Product description

Product description

Name: Opportunity Radar
One-liner: A real-time, agent-driven market radar for crypto that surfaces actionable signals (momentum, RVOL, VWAP skews, trend probability) across multiple symbols with automated alerting and lightweight posture logic.

Who it's for

- **Discretionary traders** who want timely, high-signal alerts without staring at charts.
- **Quants/PMs** who want a pluggable signal bus and a thin UI for triage.
- **Ops/Dev** who need health & diagnostics endpoints for reliability.

Problems it solves

- Consolidates heterogeneous micro-signals into a normalized "finding" stream.
- Reduces noise via configurable thresholds, persistence rules, and multi-TF features.
- Exposes fast diagnostics (`/debug/*`) so data issues are debuggable in seconds.

Key outcomes

- <1s p50 API latency for tiles/diagnostics.
- Agents run at fixed cadences (5-60s) and publish findings with consistent schemas.
- Real-time tiles per symbol: price vs VWAP (bps), momentum (1m/5m), RVOL (5 vs 20), trend score probability, last top-of-book, and "Latest Findings".

Core features

- **Multi-symbol:** BTC-USD, ETH-USD, ADA-USD (extensible via env).
- **Agent framework:** `rvol_spike`, `cvd_divergence`, `trend_score`, `llm_analyst` (+ optional reversal agents).
- **Finding stream:** normalized rows `{ts, agent, symbol, score, label, details JSON}`.
- **Tiles API:** `/signals` (legacy) and `/signals_tf` (bar-based).
- **Debug API:** `/debug/state`, `/debug/features` for direct introspection.
- **Health:** `/health`, `/db-health`, `/agents`.
- **Alerting:** optional Slack webhook with posting policy.
- **CSV export:** findings per symbol (download from UI).

Technical specification

Technical specification

Architecture (high level)

Frontend: Static HTML/JS dashboard (single page) that calls backend JSON endpoints with `?symbol=` and auto-refreshes every 5-10s. "Run agents now" triggers a backend kick.

Backend (FastAPI + Uvicorn):
- **State layer (`backend/state.py`):** in-memory deques per symbol for trades; caches for best quotes & last price; posture cache; recent in-memory findings buffer.
- **Bar/feature layer (`backend/bars.py`):** builds 1m/5m/15m bars from trade cache; computes ATR, momentum (bps), price vs VWAP (bps), RVOL ratios.
- **Signals layer (`backend/signals.py`):** assembles tile data from bars + quotes; multi-TF endpoint payloads.
- **Scheduler (`backend/scheduler.py`):** async loop that runs agents by interval, writes findings to DB (or in-mem fallback), updates posture state.
- **Agents (`backend/agents/*.py`):** each returns an optional finding dict. TrendScore computes p_up from simple features and applies persistence rules to posture.
- **DB layer (`backend/db.py`):** asyncpg connection helpers, `insert_finding`, `pg_exec/pg_fetch*`.
- **Services:** Slack webhook client (uses httpx), macro watcher (optional).

Database (PostgreSQL):
```
findings(id, ts_utc, agent, symbol, score, label, details jsonb)
position_state(symbol pk, status, qty, avg_price, updated_at, last_action, last_conf)
```

Deployment: Containerized, K8s/DO App Platform. Readiness = `/health`. Environment variables configure symbols, thresholds, LLM, Slack, DB.

