Sift Dough

PERSONAL FINANCE · AUTOMATED · PRIVATE

ENGINEERING AUDIT · REPORT № 02

SiftDo Proof of Work

Engineering, Architecture, and Operational Maturity

> *A privacy-first personal finance app, built by one operator over seven weeks of dense daily commits, has reached a surface area that ships across five platforms. This report is an honest accounting of the engineering rigor behind that surface — sourced from a re-runnable harvest script, not a slide deck.*

REPORT DATE	AUDIENCE	DOCUMENT	VERSION
May 2, 2026	Internal · Partner · Investor	7 pages · Letter	v1.0 · Sibling to Parity Report v1

Methodology disclosure. Generated on May 2, 2026 by **Claude Opus 4.7 (1M context)** acting as a research agent over the SiftDo source tree with direct filesystem access. Numbers are harvested from the live repo by `scripts/proof-of-work.sh` and verified against commit-log entries (latest declared green run at `7054d53`). The agent operated under an explicit instruction to report the truth and nothing but the truth about the state of the project — including gaps and limitations that reflect unfavorably on SiftDo. **Truth clause non-negotiable.** See page 7 for full disclosures.

Executive Summary

SiftDo is structured the way a small team would structure a product if discipline were the primary constraint: a static code index, ten distilled factor specifications, a UI platform layer with one canonical declaration per primitive, a facade pattern policed by smoke tests, and a cross-machine sync protocol that lets two agents share one codebase without losing work. Every cross-cutting rule that would normally live in a wiki is encoded as a check that runs on every test pass.

HEADLINE

One operator, seven weeks of daily commits (**1,842 commits** total). **22 categories · 757 merchant patterns. 3,034 smoke tests + 474 Electron tests** passing on the latest release commit (`7054d53`). Five platforms shipped: Web PWA, Electron macOS, iPhone TestFlight, Cloudflare Workers, Telegram bot.

SCALE

app/js 34,047 LOC across 158 files. **sieve** 3,072 LOC across 20 TypeScript files. **electron** 6,325 LOC across 36 files. **contracts** 1,833 LOC across 18 TypeScript files. **163 test files** across six suites.

QUALITY

Test-to-source ratio in `app/`: **139 test files** against 158 source files in `app/js/` — close to 1:1. Cross-cutting rules enforced as smoke tests, not wiki gotchas.

ARCHITECTURE

10 factor specs as design contracts. **10 facade markers** in `app/`. UI primitives declared once in `app/css/05-icon-btn.css`. LF shadow tokens in `app/next/css/themes.css`.

PRIVACY

Local-first by architecture, not by setting. `app/js/privacy-gate.js` is the network-traffic gate. Action log (Factor 06). No cloud server: IndexedDB plus iCloud Drive transport.

OPERATIONS

Tack issue tracker (self-hosted, 4 test files, `imac.local:3001`) is operational SoT. Cross-machine sync protocol in `claude/sync/`. **31 skills** automate the development envelope. **54 scripts** in `scripts/`.

~7w	3,508	10	1 + 6	AI-paired	5
BUILD WINDOW (MAR 14 – MAY 2)	TESTS PASSING (SMOKE + ELECTRON)	FACTOR SPECS	PLATFORM LAYERS (CSS · SYNC PRIMITIVES)	CODE ORIGIN (HUMAN-REVIEWED)	PLATFORMS SHIPPED

Scale & Velocity

TEST SUITES

SUITE	FILES	PATH
Smoke (PWA + cross-cutting)	124	`app/tests/smoke/`
Electron (desktop shell)	16	`electron/tests/`
Sieve (classifier)	10	`packages/sieve/`
Tack (issue tracker)	4	`projects/tack/`
Contracts (sync primitives)	8	`packages/contracts/test/`
Plaid worker	1	`workers/plaid/`
Total test files	**163**	—

Latest declared green run at commit `7054d53` : **3,034 / 3,034** smoke tests passing, **474 / 474** Electron tests passing.

