Codex Task Prompt (full, with appendices)

Title

Execution hardening: stabilize pending-order handling (no oscillation), reduce duplicate intents, improve pacing-cap semantics, enrich pending-order telemetry (counts/age/samples + transition-based warnings), fix noisy model-liveness alerts (phase-aware + diagnosable), and remove redundant env knobs in code (canonical keys only, fail-fast on deprecated)

Repository Context

Repo: ai-trading-bot (Dom)

Primary runtime: systemd service ai-trading.service on Ubuntu droplet using .env and generated .env.runtime.

Key modules involved:

- ai_trading/core/bot_engine.py (cycle orchestration, pending order gating, symbol blocking, signal emission hooks)
- ai_trading/execution/live_trading.py (execution engine, pending-new timeout policy, pacing cap, duplicate intent gate)
- ai_trading/main.py (cycle loop, alert emission including ALERT_MODEL_LIVENESS)
- ai_trading/monitoring/model_liveness.py (liveness evaluation, snapshot helpers)
- ai_trading/position_sizing.py + ai_trading/config/runtime.py + ai_trading/settings.py (env resolution + deprecated keys)
- ai_trading/utils/env.py, ai_trading/diagnostics/env_diag.py, ai_trading/broker/alpaca_credentials.py, ai_trading/validation/validate_env.py (Alpaca env usage + aliases)

Tests (existing targets likely touched):

- tests/bot_engine/*pending*

- tests/execution/test_execution_runtime_controls.py
- tests/execution/test_order_pacing_cap.py
- plus new tests described below

Observed operational symptoms (journalctl):

- Frequent PENDING_ORDERS_STILL_PRESENT warnings and repeated PENDING_ORDERS_POLICY_APPLIED
- Block decay “release” can occur while orders are still open (PENDING_SYMBOL_BLOCK_DECAY_RELEASED then PENDING_ORDERS_STILL_PRESENT)
- Regular cycle_duplicate_intent
- ORDER_SUBMIT_SKIPPED_DETAIL with reason=order_pacing_cap happens often
- ALERT_MODEL_LIVENESS fires constantly as ERROR, creating noise that can hide real faults

Operator debugging needs (must address):

When PENDING_ORDERS_STILL_PRESENT fires, logs must include:

- open_count
- pending_count (or whatever internal classification is used)
- oldest_open_age_s
- bounded sample_orders (ids/statuses + timestamps/age)

And ideally:

- WARN only on backlog state transitions (start), then periodic heartbeat every N minutes while backlog persists.

Droplet findings to fold in (new):

- .env and service env have been cleaned so legacy keys like ALPACA_BASE_URL, AI_TRADING_MAX_POSITION_SIZE, AI_TRADING_MAX_DRAWDOWN_THRESHOLD, and NEWS_API_KEY are no longer present. Code must now **stop accepting deprecated keys silently** and either (a) fail-fast with actionable message or (b) keep only canonical reads (NO SHIMS).
- Multiple Alpaca URL keys still exist and are being referenced across code; the droplet currently sets:

    - ALPACA_TRADING_BASE_URL=https://paper-api.alpaca.markets
    - ALPACA_API_URL=https://paper-api.alpaca.markets
    - ALPACA_BASE_URL=https://paper-api.alpaca.markets (legacy)
    - ALPACA_DATA_BASE_URL=https://data.alpaca.markets
- CLI operator attempt python -m ai_trading --print-config failed ("unrecognized arguments"), so there is currently no first-class "print resolved config" pathway.

Problem Statement

Pending-order handling is safe, but:

- Too noisy (warnings for normal open-limit behavior)
- Oscillatory (block decay can release while orders persist → reblock next cycle)
- Inefficient (duplicate intents leak into execution)
- Over-coupled to pacing (maintenance/duplicates contribute to "new order" pacing cap)

Additionally, model liveness alerting is broken operationally:

- ALERT_MODEL_LIVENESS spams ERROR repeatedly (appears ~every few minutes), likely because liveness is evaluated when signals are not expected, or signal "note" functions are not called, or liveness ignores execution phases / warmup / market-closed logic.
- Alerts are not diagnosable (no structured payload, and snapshot isn't surfaced in health/logs).