Key endpoints (selection)

Endpoint	Method	Params	Purpose	Returns (shape)
`/health`	GET	-	Liveness; trades cached; last run per agent	`{status, symbols[], trades_cached{sym:n}, agents{name:{status,last_run}}}`
`/db-health`	GET	-	DB connectivity	`{ok: bool, error?: string}`
`/agents`	GET	-	Agent registry + last run	`{agents:[{agent, last_run}]}`
`/agents/run-now`	POST	`names=csv, symbol, insert=true`	Force a run	`{ok, ran:[name], results:[{agent, error?}]}`
`/findings`	GET	`symbol, limit`	Latest findings	`{findings:[{ts_utc,agent,symbol,score,label,details}...]}`
`/signals`	GET	`symbol`	Legacy tile signals	`{mom1_bps, mom5_bps, rvol_vs_recent, px_vs_vwap_bps, best_bid, best_ask, last_price, trades_cached}`
`/signals_tf`	GET	`symbol`	Bar-based multi-TF	`{mom_bps_1m/5m/15m, px_vs_vwap_bps_*, rvol_*, atr_*}`
`/debug/state`	GET	`symbol`	Raw state	

Technical specification (cont.)

| `{symbol, trades_cached, last_trade[], last_trade_age_s, bars_1m_count, bars_1m_tail[]}` |
| `/debug/features` | GET | `symbol` | Computed features
| `{symbol, bars:{1m,5m,15m}, mom_bps_1m, mom_bps_5m, px_vs_vwap_bps_1m, atr_1m ...}` |

Data contracts

Finding
```json
{
  "ts_utc": "2025-08-14T19:27:18.854Z",
  "agent": "rvol_spike",
  "symbol": "BTC-USD",
  "score": 7.16,
  "label": "rvol_spike",
  "details": { "rvol": 7.16, "volume_5m": 533.7, "best_bid": 100.52, "best_ask": 100.59 }
}
```

Signals (legacy)
Numbers are finite floats (no NaN/Inf) or null for quotes:
```json
{
  "mom1_bps": 12.3, "mom5_bps": -8.2,
  "rvol_vs_recent": 1.45,
  "px_vs_vwap_bps": 23.1,
  "best_bid": 100.52, "best_ask": 100.59,
  "last_price": 100.57,
  "trades_cached": 482
}
```

Signals TF
```json
{
  "mom_bps_1m": -16.4, "mom_bps_5m": -19.5, "mom_bps_15m": 0.0,
  "px_vs_vwap_bps_1m": 19.4, "px_vs_vwap_bps_5m": 19.4, "px_vs_vwap_bps_15m": 19.4,
  "rvol_1m": 0.92, "rvol_5m": 1.31, "rvol_15m": 0.74,
  "atr_1m": 0.53, "atr_5m": 0.62, "atr_15m": 0.0
}
```

Configuration (env)

- **Symbols:** `SYMBOL="BTC-USD,ETH-USD,ADA-USD"`
- **DB:** `DATABASE_URL=postgres://...`
- **Slack:** `ALERT_WEBHOOK_URL`, `SLACK_ANALYSIS_ONLY=true|false`, `ALERT_VERBOSE=true|false`
- **LLM:** `LLM_ENABLE`, `OPENAI_MODEL`, `OPENAI_API_KEY`, `LLM_MIN_INTERVAL`, `LLM_ALERT_MIN_CONF`, `LLM_ANALYST_MIN_SCORE`
- **Trend/posture:** `TS_INTERVAL`, `TS_ENTRY`, `TS_EXIT`, `TS_PERSIST`, `TS_WEIGHTS`, `TS_MTF_WEIGHTS`
- **Features:** `FEATURE_BARS`, `FEATURE_NEW_TREND`, `FEATURE_REVERSAL`, `FEATURE_LIQUIDITY`
- **Macro watcher:** `MACRO_WINDOWS`, `MACRO_WINDOW_MINUTES`, `MACRO_RVOL_MIN`, `MACRO_INTERVAL_SEC`

Technical specification (cont.)

Dependencies

Python (3.11):
`fastapi, uvicorn[standard], httpx, asyncpg, psycopg2-binary, python-dateutil, pydantic (v2), numpy (optional), orjson (optional)`

(If you prefer aiohttp, add it explicitly; current Slack client uses httpx.)

