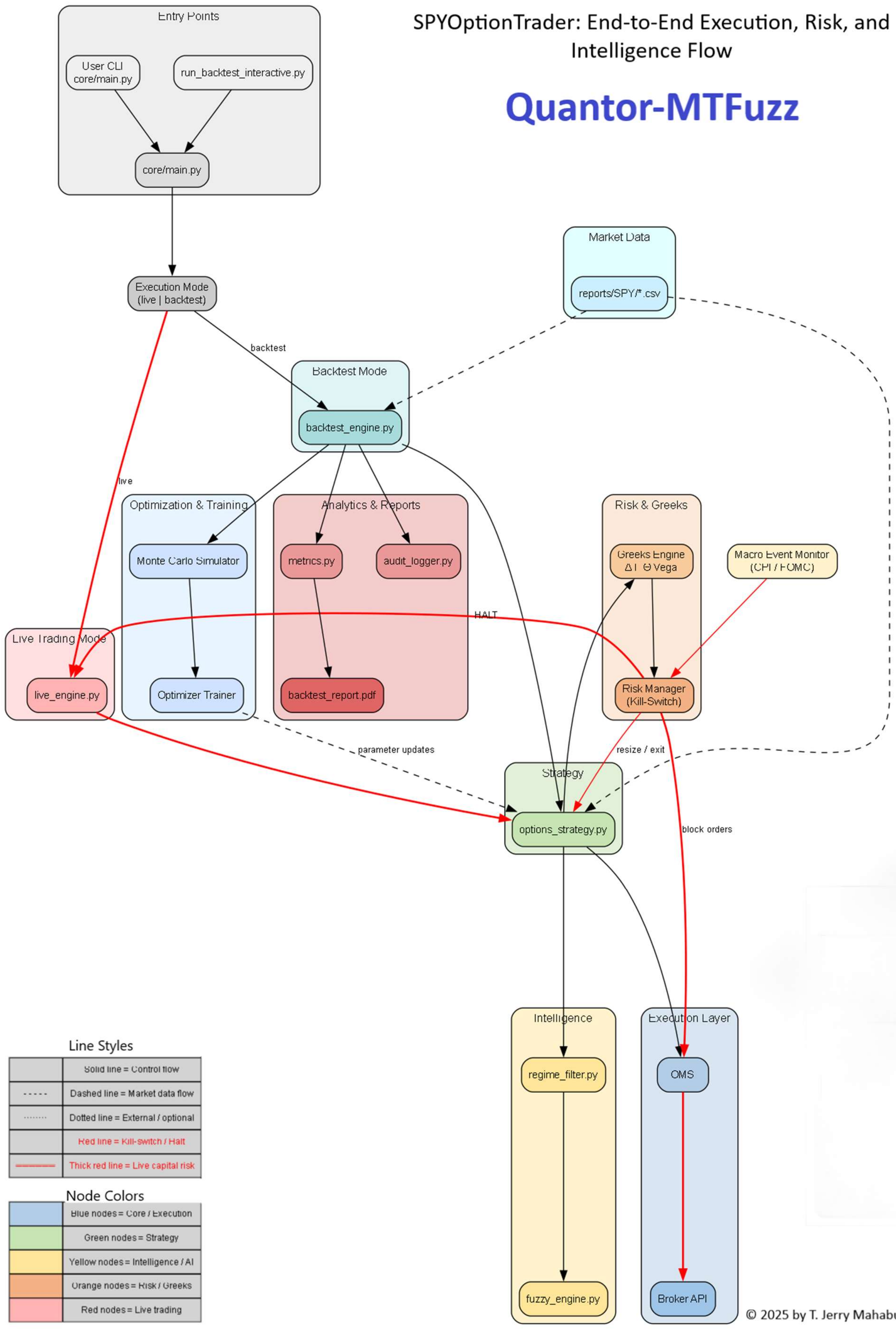
© 2025 by Jerry Mahabub
All rights reserved.