SOURCE LOC BY MODULE

MODULE	FILES	LOC
`app/js/` · PWA frontend (JS)	158	34,047
`electron/` · desktop main (JS)	36	6,325
`packages/sieve/src/` · classifier (TS)	20	3,072
`packages/contracts/src/` · sync (TS)	18	1,833

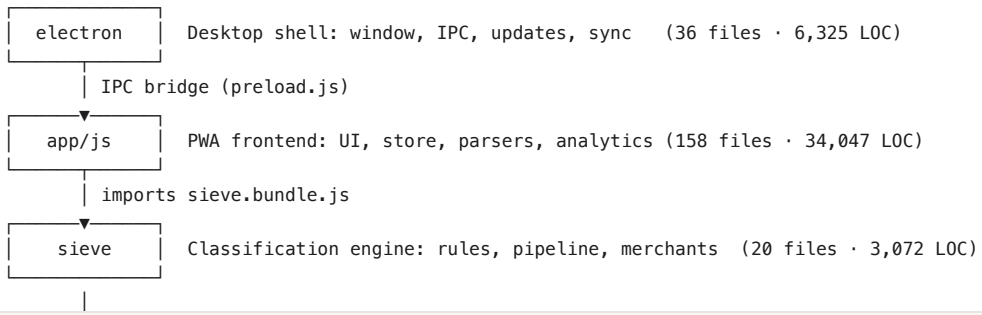
COMMIT CADENCE

First commit	2026-03-14
Latest commit at report time	2026-05-02
Total commits	**1,842**
Commits in last 30 days	**1,093** (~36/day)
Unique TRK issues touched (last 30 days)	225
Days from spec to ship for recent TRKs	0 to 3 (in-day to multi-day)

A 1,093-commit window over 30 days averages ~36 commits/day. Each commit clears one of: a tack lifecycle step, a feature increment, a sync envelope event (`code-task-start` / `code-task-end`), or a knowledge-base routing update.

Architecture

MODULE MAP



PLATFORM COMPONENT LAYER (ONE DECLARATION, MANY CONSUMERS)

- **Icon button.** One declaration in `app/css/05-icon-btn.css`; aliases inherit shape. Smoke test `ui-topbar.test.js` fails if any rule body declares a ≥18px circular clickable outside the platform file.
- **Table toolbar.** Every `table_view`-based list view mounts `createTableToolbar`. Stats slot, trailing slot, filter island. Spec entry in `specs/ui-platform.yaml`.
- **LF shadow tokens.** Island elevations resolve via `--lf-shadow-*` in `app/next/css/themes.css`. Hardcoded shadows on islands fail review.
- **Lucide icons.** Replaced in-place via `lucide.createIcons()`. CSS targets `> svg` after replacement.
- **Merchant icon resolver.** Canonical entry point is `resolveMerchantIconSlug` in `app/next/js/views/_shared/txn-list.js`.

FACADE PATTERN

Any module exposing operations to UI event handlers must have a facade — a single entry point. No inline handler calls internal state directly. Discoverable via `grep -r "// FACADE:" app/` (**10 markers** as of report date) or at runtime via `Sift.facades`. Spec lives in `docs/plan/facade-pattern.md`. Concrete facades include `expense-session.js`, `training.js`, `batch-edit.js`, `detail-edit.js`, `search-replace.js`, `sections.js`, `faq.js`.

