Synaptic Trading — Technical Evaluation (AI-First)

Candidate Pack • Last updated: October 31, 2025 14:45 IST

This evaluation explicitly expects heavy use of AI assistants (e.g., Claude Code CLI, MCP). It measures your ability to decompose problems, guide AI productively, and verify outputs with tests and benchmarks.

Format & Timebox

1. Phase 1 (Take-home, 6 hours total cap, due in 48 hours): T1, T2, T3, T4
1. Phase 2 (Live, 90 minutes): L1 (Debug & Extend), L2 (Performance + SQL)

AI Policy

Use AI tools freely. Required: submit sanitized prompt/session logs and a brief `VERIFICATION.md` describing how you audited AI outputs.

Deliverables (single repo)

1. `/src`, `/tests`, `/docs` (README.md, DESIGN.md, VERIFICATION.md), `/prompts`
1. A Makefile or run.sh to start services and tests

Phase 1 — Tasks

T1. FastAPI “Signal” Service

Goal: Ingest a simulated price stream and expose a simple trading signal with low latency.

1) Async consumer updating rolling state

2) `GET /signal?symbol=XYZ` → `trend=UP/DOWN/FLAT`, `rsi∈[0,100]`, and decision (BUY/SELL/HOLD) via MA(20/50)+RSI(14) rule

3) `WS /ws/signal` streaming latest decision

1. Input: Provided stub emits symbol, ts, price at ~50–100ms; plus a small OHLCV CSV with gaps/outliers.
1. Build:
1. Non-functional: P95 response time < 100ms locally at ~100 QPS; validation & error handling.
1. Testing: 3–5 unit tests for indicators + one async endpoint test.
1. Submit: Service code, tests, and a short perf note (how you measured latency & p95).

T2. Mini Backtest Runner (Nautilus-preferred)

`python

from nautilus_trader.backtest.engine import BacktestEngine

`

Configure: fees (~1 bps), fixed slippage (1 tick), size=1, EOD flat. Output: trades, PnL, max drawdown, daily Sharpe, equity curve CSV.

1. Track A (preferred): Nautilus Trader — Use the current import path:
1. Track B (fallback): pandas — Deterministic runner with the same rules/outputs.
1. Testing: Seeded test that reproduces identical equity for a provided CSV slice.
1. Deliberate trap: The pack includes an outdated Nautilus import snippet. Detect & correct it; note how you verified.

T3. Data Modeling + SQL (PostgreSQL)

Design a minimal schema for: `bars_1m`, `trades`, `orders`, `positions`, `pnl_daily` with partitioning/indexing hints. Provide SQL for:

1) Daily PnL rollup and max drawdown for a `strategy_id`

2) “Last known position per symbol” view

3) 30-day rolling Sharpe from `pnl_daily`

Also: explain one optimization you’d add if data grows 100×.

T4. Learning Sprint (Options & Microstructure)

Read the 1-page primer (Greeks intuition, slippage, expiry quirks, partial fills). Deliver a 1-page synthesis covering:

a) One microstructure failure mode that breaks naïve backtests and a mitigation in your runner

b) A simple slippage model you’d add next (and where it plugs into code)

c) Two sanity checks before live trading an options spread

> Cite which AI prompts you used and how you verified the answers.

Phase 2 — Live (90 min)

L1. Debug & Extend (60 min)

We provide a failing test (e.g., timezone truncation shifts equity by 1 day, or RSI NaN edge case). You: locate root cause, add a targeted test, fix it; then add a feature flag `confirm=True` that requires both MA cross and RSI divergence.

L2. Performance + SQL Drill (30 min)

Propose a latency budget & one low-effort optimization to push p95 < 80ms.

Write one SQL to return top-5 drawdowns (start, end, depth) from `pnl_daily`.

Provided Materials

1. `stream_stub.py` (async price emitter)

1. `ohlcv.csv` (with gaps/outliers)
1. `docs/nautilus_primer.md` (current imports & minimal runner outline)
1. `docs/snippet_outdated.md` (intentional wrong import to detect)
1. `tests/template_test.py` (a sanity test to extend)

Submission Notes

How to Respond (Google Doc structure)

Create one Google Doc and share a view link in your repo README and submission email.

Name: `Synaptic Trading — Tech Eval — <Your Name> — <YYYY-MM-DD>`

Top Summary:

1. GitHub repo URL (single repo), quick run command(s), environment & versions, time log.

Body (use these exact sections):

1. T1. FastAPI “Signal” Service — Approach, AI used, key files, run/test, benchmarks, improvements.
1. T2. Mini Backtest Runner — Architecture, AI used, how to run, tests/metrics, caveats.
1. T3. Data Modeling + SQL — Schema choices, 2–3 example queries, performance notes.
1. T4. Learning Sprint (Options & Microstructure) — 5–7 learnings, applied change.
1. L1. Debug & Extend (live) — hypothesis → fix path, evidence (logs/tests).
1. L2. Performance + SQL Drill (live) — bottleneck, change, measurement, result.

Appendix (optional): raw logs, prompts, diagrams.

Checklist:

1. [ ] Repo + README runnable
1. [ ] Google Doc follows T1–T4, L1–L2 order
1. [ ] Evidence (tests/benchmarks) per task
1. [ ] AI assistance and human validation noted

1. Keep Phase-1 effort within the 6-hour cap (honor system).
1. Include sanitized prompt/session logs and VERIFICATION.md outlining how you audited AI outputs.
1. Ensure the repo runs from a clean clone with one command.