC-012	NFR-019	A decision: this blueprint's 10 may **be** the answer	Small	`GAP-002`

TASK	Title	Epic	Covers	Depends on	Estimate	Opened by
TASK-019	The external-data tables, when and if a proxy lands	EPIC-006	FR-073, API-003	TASK-001	Medium	`GAP-015`
TASK-020	The third classification tier	EPIC-004	FR-060, AC-060	—	Medium — the two shipped tiers already clear the floor, so this is precision, not correctness	ADR-0015
TASK-021	Split `Entities.kt` and `Daos.kt` to one type per file	EPIC-001	BR-017, NFR-013	A mechanical refactor with a provable no-op	Medium — 40 entities and 34 DAOs, and the archive format must not move	The 500-line rule in `CLAUDE.md`; see the architecture and gap report
TASK-022	Turn household mode on	EPIC-013	FR-091, AC-091	A household UX that does not imply shared visibility	Medium	ADR-0069
TASK-023	Multi-page receipt stitching	EPIC-003	FR-023, AC-023	—	Small — and it is a `MAY`, so it may stay unbuilt	SRS §5.4
TASK-024	Play submission readiness for `READ_SMS`	MS-007	FR-003, AC-003	A policy declaration and possibly a fallback capture path	**Unestimated** — the outcome is Google's, not ours	`CON-010`, ADR-0013
TASK-025	Crash reporting, so NFR-004 becomes measurable	MS-007	NFR-004, AC-099	A privacy decision: any reporter is a network path	**Unestimated** — it is a `P-01` question before it is a task	`OQ-005`

5.1 Dependencies and the critical path Three tasks gate the rest, and two of them are **decisions rather than work**:

```
OQ-003 (host the proxy?) -> TASK-001 -+-> TASK-019 (external-data tables)
                                      +-> TASK-002 (Account Aggregator)
                                      +-> FR-078's backtests become measurable
OQ-002 (exactness vs portability) -> TASK-003 -> iOS at all
CON-010 (Play policy on READ_SMS) -> TASK-024 -> a public release
```

Everything else in §5 is independent and small. The critical path to a **public beta** is `TASK-024` and `TASK-005`, not any feature.

---

6. Definition of Ready and Definition of Done

Ready — an issue may start when it has: a requirement id that exists in 02; at least one `AC`; a planned `TC`; and a specification file plus a tracker file in `docs/issues/`.

Done — all eight, every time (SRS §4.2, `CLAUDE.md` §8):

1. The requirement is implemented and traceable — the commit references its id.
2. Unit tests for the domain logic, run with `./gradlew unitTests`; $\geq$ 85% engine coverage and **100% on money maths**.
3. The UI state is covered by at least one Compose or screenshot test.
4. It works offline, verified by an airplane-mode case.
5. The accessibility scan passes, strings are externalised, dark mode is verified.
6. No new lint or detekt warning; CI green; reviewed, or self-reviewed against the checklist.
7. SonarQube is clean — run **before** the commit, with the denominator checked.
8. A session file is written in `docs/sessions/`, any new decision is indexed in `DECISIONS.md`, and any changed call path is reflected in `FLOW.md`.

Items 7 and 8 are **agent-followed, not build-enforced**, and are documented as such rather than implying a gate that does not exist. This project has already shipped one governance gate that nothing ever ran, which is why that distinction is written down.

---

7. Environments and CI/CD

Stage	What runs
Local, before a commit	`./gradlew unitTests`, then SonarQube to zero findings
Local, before a PR	`/pre-merge` — the Definition of Done as a checklist

Stage	What runs
CI on every push (`verify` job, JDK 21)	AI data-file validation · release-metadata check · the convention-plugin ARC-002 guard · ktlint, detekt, Android lint · unit tests with coverage (money must be 100%) · frozen AI evaluation sets · the chat guardrail eval · the Paparazzi screenshot diff · the OSV dependency scan · assemble debug · release hardening and the DEX log strip
CI on a release candidate (`restore-drill` job)	The backup restore drill plus the offline end-to-end smoke, on an emulator with KVM
Promotion	`feature` → `dev` → `stage` → `main`, by reviewed PR only. `main` and `stage` are protected

8. Release and rollback

Step	Detail
Version	SemVer in `VERSION`, with `versionCode` and `CHANGELOG.md` checked against each other by a task that has already caught two historical drifts (ADR-0068)
Build	R8-minified release, debuggable and backup off, log calls stripped and **verified by reading the shipped DEX**
Gate	The restore drill is red or green. Red blocks the release — there is no override
Rollback	Reinstall the previous APK. Migrations are additive and non-destructive, so the schema survives; a downgrade is nonetheless **not** a supported user path, and the supported recovery is restore-from-backup
Data migration or backfill	None pending. Every schema change to date has been additive, by rule (BR-017)

9. Post-launch support, maintenance and ownership

Aspect	Reality
Owner	One maintainer (`STK-003`). This is the project's largest structural risk, recorded as `RISK-001`
Support model	Self-serve by design: in-app help, the chat assistant, and a published source tree. There is no server to operate and no on-call rotation to staff
Maintenance load	Dependency updates with an OSV gate; Android API-level updates; the `ai/` knowledge bases, which need a refresh every Union Budget (tax) and whenever a rulebook threshold is revisited
Data ownership	The user's, entirely. There is no copy anywhere else, which is why backup and the drill are treated as release-blocking rather than as a feature
Documentation upkeep	`DECISIONS.md`, `FLOW.md`, `docs/sessions/` and this blueprint. The first two are indexes; the session files hold the reasoning

10. Data migration and backfill

There is no predecessor system to migrate from. This is greenfield: the first user of a build is a fresh install with no profile. So the usual migration section — export, transform, reconcile, cut over — has nothing to describe.

What it *does* have is three kinds of data movement that carry the same risk as a migration and need the same discipline.

10.1 Schema migrations — 32 of them, and none destructive

Fact	Figure	Where to check it
Declared schema version	33	`CfoDatabase.VERSION`
Migrations written	32 — `MIGRATION_1_2` through `MIGRATION_32_33`	`core/database/.../migration/Migrations.kt`
Exported schema fixtures	33, one per version, contiguous	`core/database/schemas/`

Fact	Figure	Where to check it
Destructive migrations	0 — forbidden by DB-003 and BR-017	MigrationSafetyTest

Every schema change to date has been **additive**: a new table, a new nullable column, a new index. The rule is not stylistic. A user's only copy of their data is on their phone, so a migration that drops a column destroys the single copy, and there is no server-side backup to restore from.

The guard is in two halves, and the split is deliberate. Room's own MigrationTestHelper proves that *data* survives a migration, but it needs a device — and a rule enforced only when somebody plugs in a phone is a rule enforced sometimes.

Guard	Runs	Proves
MigrationSafetyTest	On the JVM, in unitTests, everywhere	The **structural** half: a fixture exists for the declared version, the fixtures are contiguous, no consecutive pair contains a dropped table, a dropped column or a changed type, and the schema-level money and time invariants hold
MigrationRoundTripTest	androidTest, on a device or emulator	The **data** half: a database written at version N opens and reads correctly at version N+1

The first of those is the one that cannot be skipped, because the exported schema JSON is just data: the destructive-change check is a comparison between two files. It also fails on a *missing* fixture, which matters more than it sounds — without that assertion every other check in the file would pass vacuously against an empty directory, which is a failure mode this repository has already been bitten by once (ADR-0081 §2).

Adding a version is therefore a four-step procedure, and the tests enforce three of them: bump CfoDatabase.VERSION, write the Migration, build so KSP exports schemas/<version>.json and **commit that file**, then add a round-trip case. Skip the fixture and the build goes red; skip the round-trip case and only a device run notices, which is the one soft edge in this gate and is worth knowing about.

10.2 The archive, which is a migration between devices FR-057's JSON archive is the user's own migration path: phone to phone, or version to version. It carries its schema version, and an import refuses an archive written by a **newer** build rather than applying part of it (VAL-020, UC-035-E1).

Risk	Control
A renamed Kotlin property silently changes the archive format	ArchiveFormatTest pins every field name; a rename fails the build (ADR-0023)
An interrupted import leaves a half-merged profile	Replace is transactional: the profile is the old one or the new one (BR-016)
An archive is restored into the wrong profile	Refused (ADR-0040)
The audit log would leak events across profiles through the archive	It is export-excluded, deliberately (UC-052-F1)

10.3 Seed data — three loads, all small, all checked

Load	Source	Needed by	Note
The category taxonomy	ai/ knowledge files	MS-002	An admin-typable set, not a migration; the user may edit it afterwards
Rulebook thresholds and the classification KB	ai/rules, ai/knowledge	MS-004	Versioned rows, each cited by id and version in evidence (BR-018). A typed mirror is checked against the data row by a drift test, so the two cannot diverge in a shipped binary
The demo dataset	Generated in code	MS-002	Realistic Indian data under its own profile_id, hard-deleted on exit (UC-003)

No backfill is pending. Two were considered and both are no:

Backfill	Status	Why
Recompute historical net-worth snapshots when the derivation changes	**No**	The series repairs the affected days on demand instead (ADR-0012), so a bulk backfill would do work the read path already does correctly
Re-run classification over every historical transaction when the KB version moves	**No**	A stored result carries the engine version that produced it (`AI-ARC-006`). Re-running would silently rewrite a figure the user has already seen, which is the opposite of reproducibility

Both of those are decisions rather than omissions, and both would be visible to a user if taken: one as a chart that changed shape overnight, the other as a category that changed without their asking.

---

11. What this plan does not say

An honest delivery plan names its own silences.

Not stated	Why, and what a reader should do instead
Estimates in days or points	There is one developer and no commitment to a date, so an estimate would be a number with no planning purpose. The sequence in §5.1 is the real constraint; `MS-007`'s exit conditions are the real gate
Team structure, roles, onboarding	`STK-003` is the whole team. `RISK-001` records what that costs, and the mitigation is mechanical enforcement rather than a second reviewer
A release date	`MS-007` depends on three open questions (`OQ-003`, `OQ-005`, `OQ-001`) and on `RISK-002`, which is Google's decision and not the project's
Velocity or burn-down	89 issue files across 13 epics are the unit of work, each with its own documentation gate (`scripts/check_issue_docs.py`). Progress is read from closed issues, not from a chart
A support SLA	§9 says self-serve, which is a position rather than an omission. There is no server to be down and no on-call to staff

---

12. The eight tasks that are not merely work

Seventeen of the twenty-five open tasks are ordinary: a known change, a known size, a known gate. Eight are not, and listing them in a table alongside the others hides what is actually true of them — **each one contains a decision that has to be made before the work can be estimated at all.** Scheduling them would be pretending otherwise.

TASK-001 — Decide the market-data proxy

- **The decision inside it:** `OQ-003`. Host the proxy, or formally descope it.
- **Why it cannot be estimated:** the two answers differ by an order of magnitude. "Build it" is a server, a hosting bill, certificate pinning and an operational story for a project that currently has no server at all. "Descope it" is a day of restating `FR-042`, `FR-073` and `FR-078` as cache-only and amending the SRS.
- **What it unblocks:** `GAP-001`, `GAP-015`, `TASK-019`, `MS-007`, and `UC-045`, which today has no data to work with at all.
- **How to tell it is done:** either six endpoints answer and the client has pins, or three requirements say "cache-only" and `GAP-001` is closed as descoped. Not answering is the one outcome with no upside.

TASK-002 — Account Aggregator integration

- **The decision inside it:** `OQ-001`. Whether AA is in the roadmap, and when.
- **Why it cannot be estimated:** it is gated on an external onboarding process nobody has started, with a sandbox, a certification and a timeline that are not this project's to set.
- **What it unblocks:** it is the only capture path that would make `READ_SMS` optional, which defuses `RISK-002` — the risk that Google refuses the permission and a specified capture path disappears.
- **How to tell it is done:** a consented AA fetch produces transactions carrying the **already reserved** `aa` source value (ADR-0074), and `AccountAggregatorReadinessTest` stops being a guard against rot and becomes a test of a real path.

TASK-003 — Unblock the KMP port

- **The decision inside it:** `OQ-002`. Does exactness yield to portability, or does iOS leave the roadmap?
- **Why it cannot be estimated:** it is a design question wearing a task's clothes. The engines are already Android-free, so the port looks close — and ADR-0075 is the record of discovering that "Android-free" is not "KMP-portable", with `Money` as the blocker and no multiplatform exact-decimal answer that `MNY-001` would accept.
- **What it unblocks:** `MS-006`, which is otherwise permanently incomplete.
- **How to tell it is done:** either `Money` compiles on a second target with its invariants intact and its property tests green, or iOS is out of scope in writing and `NFR-012` says so.

