==
==========
SECTION 1: BOOK_1_CLIENT_HANDOVER_REPORT.md
==
==========

# Axiom Quant Trading System
## Client Handover Report (Book 1)

Version: 1.0
Repository: `/Users/madhuram/trading_bot`
Prepared for: Client Handover
Prepared by: Engineering Team
Date: 2026-02-19

## 1) Executive Summary

This project delivers a production-oriented algorithmic trading platform focused on:

1. Deterministic and auditable decision-making.
2. Strict fail-closed runtime controls.
3. Operational observability for every critical gate.
4. Controlled separation between `LIVE` and `PAPER/SIM`.

The system now runs with:

1. Structured feed and gate telemetry.
2. Explicit blocker diagnostics in dashboard and logs.
3. Safer expiry selection and token/session handling.
4. Reduced ambiguity around “why no trade” outcomes.

## 2) Project Scope and Deliverables

### Delivered Scope

1. Live market data ingestion and freshness tracking.
2. Indicator + regime + strategy-gate pipeline.
3. Trade candidate generation with strict filter chain.
4. Readiness and governance fail-closed choke points.
5. Trade and audit persistence in SQLite + JSONL logs.
6. Streamlit operational dashboard.
7. Daily audit/reporting and diagnostics scripts.
8. Extensive automated tests for critical paths.

### Out of Scope

1. Exchange-side guaranteed order execution quality.
2. Market alpha guarantees.
3. Broker availability guarantees.

## 3) Current Architecture Summary

### Runtime Flow

1. `main.py` starts orchestrator and safety controls.
2. `core/orchestrator.py` runs symbol cycles and stage gates.
3. `core/market_data.py` builds immutable market snapshots.
4. `core/strategy_gatekeeper.py` evaluates strategy family eligibility.
5. `strategies/trade_builder.py` attempts candidate construction.
6. `core/readiness_gate.py` and `core/governance_gate.py` enforce runtime permissions.
7. `core/trade_store.py`, `core/tick_store.py`, `core/depth_store.py` persist state.
8. `dashboard/streamlit_app.py` provides operations and debugging views.

### Key Safety Invariants

1. `LIVE` mode remains fail-closed by default.
2. Missing critical dependencies block trades, not just warnings.
3. Every rejection path should emit an observable reason.

## 4) Major Features Delivered

### 4.1 Data and Feed Reliability

1. WebSocket feed lifecycle controls and stale detection.
2. Depth and tick freshness telemetry.
3. Future-skew tolerance for timestamp classification.
4. Feed diagnostics logs for payload schema and ingest errors.

### 4.2 Trading Decision and Risk Pipeline

1. Regime-aware strategy family routing.
2. Indicator freshness and warm-up gating.
3. Candidate filters: quote quality, spread, premium, OI/IV, score, lifecycle.
4. Controlled fallback for `PAPER/SIM` without weakening `LIVE`.

### 4.3 Governance and Auditability

1. Readiness gate + governance gate as explicit execution choke points.
2. Structured gate logs (`gate_status.jsonl`) per cycle/stage.
3. Blocked candidate logs (`blocked_candidates.jsonl`) with reason codes.
4. Decision and audit event logging pathways.

### 4.4 Operations and Dashboard

1. Unified operational dashboard sections:
   - Home readiness
   - Data & SLA
   - Execution
   - Reconciliation
   - ML/RL
2. “What Blocked Trades Today” now reads active reject logs, not only legacy files.
3. Day-type history reader compatibility improvements.

## 5) Problems Encountered and Resolution Register

This section records major production-impact defects addressed during delivery.

### 5.1 Lock and Process Ownership Issues

Problem:
1. Repeated `RUN_LOCK_ACTIVE` due to duplicate process ownership and respawn scr
ipts.

Resolution:
1. Consolidated single-owner feed behavior.
2. Strengthened run-lock handling and diagnostics.
3. Added tests around stale/dead lock scenarios.

