📌 Codex PR Prompt — v2: Production-Grade Trading System Hardening

Title

Institutional-grade platform hardening: durable OMS + simulation + cost-aware evaluation + robust RL governance + alerting + manifest signing + global data validation + DB migrations (no shims)

📌 Repository Context

You are working on the repository dmorazzini23/ai-trading-bot on the main branch. The current architecture includes:

- ML and RL modules with a demo-style environment.
- Execution engine and preliminary cost awareness.
- Basic feature building, strategy logic, and evaluation harness.

However, the system is missing several **institutional-grade operational and governance layers**, such as:

- A persistent, restart-safe OMS with exactly-once semantics.
- Execution simulation used for live evaluation and RL training.
- Strict feature/schema enforcement and manifest signing.
- RL governance with multi-seed promotion criteria.
- Automated alerting/SLO breach routing.
- Global data provenance enforcement across all runtime paths.
- Database migration/versioning workflow (Alembic-like).

Goal: Implement all missing layers in a clean, direct way without shims, abiding by your existing architectural boundaries and style.

📌 Problem Statement

Enhance the robustness, reliability, and production readiness of ai-trading-bot by implementing:

1. A **durable OMS** with persistent intents, deterministic IDs, reconciliation, and restart safety.
2. A **simulated execution layer** for realistic fills (latency, spread, slippage, partial fills).
3. **Cost-aware evaluation** using simulated executed trades for ML and RL evaluation.
4. A **portfolio-based RL environment** with real cost, constraints, and a unified state vector.
5. **RL governance hardening** with multi-seed training and promotion criteria.
6. **Artifact validation** (feature schema and dataset hash) at RL model load and fail-fast if mismatched.
7. **End-to-end alert automation** for critical SLO breaches with webhook/email and kill switch integration.
8. **Manifest signing/verification** for models and artifacts.
9. **Global data provenance enforcement** across live, replay, and evaluation pipelines.
10. A **DB migration/versioning system** for the durable OMS and other schemas.

This should be done without any fallback stubs or shim layers. Missing dependencies for a code path should fail fast or skip with explicit test markers, not silently degrade behavior.

📌 Scope of Work

1. Durable OMS with Persistent Intents

Add modules:

- ai_trading/oms/models.py
- ai_trading/oms/intent_store.py

- ai_trading/oms/idempotency.py

Requirements:

- Store intents persistently in Postgres (use psycopg or equivalent).
- client_order_id must be deterministic.
- State machine tracks:

CREATED → SUBMITTED → ACKED → PARTIAL_FILLED* → FILLED | CANCELED | REJECTED

- On restart:
    - Intents not in final state trigger reconciliation with broker state.
- Include integration tests and SQLite fallback for local testing.

2. Execution Simulator

Add module:

- ai_trading/execution/simulator.py

Features:

- Spread modeling
- Latency
- Slippage (parametric + volatility + size/participation)
- Partial fills with liquidity behavior
- Distinct arrival vs fill price

Expose:

- SimExecutionResult with fill events, realized cost, slippage, and fees.
- Deterministic simulation when seeded.

3. Cost Model & Feedback Loop

Add module:

- ai_trading/execution/cost_model.py

Features:

- Per-symbol EWMA model for spread/slippage
- Volatility conditioning
- Size/participation adjustments

Integrate with:

- Simulator
- Evaluation
- RL env
- Live execution logs

4. Evaluation — Realized Execution Backtest

Modify and add:

- ai_trading/evaluation/walkforward.py
- ai_trading/evaluation/purged_cv.py
- ai_trading/evaluation/metrics.py

Requirements:

- Use the simulator for equity generation (no proxy curves).
- Implement purged and embargoed cross-validation.
- Report cost-adjusted metrics (net returns, turnover, drawdown, sharpe, etc.).

5. Portfolio-Based RL Environment

Modify:

- ai_trading/rl_trading/env.py

Requirements:

- Action space = target position fraction.
- Portfolio state (cash, positions, leverage, exposure).
- Reward = net-of-cost realized equity delta + risk penalties.
- Use simulator for fills.
- Remove any stub RL classes; use real stable-baselines3 APIs.
- Use unified feature builder for state.

