StockSim: A Dual-Mode Order-Level Simulator for Evaluating Multi-Agent LLMs in Financial Markets

Charidimos Papadakis, Giorgos Filandrianos, Angeliki Dimitriou, Maria Lymperaiou, Konstantinos Thomas, Giorgos Stamou
School of Electrical and Computer Engineering, AILS Laboratory
National Technical University of Athens
harrypapadakis02@gmail.com,
{geofila,angelikidim, marialymp, kthomas}@ails.ece.ntua.gr,
gstam@cs.ntua.gr

Abstract

We present **StockSim**, an open-source simulation platform for systematic evaluation of large language models (LLMs) in realistic financial decision-making scenarios. Unlike previous toolkits that offer limited scope, **StockSim** delivers a comprehensive system that fully models market dynamics and supports diverse simulation modes of varying granularity. It incorporates critical real-world factors, such as latency, slippage, and order-book microstructure, that were previously neglected, enabling more faithful and insightful assessment of LLM-based trading agents. An extensible, role-based agent framework supports heterogeneous trading strategies and multi-agent coordination, making StockSim a uniquely capable testbed for NLP research on reasoning under uncertainty and sequential decision-making. We open-source all our code at https://github.com/harrypapa2002/StockSim.

1 Introduction

Financial markets present complex, dynamic environments characterized by high uncertainty, consequential decisions, and measurable outcomes (Yadav et al., 2020; Rudkin et al., 2023; Nafiu et al., 2025). As large language models (LLMs) have demonstrated significant proficiency in sequential reasoning and decision-making tasks (Chen et al., 2024; Liu et al., 2025), systematically evaluating these models within realistic financial scenarios has emerged as a crucial research direction for NLP.

However, the NLP community currently faces substantial obstacles due to a lack of standardized and openly accessible platforms specifically designed for rigorous evaluation of LLMs in realistic trading contexts (Li et al., 2024a; Lu et al., 2025). Common evaluation practices that rely on static benchmark datasets inadvertently risk data leakage, as these datasets or similar financial texts often appear in LLM training corpora (Dong et al., 2024; Singh et al., 2024; White et al., 2024). Consequently, performance metrics become inflated, and the models fail to generalize effectively to genuinely unseen scenarios, creating unrealistic expectations and potential financial risks when deployed.

Existing evaluation platforms further compound these limitations, as highlighted in Table 1. Frameworks such as Backtrader[1] and FinRL (Liu et al., 2020, 2022) offer extensive historical backtesting but abstract away crucial trading microstructure aspects like latency and detailed order-book dynamics. Conversely, platforms like ABIDES (Byrd et al., 2020), PyMarketSim (Mascioli et al., 2024), and JAX-LOB (Frey et al., 2023) simulate precise order-level market mechanics but depend heavily on expensive, limited, tick-level datasets, restricting their practical applicability and scalability. Additionally, frameworks designed for multi-agent LLM coordination, such as TradingAgents (Xiao et al., 2024), often use highly simplified market representations based solely on coarse historical data, omitting realistic execution and latency considerations critical for evaluating LLM behavior.

The fragmented landscape forces researchers either to simplify market interactions unrealistically or to invest significant resources developing custom, often proprietary pipelines, which hinder reproducibility, fair comparisons, and collective research progress in NLP-driven financial decision-making. To overcome such challenges, we introduce **StockSim**, a unified, open-source, fully preconfigured platform explicitly designed to rigorously assess LLM behavior in realistic, dynamic financial scenarios. It integrates two complementary simulation modes behind a single interface: (i) an **order-level** execution mode that simulates fine-grained market behavior capturing latency, queue dynamics, and microstructural dynamics; and (ii) a **candlestick-level** (price-bar-level) execution mode that enables scalable evaluation while abstracting

