CircuitAI - Implementation Plan

Context

Build a local-first, privacy-focused personal finance and productivity CLI app inspired by Karpathy's vision of agent-native CLIs. All data stays on the user's machine in an encrypted SQLite database. The CLI is designed to be natively usable by AI agents (via `--json` mode) while also providing a rich terminal dashboard for human use.

Designed to be generic and reusable by anyone — not hardcoded to one user's setup. All accounts, providers, and family members are configured through the CLI during setup or at any time. The goal is an open-source tool anyone can install and configure for their own household.

Key UX Principles

Morning Catchup — `circuit morning` gives you a concise briefing: bills due soon, overdue items, upcoming deadlines, today's kid activities, account alerts. This is the "what needs my attention right now" view.

Free-form Text Input — `circuit add "JCPL electric bill $142 due March 15"` or `circuit add "hockey registration for Jake $350 paid"`. The system parses natural text into structured entries. No need to remember flags.

Agent-native — Every command has `--json` mode. AI agents can use CircuitAI as a tool.

Reference Setup (author's configuration, for testing)

Banks: Chase, Bank of America, Wealthfront
Credit Cards: Amex, Citi, Bank of America
Mortgage: Townee Mortgage
Investments: Wealthfront, Titan Invest, Schwab Fund, Robinhood (recurring monthly contributions)
Retirement/Education: 401(k) accounts, Kids' 529 plans (and any other long-term savings vehicles)
Utilities: JCPL (electricity), American Water (water), Elizabethtown Gas (gas), Municipality tax (quarterly), HOA fees (yearly), Xfinity (internet)
Insurance: Auto, home, umbrella, life (yearly payments — extractable from bank/card statements)
Kids: 2 children, activities: Hockey, Soccer, Gymnastics, Tennis

Key insight: Many transactions (bill payments, investment contributions, insurance premiums, activity fees) show up on bank/credit card statements. The system supports "statement linking" — auto-matching imported transactions to known recurring items by amount + description pattern, reducing duplicate manual entry. Yearly items like insurance are particularly well-suited to this since they're easy to miss otherwise.

Tech Stack

- Python 3.11+, Click (CLI), Rich + Textual (TUI), Pydantic (models)
- SQLCipher via `sqlcipher3-binary` (encrypted SQLite)
- `keyring` for optional master password caching
- `entry_points` for plugin/adapter discovery
- Hatchling build backend, `ruff` + `mypy` for quality

Project Structure

```
CircuitAI/ ■■■ pyproject.toml ■■■ src/circuitai/ ■ ■■■ __init__.py, __main__.py ■ ■■■ cli/ # Click commands ■ ■ ■■■
main.py # Root group, --json flag ■ ■ ■■■ setup.py # First-run wizard ■ ■ ■■■ add_text.py # circuit add "free text"
(smart text parser) ■ ■ ■■■ morning.py # circuit morning (morning catchup briefing) ■ ■ ■■■ bills.py # circuit bills
{list,add,show,pay,edit,delete,summary} ■ ■ ■■■ accounts.py # circuit accounts
{list,add,show,update-balance,transactions} ■ ■ ■■■ cards.py # circuit cards
{list,add,show,update-balance,transactions} ■ ■ ■■■ mortgage.py # circuit mortgage {list,add,show,pay,amortization} ■
■ ■■■ investments.py # circuit investments {list,add,show,contribute,performance} ■ ■ ■■■ deadlines.py # circuit
deadlines {list,add,show,complete,edit,delete} ■ ■ ■■■ activities.py # circuit activities
{list,add,show,pay,schedule,costs,children} ■ ■ ■■■ dashboard.py # circuit dashboard (launches Textual TUI) ■ ■ ■■■
query.py # circuit query "..." ■ ■ ■■■ adapters_cmd.py # circuit adapters {list,info,configure,sync} ■ ■ ■■■
export.py # circuit export {csv,json} ■ ■■■ core/ # Infrastructure ■ ■ ■■■ config.py # TOML config at
~/.config/circuitai/circuit.toml ■ ■ ■■■ encryption.py # MasterKeyManager (PBKDF2-SHA512, 256K iterations) ■ ■ ■■■
database.py # DatabaseConnection (SQLCipher wrapper) ■ ■ ■■■ migrations.py # Schema versioning ■ ■ ■■■ exceptions.py
# Custom exceptions ■ ■■■ models/ # Pydantic models + Repository classes ■ ■ ■■■ base.py, bill.py, account.py,
card.py ■ ■ ■■■ mortgage.py, investment.py, deadline.py, activity.py ■ ■ ■■■ category.py, tag.py ■ ■■■ services/ #
Business logic ■ ■ ■■■ bill_service.py, account_service.py, card_service.py ■ ■ ■■■ mortgage_service.py,
investment_service.py ■ ■ ■■■ deadline_service.py, activity_service.py ■ ■ ■■■ summary_service.py # Cross-domain
aggregation for dashboard ■ ■ ■■■ morning_service.py # Morning catchup: what needs attention NOW ■ ■ ■■■
text_parser.py # Free-form text -> structured entry parser ■ ■ ■■■ query_service.py # Regex-based natural language
query parser ■ ■■■ adapters/ # Plugin system ■ ■ ■■■ protocol.py # CircuitAdapter Protocol (the contract) ■ ■ ■■■
base.py # BaseAdapter ABC (convenience) ■ ■ ■■■ registry.py # AdapterRegistry (entry_points discovery) ■ ■ ■■■
builtin/ # csv_import.py, manual.py ■ ■■■ tui/ # Textual dashboard ■ ■ ■■■ app.py # CircuitDashboard(App) ■ ■ ■■■
screens/ # unlock.py, main_dashboard.py, detail screens ■ ■ ■■■ widgets/ # summary_panel, upcoming_bills, deadlines,
accounts, cards, activities, mortgage ■ ■ ■■■ styles/dashboard.tcss ■ ■■■ output/ # Dual-mode output (Rich for
humans, JSON for agents) ■ ■■■ formatter.py # OutputFormatter ■ ■■■ json_output.py, rich_output.py ■■■ tests/
```