TASK-016 — Health-score history

- **The decision inside it:** whether it is wanted at all. ADR-0045 chose live computation **deliberately**, and `GAP-013` records the consequence: a historical score is not re-derivable from storage.
- **Why it is not simply scheduled:** the table is small and the ADR is the obstacle. Adding a series without revisiting the decision would quietly contradict a recorded one, which is the failure mode `DECISIONS.md` exists to prevent.
- **What it unblocks:** a score trend, and an answer to "why did it move since March?" that is better than "it is computed live".
- **How to tell it is done:** ADR-0045 is superseded or amended, *then* the table exists. In that order.

TASK-018 — Reconcile the promised `docs/traceability.csv`

- **The decision inside it:** `OQ-008`. Whether 10 **is** the artefact SRS §28 promised, or whether the CSV is still owed.
- **Why it is not simply work:** generating a CSV takes an hour. Deciding whether a generated, validator-checked matrix supersedes a promised file is a governance call, and the wrong answer leaves two traceability artefacts to keep in step — which is the problem, not the solution.
- **What it unblocks:** `GAP-002`, and C-17's claim that the documents agree.
- **How to tell it is done:** either §28 is amended to name this document, or a CSV is generated **from** this document so the two cannot diverge.

TASK-021 — Split `Entities.kt` and `Daos.kt`

- **The decision inside it:** none, and that is why it is here. It is a pure refactor of 4,723 and 2,666 lines, and the rule it satisfies is the project's own 500-line rule.
- **Why it is not routine:** the archive format must not move. `ArchiveFormatTest` pins every field name, so the refactor is **provable** rather than risky — but only if the proof is run, which makes this a task with a mandatory before-and-after rather than a tidy-up.
- **What it unblocks:** `RISK-013`, and the credibility of a project that enforces its rules mechanically everywhere except on its two largest files.
- **How to tell it is done:** 40 entities and 34 DAOs in their own files, every test green with no change but its imports, and `ArchiveFormatTest` untouched and passing.

TASK-024 — Play submission readiness for `READ_SMS`

- **The decision inside it:** Google's, not this project's.
- **Why it cannot be estimated:** the outcome is a policy review. The work is a declaration and possibly a fallback capture path, and which of those is needed is known only after submission.
- **What it unblocks:** `MS-007`, and it is the live half of `RISK-002`.
- **How to tell it is done:** either the permission is granted, or `FR-003`'s step becomes a no-op and the OCR and import paths carry capture — which they already can, which is why this is a risk and not a blocker.

TASK-025 — Crash reporting, so `NFR-004` becomes measurable

- **The decision inside it:** `OQ-005`. Any crash reporter is a network path, and `P-01` makes that a consent question before it is a tooling one.
- **Why it cannot be estimated:** an opt-in on-device reporter with a user-triggered upload is a different piece of work from an SDK that phones home, and the choice is a privacy decision.

- **What it unblocks:** NFR-004, which currently has a target and no measurement, and part of GAP-017.
- **How to tell it is done:** a crash-free rate exists, and the path that produced it is consented, revocable and recorded in the consents dashboard like every other optional path (FR-059).

What the eight have in common. Not one is blocked by engineering. Six are waiting on an answer from the owner, one is waiting on Google, and one — TASK-021 — is waiting on nothing at all, which makes it the only item on this list that could start today.

Part 09 — Risks, Decisions, Assumptions and Open Questions

← Master Blueprint · Related: 01 Overview & Scope · 08 Delivery · 11 Completeness

1. Risks

Exposure is likelihood × impact, on a 1–5 scale each.

RISK	Description	Category	Likelihood	Impact	Exposure	Mitigation	Contingency	Owner	Affects	Status
RISK-001	**One maintainer.** Every decision, review and release depends on one person. There is no second reader and no succession	Project	5	5	25	Mechanical enforcement instead of human review: six lint detectors, 3,986 tests, 36 drift-test classes, golden fixtures, a published decision record, and a session log that explains the reasoning rather than only the result	The source is public under AGPL, so the work is recoverable by someone else even if the author stops	STK-003	Everything	Open — accepted
RISK-002	`READ_SMS` **is refused at Play submission.** The permission is policy-sensitive, and a rejection removes a specified capture path	Compliance	3	4	12	The permission is gated behind an explicit consent, declared and argued in ADR-0013, and nothing depends on it: manual, OCR and import all work without it	Ship without SMS capture; `FR-003`'s step becomes a no-op and the OCR path carries capture	STK-004	FR-003, FR-021, INT-008	Open — `TASK-024`
RISK-003	**The device is the only copy.** No server holds the data, so a lost phone with no backup is total loss for that user	Reliability	3	5	15	E2EE local backup, a restore that is drilled on real hardware before every promotion, and a recovery screen that states irrecoverability in advance	None exists, by design. The mitigation is honesty: the UI says the passphrase cannot be recovered	STK-001	FR-087, FR-088, NFR-017	Open — mitigated
RISK-004	**Categorisation accuracy on Indian merchants.** A classifier that is wrong often enough stops being used	Product	3	4	12	A versioned merchant knowledge base, correction learning, a frozen eval set with a $\geq$ 92% floor, and a drift test that stops the KB and its mirror diverging	Lower the automation claim and lean on the correction loop	STK-001	FR-017, FR-060	Open — measured
RISK-005	**The on-device model's footprint and speed.** A model that is too big breaks NFR-005, and one that is too slow makes chat useless	Technical	3	3	9	A strict tool-call design so the model only writes prose; Apache-2.0 weights behind a swappable interface; and a fallback that returns rendered figures without prose	Drop to figures-only replies, which still answer the question	STK-003	FR-077, NFR-005	Open — mitigated
RISK-006	**An unverified figure reaches a user.** The product's central claim is that every number is checkable; one hallucinated figure undoes it	Product	2	5	10	The guardrail: one engine, a rendering allowlist, rounding that stops at the rupee, and a measured refusal rate against a frozen set with an absolute threshold	A model is disabled rather than loosened. The engines' figures still render	STK-003	FR-076, BR-011, BR-012	Open — gated
RISK-007	**The unbuilt backend becomes load-bearing by accident.** A feature that quietly needs the proxy would break `P-04`	Architecture	2	4	8	The client ships with a blank base URL, so no HTTP stack is even constructed; `offlineSmoke` runs on a device with airplane mode on	Remove the feature rather than the offline guarantee	STK-003	`GAP-001`, NFR-009	Open — gated
RISK-008	**Performance claims are unmeasured.** NFR-001…003 carry numbers that nothing checks, so a regression would be invisible	Quality	4	3	12	Architecturally the work is off the main thread and reads are local, both of which are tested — but that is not a measurement	Measure before a public beta (`TASK-005`), or restate the NFRs as untargeted	STK-003	NFR-001, NFR-002, NFR-003, `GAP-004`	Open
RISK-009	**Orphaned receipt blobs.** A deleted transaction can leave an encrypted image behind — a finding opened while closing an earlier one	Privacy	4	2	8	The blob is encrypted and unreadable without the key, so an erase still destroys it; the defect is a storage leak, not a confidentiality one	`TASK-004` sweeps them	STK-003	FR-027, ADR-0078	Open — recorded
RISK-010	**Knowledge bases go stale.** Tax parameters change every Union Budget; rulebook thresholds age	Operational	4	3	12	Every threshold is a versioned data row, every engine result cites the row id and version, and drift tests stop a mirror diverging	An engine states what it could not estimate rather than using last year's figures (ADR-0072)	STK-003	FR-081, BR-018	Open — by design
RISK-011	**An honest product is harder to sell.** "It shows its working" competes against apps that simply assert	Commercial	3	3	9	The transparency is the differentiator, and the source is public so the privacy claim is verifiable rather than marketed	`GOAL-005` is deliberately post-v1, so no revenue decision is urgent	STK-003	GOAL-004, GOAL-007	Open
RISK-012	**The education/advice boundary is crossed by a plausible-sounding engine.** The product is not SEBI-registered	Compliance	2	5	10	Every advisor output carries the informational disclaimer; the protection engine never recommends surrendering a policy; the app never moves money	Remove or reword the offending output; the boundary is documented in SRS §44	STK-005	FR-055, BR-024, BR-025	Open — gated
RISK-013	**Two large source files violate the project's own 500-line rule.** `Entities.kt` is 2,666 lines and `Daos.kt` is 4,723	Maintainability	5	2	10	Both are mechanically regular and heavily documented, and the archive format is pinned by a test, so a split is provable rather than risky	`TASK-021` splits them to one type per file	STK-003	BR-017, NFR-013	Open — see the architecture and gap report
RISK-014	**A governance gate that nobody runs.** Two items in the Definition of Done — SonarQube and the session record — are agent-followed, not build-enforced	Process	4	2	8	Both are documented **as** unenforced rather than implied to be gates, which is the lesson of the vacuous coverage gate	Add a pre-commit hook or a CI job with a server, and record that decision	STK-003	`OQ-004`, NFR-014	Open — stated
RISK-015	**iOS stays out of reach.** The blocker is `Money` itself, which is the one type that cannot yield	Technical	4	3	12	The engines are already Android-free, so the port is narrower than it would otherwise be	Accept Android-only, or accept a portability layer around exact money	STK-003	`OQ-002`, NFR-012	Open

2. Decisions

The project's decision records are **four-digit** and live in `docs/adr/` — 80 ADRs (numbered to `ADR-0082`; 0031 and 0032 were never written, and nothing references them). 05 §10 carries a digest of each one, generated from the files themselves. `DECISIONS.md` is the one-line index, and each ADR owns its own argument. This register points rather than restates — a second copy of an argument is a copy that loses.

Group	Records	What they settle
Enforcement and tooling	0001, 0063, 0064, 0065, 0066, 0067, 0068, 0076, 0081, 0082	Custom lint, the golden harness, eval datasets, screenshot coverage, the offline gate, the release train, the engine registry, static analysis, every-screen renders
Money and time	0007, 0008, 0009, 0010, 0011, 0012, 0026, 0028	Derived balances, transfers as legs, splits as a child table, future-dated posting, back-dating, amortisation, XIRR
Capture	0013, 0014, 0015, 0016, 0074, 0077, 0078	`READ_SMS`, the classification KB, Stage-1 tiers, nature by account type, the reserved `aa` source, GST storage, import batches
Planning	0004, 0005, 0017, 0019, 0020, 0021, 0033, 0034, 0035, 0036, 0037, 0038	Quick setup, deferred rulebook loaders, budget thresholds and bands, reviews, Safe-to-Spend, goals, the emergency fund, the waterfall
Wealth	0025, 0027, 0029, 0030	Card claims per cycle, asset class on the holding, the portfolio denominator, the unconfigured market client
AI	0042…0055, 0079, 0080	Stream classification, the forecast, seasonality, the health score, the orchestrator, notifications, the guardrail, the advisor, the buy list, simulators, vehicle prediction, chat, market signals, the on-device model, the verbaliser
Privacy and security	0003, 0022, 0023, 0024, 0039, 0040, 0041, 0057, 0058, 0059, 0060, 0061, 0062	The lock gate, the blur, the archive, the widget, backup crypto, restore, the drill, key management, capture guarding, consents, erase, DPDP, release hardening
Expansion	0069, 0070, 0071, 0072, 0073, 0075	Household, appliances, protection, tax, business mode, and why KMP is blocked

Two superseded records are worth naming, because a withdrawn decision is still a decision: **ADR-0011** (only a scanned receipt may be back-dated) was superseded by **ADR-0012** once the net-worth series could repair itself, and the SRS's `loan_amortization_rows` table named in ADR-0016 was refused by **ADR-0026**.

3. Assumptions

Every row is something **I** concluded rather than something the sources state. Each carries what it would cost to be wrong.