[1] https://pypi.org/project/backtrader/

Framework	**Execution Granularity**	**Async Latency**	**Real-time LOB**	**History Back-test**	**No-code Setup**	**LLM Agent Support**	**External News**	**Multi Instrument**
StockSim (ours)	Order	✓	✓	✓	✓	✓	✓	✓
ABIDES	Order	∼	✓	✓	✗	✗	✗	✓
PyMarketSim	Order	✗	✓	✗	✗	✗	✗	✓
JAX-LOB	Order	✗	✓	✓	✗	✗	✗	✓
FinRL / Meta	Bar	✗	✗	✓	∼	✗	∼	✓
TradingAgents	Daily	✗	✗	✓	✗	✓	✓	✓

Table 1: Feature comparison of open-source trading simulators. *Legend.* ✓: supported; ✗: not supported; ∼: partial or approximate support.

away low-level market effects.

StockSim shifts the research focus from evaluation infrastructure development to core NLP-driven agent design and experimentation. Specifically, it enables: 1) Comprehensive ordel-level simulation, capturing essential trading dynamics from detailed tick-level events to aggregated price bars. 2) Flexible, high-throughput bar-level execution across diverse market scenarios, assets (stocks, cryptocurrencies), and temporal resolutions. 3) Agent utilization of external multi-modal information, such as news sentiment and financial reports, facilitating realistic NLP experimentation. 4) Robust, production-grade infrastructure providing authentic market data integration, real-time technical indicator calculations, and detailed agent performance tracking.

2 Background

StockSim exposes agents to realistic trading dynamics, including order execution via a limit-order book (LOB), delays from latency, price shifts from market impact, and costs like slippage. Strategies rely on historical OHLCV (candlestick) data, derived technical indicators, and real-time market mechanics - making familiarity with these concepts essential for interpreting agent behavior.

Order types. A *market order* executes immediately against the best prices available; a *limit order* is queued and only fills at its stated price (limit) or better. Typically used to enter trades; a *stop order* is executed as a market order once a trigger price is hit, typically used to exit trades.

Limit-order book (LOB). The LOB is the continuously updated queue of pending buy (bid) and sell (ask) limit orders at each price level. It is responsible for storing these orders and facilitating their resolution when matching conditions are met. It drives price discovery and is the core of StockSim's real-time simulator.

Latency & market impact. *Latency* is the delay between submitting an order and that order reaching the order-matching system; even sub-second differences matter in modern electronic environments. *Market impact* is the price movement an order itself induces. Large market orders may clear out substantial amounts of pending orders in the limit-order book, moving the asset's price.

Slippage. The difference between the prices at which an order is submitted by the trader (or trading system) and the price it actually gets fulfilled, is called *slippage*; it is the consequence of latency and market impact effects and is a major source of hidden and unpredicted trading costs.

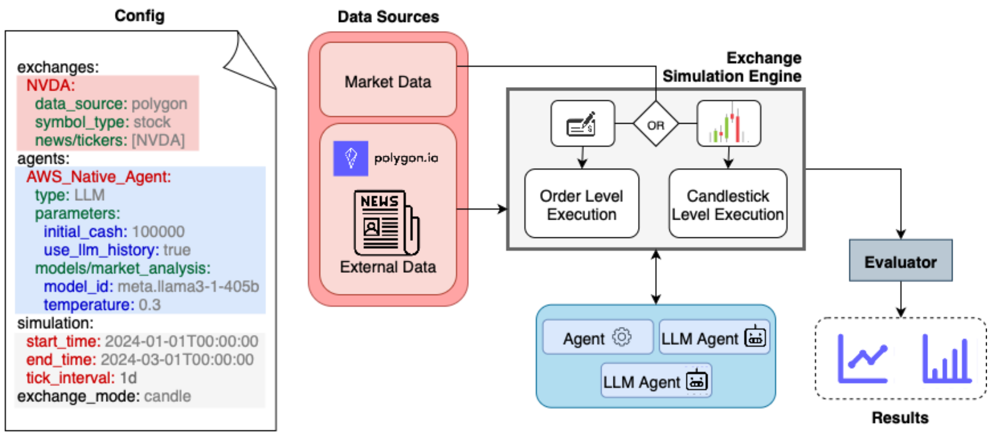
Market microstructure. The term covers statistical regularities of order placements, cancellations, spread dynamics, queue imbalance, and how they interact to form price. StockSim's order-book engine reproduces these dynamics so that agents must cope with queue position, partial fills, and other microstructure realities.