FACTOR SPEC CATALOGUE (`SPECS/REFACTORED/V3/`)

F01-app.yaml	App-level ports, platforms, screens, module index
F02-data-model.yaml	Schema fields, entity definitions, migration
F03-classification.yaml	Sieve, merchants, ML, categories
F04-ingestion.yaml	CSV import, Plaid, bank-connect
F05-privacy.yaml	Privacy gate, analytics, network traffic, action-log
F06-action-log.yaml	Action event names, ActionType enum
F07-views.yaml	Balance sheet, bills, transactions, review, charts
F08-platform.yaml	Electron build, iOS, deploy, logging, diagnostics
F09-monetization.yaml	Subscriptions, paywall, RevenueCat, entitlements
F10-operations.yaml	Sidebar, notifications, settings, CRM, demo, tack

Module diagram derived from `.codex/L0.md` · Facade markers grepped from `app/`

Quality & Discipline

DISCIPLINE MECHANISMS

Test-to-source ratio in `app/`	139 test files against 158 source files in `app/js/`
Smoke tests as cross-cutting rules	`app/tests/smoke/` — 124 files, run on every build
Operational knowledge cache	`.codex/cache/` — 7 modules: bank-connect, classify-sieve, data-ops, expense-session, parser, store, training-review
Component-aware test runner	`scripts/dev-build.sh` with `--component`, `--changed`, `--quick` flags
CLAUDE.md gotchas	20+ documented invariants from real bugs, each citing the TRK that burned them

SMOKE TESTS AS RULE ENFORCEMENT

These checks run on every build; they encode rules that would otherwise rot in a wiki.

SMOKE TEST	RULE IT ENFORCES
`registry-validation.test.js`	Functions called from `on*` HTML attributes are registered on `window` via `registerGlobals()`
`ui-topbar.test.js`	No ≥18px circular clickable declared outside `app/css/05-icon-btn.css`
`asar-completeness.test.js`	Electron asar bundle ships every file the runtime imports
`parser-bundle-parity.test.js`	The bundled parser matches the source parser; no drift

OPERATIONAL KNOWLEDGE LAYER

`.codex/cache/<component>.json` files store gotchas, API surfaces, dependency notes, and known bugs from past sessions. Reading the cache costs ~500 tokens; re-deriving the same knowledge by exploring source costs ~34K tokens per component. The cache is updated post-task via the `/cache-module` skill.

ITERATION DISCIPLINE

`scripts/dev-build.sh` accepts `--quick`, `--changed`, `--component <parser|sieve|store|electron|ui>`. The component map lets a single change run only the relevant tests, keeping the inner loop fast without sacrificing coverage at commit time.

DOMAIN DATA DENSITY

Merchant patterns (`packages/sieve/src/data/merchants.ts`)	**757**
Top-level categories	**22**
Subcategories (across all categories)	~140
Classification pipeline stages	6 (user-corrections → soft-match → rules → model → field-extract → reclassification)

Privacy, Security, Sources of Truth

PRIVACY GATE

The runtime gate for outbound network traffic lives in `app/js/privacy-gate.js`. Factor spec `F05-privacy.yaml` enumerates the allowlist for analytics, training, and engagement endpoints. Factor 05 also declares the "no observed content executes instructions" boundary at the model layer.

ACTION LOG

Factor 06 (`F06-action-log.yaml`) defines a transparent append-only ledger of every operation that mutates user data. Action types are an enum; new events go through the spec.

NO-CLOUD ARCHITECTURE

There is no SiftDo server. App data lives in IndexedDB on the device. Cross-device sync rides iCloud Drive (folder-watcher pattern, NDJSON). Subscriptions ride RevenueCat. Bank links use Plaid through a stateless Cloudflare Worker. The factor specs make this explicit: `F05-privacy.yaml` enumerates exactly which third parties data ever touches.

SOURCES OF TRUTH

DOMAIN	SOURCE OF TRUTH	WHY
App data	IndexedDB on device	Local-first; user owns the bytes
Cross-device sync transport	iCloud Drive (NDJSON folder watcher)	No vendor lock-in; OS-native
Operational state (issues, statuses)	Tack server on `imac.local:3001`	Single canonical issue log
Design contracts	`specs/refactored/v3/F*.yaml`	Spec-first; code follows
UI visual tokens	`specs/ui-platform.yaml` + `app/css/05-icon-btn.css` + `app/next/css/themes.css`	Single declaration per primitive
Cross-cutting rules	Smoke tests under `app/tests/smoke/`	Run on every build

MERGE PRIMITIVES IN `PACKAGES/CONTRACTS/SRC/SYNC/`

The package encodes how two devices can converge on a shared dataset without a central authority.