### 5.2 Feed Restart and Token Mismatch

Problem:
1. Auth validation passed while WS used stale/mismatched token source.
2. Restart storms on close/error created unstable loops.

Resolution:
1. Unified resolved token usage for WS startup.
2. Improved close/restart guard paths and policy.
3. Added tests for token path and restart behavior.

### 5.3 DB Path Drift and SQLite Reliability

Problem:
1. Inconsistent DB path defaults (`data/trades.db` vs desk DB) created split-bra
in behavior.
2. Relative path and CWD drift caused “unable to open database file”.
3. Hot-path DB usage risked file descriptor exhaustion.

Resolution:
1. Canonicalized DB path resolution behavior.
2. Added fail-fast split-brain checks.
3. Hardened connect/open diagnostics and write path handling.

### 5.4 Missing Decision and Audit Data Crashes

Problem:
1. Daily audit failed when decision events or truth dataset were absent.

Resolution:
1. Implemented skip contracts with explicit artifacts and reasons.
2. Preserved fatal behavior for unexpected/real errors.

### 5.5 Import and Script Execution Fragility

Problem:
1. Script execution failed from varying working directories.

Resolution:
1. Package markers and module-run conventions.
2. Script runtime path hardening.
3. Daily ops subprocess invocation stabilization.

### 5.6 Trade Starvation from Missing Index Bid/Ask

Problem:
1. Indices often have missing depth bid/ask; strict quote checks rejected all ca
ndidates.

Resolution:
1. In `PAPER/SIM`, synthetic index bid/ask from LTP with explicit markers.
2. In `LIVE`, still fail-closed on missing true quote fields.
3. Added unit tests for both branches.

### 5.7 Observability Gap: Blocked Candidates Not Visible

Problem:
1. Dashboard showed “No blocked candidates” despite trade-builder rejections.

Resolution:
1. Added lightweight structured logger in trade builder for rejection events.
2. Dashboard now reads desk-scoped blocked candidates log and supports legacy fallback.
3. Added tests for `missing_index_bid_ask` and `no_signal`.

### 5.8 Expiry Selection Mismatch

Problem:
1. Config weekday hints conflicted with actual available exchange expiries.

Resolution:
1. Selection now prefers actual available expiries.
2. Weekday mapping used only as fallback.
3. Defaults aligned to current weekly expiries (NSE Tuesday, Sensex Thursday).

## 6) Strengths and Improvements Achieved

1. Higher observability: root-cause logs at gate and candidate levels.
2. Better mode safety: strict `LIVE`, controlled `PAPER/SIM`.
3. Better data integrity: timestamp, DB path, and ingestion diagnostics.
4. Improved audit continuity and reduced false operational failures.
5. Better client/operator explainability: explicit reason codes in UI and logs.

## 7) Known Constraints and Residual Risks

1. Broker/exchange dependencies remain external runtime risk.
2. Large script surface area still requires disciplined release control.
3. Strategy performance depends on market conditions and model quality, not software correctness alone.
4. Runtime tuning requires controlled rollout with regression tests.

## 8) Validation and QA Coverage

The project includes broad test coverage in `/Users/madhuram/trading_bot/tests`, including:

1. Feed health and restart behavior.
2. Readiness and governance gates.
3. Strategy gatekeeper and trade-builder behavior.
4. Risk and exposure controls.
5. Auth and token consistency.
6. Data/audit/reporting contracts.