OHLCV (candlestick) bars & timeframes. Historical data are often stored as *open, high, low, close, volume* (OHLCV) tuples, commonly called *candlesticks* due to their appearance on a chart: (i) **Open** – the first traded price in the bar; (ii) **High** – the maximum traded price; (iii) **Low** – the minimum traded price; (iv) **Close** – the last traded price; (v) **Volume** – total quantity traded during the bar. The bars have a constant start/end time (i.e. 5 minutes, 1 day, etc.) referred to as the chart's *timeframe*. StockSim's candlestick engine replays any asset at resolutions from one minute to one day (and beyond).

Technical indicators. Deterministic functions of past price or volume, such as moving averages, Relative Strength Index (RSI) or Average True Range (ATR) that serve as numeric features of concentrated information, used heavily in trading. More details can be found in Appendix C.

3 System Architecture

StockSim employs a modular, asynchronous architecture designed around four core components



```
exchanges:
  NVDA:
    data_source: polygon
    symbol_type: stock
    news/tickers: [NVDA]
agents:
  AWS_Native_Agent:
    type: LLM
    parameters:
      initial_cash: 100000
      use_llm_history: true
    models/market_analysis:
      model_id: meta.llama3-1-405b
      temperature: 0.3
simulation:
  start_time: 2024-01-01T00:00:00
  end_time: 2024-03-01T00:00:00
  tick_interval: 1d
exchange_mode: candle
```

Figure 1: Overview of StockSim's system architecture and input/output scheme. Modules are color-coded by function and mapped to corresponding blocks in the centralized config file. This design supports flexible, code-free customization of simulation parameters, agent behavior, and data sources.

that enable comprehensive LLM evaluation in realistic trading environments. Figure 1 illustrates the system's data flow and component interactions, highlighting two execution mechanisms—*order level* and *candlestick level execution* —seamlessly integrated with shared modules for market data retrieval, indicator computation, news/fundamentals integration, and agent interactions. This design ensures consistency, flexibility, and scalability, supporting diverse experimental setups and facilitating reproducible experimentation on sequential decision-making in financial contexts.

3.1 Exchange Simulation Engine

The core component of StockSim is the *Exchange Simulation Engine*, which asynchronously manages and coordinates the simulated trading environment. Its primary responsibilities include: (1) receiving and processing agent actions (e.g., order placements); (2) simulating realistic market dynamics for order execution; (3) computing and disseminating relevant market indicators; and (4) providing agents with timely access to market and external information (e.g. news, corporate events).

The *Engine* acts as the central intermediary between data sources and trading agents. It does not directly store external data; in contrast, it routes the data dynamically from their respective sources to agents upon request, actively maintaining internal states related to orders and trades, including execution status and market impact. Each agent runs as a separate process and communicates asynchronously with the Engine to submit orders, request data, or receive market updates. This asynchronous communication is managed via RabbitMQ, an advanced message broker that ensures reliable message delivery and scalable communication[2]. Moreover, the *Engine* supports two distinct ways of resolving orders submitted by agents, designed to accommodate various research scenarios:

Order Level Execution emulates real market behavior by operating directly on the LOB, where the agent submits limit or market orders that interact with a stream of order book events (placements, cancellations, executions). Orders are matched based on price-time priority: e.g., a buy limit order at \$100 will execute only if a sell order exists at \$100 or lower; otherwise, it queues until matched or canceled. Execution may be full or partial, depending on available volume. The environment updates tick-by-tick, capturing fine-grained dynamics such as queue position, order interleaving with other market participants, and the impact of latency between action submission and book update. This level offers high realism and is critical for evaluating strategies sensitive to microstructure effects.