Finally, env redundancy is still present in code paths:

- Deprecated keys are still read in leaf modules (e.g., ai_trading/position_sizing.py reading deprecated AI_TRADING_MAX_POSITION_SIZE).
- Alpaca base URL keys exist in multiple forms; this makes .env harder to reason about and creates inconsistency risk.

Scope of Work

A) Make pending-new timeout policy idempotent per engine cycle

Goal: Avoid running pending-new timeout policy multiple times in one engine cycle.

In ai_trading/execution/live_trading.py:

- Add guard so _apply_pending_new_timeout_policy() runs at most once per engine_cycle_index.
- Track self._pending_new_policy_last_cycle_index: int | None.
- If equal to current cycle index, return early.
- Set it when policy runs.
- Ensure tests can call _apply_pending_new_timeout_policy() directly:
    - Provide a safe way to reset internal state in tests (e.g., a private reset helper in the class and only used by tests).

B) Fix block decay oscillation: never “decay release” while orders are still open

Goal: Stop release→reblock loop.

In ai_trading/core/bot_engine.py:

- Only release a symbol block due to decay if there are **zero open orders** for that symbol.
- If open orders still exist, keep the block and log:
    - PENDING_SYMBOL_BLOCK_DECAY_DEFERRED with structured payload:
        - symbol
        - open_orders_count
        - oldest_open_order_age_s

C) Reduce noise + add telemetry for pending/open orders (structured + severity gated)

Goal: Make warnings meaningful, and make backlog diagnosable.

In ai_trading/core/bot_engine.py pending-order gating/logging path:

C1) Compute age fields robustly

- Compute oldest_open_age_s using broker timestamps:
  - prefer updated_at, fallback submitted_at
- Parse defensively; never crash if timestamps missing/unparseable (set fields to None).

C2) Add warning threshold env var

- Add env var: AI_TRADING_PENDING_ORDERS_WARN_AFTER_SEC (default 120)
- If open orders exist but oldest_open_age_s < warn_after_sec: log at INFO (or DEBUG), not WARNING.
- WARNING only when older than threshold and policy action is being applied (or backlog “stale”).

C3) Enrich pending-order payload (required)

Update PENDING_ORDERS_DETECTED / PENDING_ORDERS_STILL_PRESENT payload to include:

- open_count (explicit definition: broker-active orders, not filled/canceled/expired)
- pending_count (explicit definition: your “not yet stable/ack” set; document which statuses count)
- counts_by_status (status → count)
- oldest_open_age_s
- affected_symbols_count
- sample_orders (bounded, N=3 or 5 max), each:
  - order_id (may be abbreviated but must be correlatable)
  - symbol
  - status

    - submitted_at and/or updated_at (optional)
    - age_s (optional)
- Sampling rule: pick **oldest orders first**.

C4) Transition-based warnings + periodic heartbeat (preferred)

Implement stateful backlog emission:

- Maintain:
    - self._pending_backlog_active: bool
    - self._pending_backlog_last_warn_ts: float | None
- On transition False -> True:
    - Emit PENDING_ORDERS_BACKLOG_STARTED (or reuse PENDING_ORDERS_STILL_PRESENT with transition="started")
    - Severity:
        - WARNING only if oldest_open_age_s >= warn_after_sec
        - else INFO
- While backlog persists:
    - Emit WARNING at most once per AI_TRADING_PENDING_ORDERS_WARN_EVERY_SEC (default 300)
    - Otherwise remain silent (or optional INFO heartbeat with cooldown)
- On transition True -> False:
    - Emit PENDING_ORDERS_CLEARED with summary payload.

Add env var:

- AI_TRADING_PENDING_ORDERS_WARN_EVERY_SEC (default 300)

Hard requirement: do not spam per-cycle.

D) Dedupe cycle intents upstream (reduce cycle_duplicate_intent)

Goal: Prevent duplicates from ever reaching execution.