Recent additions include:

1. `tests/test_trade_builder_blocked_candidates.py`
2. `tests/test_trade_builder_trend_vwap_fallback.py`
3. `tests/test_expiry_selection.py`

## 9) Deployment and Operations Handover

### Recommended Operational Commands

1. Validate environment:
   - `bash /Users/madhuram/trading_bot/scripts/ci_sanity.sh`
2. Validate auth/session:
   - `PYTHONPATH=. python /Users/madhuram/trading_bot/scripts/validate_kite_session.py`
3. Start stack:
   - `bash /Users/madhuram/trading_bot/scripts/start_live_stack.sh`
4. Start dashboard:
   - `PYTHONPATH=. streamlit run /Users/madhuram/trading_bot/dashboard/streamlit_app.py`
5. Run tests:
   - `PYTHONPATH=. pytest -q`

### Runtime Artifacts to Monitor

1. `logs/desks/<DESK_ID>/gate_status.jsonl`
2. `logs/desks/<DESK_ID>/blocked_candidates.jsonl`
3. `logs/depth_ws_watchdog.log`
4. `logs/sla_check.json`
5. `logs/incidents.jsonl`

## 10) Handover Acceptance Checklist

1. Auth validation succeeds.
2. Feed and SLA fresh during market-open windows.
3. Dashboard reflects gate reasons and blocked candidates.
4. Daily audit runs without crash.
5. Regression tests pass on release candidate.

## 11) Client Recommendations for Next Phase

1. Keep `LIVE` strictness unchanged; tune only in `PAPER/SIM` first.
2. Add weekly blocker reason analytics review.
3. Maintain release checklists with mandatory regression runs.
4. Version formal data contracts for UI and reporting artifacts.

## 12) Appendix Reference

Full repository file inventory is provided in:

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

==

SECTION 2: BOOK_2_BUILD_AND_STRATEGY_MANUAL.md

==

# 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 operators.

## 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 settings.

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 guards.
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.

==

SECTION 3: APPENDIX_FILE_INDEX.md

==

# Appendix: Repository File Index

Generated from: `/Users/madhuram/trading_bot`
Total indexed files: **424**

Scope: source, scripts, dashboard, ML/model code, and tests. Build artifacts (`__pycache__`, `*.pyc`) are excluded.

## config (1 files)

- `config/config.py`

## core (145 files)

- `core/ablation.py`
- `core/adaptive_risk.py`
- `core/alpha_ensemble.py`
- `core/approval_store.py`
- `core/audit_log.py`
- `core/auth_health.py`
- `core/auto_retrain.py`
- `core/auto_tune.py`
- `core/backtest_engine.py`
- `core/backtest_report.py`
- `core/blocked_tracker.py`
- `core/capital_allocator.py`
- `core/circuit_breaker.py`
- `core/cross_asset.py`
- `core/cross_asset_features.py`
- `core/day_type_history.py`
- `core/db_guard.py`
- `core/decision.py`
- `core/decision_builder.py`
- `core/decision_logger.py`
- `core/decision_store.py`
- `core/decision_trace.py`
- `core/depth_store.py`
- `core/desk_config.py`
- `core/execution/__init__.py`
- `core/execution/chokepoint.py`
- `core/execution_analytics.py`
- `core/execution_engine.py`
- `core/execution_guard.py`

