Global Portfolio One (GPO) ? Codebase Summary

Purpose
GPO is a Flask web app that replicates a regime-aware investment process (?Welt AG?). It combines market stress detection, allocation logic, and a dashboard UI to show portfolio targets in real time.

Runtime Architecture
? Backend: `app.py` exposes HTML and JSON APIs.
? Domain engine (`engine/`):
  - `regime.py`: classifies market state A/B/C using drawdown, credit spread, and VIX.
  - `allocator.py`: translates regime + portfolio value into position-level targets for 6-ETF and simplified 3-ETF models.
  - `market_data.py`: fetches price/VIX/spread data with cache and synthetic fallback.
  - `recovery.py`: computes C?B and B?A recovery thresholds from trough prices.
  - `config.py`: central constants (weights, ETF metadata, thresholds, symbols).
? Frontend: `templates/index.html` + `static/js/app.js` + `static/css/style.css` render dashboard cards, charts, simulator, and allocation tables.

Core Business Logic
1. Market data is loaded (`/api/dashboard`) and drawdown is computed vs running ATH.
2. Regime detection rules:
   - A (Normal): default, 80/20 equity/reserve.
   - B (Equity Scarcity): drawdown ? -20% with stress confirmation.
   - C (Escalation): drawdown ? -40% plus extreme/confirmed stress.
3. Allocation engine computes:
   - Absolute ETF target weights and EUR values.
   - Rebalance trades from optional current holdings.
   - Weighted TER estimate.
   - Simplified 3-ETF targets scaled to regime.
4. Recovery module returns price checkpoints and progress percentages for de-risking path back to A.

APIs
? `GET /` dashboard page.
? `GET /api/dashboard` market snapshot + regime + recovery levels.
? `POST /api/allocate` live-regime allocation for submitted portfolio value/holdings.
? `GET /api/reference` static ETF and weight tables.
? `POST /api/simulate` hypothetical-regime allocation without live triggers.

Data & Reliability Characteristics
? Uses `yfinance` for market prices/VIX and FRED for BBB OAS (if key exists).
? 30-minute in-memory cache reduces repeated remote calls.
? If live feeds fail and `GPO_DEMO=1`, synthetic data keeps UI/API functional.
? Unit tests in `tests/test_engine.py` validate regime, allocation, and recovery behavior.

Operational Notes
? Start: `uv run python app.py`
? Test: `uv run pytest tests/ -v`
? Deploy: `uv run gunicorn app:app --bind 0.0.0.0:8000 --workers 2`