Candlestick Level Execution places orders based on aggregated candlestick data (OHLCV). That is, if the agent submits an order at a price that falls within the range of a given candle, the order can be executed; otherwise, it cannot. We adopt this approach as it provides access to a larger dataset, enabling testing over longer historical periods. Moreover, most LLMs are evaluated under this setting (Li et al., 2024b). Despite being widely used, mainly due to its simplicity, this mode fails to capture critical dynamics, such as latency and other microstructural elements of real markets.

Both execution mechanisms consistently pro-

[2]https://www.rabbitmq.com/

vide agents with computed market indicators (e.g., SMA, EMA, RSI, VWAP) derived from real-time or historical market data, enhancing agents' decision-making capabilities (see Appendix C).

3.2 Data Sources

StockSim distinguishes between two primary categories of data: (1) **market data**, which include price, volume, and order-flow information; and (2) **external data**, such as news, corporate actions, and fundamental metrics. The *Exchange Simulation Engine* orchestrates these inputs asynchronously, delivering them to agents in simulation time.

Market Data. StockSim supports two types of market data: detailed *order-level* data and simplified *bar-level* (candlestick) data.

In the *candlestick level execution*, the data is provided as aggregated summaries i.e. OHLCV bars, obtained from general data sources like Alpha Vantage and Polygon.io[3]. Because these summaries do not include detailed, within-bar price movements, StockSim simulates realistic price paths within each bar. This allows agents to place conditional orders (like stop losses) that execute plausibly, even though exact moment-to-moment data is not available. In the *order level execution*, each market action, such as placing, changing, or cancelling an order is individually tracked. These detailed events come either from datasets like LOBSTER[4] or from logs created during the simulation. Each event has precise timestamps (milliseconds), allowing realistic simulation of latency and slippage.

External Data. Agents may request news headlines, earnings calendars, splits, dividends, or fundamental ratios at any simulation step. These streams are supplied through the same provider set (Alpha Vantage, Polygon, etc.) and exposed via a unified query interface implemented by the Exchange Simulation Engine. This abstraction lets agents reason over time-sensitive, multi-modal inputs and supports the development of more interpretable, information-driven trading strategies. Because each provider is wrapped by a lightweight adapter that maps its payloads to StockSim's canonical schema, adding a new API is as simple as contributing a single Python file, ensuring the platform can evolve alongside the data ecosystem.

[3]`https://www.alphavantage.co`; `https://polygon.io`

[4]`https://lobsterdata.com`

3.3 Agent

Agents are the research object in StockSim. Regardless of which *execution engine* mode is plugged in, every agent interacts with the simulator through the same asynchronous message API; only the *engine* decides how an order is ultimately filled. This separation allows a single agent implementation - written once in Python - to be stress-tested on both the order level and candlestick level execution mode without code changes.

Core Capabilities. Each agent may:

1) **Subscribe to data streams.** Agents request snapshots or streaming updates of *(i)* market state (order book depth or OHLCV bars), *(ii)* technical indicators produced on-the-fly, and *(iii)* external content such as news, corporate events etc.

2) **Submit and cancel orders.** The message schema supports `MARKET`, `LIMIT`, and `STOP` instructions of arbitrary size. In the **Order Level Execution** mode the order is routed to a price–time priority matcher and may experience queueing, latency, and market impact. In the **Candlestick Level Execution** mode the same message is interpreted by by whether the price action within a candle crosses the pending order price.

3) **Receive execution outcomes and portfolio updates.** Agents immediately receive feedback about their submitted orders, including confirmations of successful trades (fills), rejections if an order could not be executed, cancellations if the agent withdraws an order, and updates about their profit-and-loss (P&L). This ensures agents have timely information to adapt their decisions.