In ai_trading/core/bot_engine.py (or wherever the execution candidate list is built):

- Dedupe intents by (symbol, side) before calling execution engine.
- Keep “best” candidate (deterministic tiebreaker):
  - primary: highest score (or strongest signal metric)
  - secondary: highest notional (or stable metric)
  - tertiary: stable ordering key (symbol, side, candidate_id)
- Emit one bounded summary log per cycle:
  - CYCLE_INTENTS_DEDUPED payload: kept_count, dropped_count, dropped_examples (cap small N)

Execution engine keeps existing cycle_duplicate_intent guard as last line of defense.

E) Improve pacing-cap semantics (maintenance should not consume “new order” budget)

Goal: Pacing cap measures new order submissions, not cancel/replace maintenance.

In ai_trading/execution/live_trading.py:

- Split counters:
  - new_orders_submitted_this_cycle
  - maintenance_actions_this_cycle (cancel/replace)
- Ensure pending-new policy maintenance actions do not increment EXECUTION_MAX_NEW_ORDERS_PER_CYCLE counter.
- Update ORDER_PACING_CAP_HIT / ORDER_SUBMIT_SKIPPED_DETAIL payload so it clearly indicates:
  - cap_type=new_orders vs cap_type=maintenance (if maintenance gets separate cap)
  - include limit, used, and headroom if available

Tests must confirm the split.

NEW: F) Remove redundant env knobs in code (canonical keys only; NO SHIMS; fail-fast on deprecated keys)

Principle: pick one canonical env var per concept. Code reads only canonical key. If deprecated key is present, **fail fast** with actionable rename instructions (no silent aliases).

F1) Remove MAX_POSITION_SIZE / AI_TRADING_MAX_POSITION_SIZE split for real

Observed:

- Config/runtime indicates AI_TRADING_MAX_POSITION_SIZE is deprecated, but ai_trading/position_sizing.py still reads it.

PR change:

- Update ai_trading/position_sizing.py to read **only** MAX_POSITION_SIZE.
- If AI_TRADING_MAX_POSITION_SIZE is set:
  - do **not** silently accept it
  - raise a startup error or emit a single high-visibility WARNING and exit (depending on your existing “strict readiness”/startup validation behavior), telling operator to rename to MAX_POSITION_SIZE.

Update tests accordingly.

F2) Deduplicate other redundant knobs (repo scan + normalize)

Perform a repo-wide scan for env pairs where both old/new keys are being read (examples already found):

- MAX_DRAWDOWN_THRESHOLD vs AI_TRADING_MAX_DRAWDOWN_THRESHOLD
- AI_TRADING_ALLOW_SHORT vs TRADING__ALLOW_SHORTS
- EXECUTION_ALLOW_FALLBACK_WITHOUT_NBBO vs AI_TRADING_EXEC_ALLOW_FALLBACK_WITHOUT_NBBO

- SENTIMENT_API_KEY vs NEWS_API_KEY (your .env currently sets both to the same value; code has an alias relationship)
- Any other deprecated_env={...} entries in ai_trading/config/runtime.py where leaf modules still read the deprecated key directly

PR requirement:

- Choose canonical key for each pair (prefer what ai_trading/config/runtime.py declares canonical).
- Update leaf modules to read only canonical key.
- If deprecated keys are present: fail-fast with actionable message (NO SHIMS).
- Add a short "Env Canonicalization Map" doc/comment listing canonical key → deprecated keys and the error message behavior.

NEW: F3) Canonicalize Alpaca URL environment variables in code (trading vs data)

Right now the repo references Alpaca endpoints across multiple keys, and ops ends up setting 2–3 keys to the same value to stay safe.

Goal: Make trading base URL and data base URL explicit, reduce redundancy, and stop reading legacy keys.

Canonical keys

- Canonical trading/orders/account host: ALPACA_TRADING_BASE_URL
- Canonical data host: ALPACA_DATA_BASE_URL

Deprecated keys to eliminate from *code reads*

- ALPACA_API_URL (currently widely used)
- ALPACA_BASE_URL (legacy)

PR behavior:

- All trading client construction must use ALPACA_TRADING_BASE_URL.