Container run

```
uvicorn backend.main:app --host 0.0.0.0 --port 8080 --proxy-headers --forwarded-allow-ips='*'
```

Operational concerns

- **Readiness:** `/health` must not import heavy modules at request time; it reads last-run timestamps from scheduler memory.
- **Fallbacks:** If DB insert fails, finding is appended to in-memory `RECENT_FINDINGS` (exposed via `/findings` when DB is down).
- **Numeric hygiene:** all tile numbers are finite (no NaN/Inf); quotes may be null.
- **Backpressure:** trades deque bounded to 50k per symbol.
- **Observability:** per-agent `heartbeat(name, status)` and timestamps in `/agents` & `/health`.

Requirements

Requirements document

Functional requirements (FR)

FR1 - Symbol isolation
- The dashboard's selected symbol drives every API call via `?symbol=SYMBOL`.
- All tiles, trend score, momentum, RVOL, and "Latest Findings" update to the selected symbol.
- Browser refresh preserves the last selected symbol (hash or localStorage).

FR2 - Latest Findings stream
- Agents publish findings to DB; `/findings?symbol=...&limit=n` returns the most recent n rows for that symbol.
- When DB is unavailable, `/findings` returns from in-mem buffer.
- UI auto-refreshes findings every <=5s and prepends new rows without clearing the list.

FR3 - Momentum & VWAP tiles
- `/signals_tf` computes `mom_bps_1m` and `mom_bps_5m` from fresh bars. Both update every request.
- `/signals` exposes a single-TF "fast path" (1m) for legacy tiles.

FR4 - Trend Score (p_up)
- TrendScoreAgent runs at `TS_INTERVAL` seconds per symbol and computes `p_up`, `p_1m`, `p_5m`, `p_15m`.
- Findings are inserted with `label="trend_score"` and `details.p_up` in [0,1].
- UI tile shows `p_up` and three sub-probs.

FR5 - Run-now
- `/agents/run-now?names=list&symbol=...&insert=true` triggers immediate runs, returns per-agent results (or errors), and inserts findings when requested.

FR6 - Diagnostics
- `/debug/state` and `/debug/features` return non-error payloads for any configured symbol.

FR7 - Health
- `/health` returns `{status:"ok"}` and agent `last_run` ISO timestamps even when DB is down.

FR8 - CSV export
- UI triggers a CSV download of findings for the current symbol and date window.

Non-functional requirements (NFR)

- **Latency:** p50 < 150ms for `/signals*` and `/findings` at 3 symbols.
- **Availability:** >=99.9% for read endpoints; graceful degradation when DB/LLM/Slack are unavailable.
- **Safety:** input validation; finite numeric outputs; no unhandled exceptions leak stack traces in prod.
- **Maintainability:** linters, type hints, clear module boundaries (state/bars/signals/scheduler/agents).
- **Observability:** structured logs with agent name, symbol, and durations; `heartbeat()` status.

Acceptance criteria (quick tests)

Requirements (cont.)

Symbol wiring
```bash
curl -s "$BASE/signals_tf?symbol=BTC-USD" | jq '.mom_bps_1m,.px_vs_vwap_bps_1m'
curl -s "$BASE/signals_tf?symbol=ETH-USD" | jq '.mom_bps_1m,.px_vs_vwap_bps_1m'
# Values must differ over time; not a static copy.
```

Findings visible
```bash
curl -s "$BASE/agents/run-now?names=rvol_spike,trend_score&symbol=BTC-USD&insert=true" | jq .
curl -s "$BASE/findings?symbol=BTC-USD&limit=5" | jq '.findings[0]'
# Expect the just-inserted agent in .findings[0].agent
```

Momentum updates (1m & 5m)
```bash
for s in 1 2 3; do curl -s "$BASE/signals_tf?symbol=BTC-USD" | jq '.mom_bps_1m,.mom_bps_5m';
sleep 5; done
# 1m and 5m both change over subsequent calls.
```

Trend score populated
```bash
curl -s "$BASE/findings?symbol=ETH-USD&limit=10" | jq '.findings[] |
select(.agent=="trend_score") | .details.p_up' | head
# Values exist and are in [0,1].
```