ASM	Assumption	Why assumed	Impact if wrong	Validated by / when	Status
ASM-001	The SRS, `README.md`, `CLAUDE.md`, the 80 ADRs and the shipped code together constitute "what the user said", so facts drawn from them are Confirmed rather than assumed	The project's own `CLAUDE.md` declares the SRS binding and the ADRs authoritative; there was no interview to draw on	The whole blueprint's Confirmed/Assumed split would need re-marking	The owner, on reading this blueprint	Open
ASM-002	Where the SRS and the shipped code disagree, the **code plus its ADR** is the current truth and the SRS is the older statement	SRS §28 says the code is a bug or an ADR is missing — and in every case examined, the ADR exists	A requirement recorded here as satisfied would be unsatisfied	Spot-checked against 12 ADRs while writing 02	Confirmed by inspection
ASM-003	FR-005's three-minute onboarding target is unmeasured because the analytics consent that would measure it does not exist	The SRS conditions the measurement on consent ("measured in analytics if consented"), and no analytics path ships	A `SHOULD` requirement is unverified rather than unmet	`TASK-005`, if onboarding timing is added	Open
ASM-004	The SRS's `FR-TXN-012` is a **formatting example**, not a requirement	It appears only in §4.1 as "e.g., FR-TXN-012", and §5.3 stops at FR-TXN-010	A requirement would be missing from this blueprint	Grepped the SRS and the whole repository: the id appears nowhere else	Confirmed by inspection

ASM	Assumption	Why assumed	Impact if wrong	Validated by / when	Status
ASM-005	The 35 SRS tables that do not exist are design evolution, not omissions, except the two named as gaps	Each absence has an ADR or an obvious substitute, and 11 of them moved into `ai/` data files	Doc 04 §14 would understate the data-model debt	Entity-by-entity against SRS §20.1	Confirmed by inspection
ASM-006	The blueprint's `FR-001`…`FR-095` cover the whole shipped surface	Derived by walking the 60 modules, the 21 routes, the 9 workers and the 89 issues	A shipped capability would have no requirement, which is the reverse of the usual gap	Cross-checked against `settings.gradle.kts` and `CfoRoute`	Confirmed by inspection
ASM-007	`NFR-013`…`NFR-020` are requirements rather than merely practices	Each is enforced by a task that fails a build, which is the test of a requirement	They would belong in a process document instead	The owner	Open
ASM-008	A public release is intended at some point, which is what makes `MS-007` meaningful	`README.md` describes a product with a licence, terms and a store-facing posture	`MS-007` and `TASK-024` would be moot	The owner	Open
ASM-009	"Rekhon" (the name in `README.md`) and "AI Personal CFO" (the name in the SRS) are the same product, product name and project name respectively	The README describes exactly the SRS's product	Nothing material; the naming would need one line of clarification	The owner	Open
ASM-010	The eleven account types, 40 tables and 30 engines in the tree are complete as of `v0.13.10` and not mid-refactor	The working tree is clean of structural changes; the uncommitted files are documentation and screenshot baselines	Component counts here would be stale	`git status` and `settings.gradle.kts` at the time of writing	Confirmed by inspection
ASM-011	The 500-line and one-class-per-file rules in the global `CLAUDE.md` apply to this Kotlin tree, not only to Python and React	They are stated as global, for "ANY language"	`RISK-013` and `TASK-021` would be inapplicable	The owner	Open
ASM-012	This blueprint's 10 may stand in for the SRS's promised `docs/traceability.csv`	It holds the same mapping, in a form that a validator checks	`GAP-002` stays open as a missing artefact rather than a satisfied one	The owner — `OQ-008`	Open

4. Dependencies

DEP	Dependency	Type	Owner	Needed by	Risk if it moves
DEP-001	Android platform APIs 26+ — Keystore, StrongBox, BiometricPrompt, WorkManager, SAF	Platform	Google	FR-084, FR-085, FR-087	A behaviour change in key invalidation is the one that would hurt; ADR-0057's rotation is the hedge
DEP-002	Jetpack Compose and Material 3	Library	Google	Every `:feature:*`	A breaking release is absorbed by 115 screenshot baselines catching it
DEP-003	Room and SQLCipher	Library	Google, Zetetic	ENT-001…ENT-040	A SQLCipher major change would need a key-handling review; migrations are the safety net
DEP-004	ML Kit Text Recognition v2	Library, proprietary	Google	FR-024, FR-025	It is behind an interface precisely so a free recogniser is a contained swap
DEP-005	The on-device model's weights, Apache-2.0	Model	Upstream publisher	FR-077	Licence-clean and pinned by a manifest drift test
DEP-006	Google Tink, and BouncyCastle for Argon2id	Library	Google, BC	SEC-003, FR-087	SEC-003 forbids replacing them with anything hand-rolled
DEP-007	Hilt	Library	Google	ARC-003	A DI change is wide but mechanical
DEP-008	Paparazzi	Test library	Square	NFR-020	A rendering change invalidates baselines; re-recording is deliberate (ADR-0066)
DEP-009	Kover, detekt, ktlint, Android Lint	Build tooling	Various	NFR-013, NFR-014	A gate's own correctness is itself tested
DEP-010	GitHub Actions, JDK 21, an emulator with KVM	CI	GitHub	NFR-009, NFR-017	Robolectric needs JDK 21; CI is pinned to it
DEP-011	SonarQube, self-hosted	Tooling	STK-003	`CLAUDE.md` §8.1	Not a build gate, by design — `sonar.skipProject` is true without a token, so offline builds still work (`P-04`)

DEP	Dependency	Type	Owner	Needed by	Risk if it moves
DEP-012	An external market-data source, through our own proxy	Service	**Nobody yet**	API-001… API-006	This is `GAP-001`. Pointing the client at an exchange or AMFI directly is forbidden, and is very hard to unpick once it works
DEP-013	Google Play	Distribution	Google	MS-007	`RISK-002`

4.1 Constraints The constraint register is declared in 01 §7 as `CON-001`…`CON-014` and is not duplicated here. The three that bind hardest: `CON-002` (no server for user data) rules out whole categories of design; `CON-004` (exact money) is both the correctness guarantee and the iOS blocker; `CON-008` (one developer) is why every rule is a gate.

5. Open questions

OQ	Question	Why it matters	Blocks	Who must answer	Blocking this blueprint?	Asked	Answer
OQ-001	Will the Account Aggregator integration be built, and on what timeline?	It is the only capture path that would make `READ_SMS` optional, which defuses `RISK-002`	TASK-002, US-088	Owner	no — blocks MS-006's completion	2026-10-23	—
OQ-002	Does exactness yield to portability, or does iOS stay out of scope?	`Money` is the blocker. One of the two has to give, and neither is obviously right	TASK-003, US-089	Owner	no — blocks iOS only	2026-10-23	—
OQ-003	Will the market-data proxy be hosted at all, and by whom?	Six endpoints, the backtest feature and the external-data tables all hang off it. A "no" is a perfectly good answer — but it should be a decision, not a drift	TASK-001, TASK-019, `GAP-001`, `GAP-015`	Owner	no — blocks MS-007	2026-10-23	—
OQ-004	Which deferred thresholds should move from typed Kotlin mirrors into a runtime `ai/` loader, and what finally triggers that build?	ADR-0005's trigger list was narrowed once already (ADR-0017) because a trigger had not fired. A third deferral needs the question answered rather than re-argued	BR-018	Owner	no	2026-10-23	—
OQ-005	How is `NFR-004`'s crash-free rate to be measured without a network reporter?	Any crash reporter is a network path, which `P-01` makes a consent question before it is a tooling one	TASK-025, NFR-004	Owner	no — blocks MS-007	2026-10-23	—
OQ-006	What is `GOAL-005`'s actual target?	The SRS names a freemium shape with no number, so the goal currently has a measure and no target	GOAL-005	Owner	no — post-v1	2026-10-23	—
OQ-007	Is `FR-010`'s dashboard edit mode wanted, or should the requirement be withdrawn?	It is a `SHOULD` that is unbuilt and untested. Either is fine; both being true silently is not	FR-010, TASK-014	Owner	no	2026-10-23	—
OQ-008	Does this blueprint's 10 satisfy SRS §28's promised `docs/traceability.csv`, or is the CSV still owed?	§28 promises a file that does not exist. This document holds the same mapping in a checked form	`GAP-002`, TASK-018	Owner	no	2026-10-23	—

Blocking count: 0. Every open question gates a *future* increment — a milestone, a platform, an integration — and none of them blocks building from, or maintaining, what this blueprint describes. That is the basis for the verdict in 11, and it is the one claim in this document that a reader should check first.

---

6. How the register sorts, and what the top of it means

Exposure is likelihood × impact on a 1–5 scale each, so the figure runs 1–25 and is the single column to sort by. Reading the register that way produces four tiers, and the tiers matter more than the individual scores.

Tier	Exposure	Risks	What they have in common
Structural	25	`RISK-001`	One maintainer. Not reducible by any technical control, only by mechanical enforcement and a public source tree
High	12–15	`RISK-003`, `RISK-002`, `RISK-004`, `RISK-008`, `RISK-010`, `RISK-015`	Each is a consequence of a deliberate choice — no server, a policy-sensitive permission, on-device inference, no analytics, versioned knowledge, exact money
Moderate	8–10	`RISK-006`, `RISK-007`, `RISK-009`, `RISK-011`, `RISK-012`, `RISK-013`, `RISK-014`	All gated or recorded. Three of them (`RISK-009`, `RISK-013`, `RISK-014`) are findings this blueprint opened rather than inherited
Low	below 8	—	None. A register with no low-exposure rows is a register that has not been padded

The three that decide the verdict. `RISK-001` (one maintainer) is why every rule in this project is a build gate rather than a review convention — a convention needs a second person. `RISK-003` (the device is the only copy) is why the restore drill blocks a release and nothing else in the plan does. `RISK-008` (performance claims are unmeasured) is the one that stops `11-completeness-checklist.md` reading READY rather than READY WITH RISKS, because an unmeasured `NFR` is a requirement with no evidence behind it.

What is deliberately unmitigated. `RISK-001` and `RISK-003` have no contingency that restores the loss; they have honesty instead. The app tells the user in advance that a lost passphrase cannot be recovered, and the licence makes the work recoverable by someone else. A plan that claimed more than that would be claiming a server, a team or a backup that does not exist.

---

7. What would reopen a decision

A decision register is only useful if it says what would change its mind. These are the triggers that are already written into the records, gathered here so a reader does not have to read 80 ADRs to find them.

Trigger	Watching	Would reopen
The per-account balance subquery stops being cheap enough	Read latency on the accounts and dashboard screens	ADR-0007 — the cached column is kept, unused, precisely so the read can switch to it
A third deferral of the runtime rulebook loader	`OQ-004`	ADR-0005, narrowed once already by ADR-0017
`READ_SMS` is refused at Play submission	`RISK-002`, `TASK-024`	ADR-0013 — the capture path, not the consent model
The market-data proxy is funded, or formally descoped	`OQ-003`	ADR-0030, and `GAP-001` closes either way
Exactness yields to portability, or iOS is dropped	`OQ-002`	ADR-0075 — `Money` is the blocker, and it is the one type that cannot bend
A crash reporter becomes acceptable under `P-01`	`OQ-005`	`NFR-004` gains a measurement; nothing architectural changes
Forecast accuracy is measured against actuals	`GAP-003`	ADR-0043, which deferred the measurement and said so
A mirror and its `ai/` data row diverge	36 drift-test classes	ADR-0017 — the mirror strategy itself, not the single row

The pattern worth noticing. Six of the eight triggers are *questions awaiting an answer* rather than conditions awaiting an event. That is the honest shape of a solo project: the decisions that will change are the ones nobody has had to decide yet.

8. What writing this blueprint found

Eight findings came out of reading the code against its own documentation. They are listed here because a register that only inherits risks from a plan is a register that has not been checked against reality. Each is recorded in full in `docs/architecture-and-gap-report.md`.

Finding	Where it landed
19 files exceed the project's own 500-line rule, led by `Daos.kt` at 4,723 lines and `Entities.kt` at 2,666	`RISK-013`, `TASK-021`
A deleted transaction can leave an orphaned encrypted blob behind	`RISK-009`, `TASK-004`
Two Definition-of-Done items are agent-followed, not build-enforced, and the repository now says so plainly	`RISK-014`, `OQ-004`
The SRS has not been amended for 14 recorded deviations; each has an ADR, but §28's own rule asks for both	The gap report §4, `GAP-002`
`docs/traceability.csv`, promised by SRS §28, does not exist	`GAP-002`, `OQ-008`, and 10 stands in for it
`FR-TXN-012` is cited nowhere but a formatting example, and was nearly carried here as a requirement	`ASM-004` — recorded as an assumption rather than a requirement
The repository said "82 ADRs" in several places, read off the highest id rather than counted. There are **80**: 0031 and 0032 were never written, and nothing references them	Corrected in this blueprint and in the gap report; the digest in 05 §10 now counts the files
The repository said "29 drift tests", the figure ADR-0061 recorded on 2026-10-01. There are now **36 classes and 252 cases**	Corrected in 05, 07 and here; the ADR keeps its historical figure

Four of the eight were defects in the documentation rather than in the code, which is the result a documentation pass should expect to produce — and two of those four were **counts anybody could have checked**, which is the more uncomfortable kind. The other four are in the code, and three of them now have a task.