- `core/execution_quality.py`
- `core/execution_router.py`
- `core/exposure_ledger.py`
- `core/feature_builder.py`
- `core/feature_contract.py`
- `core/feature_flags.py`
- `core/feed_circuit_breaker.py`
- `core/feed_freshness.py`
- `core/feed_health.py`
- `core/feed_restart_guard.py`
- `core/fill_model.py`
- `core/fill_quality.py`
- `core/filters.py`
- `core/freshness_sla.py`
- `core/gate_status_log.py`
- `core/governance.py`
- `core/governance_gate.py`
- `core/gpt_advisor.py`
- `core/greeks.py`
- `core/hedge_exposure.py`
- `core/hypothesis_engine.py`
- `core/incidents.py`
- `core/indicators_live.py`
- `core/invalidation.py`
- `core/kite_client.py`
- `core/kite_depth_ws.py`
- `core/log_lock.py`
- `core/log_writer.py`
- `core/market_calendar.py`
- `core/market_data.py`
- `core/market_regime.py`
- `core/meta_model.py`
- `core/ml_dataset.py`
- `core/ml_governance.py`
- `core/model_health.py`
- `core/model_registry.py`
- `core/news_calendar.py`
- `core/news_encoder.py`
- `core/news_ingestor.py`
- `core/news_shock_encoder.py`
- `core/no_trade_engine.py`
- `core/ohlc_buffer.py`
- `core/oi_confirmation.py`
- `core/option_chain.py`
- `core/orchestrator.py`
- `core/orchestrator_parts/__init__.py`
- `core/orchestrator_parts/cycle.py`
- `core/orchestrator_parts/data.py`
- `core/orchestrator_parts/decisions.py`
- `core/orchestrator_parts/finalize.py`
- `core/orders/__init__.py`
- `core/orders/order_intent.py`
- `core/paper_fill_simulator.py`
- `core/paths.py`
- `core/portfolio_allocator.py`
- `core/portfolio_manager.py`
- `core/portfolio_risk_allocator.py`
- `core/position_sizer.py`
- `core/post_trade_labeler.py`
- `core/premarket.py`
- `core/probability_engine.py`
- `core/pullback.py`

- `core/readiness_gate.py`
- `core/readiness_state.py`
- `core/reason_codes.py`
- `core/regime.py`
- `core/regime_detection.py`
- `core/regime_detector.py`
- `core/regime_prob_model.py`
- `core/replay_engine.py`
- `core/reports/__init__.py`
- `core/reports/daily_audit.py`
- `core/reports/decay_report.py`
- `core/reports/execution_report.py`
- `core/reports/promotion_report.py`
- `core/reports/rl_shadow_report.py`
- `core/research_pipeline.py`
- `core/retrain_manager.py`
- `core/review_packet.py`
- `core/review_queue.py`
- `core/risk.py`
- `core/risk_engine.py`
- `core/risk_halt.py`
- `core/risk_manager.py`
- `core/risk_state.py`
- `core/risk_utils.py`
- `core/run_backtest.py`
- `core/run_lock.py`
- `core/runtime_paths.py`
- `core/scorecard.py`
- `core/security_guard.py`
- `core/session_calendar.py`
- `core/session_guard.py`
- `core/strategy_allocator.py`
- `core/strategy_decay.py`
- `core/strategy_gatekeeper.py`
- `core/strategy_lifecycle.py`
- `core/strategy_tracker.py`
- `core/stress_generator.py`
- `core/strike_selector.py`
- `core/synthetic_market.py`
- `core/telegram.py`
- `core/telegram_alerts.py`
- `core/tf_utils.py`
- `core/tick_store.py`
- `core/time_sanity.py`
- `core/time_utils.py`
- `core/trade_journal.py`
- `core/trade_logger.py`
- `core/trade_schema.py`
- `core/trade_scoring.py`
- `core/trade_store.py`
- `core/trade_ticket.py`
- `core/tv_queue.py`
- `core/walk_forward.py`
- `core/walk_forward_ml.py`

## dashboard (5 files)

- `dashboard/__init__.py`
- `dashboard/streamlit_app.py`
- `dashboard/ui/__init__.py`
- `dashboard/ui/components.py`
- `dashboard/ui/tokens.py`

## ml (9 files)

- `ml/alpha_factory.py`
- `ml/decay_dataset.py`
- `ml/deep_predictor.py`
- `ml/microstructure_model.py`
- `ml/microstructure_predictor.py`
- `ml/strategy_decay_predictor.py`
- `ml/trade_predictor.py`
- `ml/train_decay_model.py`
- `ml/truth_dataset.py`

