Flagium - One-Page App Summary

Evidence source: repository files (README.md, api/, ingestion/, engine/, db/, ui/).

What It Is

Flagium is a rule-based financial risk intelligence application that flags early signs of deterioration in listed companies using ingested financial filings and deterministic checks. It combines a FastAPI backend, data ingestion pipeline, and React dashboard for analysis and monitoring.

Who It Is For

Primary users are long-term investors, portfolio managers, and analysts monitoring company-level and portfolio-level financial risk.

What It Does

- Ingests company filings (NSE first, BSE fallback), parses XBRL, and stores normalized financial records.
- Runs modular red-flag checks (for example: profit collapse, low interest coverage, OCF vs PAT divergence).
- Exposes REST APIs for dashboard, companies, flags, scans, and ingestion triggers.
- Supports user registration/login, JWT auth, and profile/password management.
- Provides portfolio CRUD, holdings-level risk scoring, deltas, and escalation feeds.
- Includes admin endpoints for ingestion coverage checks and data sanity reporting.
- Serves a React SPA with public pages plus protected views (dashboard, market monitor, companies, flags, portfolio, reports).

How It Works (Compact Architecture)

UI: React + Vite app (`ui/src/App.jsx`) calls backend at `http://localhost:8000/api` (`ui/src/api.js`).

API Layer: FastAPI app (`api/server.py`) mounts routers for core data routes, auth, portfolios, and admin.

Ingestion: `ingestion/ingest.py` orchestrates ticker fetch, filing download, XBRL parse (`ingestion/xbrl_parser.py`), and DB writes (`ingestion/db_writer.py`).

Flag Engine: `engine/runner.py` loads flag modules from `flags/`, executes checks per company/period, and inserts `flags` rows.

Data Store: MySQL schema in `db/bootstrap.sql` with `companies`, `financials`, `flags`, `users`, `portfolios`, and `portfolio_items`.

Data Flow: Exchange filings -> parser -> `financials` table -> flag runner -> `flags` table -> API queries -> dashboard/portfolio screens.

Not found in repo: production orchestration manifests (for example Docker Compose/Kubernetes files).

How To Run (Minimal Getting Started)

- Provision MySQL locally and initialize schema: run `db/bootstrap.sql`.
- Create backend venv and install deps: `python3 -m venv venv` and `pip install -r requirements.txt`.
- Install frontend deps: `cd ui && npm install`.
- Start both apps with bundled script: `./start.bash` (backend :8000, frontend :5173).
- Alternative manual start: `uvicorn api.server:app --reload --port 8000` and `npm run dev -- --host` in `ui`.

Note: DB credentials are hardcoded in `db/connection.py` for local setup (`localhost`, `flagium_user`, `flagium`).