Part 10 — Traceability Matrix

← Master Blueprint · Related: 02 Requirements · 07 Tests · 08 Delivery · 11 Completeness

> `--` *means* ***not applicable****, with the reason in the row or the notes.* ***none*** *means the link* should *exist and does not; every* ***none*** *has a matching* `GAP-xxx` *in §6. §1's forward matrix is* ***generated from 02****, not transcribed, so the two documents cannot drift apart on a requirement's source, use case or as-built status.*

0. ID conventions in use

This project already had an identifier convention, and two of them now coexist on purpose. Nothing here is renumbered: an id that commits, code comments and ADRs already cite keeps its meaning.

Family	Form	Declared where	Status
`FR-<area>-<nnn>` (`FR-TXN-004`)	The SRS's own functional requirement ids	`docs/init/AI_Personal_CFO_SRS_v1.7.pdf` §5	**Authoritative for commits and code.** Carried in the "SRS ID" column of 02 §1 and in §1 below
`FR-nnn` (`FR-016`)	This blueprint's local handle for the same requirement	02 §1	A stable three-digit handle so the graph below is machine-checkable
`NFR-nnn`	The SRS's numbering, **unchanged** — `NFR-001`…`NFR-012` are the SRS's own	SRS §6, extended here to `NFR-020`	One family, no mapping needed
`AI-<engine>-<nnn>`, `SEC-nnn`, `DB-nnn`, `ARC-nnn`, `MNY-nnn`, `TIM-nnn`, `DRL-nnn`, `P-0n`	The SRS's architecture, security, data and principle rules	SRS §4.1	Cited throughout as the source of a `BR` or an `NFR`. Never re-declared here
`ADR-nnnn` (**four** digits)	The project's architecture decision records	`docs/adr/`, indexed in `DECISIONS.md`	Used as-is. Deliberately outside this blueprint's three-digit scheme, so there is exactly one `ADR-0057` and no ambiguity
`RULE-<name>` (`RULE-EMF-MULT`)	Versioned rulebook rows	`ai/rules/`	Cited by engines and by `BR-018`…`BR-023`
`<epic>.<issue>` (`9.2`)	The project's work items	`docs/issues/`	Mapped to `EPIC`/`US` in 08
`GOAL`, `STK`, `CON`, `BR`, `VAL`, `AC`, `FLOW`, `UC`, `ENT`, `CMP`, `INT`, `API`, `TC`, `MS`, `EPIC`, `US`, `TASK`, `RISK`, `ASM`, `DEP`, `OQ`, `GAP`	This blueprint's own families	The owning document	New — the SRS had no equivalent

One collision is worth naming. SRS §22 labels four *API rules* `API-001`…`API-004`. This blueprint uses `API-nnn` for an API **operation**, as the skill's scheme requires. To avoid two meanings for one id, the SRS's four rules are referred to in prose as "SRS §22 rules" and are never cited by id. They appear here as `BR-028` (the consent asymmetry), the error-envelope and rate-limit conventions in 06 §4, and `NFR-009`.

Also deliberately not re-declared: the SRS's `BG-01`…`BG-06` business goals, which appear as `GOAL-001`…`GOAL-006` with the mapping in 01 §2.

1. Forward matrix — requirement outward

Functional requirements

Req	SRS ID	Source	AC	UC	ENT	API	TC	TASK	As built	Status
FR-001	FR-ONB-001	GOAL-001	AC-001	UC-001	ENT-002, ENT-003	—	TC-001	—	Yes	Traced
FR-002	FR-ONB-002	GOAL-001	AC-002	UC-002	ENT-009, ENT-012	—	TC-002	—	Yes (ADR-0004)	Traced
FR-003	FR-ONB-003	STK-005	AC-003	UC-001	— (consents live in DataStore, not the database)	—	TC-003	TASK-024	Yes	Traced
FR-004	FR-ONB-004	GOAL-003	AC-004	UC-003	ENT-002	—	TC-004	—	Yes	Traced
FR-005	FR-ONB-005	GOAL-003	AC-005	UC-001	—	—	TC-005	—	Partial	Traced
FR-006	FR-HOME-001	GOAL-001	AC-006	UC-004	ENT-004, ENT-009, ENT-014, ENT-025	—	TC-006	—	Yes	Traced
FR-007	FR-HOME-002	GOAL-004	AC-007	UC-004	—	—	TC-007	—	Yes	Traced
FR-008	FR-HOME-003	STK-001	AC-008	UC-005	—	—	TC-008	—	Yes	Traced
FR-009	FR-HOME-004	NFR-001	AC-009	UC-004	ENT-004, ENT-009, ENT-014, ENT-025	—	TC-009	—	Partial	Traced
FR-010	FR-HOME-005	STK-001	AC-010	UC-004	—	—	**none**	TASK-014	No	**GAP-016**
FR-011	FR-HOME-006	GOAL-003	AC-011	UC-006	— (Glance state, never Room)	—	TC-011	—	Yes	Traced
FR-012	FR-HOME-007	P-04	AC-012	UC-004	ENT-004, ENT-009, ENT-014, ENT-025	—	TC-012	—	Yes	Traced
FR-013	FR-TXN-001	GOAL-001	AC-013	UC-007	ENT-004, ENT-006, ENT-007, ENT-015	—	TC-013	—	Yes	Traced
FR-014	FR-TXN-002	GOAL-003	AC-014	UC-007	ENT-004, ENT-006, ENT-007, ENT-015	—	TC-014	—	Yes	Traced
FR-015	FR-TXN-003	GOAL-001	AC-015	UC-008	ENT-004	—	TC-015	—	Yes	Traced
FR-016	FR-TXN-004	BR-008	AC-016	UC-009	ENT-004, ENT-005	—	TC-016	—	Yes	Traced
FR-017	FR-TXN-005	GOAL-002	AC-017	UC-010	ENT-004, ENT-008	—	TC-017	TASK-012	Partial	Traced
FR-018	FR-TXN-006	P-05	AC-018	UC-011	ENT-012	—	TC-018	—	Yes	Traced
FR-019	FR-TXN-007	GOAL-001	AC-019	UC-012	ENT-004	—	TC-019	—	Yes	Traced
FR-020	FR-TXN-008	STK-001	AC-020	UC-012	ENT-004	—	TC-020	TASK-008	Partial	Traced
FR-021	FR-TXN-009	GOAL-004	AC-021	UC-007	ENT-004, ENT-006, ENT-007, ENT-015	—	TC-021	—	Yes	Traced
FR-022	FR-TXN-010	FR-063	AC-022	UC-013	ENT-004	—	TC-022	—	Yes	Traced
FR-023	FR-OCR-001	P-05	AC-023	UC-014	ENT-004, ENT-015	—	TC-023	TASK-023	Partial	Traced
FR-024	FR-OCR-002	P-01	AC-024	UC-014	ENT-004, ENT-015	—	TC-024	—	Yes	Traced
FR-025	FR-OCR-003	P-05	AC-025	UC-014	ENT-004, ENT-015	—	TC-025	TASK-013	Partial	Traced
FR-026	FR-OCR-004	VAL-010	AC-026	UC-014	ENT-004, ENT-015	—	TC-026	—	Yes	Traced
FR-027	FR-OCR-005	P-01	AC-027	UC-014	ENT-015	—	TC-027	TASK-004	Partial	Traced
FR-028	FR-OCR-006	GOAL-001	AC-028	UC-014	ENT-004, ENT-015	—	TC-028	—	Yes	Traced
FR-029	FR-BUD-001	GOAL-001	AC-029	UC-015	ENT-009	—	TC-029	TASK-009	Partial	Traced
FR-030	FR-BUD-002	GOAL-002	AC-030	UC-015	ENT-009	—	TC-030	—	Yes	Traced
FR-031	FR-BUD-003	P-02	AC-031	UC-015	ENT-009	—	TC-031	—	Yes	Traced
FR-032	FR-BUD-004	GOAL-002	AC-032	UC-016	ENT-010	—	TC-032	—	Yes	Traced
FR-033	FR-BUD-005	GOAL-002	AC-033	UC-017	ENT-011	—	TC-033	—	Yes	Traced
FR-034	FR-GOAL-001	GOAL-002	AC-034	UC-018	ENT-022	—	TC-034	—	Yes	Traced
FR-035	FR-GOAL-002	GOAL-002	AC-035	UC-018	ENT-022	—	TC-035	—	Partial	Traced
FR-036	FR-GOAL-003	P-02	AC-036	UC-019	ENT-022	—	TC-036	—	Yes	Traced
FR-037	FR-GOAL-004	GOAL-004	AC-037	UC-020	ENT-023, ENT-024	—	TC-037	—	Yes	Traced
FR-038	FR-GOAL-005	GOAL-002	AC-038	UC-019	ENT-022	—	TC-038	—	Yes	Traced
FR-039	FR-ACC-001	GOAL-001	AC-039	UC-021	ENT-003	—	TC-039	—	Yes	Traced
FR-040	FR-ACC-002	GOAL-001	AC-040	UC-022	ENT-017, ENT-019	—	TC-040	—	Yes	Traced
FR-041	FR-ACC-003	GOAL-002	AC-041	UC-023	ENT-018	—	TC-041	—	Yes	Traced
FR-042	FR-ACC-004	GOAL-001	AC-042	UC-024	ENT-020, ENT-036	API-001, API-002	TC-042	TASK-001	Partial	Traced
FR-043	FR-ACC-005	GOAL-001	AC-043	UC-025	ENT-014	—	TC-043	—	Yes	Traced
FR-044	FR-ACC-006	BR-007	AC-044	UC-026	ENT-003, ENT-004	—	TC-044	—	Yes	Traced
FR-045	FR-ACC-007	GOAL-001	AC-045	UC-021	ENT-003	—	TC-045	—	Yes	Traced
FR-046	FR-INV-001	GOAL-001	AC-046	UC-027	ENT-020, ENT-021	API-002	TC-046	—	Yes	Traced
FR-047	FR-INV-002	GOAL-002	AC-047	UC-028	ENT-020	—	TC-047	—	Yes	Traced
FR-048	FR-VEH-001	GOAL-001	AC-048	UC-029	ENT-031, ENT-032, ENT-033, ENT-034	—	TC-048	TASK-007	Partial	Traced
FR-049	FR-VEH-002	GOAL-002	AC-049	UC-029	ENT-031, ENT-032, ENT-033, ENT-034	—	**none**	TASK-007	No	**GAP-009**
FR-050	FR-VEH-003	P-02	AC-050	UC-029	ENT-031, ENT-032, ENT-033, ENT-034	API-004	TC-050	TASK-007	Partial	Traced
FR-051	FR-AI-001	GOAL-002	AC-051	UC-030	ENT-025	—	TC-051	—	Yes	Traced
FR-052	FR-AI-002	P-02	AC-052	UC-031	ENT-025	—	TC-052	—	Yes	Traced
FR-053	FR-AI-003	GOAL-002	AC-053	UC-032	ENT-027, ENT-028	—	TC-053	—	Yes	Traced
FR-054	FR-AI-004	P-03	AC-054	UC-033	ENT-035	—	TC-054	—	Yes	Traced
FR-055	FR-AI-005	STK-005	AC-055	UC-030	—	—	TC-055	—	Yes	Traced
FR-056	FR-SET-001	STK-001	AC-056	UC-034	ENT-008, ENT-012	—	TC-056	TASK-011	Partial	Traced
FR-057	FR-SET-002	STK-005	AC-057	UC-035	ENT-001… ENT-040 (every serialisable entity)	—	TC-057	TASK-010	Partial	Traced
FR-058	FR-SET-003	STK-005	AC-058	UC-036	ENT-001… ENT-040	—	TC-058	—	Yes	Traced

