Axiom Quant Trading System
Build and Strategy Manual (Book 2)

Version: 1.0
Repository: `/Users/madhuram/trading_bot`
Date: 2026-02-19

---

1) Purpose of This Book

This document explains:

1. How to build and operate the system step-by-step.
2. Why each logic layer exists.
3. What safety and reliability constraints are mandatory.
4. How to debug “no trade”, stale feed, and gate-block conditions.

This is implementation-oriented and intended for engineers and technical operato
rs.

---

2) System Design Principles

1. Fail closed over fail open.
2. Mode-aware behavior:
   - `LIVE` must remain conservative.
   - `PAPER/SIM` may run controlled experimentation.
3. Every decision path must be observable.
4. Every fallback must be explicit, logged, and reversible.
5. No silent error suppression in critical data paths.

---

3) Build Setup (Environment and Baseline)

3.1 Prerequisites

1. Python 3.12 (as used by project runtime).
2. Repo cloned at `/Users/madhuram/trading_bot`.
3. Dependencies installed from:
   - `/Users/madhuram/trading_bot/requirements.txt`

3.2 Baseline Setup

1. `cd /Users/madhuram/trading_bot`
2. `pip install -r requirements.txt`
3. `bash scripts/ci_sanity.sh`

Expected:
1. Compile checks pass.
2. Unit tests execute.

---

4) Step-by-Step Build of Core Runtime

Step 1: Configuration Layer

File:
1. `/Users/madhuram/trading_bot/config/config.py`

Build tasks:
1. Define all runtime controls as flags.
2. Centralize mode (`EXECUTION_MODE`), risk limits, feature flags, and path sett
ings.

Why:
1. Prevent hardcoded behavior drift.
2. Enable safe rollout and rollback via config changes.

Step 2: Session and Expiry Logic

Files:
1. `/Users/madhuram/trading_bot/core/market_calendar.py`
2. `/Users/madhuram/trading_bot/core/option_chain.py`
3. `/Users/madhuram/trading_bot/core/kite_client.py`

Build tasks:
1. Compute market session state and holiday-safe checks.
2. Select expiry from actual available exchange expiries first.
3. Use weekday fallback only when exchange list unavailable.

Why:
1. Wrong expiry selection invalidates option resolution and trading intent.
2. Exchange calendars/expiry rules change and must be source-driven.

Step 3: Auth and Broker Session

Files:
1. `/Users/madhuram/trading_bot/core/kite_client.py`
2. `/Users/madhuram/trading_bot/scripts/generate_kite_access_token.py`
3. `/Users/madhuram/trading_bot/scripts/validate_kite_session.py`

Build tasks:
1. Validate token against profile endpoint before reporting success.
2. Refresh access token on client ensure, even with cached client instance.
3. Support local token store as single source fallback.

Why:
1. Token/profile drift creates false-ready states and runtime confusion.

Step 4: Feed Ingestion and Freshness

Files:
1. `/Users/madhuram/trading_bot/core/kite_depth_ws.py`
2. `/Users/madhuram/trading_bot/core/feed_health.py`
3. `/Users/madhuram/trading_bot/core/sla_check.py`

Build tasks:
1. Receive WS ticks/depth and store freshness epochs.
2. Distinguish true stale from off-hours and minor skew.
3. Log payload shape and tick ingest errors to dedicated files.

Why:
1. Feed status is a first-order risk input.
2. False stale/future alerts produce unsafe operator actions.

Step 5: Persistence and State Stores

Files:
1. `/Users/madhuram/trading_bot/core/trade_store.py`
2. `/Users/madhuram/trading_bot/core/tick_store.py`
3. `/Users/madhuram/trading_bot/core/depth_store.py`
4. `/Users/madhuram/trading_bot/core/run_lock.py`

Build tasks:
1. Use canonical DB path, no ambiguous defaults.
2. Ensure parent directories and write permissions.
3. Keep lock behavior deterministic with stale detection.

Why:
1. DB/open failures and stale locks are common production blockers.

Step 6: Market Snapshot and Indicator Pipeline

Files:
1. `/Users/madhuram/trading_bot/core/market_data.py`
2. `/Users/madhuram/trading_bot/core/indicators_live.py`
3. `/Users/madhuram/trading_bot/core/regime.py`

Build tasks:
1. Build one snapshot per symbol per cycle.
2. Compute indicators once and maintain freshness age.
3. Warm-seed OHLC when possible.
4. Emit explicit missing input reasons.

Why:
1. Most no-trade scenarios are indicator readiness issues.
2. Duplicate or mutable snapshots create contradictory gate outcomes.

Step 7: Strategy Gating and Candidate Construction

Files:
1. `/Users/madhuram/trading_bot/core/strategy_gatekeeper.py`
2. `/Users/madhuram/trading_bot/strategies/trade_builder.py`
3. `/Users/madhuram/trading_bot/strategies/ensemble.py`

Build tasks:
1. Route strategy families by regime/day-type.
2. Apply liquidity, premium, and confidence controls.
3. Enforce lifecycle and cooldown gates.
4. Log all candidate rejections with structured reason codes.

Why:
1. Trade quality depends on strict pre-execution filtering.
2. Without reason logs, tuning becomes guesswork.

Step 8: Controlled Fallbacks (Without Degrading Safety)

Files:

1. `/Users/madhuram/trading_bot/strategies/trade_builder.py`
2. `/Users/madhuram/trading_bot/config/config.py`

Build tasks:
1. Add controlled fallback `trend_vwap_fallback` with fixed score and strict gua
rds.
2. Disable in `LIVE` by default unless explicitly enabled.
3. Keep lifecycle and premium checks active.
4. Log fallback usage to signal and blocked-candidate streams.