Data Model

All amounts stored as INTEGER cents. All IDs are UUIDs. All dates ISO 8601.

Tables: `bills`, `bill_payments`, `accounts`, `account_transactions`, `cards`, `card_transactions`, `mortgages`, `mortgage_payments`, `investments`, `investment_contributions`, `deadlines`, `children`, `activities`, `activity_payments`, `tags`, `adapter_state`, `schema_version`

Key relationships:
- `bill_payments.bill_id` -> `bills.id`
- `activities.child_id` -> `children.id`
- `bills.account_id` -> `accounts.id` (linked payment account)
- `deadlines.linked_bill_id` -> `bills.id` (auto-created from bill due dates)
- `investment_contributions.investment_id` -> `investments.id`
- `investment_contributions.source_account_id` -> `accounts.id` (which bank account funds it)

Investment accounts (`investments` table) use an `account_type` field to cover all investment vehicles: `brokerage`, `401k`, `529`, `ira`, `roth_ira`, `hsa`, `crypto`, `other`. Each tracks: institution, current value, recurring contribution amount/frequency, and linked source bank account. The `investment_contributions` table logs each contribution. For 529 plans, a `beneficiary_child_id` links to the `children` table.

When bank/card statements are imported, the system can auto-match known recurring transactions (bill payments, investment contributions, activity fees) by amount + description patterns. This "statement linking" reduces manual entry over time.

Encryption

- Master password -> PBKDF2-SHA512 (256K iterations) -> hex key
- SQLCipher `PRAGMA key` with derived key
- Verification hash stored separately (different salt)
- Optional keyring caching for session convenience
- Password never stored on disk

Agent-Native Design

- Every command supports `--json` (JSON to stdout, human messages to stderr)
- Deterministic command structure with stable UUID IDs
- Exit codes: 0 success, 1 user error, 2 system error
- No interactive prompts in `--json` mode
- Structured error output in JSON mode

Adapter/Plugin Architecture

Third-party adapters are pip packages with `entry_points`:

```
[project.entry-points."circuitai.adapters"] chase = "circuitai_adapter_chase.adapter:ChaseAdapter"
```

Protocol: `metadata()`, `configure()`, `validate_config()`, `sync(db)`, `test_connection()`

Built-in: `manual` (CLI entry), `csv-import` (CSV file import)

Morning Catchup (`circuit morning`)

The "open your terminal and know what needs attention" command:

```
$ circuit morning Good morning! Tuesday, Feb 24, 2026 NEEDS ATTENTION (3 items) [!] JCPL Electric — $142.30 due in 2 days (Feb 26) [!] Hockey registration — $350 due tomorrow (Feb 25) [!] Amex minimum payment — $89 due Mar 5 TODAY'S SCHEDULE 4:00 PM Hockey practice (Jake) — Ice Arena 5:30 PM Gymnastics (Emma) — YMCA THIS WEEK $2,890 in bills due 3 deadlines upcoming 4 activities scheduled ACCOUNTS SNAPSHOT Chase Checking: $5,200 | BofA Savings: $12,400 Amex: $1,205 / $10K | Citi: $890 / $5K
```

In `--json` mode, returns structured data with `attention_items[]`, `todays_schedule[]`, `week_summary{}`, `accounts_snapshot[]` — perfect for an AI agent to summarize or act on.