Req	SRS ID	Source	AC	UC	ENT	API	TC	TASK	As built	Status
FR-059	FR-SET-004	P-01	AC-059	UC-037	— (consents live in DataStore, not the database)	—	TC-059	—	Yes	Traced
FR-060	AI-CLSN-001	GOAL-002	AC-060	UC-010	ENT-004, ENT-008	—	TC-060	TASK-020	Partial	Traced
FR-061	—	GOAL-002	AC-061	UC-038	ENT-004	—	TC-061	—	Yes (ADR-0042)	Traced
FR-062	—	P-02	AC-062	UC-038	ENT-004	—	TC-062	—	Partial	Traced
FR-063	AI-FCT-001	GOAL-002	AC-063	UC-039	ENT-004, ENT-012	—	TC-063	TASK-006	Yes	Traced
FR-064	—	P-08	AC-064	UC-039	ENT-004, ENT-012	API-008	TC-064	—	Yes	Traced
FR-065	—	GOAL-002	AC-065	UC-040	ENT-003, ENT-022	—	TC-065	—	Partial	Traced
FR-066	—	GOAL-002	AC-066	UC-028	ENT-020	—	TC-066	—	Yes	Traced
FR-067	—	GOAL-002	AC-067	UC-029	ENT-031, ENT-032, ENT-033, ENT-034	API-009	TC-067	—	Yes	Traced
FR-068	—	GOAL-001	AC-068	UC-041	ENT-037, ENT-038, ENT-039	API-009	TC-068	—	Yes	Traced
FR-069	—	P-02	AC-069	UC-032	ENT-027, ENT-028	—	TC-069	—	Yes (ADR-0049)	Traced
FR-070	—	GOAL-002	AC-070	UC-042	ENT-029, ENT-030	—	TC-070	—	Yes (ADR-0050)	Traced
FR-071	—	P-02	AC-071	UC-043	ENT-003, ENT-004, ENT-017	—	TC-071	TASK-016	Partial	Traced
FR-072	—	GOAL-002	AC-072	UC-018	ENT-022	—	TC-072	—	Yes	Traced
FR-073	—	P-04	AC-073	UC-024	ENT-020, ENT-036	API-001…API-006	TC-073	TASK-001, TASK-019	Partial	Traced
FR-074	—	STK-001	AC-074	UC-044	ENT-026	—	TC-074	TASK-017	Partial	Traced
FR-075	—	GOAL-002	AC-075	UC-031	ENT-025	—	TC-075	—	Yes (ADR-0046)	Traced
FR-076	AI-ARC-004	P-03	AC-076	UC-033	ENT-035	—	TC-076	—	Yes	Traced
FR-077	—	P-03	AC-077	UC-033	ENT-035	API-011	TC-077	—	Yes (ADR-0053, ADR-0079)	Traced
FR-078	—	P-02	AC-078	UC-045	ENT-036	API-005	TC-078	—	Yes (ADR-0055)	Traced
FR-079	—	GOAL-002	AC-079	UC-046	ENT-022	—	TC-079	—	Yes	Traced
FR-080	—	GOAL-002	AC-080	UC-047	ENT-018	—	TC-080	—	Yes	Traced
FR-081	—	P-02	AC-081	UC-048	ENT-004, ENT-020	API-007	TC-081	—	Partial	Traced
FR-082	—	GOAL-002	AC-082	UC-049	ENT-004, ENT-020	—	TC-082	—	Yes	Traced
FR-083	—	P-02	AC-083	UC-004	ENT-004, ENT-009	—	TC-083	—	Yes	Traced
FR-084	SEC-002	STK-001	AC-084	UC-050	— (keys live in the Keystore)	—	TC-084	—	Yes	Traced
FR-085	SEC-001	P-01	AC-085	UC-050	— (keys live in the Keystore)	—	TC-085	—	Yes	Traced
FR-086	—	STK-001	AC-086	UC-005	—	—	TC-086	—	Yes	Traced
FR-087	SEC-005	STK-001	AC-087	UC-051	ENT-001…ENT-040	API-010	TC-087	—	Yes	Traced
FR-088	DRL-001	GOAL-004	AC-088	UC-051	ENT-001…ENT-040	—	TC-088	—	Yes	Traced
FR-089	—	STK-005	AC-089	UC-052	ENT-013	—	TC-089	—	Yes	Traced
FR-090	NFR-011	STK-001	AC-090	UC-053	—	—	TC-090	—	Yes	Traced
FR-091	—	STK-002	AC-091	UC-054	ENT-001, ENT-002	—	TC-091	TASK-022	Partial	Traced
FR-092	—	STK-001	AC-092	UC-054	ENT-001, ENT-002	—	TC-092	—	Yes (ADR-0073)	Traced
FR-093	—	GOAL-004	AC-093	UC-055	ENT-040	—	TC-093	TASK-002	Yes	Traced
FR-094	—	BR-010	AC-094	UC-013	ENT-004	—	TC-094	—	Yes (ADR-0010)	Traced
FR-095	DB-001	BR-007	AC-095	UC-026	ENT-003, ENT-004	—	TC-095	—	Yes	Traced

Non-functional requirements

Req	Category	Target is a number?	AC	TC	Test approach	TASK	Status
NFR-001	Performance	yes — 2.0 s P90	AC-096	**none**	Device timing run; not yet automated	TASK-005	**GAP-004**
NFR-002	Performance	yes — 100 ms	AC-097	**none**	Instrumented timing of the write path	TASK-005	**GAP-004**
NFR-003	Performance	yes — 3 s / 50k	AC-098	**none**	Benchmark over a generated fixture	TASK-005	**GAP-004**
NFR-004	Reliability	yes — 99.8%	AC-099	**none**	Play vitals, post-launch	TASK-025	**GAP-017**
NFR-005	Footprint	yes	AC-100	TC-100	`assembleRelease` artefact size in CI	—	Traced
NFR-006	Battery	yes	AC-101	TC-101	Code review plus the nine declared workers	—	Traced
NFR-007	Durability	yes	AC-102	TC-102	Room in-memory tests plus the multi-row transaction rule (BR-016)	—	Traced
NFR-008	Compatibility	yes	AC-103	TC-103	`minSdk` in the build	—	Traced
NFR-009	Offline	yes	AC-104	TC-104	`offlineSmoke` on an emulator in airplane mode (ADR-0067)	—	Traced
NFR-010	Accessibility	yes	AC-105	TC-105	Accessibility scan plus Paparazzi renders at 200% font in light and dark	—	Traced
NFR-011	Localisation	yes	AC-106	TC-106	`CfoHardcodedUiString` fails the build; a drift test fails on a missing translation	—	Traced
NFR-012	Correctness	yes	AC-107	TC-107	`CfoMoneyAsFloatingPoint` fails the build; `:core:model` holds 100% line coverage	—	Traced
NFR-013	Maintainability	yes	AC-108	TC-108	`koverVerify` with a real `minBound`, after the vacuous version was found (audit G-01)	—	Traced
NFR-014	Maintainability	yes	AC-109	TC-109	`./gradlew unitTests` plus `lintDebug` in CI	—	Traced
NFR-015	Security	yes	AC-110	TC-110	OSV scan in CI, querying resolved coordinates (SEC-007)	—	Traced
NFR-016	Security	yes	AC-111	TC-111	`verifyReleaseLogStripping` reads the dex rather than trusting the config (ADR-0062)	—	Traced
NFR-017	Recoverability	yes	AC-112	TC-112	`restoreDrill` on an emulator in CI (ADR-0041)	—	Traced
NFR-018	Privacy	yes	AC-113	TC-113	`CfoPiiInLogs` fails the build	—	Traced
NFR-019	Reproducibility	yes	AC-114	TC-114	`engine_version` stored with every result (AI-ARC-006); golden files frozen per version (ADR-0064)	—	Traced
NFR-020	Verifiability	yes	AC-115	TC-115	`verifyPaparazziDebug` inside `unitTests`; 115 committed renders (ADR-0082)	—	Traced

2. Goals and stakeholders outward

GOAL	Measure has a target?	Requirements serving it	Milestone
GOAL-001	yes — ≥ 90% of spend captured by month 3	FR-001, FR-002, FR-006, FR-013, FR-019, FR-021, FR-039, FR-042, FR-043, FR-045, FR-046	MS-002
GOAL-002	yes — ≥ 40% of recommendations acted on	FR-017, FR-030, FR-032, FR-033, FR-034…FR-038, FR-041, FR-047, FR-051, FR-053, FR-060, FR-063, FR-065…FR-072, FR-074, FR-075, FR-079…FR-082	MS-004, MS-005
GOAL-003	yes — day-30 retention ≥ 35%	FR-004, FR-011, FR-014	MS-002
GOAL-004	yes — ≥ 4.5 trust score, < 1% data complaints	FR-007, FR-021, FR-037, FR-043, FR-052, FR-088, FR-093	MS-003
GOAL-005	**no target yet** — `OQ-006`	—	MS-007
GOAL-006	yes — ≥ 90% self-serve	FR-054, FR-056	MS-005
GOAL-007	yes — source published under an inspectable licence	FR-024, FR-058, FR-086, NFR-018	MS-001

STK	Requirements that exist for them
STK-001	Nearly all of `FR-001`…`FR-095`; named explicitly in FR-008, FR-056, FR-084, FR-086, FR-090
STK-002	FR-091, FR-092
STK-003	NFR-013…NFR-020 — the requirements that exist because there is no second reviewer
STK-004	FR-003, NFR-005, NFR-016
STK-005	FR-003, FR-055, FR-057, FR-058, FR-089, SEC-008
STK-006	GOAL-007, and the licence posture in 01 §7 (`CON-011`)
STK-007	NFR-019, NFR-020, and this blueprint

3. Backward matrices — catching orphans

3.1 Test cases to requirements

TC range	Covers	Justified?
TC-001…TC-095	`FR-001`…`FR-095`, one to one	Yes — every row names its `FR` and its `AC`
TC-096…TC-115	`NFR-001`…`NFR-020`, one to one	Yes. Four of them (TC-096…TC-099) are **recorded as not implemented** rather than listed as passing
TC-116	BR-006, FR-091	Yes — profile scoping is a rule, not a feature
TC-117	BR-018, FR-060	Yes — drift between a mirror and its data row
TC-118	BR-017, FR-093	Yes — migration safety
TC-119	BR-013, NFR-019	Yes — the ARC-002 guard
TC-120	BR-013, FR-075	Yes — engine-registry consistency (ADR-0076)
TC-121	BR-013, FR-063	Yes — the golden harness testing itself
TC-122	NFR-019, FR-060	Yes — an eval run must report what it measured
TC-123	FR-059, NFR-015	Yes — DPDP matrix alignment
TC-124	BR-015, FR-085	Yes — the SEC-003 lint rule

Tests in the tree with no requirement are enumerated and justified in 07 §8. All seven groups are infrastructure whose parent is a binding rule rather than a feature. None is an orphan to delete.

3.2 Tasks to requirements

TASK	Implements	Justified?
TASK-001	FR-042, FR-073, API-001	Yes
TASK-002	FR-093, AC-021	Yes
TASK-003	NFR-012, AC-107	Yes
TASK-004	FR-027, AC-027	Yes
TASK-005	NFR-001, NFR-002, NFR-003	Yes
TASK-006	FR-063, AC-063	Yes
TASK-007	FR-048, FR-049, FR-050	Yes
TASK-008	FR-020, AC-020	Yes
TASK-009	FR-029, AC-029	Yes
TASK-010	FR-057, AC-057	Yes
TASK-011	FR-056, AC-056	Yes
TASK-012	FR-017, AC-017	Yes
TASK-013	FR-025, AC-025	Yes
TASK-014	FR-010, AC-010	Yes
TASK-015	NFR-010, AC-105	Yes
TASK-016	FR-071, AC-071	Yes — but it needs ADR-0045 revisited before it is work
TASK-017	FR-074, AC-074	Yes
TASK-018	NFR-019	Yes — an SRS §28 artefact that does not exist
TASK-019	FR-073, API-003	Yes
TASK-020	FR-060, AC-060	Yes
TASK-021	BR-017, NFR-013	Yes — the project's own 500-line rule, not a feature
TASK-022	FR-091, AC-091	Yes
TASK-023	FR-023, AC-023	Yes — a `MAY`, so it may stay unbuilt
TASK-024	FR-003, AC-003	Yes
TASK-025	NFR-004, AC-099	Yes

Every task names a parent. None is an unjustified orphan.