## models (3 files)

- `models/__init__.py`
- `models/tick_dataset.py`
- `models/train_from_ticks.py`

## scripts (132 files)

- `scripts/activate_model.py`
- `scripts/adaptive_risk_status.py`
- `scripts/analyze_experiment.py`
- `scripts/approve_trade.py`
- `scripts/arm_trade.py`
- `scripts/audit_market_data.py`
- `scripts/auth_warmup.py`
- `scripts/backfill_epoch_columns.py`
- `scripts/backfill_trades_db.py`
- `scripts/bootstrap.py`
- `scripts/build_decision_dataset.py`
- `scripts/build_training_dataset.py`
- `scripts/build_truth_dataset.py`
- `scripts/canary_status.py`
- `scripts/capital_committee_report.py`
- `scripts/check_cross_asset.py`
- `scripts/check_kite_auth.py`
- `scripts/check_openai_key.py`
- `scripts/check_option_tokens.py`
- `scripts/ci_sanity.sh`
- `scripts/clear_feed_breaker.py`
- `scripts/cross_asset_status.py`
- `scripts/daily_ops.py`
- `scripts/daily_ops_ist.sh`
- `scripts/daily_report.py`
- `scripts/daily_rollup.py`
- `scripts/daily_scorecard.py`
- `scripts/data_manifest.py`
- `scripts/data_qc.py`
- `scripts/desk_status.py`
- `scripts/disable_boot_launch.sh`
- `scripts/download_instruments.py`
- `scripts/dr_backup.py`
- `scripts/dr_failover_drill.py`
- `scripts/dr_restore.py`
- `scripts/dr_verify.py`
- `scripts/enable_boot_launch.sh`
- `scripts/execution_diagnostics.py`
- `scripts/export_audit_bundle.py`
- `scripts/export_trade_log_csv.py`
- `scripts/export_trades_excel.py`

- `scripts/flatten_positions.py`
- `scripts/freeze_dataset.py`
- `scripts/generate_hypotheses.py`
- `scripts/generate_kite_access_token.py`
- `scripts/generate_sample_trades.py`
- `scripts/generate_synthetic_sessions.py`
- `scripts/hash_trade_log.py`
- `scripts/import_sanity.py`
- `scripts/import_signals_excel.py`
- `scripts/incident_bundle.py`
- `scripts/kill_switch.sh`
- `scripts/kite_autologin_localhost.py`
- `scripts/launchagent_status.sh`
- `scripts/live_fills_sync.py`
- `scripts/lock_trade_log.py`
- `scripts/meta_shadow_eval.py`
- `scripts/micro_accuracy.py`
- `scripts/migrate_review_queue.py`
- `scripts/migrate_strike.py`
- `scripts/migrate_timestamps.py`
- `scripts/ops_summary.py`
- `scripts/paper_tournament.py`
- `scripts/partial_profit_smoketest.py`
- `scripts/premarket_plan.py`
- `scripts/readiness_gate.py`
- `scripts/reconcile_fills.py`
- `scripts/recover_db_and_clear_halt.py`
- `scripts/refresh_option_chain.py`
- `scripts/register_experiment.py`
- `scripts/register_model.py`
- `scripts/regression_gate_12_14.py`
- `scripts/regression_gate_final.py`
- `scripts/regression_gate_manual_approval.py`
- `scripts/regression_gate_stable.py`
- `scripts/repair_tick_timestamps.py`
- `scripts/repair_ticks.py`
- `scripts/replay_day.py`
- `scripts/reset_risk_halt.py`
- `scripts/risk_monitor.py`
- `scripts/rollback_flags.py`
- `scripts/rollback_model.py`
- `scripts/run_all.sh`
- `scripts/run_alpha_factory.py`
- `scripts/run_daily_audit.py`
- `scripts/run_daily_scorecard.sh`
- `scripts/run_decay_daily.py`
- `scripts/run_execution_analytics.py`
- `scripts/run_model_promotion.py`
- `scripts/run_pilot_checklist.py`
- `scripts/run_retrain_with_gates.py`
- `scripts/run_rl_shadow_report.py`
- `scripts/run_shadow_day.py`
- `scripts/run_stress_tests.py`
- `scripts/run_tests.sh`
- `scripts/run_walk_forward.py`
- `scripts/scheduler.py`
- `scripts/sla_check.py`
- `scripts/start_depth_ws.py`
- `scripts/start_experiment.py`
- `scripts/start_live_stack.sh`
- `scripts/status_all.sh`
- `scripts/status_live_stack.sh`