6. Unified Feature Builder with Governance

Add:

- ai_trading/features/builder.py

Requirements:

- Central feature builder for live, training, RL env, evaluation.
- Generate feature_schema_hash and dataset_fingerprint.
- Feature schema changes must be versioned.
- Validate feature hash in live, eval, RL load — fail fast on mismatch.

7. RL Governance & Promotion Criteria

Add:

- ai_trading/rl/governance.py

Define criteria:

- Multi-seed runs (default 5–10 seeds)
- Aggregated:
    - mean Sharpe
    - std(Sharpe) < threshold

    - cost-adjusted net returns positive
    - max drawdown below threshold
    - turnover within bounds

Behavior:

- Pass/fail gating for promotion
- Output structured metrics and reasons for rejection.

8. Alert Routing Automation

Add:

- ai_trading/monitoring/alerts.py

Trigger on:

- OMS reconciliation mismatches
- Execution cost spikes
- Evaluation anomalies
- RL governance fails
- Backtest/eval SLO breaches

Actions:

- Webhooks (configurable)
- Email/SMTP
- Kill switch or auto-rollback to previous model

9. Manifest Signing & Verification

Implement:

- Ed25519/HMAC signing for model manifests.

Behavior:

- Sign manifest on training/eval completion.

- Verify on model load.
- Fail fast on invalid signature.

10. DB Migration Hardening

Add:

- Alembic or equivalent migration tooling for Postgres schemas.

Requirements:

- Migrations for OMS tables
- Version table
- Upgrade/downgrade scripts
- Test migrations

📌 Acceptance Criteria

No shims

- No stub or fake behavior anywhere.
- Code paths missing dependencies must fail with clear errors.

OMS

- Fully persistent store
- Restart safety and reconciliation
- Deterministic client IDs
- Verified with tests

Simulator

- Realistic fills
- Configurable cost parameters
- Used in RL and evaluation

Evaluation

- Walk-forward + purged CV
- Cost-adjusted metrics
- No proxy curves

RL

- Portfolio env
- Real cost reward
- Multi-seed governance
- Fail-fast feature hash checks

Alerting

- Alerts wired to webhook/email
- Kill switch integration

Manifest signing

- Verified on load
- Signed at training

Provenance

- Feature hash everywhere
- Fail-fast everywhere

DB Migrations

- All Postgres schemas versioned
- Migration scripts included

📌 Change Details (Files)

New Modules

- ai_trading/oms/*
- ai_trading/execution/simulator.py
- ai_trading/execution/cost_model.py
- ai_trading/evaluation/*.py
- ai_trading/rl/governance.py
- ai_trading/features/builder.py
- ai_trading/monitoring/alerts.py
- Migration set (e.g., migrations/)

Modified

- ai_trading/execution/engine.py
- ai_trading/rl_trading/env.py
- ai_trading/rl_trading/train.py
- ML training/inference entrypoints
- Any live startup scripts loading models
- README/Docs

📌 Constraints & Standards

- Follow existing logging practices.
- No broad exception swallows.
- Use required imports only; skip tests via markers if RL extras missing.
- Fail fast on mismatches.

📌 Implementation Requirements

client_order_id Standard

CIDv2:{strategy}:{symbol}:{side}:{session}:{horizon}:{signal_hash}:{qty}:{version}

Reconciliation Logic

- Persist intents
- On startup, resolve via broker open orders/fills
- Mark foreign orders carefully

Simulator Models

- Spread + slippage + latency + partials
- Predictable via seed

📌 Testing

Example CLI:

```
pytest -q --disable-warnings
PYTEST_DISABLE_PLUGIN_AUTOLOAD=1 pytest -q -k "oms or simulator or rl"
```

Postgres:

```
export AI_TRADING_OMS_DSN="postgresql://..."
pytest -q -k "postgres"
```

Evaluation:

```
python -m ai_trading.evaluation.walkforward --symbols SPY --start
2024-01-01 --end 2024-07-01
```

RL:

```
python -m ai_trading.rl_trading.train --algo sac --seeds 5 --env-
config ...
```

📌 Non-Goals

- GUI dashboards
- Exchange live simulators beyond fill realism
- External dependencies beyond approved stack

End of Prompt