Author: Jerry Mahabub

SPYOptionTrader: End-to-End Execution, Risk, and Intelligence Flow

Quantor-MTFuzz



Line Styles

	Solid line = Control flow
- - - - -	Dashed line = Market data flow
········	Dotted line = External / optional
	Red line = Kill-switch / Halt
━━━━	Thick red line = Live capital risk

Node Colors

🟦	Blue nodes = Core / Execution
🟩	Green nodes = Strategy
🟨	Yellow nodes = Intelligence / AI
🟧	Orange nodes = Risk / Greeks
🟥	Red nodes = Live trading

© 2025 by T. Jerry Mahabub – All rights reserved

SPYOptionTrader is designed to **manage and bound risk**, not to eliminate it. Capital should only be deployed by parties who understand the risks of options trading and automated execution systems.

1. High-Level System Overview

Quantor-MTFuzz™ is a **modular quantitative trading framework** designed to:

- Backtest and eventually trade **SPY options (Iron Condors)**
- Use **volatility, regime detection, and multi-timeframe (MTF) intelligence**
- Enforce **liquidity, capital, and risk constraints**
- Generate **professional-grade analytics and PDF reports**

At present, it functions primarily as:

A complete execution scaffold with incomplete signal → trade conversion

- Backtests run cleanly
- Reports generate correctly
- Currently, for MTF, No trades are executed

---

2. Directory & Module Responsibilities

Root-Level Scripts

File	Purpose
core/main.py	**Primary CLI entry point**
run_backtest_interactive.py	Interactive wrapper for backtests
verify_structure.py	Validates directory + module integrity
project_structure_setup.py	Bootstraps directory layout
convertpytotxt.py	Developer tooling (archival / audit)
README.md	Documentation scaffold
config.template.py	User-configurable defaults

---

3. Core Engine Architecture

3.1 core/main.py — Orchestration Layer

This is the **command router**:

1. Parses CLI arguments
2. Loads config (CLI > config file > defaults)
3. Chooses execution mode:
    - live
    - backtest
4. Instantiates:
    - BacktestEngine
    - OptionsStrategy
    - Broker
    - Filters (MTF, regime, liquidity)

Important observation
No fatal errors occur → control flow is valid.

3.2 core/backtest_engine.py

Responsibilities:

- Load historical bar data
- Step through bars sequentially
- Track:
    - Cash
    - Equity
    - Positions
    - Drawdowns

- Call the strategy at each bar

Core loop (conceptual):

```
for bar in bars:
  strategy.evaluate(bar)
  broker.update_positions(bar)
  analytics.update(bar)
```

Current behavior
The engine runs — but the strategy never returns actionable trades.

3.3 core/broker.py

Simulates:

- Order fills
- Position PnL
- Capital allocation
- Max position enforcement

Key assumptions:

- Zero slippage
- Immediate fills
- No assignment risk
- No early exercise

These are acceptable **for Phase 1 backtesting**, but must be revisited for live trading.

3.4 core/liquidity_gate.py

Intended to block trades when:

- Bid/ask spreads too wide
- Volume too low

- IV too thin
- Option chain incomplete

Likely blocking everything
If default thresholds are too strict and no real option chain exists in backtest mode → *no trades*.

---

4. Strategy Layer

strategies/options_strategy.py

This is the **heart of the system** — and the current bottleneck.

