AI Personal CFO (Rekhon)

Architecture and Gap Report

23 October 2026 · v0.13.10 · Where the code differs from the SRS, and from this project's own rules

The companion to the blueprint, and deliberately the least flattering document in the set

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

Claim	Verified	How
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`
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

SRS says	The code does	Record	Why it is better
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
`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`

#	Action	Why first	Tracked as
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.