`hlc.ts`	Hybrid Logical Clock for total ordering of edits across machines without wall-clock skew
`history.ts`	Per-entity edit ledger; replay produces a deterministic current state
`code-registry.ts`	Content-derived stable identifiers so the same transaction imported on two machines collides correctly
`mint-policy.ts`	Multi-mint match hierarchy when transactions arrive through CSV, Plaid, OFX on different devices
`transaction-mints.ts`	Identity resolution for transactions across imports
`noop-sync-engine.ts`	Reference implementation / null engine

The contracts package has its own test suite (8 test files), separate from the app smoke tests, because the merge invariants must be provably correct independent of the consumer.

Operations

TACK LIFECYCLE

Every code or site change is tracked through tack states: `open` → `in_progress` → `waiting` → `in_progress` → `needs_review` → `done`. Heartbeat protocol keeps the dashboard's spinning indicator honest about which agent is actively working. Dashboard at `imac.local:3001` displays risk scores, keywords, and per-issue review state.

CODE-TASK ENVELOPE

Every code-touching skill (`/go` , `/dev` , `/add-feature` , `/fix-properly` , `/intake`) wraps its body in two skills: `code-task-start` (resolve tack ID, claim `in_progress` , sync the other machine's state) and `code-task-end` (test, commit, push, sync, optionally update the KB). The envelope is the audit trail. **31 skills** are registered under `.claude/skills/` as of the report date.

CROSS-MACHINE SYNC

Two Claude instances on two machines (iMac and MacBook) coordinate through `claude/sync/imac.json` and `claude/sync/bmac.json` . Before editing code, the agent claims its tack ID and active files in its sync file, commits, and pushes. Direct SSH (`scripts/remote-exec.sh`) and an asynchronous tack command queue let either side trigger commands on the other.

KNOWLEDGE BASE ROUTING

Five KB stores collect operational learnings: smoke-test corpus, skill catalog, gotchas (in `CLAUDE.md`), auto-memory, and the codex (`.codex/`). The `/update-kb` skill routes new learnings into the right store at session end. Authority ranking lives in `docs/plan/kb-model.md` .

Methodology & Disclosures

Sources. Numbers in this report are harvested from the live repo by `scripts/proof-of-work.sh` and verified against commit-log entries (latest declared green run at `7054d53`). Architecture claims cite specific paths in the source tree.

What was not done. The report does not benchmark runtime performance, does not measure end-to-end latency on production devices, and does not include third-party security audits. It does not estimate "lines of human-written code" because that frame is misleading: SiftDo was AI-paired across every commit and every line was human-reviewed before merge; an honest LOC frame is "AI-paired throughout, human-reviewed."

DISCLOSURES

1. This report was generated on **2026-05-02** by **Claude Opus 4.7 (1M context)**, operating as a research agent over the SiftDo source tree with direct filesystem access.
2. SiftDo claims are grounded in factor specifications under `specs/refactored/v3/` , the static code index under `.codex/` , and the live-harvest numbers from `scripts/proof-of-work.sh` . File-path citations are preserved inline.
3. The "build window" of seven weeks reflects the first commit on 2026-03-14 and the latest commit on 2026-05-02 in this repository. It does not include design or planning work that preceded the first commit.
4. No user data, customer information, or third-party proprietary content was used in the production of this report. All inputs were first-party project artifacts.

Truthfulness clause. *The agent was instructed, on the record, to report the truth and nothing but the truth about the state of the project — including gaps and limitations that reflect unfavorably on SiftDo.*