3.3 Components to requirements Each of the 60 `CMP` rows in 05 §2 names the requirements it serves. Four components serve a **rule** rather than a requirement, and that is recorded rather than left to look like an accident:

CMP	Serves	Instead of
CMP-059 `:lint`	NFR-012, NFR-014, NFR-018, NFR-011, BR-001, BR-003	A user-facing feature. It ships in no APK
CMP-060 `:spike:kmp`	`OQ-002`	Anything. It is a time-boxed answer to a question, and was never shipped
CMP-003 `:core:common`	BR-003	A feature. It exists so that `Clock` can be injected
CMP-002 `:core:model`	BR-001, BR-002, NFR-012	A feature. It exists so that money cannot be a `Double`

4. Coverage summary

Counted from the tables above, not estimated.

Metric	Count	%	Target
`FR` with ≥ 1 `AC`	95 of 95	100%	100%
`FR` with ≥ 1 `TC`	93 of 95	97.9%	100%
`FR` with ≥ 1 `UC`	95 of 95	100%	—
`FR` traced to a `GOAL`, `STK`, `NFR` or `BR` as its source	95 of 95	100%	100%
`NFR` with a measurable target	20 of 20	100%	100%

Metric	Count	%	Target
NFR with a working test approach	16 of 20	80%	100%
UC with an exception **and** a failure path	55 of 55	100%	100%
ENT governed by at least one BR or VAL	40 of 40	100%	—
API with error cases documented	12 of 12 by the shared envelope; 2 of 12 with per-operation status codes	100% / 17%	100%
TC with a parent requirement or a stated justification	124 of 124	100%	—
TASK with a parent requirement or a justification	25 of 25	100%	100%
CMP with a named responsibility and dependencies	60 of 60	100%	100%
INT with protocol, auth and failure behaviour	11 of 11	100%	100%
Blocking OQ outstanding	0 of 8	0%	0
FR shipped (As built = Yes)	66 of 95	69.5%	—
FR shipped with a named limitation (Partial)	26 of 95	27.4%	—
FR not built (No)	3 of 95	3.2%	—

Two notes on the honest reading of the last three rows. **"Partial" is not a synonym for broken** — in 22 of the 26 cases the limitation is a *decision* with an ADR behind it (the goals engine assumes no return rate; nature fires on account type; the health score keeps no history), not an unfinished job. And the 40 entities counted as "governed by a rule" are governed mostly by the **universal** rules — BR-001…BR-006 and BR-017 apply to every table; only six entities are named individually by a rule. Those universal rules are the ones that actually bind the schema, which is why the metric reads 100% rather than 15%.

5. SRS coverage

Every requirement in the SRS, accounted for.

SRS family	Count	In this blueprint	Not carried
FR-ONB-*	5	FR-001…FR-005	—
FR-HOME-*	7	FR-006…FR-012	—
FR-TXN-*	10	FR-013…FR-022	FR-TXN-012 is a **formatting example** in SRS §4.1, not a requirement (ASM-004)
FR-OCR-*	6	FR-023…FR-028	—
FR-BUD-*	5	FR-029…FR-033	—
FR-GOAL-*	5	FR-034…FR-038	—
FR-ACC-*	7	FR-039…FR-045	—
FR-INV-*	2	FR-046, FR-047	—
FR-VEH-*	3	FR-048…FR-050	—
FR-AI-*	5	FR-051…FR-055	—
FR-SET-*	4	FR-056…FR-059	—
NFR-*	12	NFR-001…NFR-012, unchanged	—
AI-* engine requirements (§7–§19, §30, §36–§43)	—	FR-060…FR-083	The engines of §32 (Worth-It), §33 (Growth Optimizer beyond tax), §37 (FI), §41 (Rent-vs-Buy), §42 (glide paths), §43 (behaviour dashboard) are **specified and not built** — GAP-019
SEC-001…SEC-008	8	FR-084…FR-089, NFR-015, NFR-016, NFR-018	—
DB-001…DB-006	6	BR-005, BR-007, BR-016, BR-017, and doc 04 §1, §7	—
ARC-001…ARC-006	6	05 §1, §3	—

SRS family	Count	In this blueprint	Not carried
MNY-001, MNY-002, TIM-001, TIM-002	4	BR-001…BR-004	—
P-01…P-08	8	Cited as the source of CON-002…CON-007, BR-011, BR-024	—
DRL-001	1	FR-088, NFR-017	—
BG-01…BG-06	6	GOAL-001…GOAL-006	—

6. Gaps

Severity: **Critical** blocks readiness · **High** · **Medium** · **Low**.

GAP	Missing link	Severity	Affected IDs	Action	Owner
GAP-001	The backend specified in SRS §22 is not built. Eleven of twelve operations have no implementation on either side, and the one client method ships against a blank base URL	**High**	API-001…API-012, FR-042, FR-073, FR-078, DEP-012, OQ-003	Answer OQ-003, then TASK-001: build it, or formally descope it and restate the affected requirements	Owner
GAP-002	SRS §28 promises docs/traceability.csv, which does not exist	Medium	NFR-019, OQ-008	TASK-018 — decide whether this document satisfies it, and either generate the CSV from it or amend §28	Owner
GAP-003	Forecast accuracy (MAPE) is not tracked, so a KPI the SRS sets a target for is unmeasured	Medium	FR-063, GOAL-002	TASK-006	Owner
GAP-004	NFR-001, NFR-002 and NFR-003 carry numbers that nothing measures. TC-096…TC-098 are **none**	**High**	NFR-001, NFR-002, NFR-003, FR-009, RISK-008	TASK-005 — a benchmark harness, or restate the NFRs without targets	Owner
GAP-005	"Apply to similar past transactions" after a correction is not built	Low	FR-017	TASK-012	Owner
GAP-006	The seven-day trash compactor is not built; soft-deleted rows are kept indefinitely	Low	FR-020, doc 04 §11	TASK-008	Owner
GAP-007	Receipt line items are not extracted	Low	FR-025	TASK-013	Owner
GAP-008	Budget rollover of an unused amount is not built	Low	FR-029	TASK-009	Owner
GAP-009	The vehicle fuel log is not built, so FR-049 has **no test and no implementation**, and FR-050's cost figure omits fuel	Medium	FR-048, FR-049, FR-050, TC-049	TASK-007	Owner
GAP-010	There is no recurring-rules manager screen, although FR-SET-001 lists one	Low	FR-056	TASK-011	Owner
GAP-011	CSV export is not built; only the JSON archive is	Low	FR-057	TASK-010	Owner
GAP-012	TalkBack completeness is asserted by renders and tokens, not by a screen-reader pass	Medium	NFR-010	TASK-015	Owner
GAP-013	The health score keeps no history, so a past score cannot be re-derived from storage	Low	FR-071	TASK-016, and ADR-0045 revisited first — the live computation was deliberate	Owner
GAP-014	The notification digest is deferred, so several same-day candidates are capped rather than batched	Low	FR-074	TASK-017	Owner
GAP-015	The external-data tables (macro, fuel, FX, calendar) do not exist, because the feeds do not	Low	FR-073, doc 04 §14	TASK-019, after TASK-001	Owner

GAP	Missing link	Severity	Affected IDs	Action	Owner
GAP-016	FR-010 (dashboard edit mode) has no implementation and no test	Low	FR-010, TC-010, OQ-007	TASK-014, or withdraw the requirement	Owner
GAP-017	NFR-004 (crash-free sessions) has no measurement path, because any reporter is a network path	Medium	NFR-004, TC-099, OQ-005	TASK-025	Owner
GAP-018	~~API-002… API-009, API-011 and API-012 rely on the shared error envelope and do not list per-operation status codes~~	**Closed 2026-10-23**	API-002… API-012	Done: 06 §6 gives every operation its request table, response body, field semantics, status codes and client behaviour on failure; 06 §7 lists the operation-specific error cases; 06 §8 defines all twelve paths	—
GAP-019	Six SRS engine sections are specified and unbuilt: §32 Worth-It, §33 Growth Optimizer beyond tax, §37 Financial Independence, §41 Rent-vs-Buy, §42 glide paths and rebalancing, §43 the behaviour and cost-drag dashboard	Medium	SRS §32, §33, §37, §41, §42, §43	Scope them into a milestone after MS-007, or record them as descoped. They are **not** in FR-001…FR-095, so this blueprint does not claim them	Owner

19 gaps, 0 Critical, 1 closed. GAP-018 was closed on 2026-10-23 by completing 06's operation detail, error cases and OpenAPI document — the only one of the nineteen that this blueprint could close by itself, because it was a gap in the documentation rather than in the code. Of the eighteen that remain, two are High: the unbuilt backend (GAP-001) and the unmeasured performance targets (GAP-004). Neither prevents the product from working — the first is a degradation the architecture was built for, and the second is a measurement rather than a defect — which is why neither is Critical and why the verdict in 11 is READY WITH RISKS rather than NOT READY.

What would make a gap Critical: a requirement with no acceptance criterion, a money rule with no test, a blocking open question, or a claim in this blueprint that the code does not support. None of those is present, and GAP-019 exists precisely so the last one stays true — six specified engines are missing, and this blueprint says so rather than quietly scoping them out.

Part 11 — Completeness Checklist

← Master Blueprint · Related: 10 Traceability · 09 Risks & Open Questions

> ***What is being judged.*** *Not "is the software good" — that is what 3,986 tests and thirteen CI gates answer. This is:* ***is the documentation complete and honest enough to build from, maintain, and hand to someone else?*** *The product already exists, so the usual question ("is it safe to start building?") is replaced by "does this blueprint describe what was actually built, including where it fell short?"* `PASS` *means satisfied with evidence that can be pointed at by ID.* `PARTIAL` *means satisfied with named exceptions. "We did not get to it" is* `FAIL`*, never* `NOT APPLICABLE`*.*

Checks

#	Check	Status	Evidence	Remaining gap	Next action
C-01	Business problem and success measures are clear	**PARTIAL**	01 §1 (five named failures, with the current workaround's cost); 01 §2, `GOAL-001`…`GOAL-007`, six of seven carrying a target number	`GOAL-005` (monetisation) has a measure and **no target**	Answer `OQ-006`, or withdraw `GOAL-005` until v1 ships
C-02	Stakeholders, actors and user roles identified	**PASS**	01 §3, `STK-001`…`STK-007` with role, interest, influence and need; 01 §3.1 lists six actors including the three system actors. Every actor named in a `UC` appears there	None	—
C-03	Scope, exclusions and boundaries documented	**PASS**	01 §5 (in-scope, grouped by capability); 01 §6 (nine exclusions, **each with its reason**); 01 §4 plus `diagrams/context.dot` for the boundary; `INT-001`…`INT-011` in 05 §4	None	—
C-04	Requirements are specific and testable	**PASS**	02 §1–§2, 115 requirement rows. Checked mechanically: **zero** rows contain an unquantified *fast, easy, secure, scalable, modern, intuitive, user-friendly, robust, performant* or *simple*. Every `NFR` carries a number	None	—
C-05	Functional and non-functional requirements covered	**PARTIAL**	`FR-001`…`FR-095` cover every shipped capability, cross-checked against the 60 modules, 21 routes and 9 workers (`ASM-006`); `NFR-001`…`NFR-020` cover performance, footprint, battery, durability, compatibility, offline, accessibility, localisation, correctness, maintainability, security, recoverability, privacy, reproducibility and verifiability	Six SRS engine sections are specified and carried as **no** requirement here: §32, §33 (beyond tax), §37, §41, §42, §43 — `GAP-019`	Scope them into a milestone after `MS-007`, or record them as descoped in the SRS
C-06	Business rules and validation rules explicit	**PASS**	02 §3, `BR-001`…`BR-028`, each naming what it applies to **and who may change it** — which for every financial threshold is a data row, not a developer; 02 §4, `VAL-001`…`VAL-022`, each with its message and its enforcement point	None	—
C-07	Acceptance criteria defined	**PASS**	02 §5, `AC-001`…`AC-115`, Given/When/Then, one per `FR` and per `NFR`. 02 §2.1 is empty by construction, and 10 §4 recomputes it at 95 of 95 and 20 of 20	None	—