- `scripts/stop_all.sh`
- `scripts/stop_experiment.py`
- `scripts/stop_live_stack.sh`
- `scripts/strategy_promote.py`
- `scripts/strategy_status.py`
- `scripts/telegram_alert_smoketest.py`
- `scripts/time_sanity_check.py`
- `scripts/tradingview_webhook.py`
- `scripts/trigger_test_incident.py`
- `scripts/tv_test_client.py`
- `scripts/unlock_trade_log.py`
- `scripts/update_scorecard.py`
- `scripts/update_trade_outcome.py`
- `scripts/validate_kite_session.py`
- `scripts/verify_audit_chain.py`
- `scripts/verify_decision_chain.py`
- `scripts/verify_decision_quote_age.py`
- `scripts/verify_desk_paths.py`
- `scripts/verify_execution_stat.py`
- `scripts/verify_feed_sla.py`
- `scripts/verify_ops_summary.py`
- `scripts/verify_outcome_labels.py`
- `scripts/verify_queue_filter.py`
- `scripts/verify_risk_profiles.py`
- `scripts/verify_risk_units.py`
- `scripts/verify_trade_fill_db.py`
- `scripts/verify_trade_identity.py`
- `scripts/verify_trailing_fields.py`
- `scripts/watchdog.sh`

## strategies (10 files)

- `strategies/__init__.py`
- `strategies/banknifty_intraday.py`
- `strategies/ensemble.py`
- `strategies/nifty_intraday.py`
- `strategies/position_sizer.py`
- `strategies/risk_manager.py`
- `strategies/sensex_intraday.py`
- `strategies/trade_builder.py`
- `strategies/vwap_orb.py`
- `strategies/zero_hero.py`

## tests (119 files)

- `tests/conftest.py`
- `tests/test_adaptive_risk_bounds.py`
- `tests/test_alpha_ensemble.py`
- `tests/test_alpha_factory_smoke.py`
- `tests/test_approval_binding.py`
- `tests/test_auth_health.py`
- `tests/test_auth_health_cache.py`
- `tests/test_auto_retrain_gates.py`
- `tests/test_capital_committee_report.py`
- `tests/test_circuit_breaker.py`
- `tests/test_confidence_weighted_sizing.py`
- `tests/test_cross_asset_stale.py`
- `tests/test_data_root_contract.py`
- `tests/test_day_type_history.py`
- `tests/test_db_guard.py`
- `tests/test_decay_dataset.py`
- `tests/test_decision_chain.py`