DB fallback
```
# Stop DB or set invalid DATABASE_URL; run an agent; verify /findings still returns
# a recent item (from in-mem buffer) and /db-health says {ok:false}.
```

Status & next steps

What has been built so far

Working
- Backend boots and serves `/health`, `/agents`, `/signals_tf`, `/debug/state`, `/debug/features`.
- Trade cache fills; bars build for 1m/5m/15m; features (momentum, VWAP deviation, ATR) compute and are finite.
- Agents framework runs on schedule; `rvol_spike` and `cvd_divergence` execute and can emit findings.
- TrendScore agent implemented to use `compute_signals()` (no private imports); posture guard scaffolding present.
- `/health` summarizes agent `last_run` without importing scheduler state at request time.
- Frontend dashboard renders tiles, "Latest Findings" list, top-of-book, trades cached, basic agent health.

Partially working / open issues
- **Symbol selection in UI** - selection doesn't consistently propagate to all fetches. Some tiles appear static across symbols.
  Likely cause: one or more fetches omit `?symbol=${current}` or state is not synchronized before refresh.
- **"Latest Findings" not updating** - agents run but list doesn't reflect new rows promptly.
  Causes: DB inserts failing (e.g., connectivity) and UI not falling back to in-mem results; fetch interval/debouncing; response filtered for wrong symbol.
- **Momentum (1m) tile** - previously only 5m updated.
  Root cause was building/reading bars only at 5m for that tile; fix is to call `momentum_bps` on the 1m bar set in `/signals_tf` and surface it in the UI.
- **Trend Score empty** - originally due to TrendScore importing non-existent private helpers (`_dcvd`/`_rvol`/`_mom_bps`).
  Now fixed to use `compute_signals()`, but verify it's in the agent registry and running per symbol with inserts enabled.
- **Slack dependency** - import error (`aiohttp`) surfaced. Slack service now uses **httpx** to avoid extra runtime deps; confirm httpx is in requirements.
- **DB health** - `/db-health` returned `no_connection` earlier; when DB is down, findings should still appear via `RECENT_FINDINGS` fallback.
- **State globals** - historical issues with `_best_quotes` and duplicate `best_px` definitions are resolved by centralizing in `state.py`.

Housekeeping done
- Defensive numeric casting (`_num`) across signals/TF endpoints (never NaN/Inf).
- RVOL window corrected to (5 vs 20) with safe division: `return (v_recent / v_base) if v_base else 0.0`
- Scheduler's agent list unified; `list_agents_last_run()` used by `/health`.
- In-memory fallback for findings when DB insert fails.

Recommended immediate next steps
1. **Front-end symbol plumbing audit**
   Ensure every fetch appends `?symbol=${selected}`; centralize `getSymbol()`; store in hash or localStorage; re-render on change. Add a debug badge that echoes the symbol each tile used.
2. **Findings pipeline**
   Log DB insert errors; on UI, if `/findings` is empty but `/agents/run-now` shows success, hit a `/findings?source=fallback` path or let the server return merged (DB + in-mem) results by default.
3. **TrendScore visibility**

Status & next steps (cont.)

Confirm it's in `AGENTS`, runs every `TS_INTERVAL`, and inserts (`insert_finding`) with `details.p_up`. Add tile call to a dedicated endpoint (or reuse `/findings` with last `trend_score`) for the top-left card.

4. **Requirements file**
   Pin: `fastapi~=0.111`, `uvicorn[standard]~=0.30`, `httpx~=0.27`, `asyncpg~=0.29`, `psycopg2-binary~=2.9`, `python-dateutil~=2.9`, `pydantic~=2.8`, `orjson~=3.10` (optional).
5. **Playbooks**
   Add a Make target: `make diag` to run the import/symbol auditor and curl smoke tests from the Requirements' acceptance section.