4) **Log reasoning.** Optional free-form "explanation" strings can accompany every order; these are preserved by the engine and can be inspected offline to analyse LLM rationale and decision trace.

Multi-Agent and Specialist Roles. StockSim includes a modular `LLMTradingAgent` that delegates decision-making to a team of specialist LLMs (e.g., market-technical analyst, news analyst, fundamental analyst). Each analyst operates with its own prompt template, memory context, and reasoning function. While adding new analyst roles does require lightweight code changes, such as implementing a new analyst class and registering it in the coordinator, the process is intentionally simple, well-documented, and configuration-driven. The modular structure ensures that agent internals remain decoupled from the simulation engine, making it easy to test new multi-agent frameworks or LLM coordination setups with minimal friction.

This design enables researchers to: *(i)* rapidly prototype new agent structures, including macroeconomic, sentiment, or reflection-based analysts; *(ii)* experiment with different backbones or prompting techniques per role; *(iii)* plug analysts into different coordination strategies, such as voting, tree-of-thought, or chain-of-experts; *(iv)* conduct clean ablation studies by toggling analysts or modifying their configuration in one place.

Research Convenience. StockSim maintains a unified interface across simulation engines: switching between order level and candlestick level execution requires only a configuration change, not architectural rewrites, isolating research focus on core questions like prompt engineering, reasoning, or analyst collaboration, without being burdened by low-level simulation mechanics. In technical terms, we implement a base class per agent, enabling easy adaptation to specific needs. We also provide pre-configured wrappers for widely used LLMs, including LLaMA, OpenAI's offerings (e.g., GPT-o4, o3), and Anthropic's models (e.g., Claude Sonnet, Haiku, Opus).

3.4 Evaluator

The *Evaluator* component subscribes to all trade executions, recording a complete history of positions, cash and realized P&L (Profit & Loss). Once the simulation finishes, it computes a concise set of core performance metrics, such as overall return, risk-adjusted ratios (e.g. Sharpe), drawdown, and basic trade statistics, finally packaging them into a uniform report. A list of predefined metrics and their definitions can be found in Appendix B. Crucially, the metric evaluation system is designed to be **fully extensible**. Users can seamlessly integrate custom performance measures, such as tail-risk, turnover, or regime-specific statistics, by registering additional evaluation components, all without modifying the core simulation engine.

For visual diagnostics, the *Evaluator* provides several useful outputs, such as equity curves showing portfolio value changes over time, candlestick charts highlighting executed trade entries and exits clearly marked on the price data, and comprehensive summary tables of key trading performance metrics. These outputs can be directly generated by StockSim's built-in plotting utilities or exported in JSON format for further analysis. An example

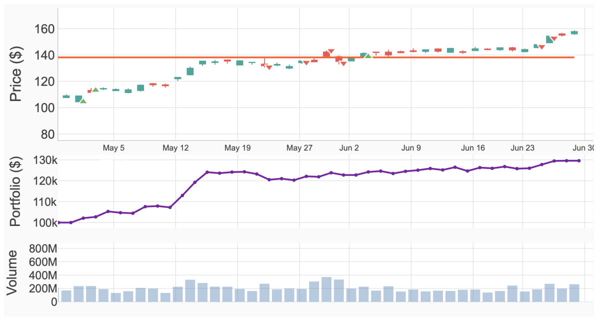


Figure 2: Performance of GPT-o3, showing executed trades and portfolio value evolution. Sell actions are marked with ▼, while buy actions are marked with ▲.

of these diagnostics for a trading session involving NVIDIA (NVDA) using the o3 model evaluated at the candlestick level is shown in Figure 2. Additionally, all figures provided by StockSim offer interactive features (zoom-in/out and hover-to-display details like OHLCV data) and model explanations, advancing interpretability of decision-making. Examples can be found in Appendix D.