Why:
1. Need sensitivity in paper testing without opening unsafe live behavior.

Step 9: Readiness and Governance Choke Points

Files:
1. `/Users/madhuram/trading_bot/core/readiness_gate.py`
2. `/Users/madhuram/trading_bot/core/governance_gate.py`
3. `/Users/madhuram/trading_bot/core/risk_halt.py`

Build tasks:
1. Enforce hard blockers for auth/feed/risk-halt/manual-approval.
2. Return explicit reasons and states.

Why:
1. Single choke point prevents accidental order path bypass.

Step 10: Dashboard and Operator Visibility

Files:
1. `/Users/madhuram/trading_bot/dashboard/streamlit_app.py`
2. `/Users/madhuram/trading_bot/dashboard/ui/components.py`

Build tasks:
1. Show gate, SLA, and feed states.
2. Show blocked candidates from desk-scoped logs.
3. Preserve backward compatibility with legacy logs.

Why:
1. Operators need exact causes, not generic status banners.

Step 11: Daily Ops and Reporting

Files:
1. `/Users/madhuram/trading_bot/scripts/daily_ops.py`
2. `/Users/madhuram/trading_bot/scripts/run_daily_audit.py`
3. `/Users/madhuram/trading_bot/ml/truth_dataset.py`
4. `/Users/madhuram/trading_bot/core/reports/rl_shadow_report.py`

Build tasks:
1. Run data quality, repair, backfill, and audit stages.
2. Skip gracefully when decision/truth data is legitimately absent.
3. Keep non-empty-data errors fatal.

Why:
1. Daily workflow must not crash on no-trade days.

2. Real schema/data bugs must still fail fast.

Step 12: Test Strategy and CI

Files:
1. `/Users/madhuram/trading_bot/tests/*`
2. `/Users/madhuram/trading_bot/scripts/ci_sanity.sh`

Build tasks:
1. Add unit tests for each high-risk branch.
2. Keep integration concerns explicit.
3. Enforce compile + pytest baseline before release.

Why:
1. Regression prevention is mandatory in mission-critical systems.

---

5) Strategy and Logic Rationale

5.1 Why Regime-Based Routing

1. Regime mismatch causes false positives in strategy triggers.
2. Routing aligns strategy family with market structure.

5.2 Why ORB and Day-Type Gates

1. Early session noise causes unstable directional signals.
2. ORB/day-type gates delay aggressive calls until structure stabilizes.

5.3 Why Quote and Liquidity Filters

1. Options with poor spread/depth are untradeable in practice.
2. Filters reduce slippage-driven PnL distortion.

5.4 Why Candidate Rejection Logging

1. “No trade” is valid, but must be explainable.
2. Logs allow objective tuning rather than intuition.

5.5 Why Strict `LIVE` / Flexible `PAPER`

1. Live risk must remain bounded by conservative controls.
2. Paper mode is used for controlled sensitivity experiments.

---

6) Bug-to-Fix Mapping (Engineering Lessons)

1. Duplicate process ownership -> unified owner and lock discipline.
2. Token source inconsistency -> deterministic token refresh and validation.
3. DB path drift -> canonical resolver and split-brain guard.
4. Audit crashes on empty data -> explicit skip artifacts.
5. Missing index depth -> synthetic quote only in non-live modes.
6. Dashboard blind spots -> desk-scoped blocked candidate logs.
7. Expiry mismatch -> exchange-list-first expiry selection.

---

7) Operational Debug Playbook

7.1 If No Trades Are Suggested

Check:
1. `logs/desks/<DESK_ID>/gate_status.jsonl`
2. `logs/desks/<DESK_ID>/blocked_candidates.jsonl`
3. `logs/sla_check.json`

Interpret:
1. `indicators_missing_or_stale` -> check OHLC seed and indicator freshness.
2. `missing_live_bidask` in `LIVE` -> quote source issue, do not bypass.
3. `no_signal` with healthy indicators -> strategy thresholds, regime alignment.

7.2 If Feed Seems Healthy but UI Contradicts

Check:
1. Ensure dashboard reads latest desk log files.
2. Verify stale file mtime and correct source path.

7.3 If Daily Ops Fails

Check:
1. import path/cwd context for scripts.
2. decision/truth artifacts presence.
3. DB path consistency and permissions.

---

8) Configuration Controls Added for Sensitivity

From `/Users/madhuram/trading_bot/config/config.py`:

1. `TREND_VWAP_FALLBACK_ENABLE`
2. `TREND_VWAP_FALLBACK_LIVE_ENABLE`
3. `TREND_VWAP_FALLBACK_SCORE`
4. `TREND_VWAP_FALLBACK_SLOPE_ABS_MIN`

Default policy:
1. Enabled for paper/sim experimentation.
2. Disabled in live unless explicitly opted in.

---

9) Release and Rollout Guidance

Stage 1: Paper Soak

1. Run with full logs enabled.
2. Verify blocked reasons distribution.
3. Validate fallback usage frequency and quality.

Stage 2: Controlled Live Dry-Run

1. Keep fallback live-disabled.
2. Verify no unexpected blocker storms.
3. Confirm readiness and governance pass only on valid conditions.

Stage 3: Production Monitoring

1. Track gate and blocked-candidate trend metrics.
2. Trigger incident workflow on repeated critical blockers.

---

10) Complete File Coverage Reference

Complete file inventory for this codebase is provided in:

`/Users/madhuram/trading_bot/docs/APPENDIX_FILE_INDEX.md`

Use this appendix with Book 1 and Book 2 for full client and engineering handover.