- All data client construction must use ALPACA_DATA_BASE_URL.
- If code currently expects ALPACA_API_URL / ALPACA_BASE_URL, migrate it.
- If deprecated keys are set at runtime:
  - do **not** silently accept
  - fail fast with a clear message:
    - "Set ALPACA_TRADING_BASE_URL for trading endpoints; remove ALPACA_API_URL/ALPACA_BASE_URL"
- Update validation and diagnostics modules (ai_trading/validation/validate_env.py, ai_trading/diagnostics/env_diag.py, config management helpers) to reflect the new canonical rules.

Important: keep paper/live correctness checks:

- paper must be https://paper-api.alpaca.markets
- live must be https://api.alpaca.markets (or your configured live host)

(Enforce based on EXECUTION_MODE or equivalent existing switch.)

NEW: F4) Fix the operator "print config" / introspection gap

Observed on droplet:

- python -m ai_trading --print-config fails (arg not recognized). Operators need a stable way to see resolved env/config, especially after canonicalization.

PR requirement:

- Add a supported CLI option (pick one):
  - python -m ai_trading --print-config **OR**
  - python -m ai_trading --diagnostics env / --dump-config
- It must:
  - Print resolved configuration **without secrets**
  - Show canonical key names (not deprecated ones)

    - If deprecated keys are present, print the fail-fast message (and exit non-zero)

Add tests for this option if you have CLI parsing tests; otherwise add a small new test module that runs the argument parser.

NEW: G) Fix ALERT_MODEL_LIVENESS so it only fires when meaningful (and becomes diagnosable)

Treat as a real bug.

G1) Make liveness evaluation phase-aware

In ai_trading/main.py / ai_trading/monitoring/model_liveness.py implement:

Liveness should be evaluated only when ALL are true:

1. Market is open (already passed in; must be respected).
2. Bot is in a phase where signals are expected:
    a. post-bootstrap
    b. post-reconcile / execution gate open
    c. warmup complete
    d. not in "market closed skip cycle"
3. Signals are expected given enabled modules:
    a. If RL disabled (USE_RL_AGENT=0), don't enforce RL heartbeat.
    b. If ML disabled/model not loaded, don't enforce ML heartbeat.
    c. If execution is blocked for legitimate reasons, don't alert liveness.

Implementation approach:

- In main.py, derive a boolean like signals_expected_now and pass into liveness evaluation (or compute inside liveness using shared state).
- In model_liveness.py, if signals_expected_now is false: return no breaches.

G2) Ensure signals are actually “noted” correctly

Likely root causes:

- note_ml_signal / note_rl_signals_emitted not called on the “real” signal emission path.
- Liveness expects a signal too frequently even when strategy legitimately produces none.

PR requirement:

- Identify where ML and RL signals are considered “emitted” in the actual flow.
- Decide and document semantics:
    - If “signal liveness” means “model produced a decision record (even no-trade)”, then note on decision production.
    - If it means “we produced at least one actionable candidate”, note only then.
- Enforce thresholds using:
    - AI_TRADING_ML_SIGNAL_MAX_AGE_SECONDS
    - AI_TRADING_RL_SIGNAL_MAX_AGE_SECONDS

G3) Alert behavior: rate-limited + correct severity + structured payload

Acceptance:

- Rate-limit via AI_TRADING_MODEL_LIVENESS_ALERT_COOLDOWN_SECONDS
- Log level:
    - WARNING for breach
    - ERROR only if automated action taken (kill switch / canary rollback)

Every alert must include structured payload:

- last_ml_signal_ts, last_rl_signal_ts (or None)
- ml_age_s, rl_age_s
- ml_max_age_s, rl_max_age_s
- market_open
- signals_expected_now

- phase / execution gate state / warmup state (whatever exists)

G4) Add liveness snapshot to logs + health endpoint

get_model_liveness_snapshot() exists — wire it into:

- periodic health tick logs (cooldown; not per cycle)
- health endpoint payload (or diagnostics endpoint)

Snapshot must not leak secrets.