- `tests/test_decision_logger.py`
- `tests/test_decision_logger_schema.py`
- `tests/test_decision_store.py`
- `tests/test_decision_traceability.py`
- `tests/test_depth_subscription_tokens.py`
- `tests/test_ensemble.py`
- `tests/test_execution_alpha.py`
- `tests/test_execution_determinism.py`
- `tests/test_execution_readiness_guard.py`
- `tests/test_execution_refresh.py`
- `tests/test_execution_simulation.py`
- `tests/test_experiment_flags.py`
- `tests/test_experiments_smoke.py`
- `tests/test_expiry_selection.py`
- `tests/test_exposure_limits.py`
- `tests/test_feature_builder.py`
- `tests/test_feature_contract_integrity.py`
- `tests/test_feed_circuit_breaker.py`
- `tests/test_feed_freshness_units.py`
- `tests/test_feed_health_epoch_missing.py`
- `tests/test_feed_health_market_closed.py`
- `tests/test_feed_restart_breaker.py`
- `tests/test_feed_restart_guard.py`
- `tests/test_feed_sla_epoch_units.py`
- `tests/test_fill_model.py`
- `tests/test_gate_status_log.py`
- `tests/test_gatekeeper_cross_asset.py`
- `tests/test_governance_gate.py`
- `tests/test_hypotheses_smoke.py`
- `tests/test_incident_triggers.py`
- `tests/test_invalid_ltp.py`
- `tests/test_kite_auth_consistency.py`
- `tests/test_kite_depth_restart.py`
- `tests/test_kite_depth_ws_stability.py`
- `tests/test_legacy_scripts_guard.py`
- `tests/test_log_writer.py`
- `tests/test_manual_approval_enforcement.py`
- `tests/test_manual_approval_invariant.py`
- `tests/test_manual_approval_sample_run.py`
- `tests/test_market_data_index_quote_cache.py`
- `tests/test_market_data_indicator_freshness.py`
- `tests/test_market_data_indicator_mode_policy.py`
- `tests/test_market_data_minutes_since_open.py`
- `tests/test_market_data_unstable_reasons.py`
- `tests/test_market_data_warm_seed.py`
- `tests/test_meta_model_smoke.py`
- `tests/test_model_promotion_stub.py`
- `tests/test_news_calendar.py`
- `tests/test_news_encoder.py`
- `tests/test_news_ingestor.py`
- `tests/test_no_order_bypass_static.py`
- `tests/test_option_chain_refresh_filters.py`
- `tests/test_orchestrator_decision_event.py`
- `tests/test_orchestrator_pilot_unlock.py`
- `tests/test_orchestrator_reports_finally.py`
- `tests/test_orchestrator_strategy_gate_once.py`
- `tests/test_orchestrator_target_points_ideas.py`
- `tests/test_order_approval_store.py`
- `tests/test_order_intent_canonical.py`
- `tests/test_outcome_classification.py`
- `tests/test_paper_fill_simulator.py`
- `tests/test_paper_tournament.py`

- `tests/test_partial_profit_logic.py`
- `tests/test_position_sizer.py`
- `tests/test_post_trade_labeler.py`
- `tests/test_quote_freshness.py`
- `tests/test_readiness_state_machine.py`
- `tests/test_regime_canonical.py`
- `tests/test_regime_classifier.py`
- `tests/test_regime_wrappers.py`
- `tests/test_retrain_manager.py`
- `tests/test_review_packet.py`
- `tests/test_risk_engine.py`
- `tests/test_rl_env_sizing.py`
- `tests/test_rl_shadow_report.py`
- `tests/test_rl_trading_env.py`
- `tests/test_run_lock.py`
- `tests/test_scheduler_once_per_day.py`
- `tests/test_scheduler_time_utils.py`
- `tests/test_security_guard.py`
- `tests/test_session_calendar.py`
- `tests/test_session_guard.py`
- `tests/test_strategy_decay_gating.py`
- `tests/test_strategy_gatekeeper_mode_thresholds.py`
- `tests/test_strategy_lifecycle.py`
- `tests/test_stress_generator.py`
- `tests/test_synthetic_chain.py`
- `tests/test_telegram_tradable_mode.py`
- `tests/test_tick_module_imports_clean.py`
- `tests/test_tick_training.py`
- `tests/test_time_sanity_staleness.py`
- `tests/test_tradable_flag.py`
- `tests/test_trade_builder.py`
- `tests/test_trade_builder_blocked_candidates.py`
- `tests/test_trade_builder_index_bidask.py`
- `tests/test_trade_builder_trend_vwap_fallback.py`
- `tests/test_trade_identity_complete.py`
- `tests/test_trade_store_identity.py`
- `tests/test_trade_ticket.py`
- `tests/test_trailing_stop_persistence.py`
- `tests/test_truth_dataset_leakage.py`
- `tests/test_walk_forward_framework.py`