Free-Form Text Input (`circuit add "..."`)

Parse natural language into structured entries:

```
# These all work: circuit add "JCPL electric bill $142 due March 15" circuit add "paid hockey registration $350 for Jake" circuit add "dentist appointment March 20 high priority" circuit add "xfinity internet $89.99 monthly due on the 5th" circuit add "soccer practice for Emma Monday Wednesday 5-6pm at Fields Park"
```

The `TextParser` service uses regex + keyword matching to detect:
- **Entity type**: bill, payment, deadline, activity (from keywords like "bill", "paid", "due", "practice")
- **Amount**: dollar amounts (`$142`, `$89.99`)
- **Dates**: relative and absolute (`March 15`, `tomorrow`, `next Friday`, `on the 5th`)
- **Names**: matched against existing children, bills, activities
- **Recurrence**: keywords like "monthly", "quarterly", "weekly"

If ambiguous, the CLI confirms: "I'll add this as a bill: JCPL Electric, $142, due Mar 15. Correct? [Y/n]"
In `--json` mode, returns the parsed result without confirmation (agent decides).

Implementation Phases

Phase 1: Foundation

Create project skeleton, pyproject.toml, encryption, database connection, schema migrations, root CLI group, `circuit setup` wizard, OutputFormatter. Verify `pip install -e .` works.

Files: `pyproject.toml`, `src/circuitai/core/*`, `src/circuitai/cli/main.py`, `src/circuitai/cli/setup.py`, `src/circuitai/output/*`, all `__init__.py`

Phase 2: Bill Tracking

Full bill CRUD (add, list, show, pay, edit, delete, summary). Pydantic models, Repository, Service, CLI commands. Wire up `--json` for all commands.

Files: `src/circuitai/models/base.py`, `src/circuitai/models/bill.py`, `src/circuitai/services/bill_service.py`, `src/circuitai/cli/bills.py`

Phase 3: Accounts, Cards, Mortgage, Investments

All financial entity types: bank accounts with transactions, credit cards with transactions, mortgage with amortization, investment accounts with recurring contributions. Statement linking: auto-match imported transactions to known recurring items.

Files: `src/circuitai/models/{account,card,mortgage,investment}.py`, `src/circuitai/services/{account,card,mortgage,investment}_service.py`, `src/circuitai/cli/{accounts,cards,mortgage,investments}.py`

Phase 4: Deadlines & Kids Activities

Deadline tracking with urgency/priority. Kids activities with schedule, per-child/per-sport cost tracking. Auto-create deadlines from bill due dates.

Files: `src/circuitai/models/{deadline,activity}.py`, `src/circuitai/services/{deadline,activity}_service.py`, `src/circuitai/cli/{deadlines,activities}.py`

Phase 5: Summary, Query, Morning Catchup & Text Input

- Cross-domain summary aggregation
- Regex-based natural language query: "What's my electricity bill?", "When is hockey practice?"
- `circuit morning` — the attention-needed briefing
- `circuit add "free text"` — smart text parser for quick entry
- `TextParser` service with regex + keyword detection for entity type, amounts, dates, names

Files: `src/circuitai/services/{summary,query,morning}_service.py`, `src/circuitai/services/text_parser.py`, `src/circuitai/cli/{query,morning,add_text}.py`

Phase 6: Adapter/Plugin System

Protocol class, BaseAdapter, AdapterRegistry with entry_points discovery. Built-in CSV import adapter. `circuit adapters` CLI.

Files: `src/circuitai/adapters/*`, `src/circuitai/cli/adapters_cmd.py`

Phase 7: Textual Dashboard

Full TUI dashboard with panels: Financial Summary, Upcoming Bills, Deadlines, Bank Accounts, Credit Cards, Kids Activities, Mortgage. Keyboard navigation, drill-down screens, auto-refresh.

Files: `src/circuitai/tui/*`, `src/circuitai/cli/dashboard.py`

Phase 8: Export & Polish

CSV/JSON export, comprehensive error handling, tests.

Files: `src/circuitai/cli/export.py`, `tests/*`

Verification

---

1. `pip install -e .` succeeds
2. `circuit --help` shows all subcommands
3. `circuit setup` creates encrypted DB and config
4. `circuit bills add --name "JCPL Electric" --provider "JCPL" --category electricity --amount 150 --due-day 15` works
5. `circuit bills list --json` returns valid JSON
6. `circuit dashboard` launches TUI with all panels
7. `circuit query "what's my electricity bill?"` returns JCPL bill info
8. External adapter can be installed and discovered via `circuit adapters list`

Seed Data (User's Accounts)

On first setup or via a seed script, we'll pre-populate with the user's known accounts:

Bills: JCPL Electric, American Water, Elizabethtown Gas, Municipality Tax (quarterly), HOA Fees (yearly), Xfinity Internet, Auto Insurance (yearly), Home Insurance (yearly), Umbrella Insurance (yearly), Life Insurance (yearly)
Accounts: Chase (checking/savings), Bank of America, Wealthfront
Cards: Amex, Citi, Bank of America
Mortgage: Townee Mortgage
Investments: Wealthfront (brokerage), Titan Invest (brokerage), Schwab Fund (brokerage), Robinhood (brokerage), 401(k), Kids' 529 plans
Children: 2 kids (names to be provided during setup)
Activities: Hockey, Soccer, Gymnastics, Tennis