Acceptance Criteria

Pending-orders / execution:

1. Pending-new timeout policy runs once per engine cycle (idempotent).
2. Symbol block decay does not release while open orders exist; emits PENDING_SYMBOL_BLOCK_DECAY_DEFERRED.
3. Pending backlog logs are severity-gated by age threshold:
    a. WARNING only when oldest_open_age_s >= AI_TRADING_PENDING_ORDERS_WARN_AFTER_SEC
    b. and warnings are transition/periodic, not per-cycle.
4. PENDING_ORDERS_* logs include:
    a. open_count, pending_count, counts_by_status, oldest_open_age_s, bounded sample_orders.
5. Upstream dedupe reduces cycle_duplicate_intent materially.
6. Pacing semantics split works:
    a. maintenance actions do not consume new-order cap.

Env canonicalization:

7. ai_trading/position_sizing.py reads only MAX_POSITION_SIZE.
8. Deprecated env keys are not silently accepted:
    a. if set, bot fails fast with clear rename instructions (NO SHIMS).
9. Similar dedupe performed for other redundant env pairs discovered.

10. Alpaca URLs:

- trading uses ALPACA_TRADING_BASE_URL
- data uses ALPACA_DATA_BASE_URL
- deprecated ALPACA_API_URL / ALPACA_BASE_URL are not read (fail-fast if present).

Operator introspection:

11. There is a supported CLI option to print resolved config (sanitized), including canonical env key names, and it exits non-zero if deprecated keys are set.

Model liveness:

12. ALERT_MODEL_LIVENESS no longer spams:

- only evaluated when signals expected and market open
- rate-limited
- WARNING unless action taken
- structured payload included

13. Health/logging includes a liveness snapshot so ops can debug quickly.

All tests pass.

Change Details (Files / Touchpoints)

- ai_trading/execution/live_trading.py
    - idempotency guard for _apply_pending_new_timeout_policy
    - split new-order pacing vs maintenance counters
    - improve ORDER_PACING_CAP_HIT / ORDER_SUBMIT_SKIPPED_DETAIL payload
- ai_trading/core/bot_engine.py
    - pending-order severity gating by age
    - telemetry enrichment (counts/age/samples)
    - state transition + heartbeat emission
    - block decay deferral
    - upstream intent dedupe
- ai_trading/monitoring/model_liveness.py

    - phase-aware / signals-expected gating
    - cooldown correctness
    - structured payload + improved snapshot
- ai_trading/main.py
    - derive signals_expected_now and apply gating
    - correct severity semantics (WARN vs ERROR on action)
    - periodic snapshot emission
- ai_trading/position_sizing.py
    - canonical MAX_POSITION_SIZE only; fail-fast if deprecated present
- Alpaca env canonicalization touchpoints (likely):
    - ai_trading/utils/env.py
    - ai_trading/broker/alpaca_credentials.py
    - ai_trading/diagnostics/env_diag.py
    - ai_trading/validation/validate_env.py
    - any module that constructs Alpaca clients or uses base URLs
- CLI / diagnostics:
    - ai_trading/__main__.py (argparse)
    - or a dedicated diagnostics module

Tests (update/add)

Update or add tests to cover:

Pending orders:

- idempotency guard
- decay deferral
- telemetry payload fields present
- transition/heartbeat gating

Pacing cap:

- maintenance does not consume new-order cap

Env canonicalization:

- MAX_POSITION_SIZE works

- deprecated key present ⇒ fails fast with actionable message
- Alpaca deprecated URL key present ⇒ fails fast

Model liveness:

- does not evaluate/alert when market closed or phase not expecting signals
- alerts when market open + signals expected + last signal older than threshold
- includes payload fields
- respects cooldown
- WARNING vs ERROR semantics

CLI config dump:

- --print-config (or chosen flag) works, is sanitized, and honors fail-fast behavior

Constraints & Standards

- **NO SHIMS** anywhere (no compatibility wrappers, no silent alias fallbacks).
- Fail-safe behavior (prefer blocking trading over unsafe repeated submissions).
- Logs structured and bounded; avoid per-cycle noise.
- Deterministic dedupe behavior.
- Use broker timestamps defensively; never crash on missing data.
- Preserve .env → .env.runtime workflow.
- Do not delete .jsonl runtime logs (operator explicitly wants them retained).