#	Check	Status	Evidence	Remaining gap	Next action
C-08	Main, alternate, exception and failure flows documented	**PASS**	03 §3, `UC-001`…`UC-055`. Every one has a main path, ≥ 1 alternate, ≥ 1 exception **and** ≥ 1 failure path with its recovery behaviour; 03 §4 counts it; `FLOW-001`…`FLOW-010` hold the end-to-end processes	None	—
C-09	Entities, attributes, relationships and data dictionary covered	**PASS**	04 §2, `ENT-001`…`ENT-040` with purpose, owner and lifecycle; 04 §3, **435 columns** with type, required-ness, default, sensitivity and notes, generated from the schema at version 33; 04 §4 gives every relationship a cardinality and an on-delete	None	—
C-10	Database choices and unresolved design questions visible	**PASS**	04 §13 (Room + SQLCipher, with the options considered and the negative consequences stated); 04 §5 keys; 04 §8 indexes, each justified by a named query; 04 §9 transaction boundaries; 04 §12 migrations, 32 of them, non-destructive by rule; 04 §14 names all 35 SRS tables that do not exist and why	None	—
C-11	Architecture, component responsibilities and integrations documented	**PASS**	05 §2, `CMP-001`…`CMP-060`, one responsibility each with its allowed dependencies; 05 §4, `INT-001`…`INT-011` each with protocol, auth, failure behaviour and owner; 05 §6 every technology choice with its rule; 05 §3 names the **mechanism** that enforces each architectural rule, which is the part that usually goes missing	None	—
C-12	API contracts and error handling documented	**PARTIAL**	06 §4.1 (the error envelope, eight codes, each with its client behaviour), §4.4 idempotency, §4.5 versioning; 06 §5 `API-001`…`API-012`; 06 §6 per-operation detail — every operation with its request table, response body, field semantics, status codes and behaviour on failure; 06 §7 the operation-specific error cases; 06 §8 a complete OpenAPI document, twelve paths and sixteen schemas, with no dangling `$ref`	**The backend does not exist** — `GAP-001`. The specification gap that stood here (`GAP-018`) is **closed**: the contract is now complete enough to build the proxy from this document alone	Answer `OQ-003`. The document is deliberately *not* NOT APPLICABLE: the client contract is real and one operation is consumed by shipped code
C-13	Security, privacy, accessibility, performance, reliability, operations considered	**PARTIAL**	Security — 05 §5, 13 controls, `SEC-001`…`SEC-008`, `NFR-015`, `NFR-016`, `NFR-018`. Privacy — `P-01`, `BR-014`, `FR-059`, `FR-086`, DPDP alignment as a **checked** document. Reliability — 05 §9 with RPO/RTO and the restore drill as a release gate. Operations — 05 §8, shaped by the fact that the app may not phone home	Performance has numbers nothing measures (`GAP-004`); accessibility lacks a screen-reader pass (`GAP-012`); crash-free rate has no measurement path (`GAP-017`)	`TASK-005`, `TASK-015`, `TASK-025`
C-14	Test scenarios and coverage are traceable	**PARTIAL**	07 §5, `TC-001`…`TC-124`, each naming its type, its parent and the **real test class** in the tree; 07 §1 maps twelve test levels to what each proves; 10 §4 computes coverage	93 of 95 `FR` have a `TC` (`GAP-016`, `GAP-009`); 16 of 20 `NFR` have a working test approach (`GAP-004`, `GAP-017`). 07 §7 lists every one of them rather than omitting them	`TASK-005`, `TASK-007`, `TASK-014`, `TASK-025`

#	Check	Status	Evidence	Remaining gap	Next action
C-15	Delivery tasks, dependencies, milestones, release covered	**PASS**	08 §2, `MS-001`…`MS-007`, each with its exit criteria and its real status — including `MS-004`'s **unmet** criterion; 08 §3 thirteen epics; 08 §4, `US-001`…`US-089`, one per issue file; 08 §5, 25 open tasks each tied to the gap that opened it; §5.1 the critical path; §8 release and rollback	None	—
C-16	Risks, assumptions, decisions and open questions recorded	**PASS**	09 §1, `RISK-001`…`RISK-015` with likelihood, impact, exposure, mitigation, contingency and owner; 09 §2 and 05 §10 the 80 four-digit ADRs — 33 digested in full and all 80 indexed, generated from the files so the digest cannot drift from the records; 09 §3, `ASM-001`…`ASM-012`, **each with its impact if wrong**; 09 §4, `DEP-001`…`DEP-013`; 09 §5, `OQ-001`…`OQ-008`, each marked blocking or not	None	—
C-17	Master and detailed documents agree	**PASS**	00 §10's counts are computed from the detailed documents, not typed; 00 §8's verdict is copied verbatim from this document; 10 §1's forward matrix is **generated from** 02, so a requirement's source, use case and as-built status cannot differ between them	None	—
C-18	No invented requirements or unapproved decisions presented as fact	**PASS**	Every `FR` and `NFR` cites its SRS id or is marked `--` where it is a project-enforced requirement with a named gate. Everything I concluded rather than read is an `ASM` (09 §3). **Five spot-checks**, each traced to its source: `FR-016` → SRS §5.3 FR-TXN-004 + ADR-0009; `NFR-012` → SRS §6 + the lint detector; `FR-065`'s two-of-five multiplier terms → ADR-0034; `FR-071`'s six signals → ADR-0045 (and the SRS's "five pillars" is recorded as superseded, not as satisfied); `FR-TXN-012` → **not** carried, because it is a formatting example (`ASM-004`)	None	—

Verdict

READY WITH RISKS

Test	Result
Any `FAIL`?	No. Five `PARTIAL`, thirteen `PASS`, none `FAIL`, none `NOT APPLICABLE`
`FAIL` on C-01, C-04, C-07 or C-18?	No. C-01 is `PARTIAL` over one goal's missing target; the other three `PASS`
Blocking open questions?	**0 of 8.** Every `OQ` gates a future increment — a milestone, a platform, an integration — not the current one (09 §5)
Critical gaps?	**0 of 19**, and one is now closed: `GAP-018` (incomplete API operation detail). Two are High: `GAP-001` (the unbuilt backend) and `GAP-004` (unmeasured performance targets)
Every `PARTIAL` has an owner and an action?	Yes — C-01 → `OQ-006`; C-05 → `GAP-019`; C-12 → `OQ-003`/`TASK-001`; C-13 → `TASK-005`/`TASK-015`/`TASK-025`; C-14 → `TASK-005`/`TASK-007`/`TASK-014`/`TASK-025`

Why not `READY`. C-14 is `PARTIAL`: two requirements have no test and four non-functional targets have no measurement. A blueprint that claimed full test traceability with `TC-096`…`TC-099` reading **none** would be exactly the kind of green gate this project has already caught itself shipping once.

Why not `NOT READY`. Nothing is missing that would make the work unsafe to continue. The requirements are specific, every one has an acceptance criterion, the data model is read off the schema rather than imagined, 93 of 95 requirements have a test, and the two High gaps are a *decision not yet taken* and a *measurement not yet made* — neither is a defect in what exists.

Answer these before readiness can be approved

In priority order. Each is phrased as a question, because each is the owner's to answer rather than mine to assume.

1. `OQ-003` — **Will the market-data proxy be hosted, by whom, and with what certificate pins?** A "no" is a perfectly good answer and would close `GAP-001` and `GAP-015` at once by letting `FR-042`, `FR-073` and `FR-078` be restated as cache-only. Letting it stay unanswered is the only outcome with no upside.
2. **Will `NFR-001`, `NFR-002` and `NFR-003` be measured, or restated without targets?** (`GAP-004`, `TASK-005`.) Three numbers that nothing checks are three claims a reviewer cannot verify — and a regression in any of them would currently be invisible.
3. `OQ-002` — **Does exactness yield to portability, or does iOS leave the roadmap?** `Money` is the blocker and one of the two has to give. Leaving it open keeps `MS-006` permanently incomplete.
4. `OQ-005` — **How should crash-free sessions be measured, given that any reporter is a network path?** This is a `P-01` decision before it is a tooling one, and `NFR-004` cannot be verified until it is made.
5. `GAP-019` — **Are the six unbuilt SRS engine sections (§32, §33, §37, §41, §42, §43) still in scope?** If yes they need requirements and a milestone; if no they need a line in the SRS. They are currently in neither state.
6. `OQ-007` — **Is `FR-010`'s dashboard edit mode wanted, or withdrawn?** It is a `SHOULD` that is unbuilt and untested; either answer is fine, and both being silently true is not.
7. `OQ-008` — **Does 10 satisfy SRS §28's promised `docs/traceability.csv`, or is the CSV still owed?**
8. `OQ-004` — **What finally triggers moving the deferred thresholds into the runtime `ai/` loader?** The trigger list has already been narrowed once because a trigger had not fired; a third deferral should answer the question rather than re-argue it.
9. `OQ-006` — **What is `GOAL-005`'s target?** It is the one business goal with a measure and no number, which is why C-01 is `PARTIAL`.

Appendix A — Architecture and Gap Report

Date: 2026-10-23 · **Version examined:** `v0.13.10` · **Author:** written alongside `docs/project-blueprint/`

> ***What this is.*** *The blueprint describes what the system* is. *This report is the other half: where the shipped code* ***differs from the SRS****, where it* ***differs from the project's own rules****, and what each difference costs. It is deliberately the least flattering document in the set.* ***How to read a finding.*** *Every one is either a* ***deviation with an ADR*** *(a decision that was made and recorded — these are not defects), a* ***gap*** *(something specified and absent — tracked as* ***GAP-xxx*** *in 10 §6), or a* ***conformance finding*** *(the code disagreeing with a rule this project wrote for itself). Only the third category is a criticism.*

1. Summary

Dimension	Finding
Requirements	59 of the SRS's 60 screen requirements are implemented; `FR-TXN-012` turned out not to be a requirement at all. 26 of 95 ship with a named limitation, and **22 of those 26 are decisions with an ADR**, not unfinished work
Architecture	The one-way module graph, pure-Kotlin engines and the layered AI pipeline are real and **mechanically enforced**, not merely documented. This is the strongest part of the project
Data model	40 tables against the SRS's ~75. 35 are absent, and 33 of those absences are design evolution with a reason; **two are genuine gaps**
The backend	Specified in SRS §22, **not built**, and one client method ships against a blank base URL. The largest single gap
Testing	3,986 tests, thirteen CI gates, and gates that have been proved able to fail. Performance is the hole: three NFRs carry numbers that nothing measures
Project's own rules	Six findings, §5. The significant one is that **19 production files exceed the 500-line rule**, led by two at 4,723 and 2,666 lines
Documentation	80 ADRs, a decision index, a traced call-path document, 89 issue files with trackers, 30 of 30 `ENGINE.md` files. Unusually complete; the weak point is that the SRS has not been amended as the code moved past it

The single most useful sentence in this report: nearly every difference between the SRS and the code is *written down somewhere* — but it is written down in an ADR, and the SRS itself has never been updated. A reader who trusts the SRS alone will be wrong about the health score's five pillars, the goals engine's return assumption, the amortisation table, and about thirty other things.

2. What the architecture actually is

Verified against `settings.gradle.kts`, the convention plugins, `FLOW.md` and the tree.

Claim	Verified	How
60 Gradle modules, one-way `feature` → `domain` → `data/core`	Yes	`settings.gradle.kts`; no feature module depends on another
Engines are pure Kotlin with no Android imports	Yes	A convention-plugin TestKit guard runs in CI (`Arc002GuardTest`)
Repositories are the only DAO touchers	Yes	`:data:repository` holds every DAO reference; ViewModels see domain types
One immutable `UiState` per screen as `StateFlow`	Yes	Every feature module, Turbine-tested
`GlobalScope` is banned	Yes	`CfoGlobalScope` fails the build
Money is `Long` paise everywhere	Yes	`CfoMoneyAsFloatingPoint` fails the build; `:core:model` at 100% line coverage
Time is UTC millis through an injected `Clock`	Yes	`CfoWallClockInDomain` fails the build in `:domain:*` and `:core:model`

Claim	Verified	How
Every engine result carries provenance	Yes	`EngineProvenance` in `:core:model`; `engine_version` stored with results
Thresholds are data rows, not code	Yes	`ai/rules/`, mirrored into typed Kotlin and guarded by 36 drift-test classes whose data files are declared Gradle inputs
Every engine has an `ENGINE.md`	Yes — **30 of 30**	—

What makes this project unusual is not that it has rules; it is that the rules are *tasks that fail*. Six custom lint detectors, a TestKit architecture guard, 36 drift tests, 115 screenshot baselines, an offline smoke on a real device, and a restore drill that blocks a release. The project has also twice found and fixed a gate that could not fail — `koverVerify` with zero rules, and a Paparazzi baseline nothing compared against. That history is why the Definition of Done now says plainly which two items are *agent-followed rather than build-enforced* instead of implying a gate that does not exist.

