AYUSH RAWAT / FIELD NOTES

The AI Trading Bot Did Not Make Me Rs 10 Lakh.

It gave me a safer experiment.

What a six-prompt paper-trading workflow can prove - and what it cannot.

By Ayush Rawat / 19 Aug 2026 / 8 min read

The internet has a familiar trading-bot story. Feed in a little money, let AI work while you sleep, and wake up to a five-figure profit. The title says Rs 10 lakh a month. The screenshots never show the capital, fees, slippage, drawdown, or the losing days.

I reviewed a different kind of build: a six-prompt workflow for a TypeScript and Node.js paper-trading bot. It uses real market data, keeps live orders out of the project, builds a no-memory baseline first, and only then adds a small memory layer. That is a good engineering sequence. It is not proof of a profitable strategy.

> **THE HONEST HEADLINE**
>
> The workflow can make a trading experiment more controlled and auditable. It cannot manufacture an edge, predict the next candle, or guarantee a monthly return.

1. Start with the evidence, not the thumbnail

The source material is a prompt pack, not a verified performance report. It specifies what an agent should build and what it should report. It does not provide a completed out-of-sample track record, a live account statement, or a capital-and-return calculation that supports Rs 10 lakh per month.

That distinction matters. A design can be careful and still produce a losing strategy. A clean dashboard can make a bad backtest look respectable. Before believing any performance claim, ask four questions:

- How much capital was used?
- What exact market, timeframe, and date range were tested?
- Were fees, slippage, spread, and rejected orders included?
- What happened on data the strategy did not use to tune itself?

If those answers are missing, the right label is **unverified experiment**, not passive income.

2. What the v2 workflow actually builds

The workflow is deliberately gated. Each prompt produces an artifact that can be checked before the next stage begins:

- **1. Connection check.** Verify the venue and MCP or API in paper mode with read-only smoke tests. No order is placed.
- **2. Strategy backtest.** Define the market, timeframe, entry and exit rules, risk rules, fees, and slippage before code becomes the source of truth.
- **3. Instructions file.** Convert the trader's actual preferences into explicit rules, limits, and a definition of done.
- **4. Raw build.** Build a complete paper bot with real public market candles and no memory layer.
- **5. Memory comparison.** Add a ledger and plain-English lessons, then run the raw and memory paths side by side.
- **6. Experiment-ready finish.** Add a README, environment template, reset command, and guardrails against accidental live orders.

The default example is BTCUSDT on a 5-minute interval with a 9-period and 21-period moving-average crossover. That is a starting hypothesis, not a discovered edge.

6	0	2
gated prompts from connection check to safe experimentation	live orders in the described workflow	replay paths: raw baseline and memory-enabled

3. Why the no-memory baseline comes first

Adding memory before measuring the raw strategy makes it impossible to tell what actually helped. The first run therefore has no ledger.csv, no learnings.md, and no adaptive filter. It fetches real candles, detects actual crossover events, simulates paper execution, and reports what the data says.

The required output is boring by design: total setups, wins, losses, win rate, average PnL, best trade, worst trade, and drawdown where the data supports it. If there are not enough setups, the bot should say that. If the strategy loses, it should say that. A system that cannot admit "not enough evidence" is not ready to trade.

4. Memory is a journal, not intelligence

The memory layer is the most useful idea in the build, but it needs a precise name. It is not a mind that understands markets. It is an auditable journal plus a filter.

ledger.csv records the timestamp, symbol, action, price, quantity, reason, mode, outcome, and PnL. learnings.md stores plain-English lessons from real paper or replay outcomes. Before a future BUY or SELL, the bot checks whether a similar setup has lost before and whether the lessons file contains a matching warning.

If the evidence matches, the final decision becomes **SKIP** and the reason is logged. If no real warning exists, the memory path should HOLD or tell you to keep paper testing. It must never seed fake losses to make the new version look smarter.

That is disciplined journaling. Calling it "the bot learned a strategy" would overstate what is happening. A skip filter can also over-block and miss a future winner, so it needs its own out-of-sample review.

5. The risk model is the part thumbnails hide

A win rate without a risk model is almost meaningless. The project needs configurable quantity, maximum position, stop or invalidation rules, maximum daily loss, and a drawdown limit before anyone discusses live money.

Fees and slippage belong in the test. So do spread, partial fills, latency, exchange outages, and the tax treatment that applies to the trader's jurisdiction. These are not polish items. They decide whether a small theoretical edge survives contact with execution.

There is also a simple capital reality check. A target return is inseparable from the capital and risk required to chase it. If someone promises Rs 10 lakh per month without stating both, they are selling an outcome, not explaining a system.

6. What this bot can prove

After a clean run, the workflow can give you evidence about process:

- The connection stayed in paper or test mode.
- The strategy rules were explicit and reproducible.
- The raw path used real market data rather than invented candles.
- The memory path changed decisions only when real prior evidence supported the change.
- The final project has visible guardrails and no secret keys in frontend code.

It still cannot prove that the next month will resemble the last one. Markets change. A strategy can decay. A backtest can be overfit. Paper fills can be kinder than live fills. The responsible conclusion is therefore narrow: **the experiment is better controlled**. The profit question remains open.

7. The build order I would keep

I would keep the six-stage order and change the marketing around it. Call the first run a baseline. Call the memory layer an evidence-backed filter. Publish the exact test window and assumptions. Show the losing trades as clearly as the winning ones. Then run a fresh date range that the strategy did not use while it was being tuned.

The strongest output may be a string most trading content avoids:

> **DECISION: HOLD.**
>
> No live order was sent. The system does not have enough evidence to justify a new risk.

Final take

An AI trading bot cannot create an edge because the prompt sounds confident. It can, however, make the experiment repeatable, keep a record of decisions, and stop the same mistake from being repeated without explanation.

That is a worthwhile project. It is also a much more honest promise than Rs 10 lakh while you sleep.

> **EDUCATIONAL DISCLAIMER**
>
> This article is for education and software review only. It is not financial advice, a recommendation to trade, or evidence of future performance. The underlying workflow is paper-trading first and contains no live-order path by default.