Validation Steps

```
cd /home/aiuser/ai-trading-bot
source venv/bin/activate
```

```
pytest -q tests/bot_engine/test_order_pending_severity.py
pytest -q tests/bot_engine/test_pending_orders_cleanup.py
pytest -q tests/execution/test_execution_runtime_controls.py
pytest -q tests/execution/test_order_pacing_cap.py

# new tests you add (names TBD)
pytest -q tests/monitoring/test_model_liveness.py
pytest -q tests/config/test_env_canonicalization.py
pytest -q tests/cli/test_print_config.py

pytest -q
```

Operational check (droplet):

```
journalctl -u ai-trading.service --since "60 min ago" -o short-iso -l
| \
  grep -E
'PENDING_ORDERS_|PENDING_SYMBOL_BLOCK_DECAY_|CYCLE_INTENTS_DEDUPED|cyc
le_duplicate_intent|order_pacing_cap|ORDER_SUBMIT_SKIPPED_DETAIL|ALERT
_MODEL_LIVENESS|MODEL_LIVENESS' | \
  tail -n 400
```

Confirm:

- Pending backlog warnings are not spammy and include enriched payload.
- Liveness alerts are not constant; when they fire they’re meaningful and include snapshot/payload.
- Deprecated env keys cause a clear, actionable startup failure (no silent fallback).
- Alpaca URL keys are canonicalized (trading vs data), and old keys are rejected.

Risk & Rollback

Risk: pending-order semantics changes can alter how aggressively trading is blocked during open-limit workflows; liveness gating could suppress alerts if implemented incorrectly; env canonicalization could fail startup if operator forgets to rename a key.

Rollback:

- revert PR
- sudo systemctl restart ai-trading.service
- restore previous .env if needed (PR must make rename requirements obvious and explicit)

Non-Goals

- No strategy logic changes (signals/scoring/model weights)
- No broker/provider rewrites
- No new compatibility shims or facades
- No deleting .jsonl runtime logs

Appendices

Appendix A: Target build/runtime combo (verbatim)

Ubuntu 24.04 (Noble) x86_64, glibc 2.39, CPython 3.12.3 in venv at /home/aiuser/ai-trading-bot/venv. Pip compatible top tag: cp312-cp312-manylinux_2_39_x86_64 (and accepts older manylinux baselines). Dry-run shows runtime/dev reqs satisfied; notable heavy deps present: torch 2.3.1, stable-baselines3 2.3.2, gymnasium 0.29.1, scikit-learn 1.5.2, scipy 1.13.1, matplotlib 3.9.2, hypothesis 6.138.2. Alpaca SDK: alpaca-trade-api 3.2.0 (runtime) and alpaca-py 0.42.0 (dev-only).

Appendix B: Current “known good” runtime knobs you’re using

- AI_TRADING_EXECUTION_PHASE_GATE_ENABLED=1
- AI_TRADING_BLOCK_ENTRIES_ON_FALLBACK_MINUTE_DATA=1
- AI_TRADING_EVENT_DRIVEN_NEW_BAR_ONLY=1
- EXECUTION_MAX_NEW_ORDERS_PER_CYCLE=10
- Debug noise removed (AI_TRADING_EXEC_DEBUG, AI_TRADING_EXEC_LOG_REQUESTS, etc.)

Appendix C: Why this PR is worth it

- Less churn from block decay oscillation
- Cleaner logs (warnings mean “abnormal”, not “open orders exist”)
- Higher throughput under normal limit-order behavior
- Same safety posture, fewer self-inflicted throttles
- Better ops visibility: pending backlog payload includes counts/ages/samples
- Liveness monitoring becomes trustworthy: fewer false alarms, more diagnosable alerts, less masking of real issues
- Env becomes maintainable: canonical keys only, and the bot tells you exactly what to fix if a deprecated key appears