3. SRS versus the code — deviations, each with its record

These are **not** defects. Each is a decision whose argument is on file; they are listed because a reader of the SRS alone would be misled.

SRS says	The code does	Record	Why it is better
Onboarding is four steps in a given order	SMS consent is promoted to its own step	ADR-0002	A consent granted in passing is not informed consent
`accounts.current_balance` is a column	It is a **cache**; the derived balance is the truth	ADR-0007, DB-001	A stored balance is a second source of truth that drifts
A `transfers` table	Two legs sharing a `transfer_id`, type derived	ADR-0008	Both balances stay correct from one ledger, with no table to fall out of step
`loan_amortization_rows` table	No table — the schedule is derived from five columns	ADR-0026	240 stored rows are a cache of a pure function; edit the rate and the copy silently disagrees
The health score has five fixed pillars	Six live signals, reweighted over what is present, no snapshots	ADR-0045	A fixed weighting scores a missing signal as zero, which is a lie about a sparse profile
Goals carry an expected return	The engine assumes **no** return and says so	ADR-0033	No rate exists anywhere in `ai/`; inventing one is the hardcoded financial number §6 forbids
The emergency fund uses five multiplier terms	Two of the five; the other three are asserted **absent**	ADR-0034	Three terms have no field anywhere in the app. A threshold nothing reads is indistinguishable from one that works
XIRR, unspecified method	A fixed 128 bisections over the daily growth factor, no early exit	ADR-0028	The textbook form needs a fractional power, which pulls in the `Double` that MNY-001 bans; Newton is guess-dependent and tolerance-terminated
The privacy toggle "blurs" amounts	A fixed-width mask plus `FLAG_SECURE`	ADR-0022	`Modifier.blur` is a no-op below API 31 and stops no screenshot; a mask also refuses to leak the order of magnitude
Classification has three tiers	Two ship	ADR-0015	The two already clear the $\geq 92\%$ eval gate
Nature classification from the transaction	From the **account type**, override stored, spend understated	ADR-0016	Account type is a fact the user entered — stronger than inference, weaker than data the app does not have
The market API is used	The client ships **unconfigured**; no socket is ever opened	ADR-0030	The alternative was pointing at an exchange directly, which EXT-001 forbids and which is hard to unpick once it works
`household_members` join table	A `household` row *above* the profile	ADR-0069	Nothing asks for a profile in two households, and a join turns a loaded fact into a query
Chat sends context to a relay	An on-device model; the relay is unbuilt	ADR-0079	Strictly better for `P-01`, and it shipped first

Fourteen listed; there are more. The pattern is consistent and worth naming: in every case the code refused to store a derivable figure, refused to invent a number it did not have, or refused a control that would protect nothing. That is the SRS's own P-02 and P-03 applied harder than the SRS applied them.

---

4. Gaps — specified and absent

Full table with severities and actions: 10 §6. The shape of it:

Severity	Count	The ones that matter
High	2	GAP-001 the unbuilt backend · GAP-004 three unmeasured performance targets
Medium	6	GAP-002 the missing traceability.csv · GAP-003 forecast accuracy untracked · GAP-009 the vehicle fuel log · GAP-012 no TalkBack pass · GAP-017 no crash measurement · GAP-019 six unbuilt SRS engine sections
Low	11	Rollover, CSV export, trash compactor, line items, apply-to-similar, recurring-rules screen, health history, digest, dashboard edit mode, external-data tables, per-operation API error codes
Critical	**0**	—

4.1 The two that deserve a paragraph GAP-001 **— the backend.** Eleven of twelve specified operations have no implementation on either side, and the twelfth ships against a blank base URL. The important thing is what this does *not* break: because P-04 forced every external feed to be optional from the start, the absence degrades rather than fails. Prices come from cache with a staleness label; unpriced holdings are reported as missing coverage rather than counted as Rs. 0; the tax engine states what it could not estimate; the signal engine says "not measurable yet" rather than scoring. **One feature is genuinely removed** — the market backtest, which needs five years of history there is no way to fetch. The decision to make is OQ-003, and "we will not host it" is a perfectly good answer that would let three requirements be restated honestly.

GAP-004 **— unmeasured performance.** NFR-001 (cold start < 2.0 s), NFR-002 (save < 100 ms) and NFR-003 (recompute < 3 s over 50k transactions) each carry a number, and nothing measures any of them. The architecture is right — work is off the main thread, reads are local, and both of those *are* tested — but a bound nobody measures is a claim, not a requirement. A regression in any of the three would currently be invisible. This is the gap most out of keeping with the rest of the project's standards, which is precisely why it is called out here.

4.2 Six SRS engines that are specified and unbuilt GAP-019: §32 Worth-It, §33 Growth Optimizer beyond its tax half, §37 Financial Independence, §41 Rent-vs-Buy, §42 glide paths and rebalancing, and §43 the behaviour and cost-drag dashboard. These are **not** carried as requirements in 02 — the blueprint describes what was built — so without this finding they would have disappeared silently. They need either a milestone or a line in the SRS saying they were descoped.

---

5. Conformance findings — the code against this project's own rules

This is the section that is actually criticism. Each is measured, not asserted.

5.1 Nineteen production files exceed the 500-line rule CLAUDE.md (global) — *"No file over 500 lines. Counting everything."*

File	Lines	Reading
core/database/.../dao/Daos.kt	**4,723**	34 DAO interfaces in one file
core/database/.../entity/Entities.kt	**2,666**	40 entity classes in one file
core/database/.../migration/Migrations.kt	2,057	32 migrations. Arguably one responsibility — "the migration history" — and splitting it by version range is the obvious seam
data/repository/.../TransactionRepository.kt	1,883	One class, and the ledger genuinely is the biggest thing in the app
app/.../di/RepositoryModule.kt	1,209	A Hilt module — wiring, not logic

File	Lines	Reading
`data/repository/.../RepositoryFactory.kt`	1,132	Construction
`data/repository/.../BudgetRepository.kt`	1,019	—
`feature/transactions/.../TransactionsUiState.kt`	980	One screen's state
11 more between 500 and 800	—	`TransactionsViewModel`, `InvestmentRepository`, `AccountRepository`, `BudgetEngine`, `DashboardViewModel`, `DemoDataset`, `TransactionsScreen`, `ArchiveRepository`, `AddTransactionViewModel`, `AccountEditorViewModel`

The two at the top also violate the companion rule — *"One entity, one class, one file… Do not collect many entity classes into one shared `models/entities/types` file."* Forty entity classes and 34 DAO interfaces are collected into exactly the two files that rule names.

The honest counter-argument, which the source itself makes: both files are mechanically regular, every class carries a full doc comment, and the archive format is pinned field-by-field by `ArchiveFormatTest`, so a reader navigating by class name is not badly served. **The counter to the counter:** a 4,723-line file is where merge conflicts live, it defeats a reviewer's ability to see a change in context, and the rule exists precisely because "it is regular" is what every oversized file says about itself.

Recommended: `TASK-021` — split both to one type per file with package `__init__`-style re-exports, which in Kotlin means the import sites do not change at all. The archive format test makes this a *provable* no-op: the field names are pinned, so if the split changed anything observable the build goes red. That is an unusually safe refactor, and it is why this finding is actionable rather than merely true.

5.2 The SRS has not been amended as the code moved past it SRS §28: *"This document is the single source of truth. If code and SRS disagree, either the code is a bug or an ADR is missing."* The ADRs exist — all 80 of them — so the letter of the rule is satisfied. But the SRS still says the health score has five pillars, that goals carry an expected return, and that `loan_amortization_rows` exists. A reader who starts at the "single source of truth" is misled roughly thirty times.

Recommended: either amend the SRS at its next version bump, or demote it explicitly to "the original specification, superseded where an ADR says so" — and say that in §28 itself. This blueprint takes the second reading (`ASM-002`).

5.3 The promised traceability artefact does not exist SRS §28 promises `docs/traceability.csv`, *"updated in the same PR as the code."* It has never existed. `GAP-002`, `OQ-008`. This blueprint's 10 holds the same mapping in a form a validator checks, and may simply be the answer — but that should be a decision rather than a silence.

5.4 Two Definition-of-Done items are unenforced, and say so SonarQube cleanliness and the session record are agent-followed, not build-enforced. **This is not a finding against the code** — both are documented *as* unenforced, which is the correct handling and visibly the product of having been burned once. It is listed here only so that a reader does not mistake the Definition of Done for a set of gates that all run. If enforcement is wanted, the stated route is a pre-commit hook or a CI job with a server, recorded as a decision.

5.5 A known defect is open and recorded `RISK-009` — a deleted transaction can leave an orphaned encrypted receipt blob (ADR-0078). It is a storage leak rather than a confidentiality one: the blob is encrypted and an erase destroys the key first, so the data is unrecoverable regardless. `TASK-004`.

5.6 One SRS identifier does not exist `FR-TXN-012` appears in SRS §4.1 as a formatting example and is cited nowhere else; §5.3 stops at `FR-TXN-010`. A governance audit in this repository already predicted this exact failure mode — *"a plausible-looking `FR-TXN-011` that does not exist will pass"* — and the blueprint records it as `ASM-004` rather than inventing a requirement to match.

5.7 Two published counts were wrong, and both were checkable Neither is serious on its own. Both are worth recording because the project's whole argument is that a figure should be checkable, and these two were published for months without being checked.

Claim	Reality	How it happened
"82 ADRs" — in `00`, `01`, `09`, this report and elsewhere	**80 records.** The numbers 0031 and 0032 were never written, and nothing in the repository references them	The figure was read off the highest id, `ADR-0082`, rather than counted. The highest id and the count differ the moment a number is skipped
"29 drift tests" — in `05`, `07`, `09` and the summary	**36 classes, 252 cases**	29 was correct when ADR-0061 counted them on 2026-10-01. Seven classes have been added since, and the prose did not follow

The second is the more instructive. A count copied from an ADR is a snapshot with a date on it; a count in a document that claims to describe the system as built has to be re-derived or it decays. The ADR keeps its 29 — it is a historical record and was right when written. The blueprint now says 36, and `scripts/gen_adr_digest.py` exists so that the decision counts in `05 §10` are generated from the files rather than typed beside them.

6. What is notably strong

A gap report that only lists faults misrepresents the thing it examines.

1. **Enforcement over intention.** Six custom lint detectors, an architecture guard, 29 drift tests with their data declared as build inputs, 115 screenshot baselines, an offline smoke on a real device, and a release-blocking restore drill. Most projects with these rules enforce none of them.
2. **Gates that have been proved able to fail.** The coverage bound was raised to an impossible 101% to watch it go red. The golden harness refuses to rewrite its own fixture. This is the discipline that separates a gate from a decoration.
3. **Refusing to store derivable figures.** Balances, amortisation schedules, holding values and health scores are all computed, never cached as truth. Each refusal is argued in its own ADR, and together they eliminate a whole class of drift bug.
4. **Refusing to invent numbers.** No assumed return rate. No substituted zero for a missing price. No surrender recommendation. No tax slab carried over from last year. Each absence is asserted by a test, so it cannot quietly become a default.
5. **The guardrail.** A language model that may never produce a figure, enforced by a check measured against a frozen set with an absolute threshold. The product's central claim has a test behind it.
6. **Documentation that is maintained rather than written once.** 80 ADRs, a decision index, a traced call-path document, 89 issue files with trackers, 30 of 30 `ENGINE.md`, and session records. The weakness in §5.2 is the exception that stands out because the rest is kept current.

7. Recommended order of work

#	Action	Why first	Tracked as
1	Answer `OQ-003` — host the proxy, or formally descope it	It is a **decision**, costs nothing to make, and closes or reshapes four gaps at once	`TASK-001`
2	Measure `NFR-001`…`NFR-003`, or restate them without targets	Three unverifiable claims in a project whose whole thesis is verifiability	`TASK-005`
3	Split `Entities.kt` and `Daos.kt`	The archive format test makes it a provable no-op, so the risk is near zero and the benefit compounds	`TASK-021`
4	Amend SRS §28, or demote the SRS explicitly	Cheap, and it stops the single-source-of-truth claim being false	§5.2
5	Sweep orphaned receipt blobs	A known open defect	`TASK-004`
6	Decide `GAP-019` — six unbuilt engines, in scope or descoped	They are currently in neither state	`GAP-019`
7	Answer `OQ-002` — exactness or portability	`MS-006` cannot complete until it is answered	`TASK-003`

Everything else in 08 §5 is small and independent.