Intended responsibilities:

- Select expiration (DTE window)
- Choose delta range
- Build Iron Condor:
  - Short put
  - Long put
  - Short call
  - Long call
- Validate:
  - Credit / width ratio
  - Margin usage
  - Regime conditions
- Emit a **trade object**

Current state (inferred from behavior):

- Entry conditions are *never satisfied*
- OR trade creation is stubbed / incomplete
- OR option chain data is unavailable in backtest

<u>This is why:</u>

Total Trades: 0

---

5. Intelligence Layer

5.1 intelligence/regime_filter.py

Detects macro regime using:

- IV Rank (IVR)
- VIX thresholds
- Possibly trend / volatility expansion

Mathematics (typical):

- IV Rank:

$$IVR = \frac{IV - IV_{min}}{IV_{max} - IV_{min}}$$

Used to:

- Widen wings
- Reduce size
- Block entries entirely

5.2 intelligence/fuzzy_engine.py

A **fuzzy logic decision layer**:

- Inputs:
    - Volatility
    - Trend
    - Momentum
    - Regime
- Outputs:

- Confidence score
- Bias (bullish / neutral / bearish)

This is ideal for:

- Iron Condor skewing
- Dynamic delta targeting

5.3 Multi-Timeframe (MTF) System

When --use-mtf is enabled:

- Multiple datasets are loaded (5m, 15m, 60m, D)
- Signals are computed independently
- A **consensus threshold** must be met

Why it "hangs"
Most likely causes:

1. Infinite loop waiting for aligned timestamps
2. Blocking I/O while resampling
3. No termination condition when one TF lacks data
4. Consensus never converges → loop never exits

This is a **logic bug**, not performance.

---

6. Data Layer

data_factory/*

- AlpacaGetData.py
  Pulls historical OHLCV
- polygon_client.py
  Intended for options / IV data
- sync_engine.py
  Aligns timestamps across timeframes

Strong data hygiene

- Interpolation
- Decimal normalization
- Missing-bar repair

Critical missing piece
Backtests appear to use **underlying bars only**, not **options chain snapshots**.

Iron Condors require:

- Strike-level IV
- Greeks
- Bid/ask
- Open interest

Without these → strategy can never construct trades.

---

7. Analytics & Reporting

analytics/metrics.py

Computes:

- Net PnL
- CAGR
- Max drawdown
- Sharpe ratio:

$$Sharpe = \frac{E[R] - R_f}{\sigma_R}$$

- Win rate
- Expectancy:

$$E = (W \cdot AvgWin) - (L \cdot AvgLoss)$$

---

analytics/audit_logger.py

Tracks:

- Trade lifecycle
- Rule violations
- Debug traces

PDF Reports

Generated successfully → confirms:

- Data pipeline
- Metrics pipeline
- Visualization pipeline

---

8. Why You Get Zero Trades (Root Causes)

Guaranteed contributors (ranked):

1. **No options chain data in backtest**
2. **Liquidity gate blocking entries**
3. **Entry logic incomplete or overly strict**
4. **MTF consensus never satisfied**
5. **Regime filter defaulting to "no trade"**

This is expected at this stage — not a failure.

---

9. Live Trading Risk Assessment

Major Risks

Risk	Impact
Slippage	High
Assignment risk	Severe
Volatility spikes	Severe
Correlated losses	Severe
Over-optimization	High
Latency	Medium
API outages	High

---

10. Risk Mitigation Enhancements (Recommended)

Position-Level

- SPX-style cash-settled logic
- Max gamma exposure
- Dynamic stop-loss by IV expansion
- Early exit before CPI / FOMC

Portfolio-Level

- Volatility targeting
- Correlation-adjusted sizing
- Kill switch on drawdown slope

---

11. Advanced AI & Optimization Roadmap

Phase 1 — Deterministic

- Complete option chain simulator
- Enable trades
- Validate logic

Phase 2 — Statistical

- Monte Carlo PnL distributions
- VaR / CVaR
- Regime-conditioned expectancy

Phase 3 — ML / AI

- LSTM for volatility forecasting
- Transformer-based regime classification
- Reinforcement learning for wing width
- Bayesian optimization for parameter search

---

12. Real-Time Analytics & Observability

Console

- Rolling Greeks
- Exposure heatmap
- Confidence scores

Webhooks

- FastAPI + WebSockets
- Real-time dashboards
- Trade audit trails

Client/Server

- Immutable trade ledger
- Compliance-grade logs
- Replayable sessions