4 Evaluation

Scalability and Consistency of StockSim are evaluated through a series of controlled simulation tests using varying numbers of *deterministic* agents. Each agent follows predefined strategies, such as moving average crossovers or buy-and-hold, allowing us to observe the simulation engine's behavior under repeatable conditions. To ensure that the evaluation reflects only the core behavior of the engine, we exclude LLMs, which introduce variability in latency, resource usage, and output consistency due to differences in deployment mode, reasoning strategy, and stochastic outputs. The results confirm StockSim's consistency: across all runs, simulation outputs (including order placements, executions, and performance metrics) **remain identical**. This repeatability empirically verifies the platform's deterministic behavior and validates its correctness, since any deviation would indicate flaws in the design or execution logic.

Scalability is assessed by monitoring system-level metrics during each run, including CPU utilization across all cores and memory usage (in MB) for both the simulation engine and RabbitMQ. Results across agent configurations are presented in Figure 3, confirming that StockSim **scales almost linearly** up to $\sim$150 agents: the simulation container's mean CPU load increases from 8% to 27%, while memory usage rises from 0.8 GB to 2

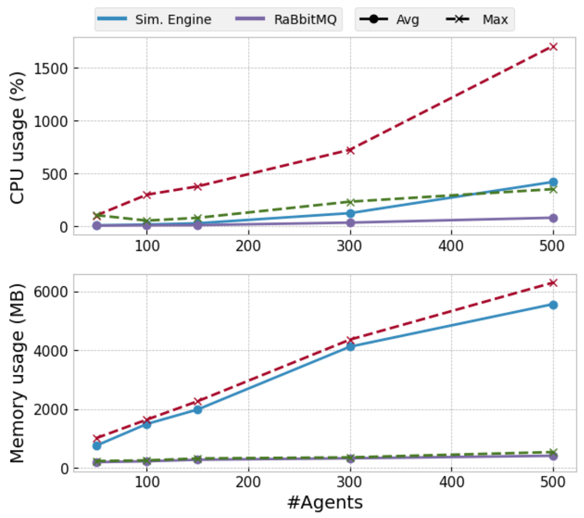


Figure 3: System performance metrics (memory/CPU usage) for varying numbers of deterministic agents.

Metric	o4-mini	o3
ROI (↑)	0.0734	0.2956
Sharpe Ratio - SR (↑)	0.1652	0.376
Annualized SR (↑)	2.6218	5.9682
Sortino Ratio (↑)	0.2868	1.0587
Win Rate (↑)	0.6667	1.0
Profit Factor (↑)	2.3691	999.0
Max Drawdown (↓)	0.0306	0.0323
Num Trades	31	9
Num Closed Trades	21	6
Total Traded Volume	931,416.775	368,306.25
Average Trade Size	30,045.70	40,922.92
ROIC	0.0151	0.1633
Profit per Trade (↑)	258.47	4,520.13
Last Portfolio Value (↑)	107,338.30	129,556.75
Realized P&L	5,427.80	27,120.75

Table 2: Summary of trading performance metrics (Appendix B) for GPT-o4-mini and GPT-o3.

GB, both roughly proportional to the agent count. Beyond this point, the workload becomes super-linear: at 300 and 500 agents, mean CPU usage surges to 123% and 418%, and memory climbs to 4.1 GB and 5.6 GB, respectively, with peak values reaching nearly four times the averages.

Despite this growth, the resource demands of the simulation framework remain modest; even at maximum load, usage peaks at 5.6 GB of RAM and a few CPU cores. All experiments are conducted on a MacBook Pro with an Apple M3 Pro chip (11-core CPU) and 18 GB of unified memory, underscoring StockSim's efficiency. Running 500 concurrent LLM agents in parallel is practically infeasible on such hardware, whereas this analysis demonstrates that StockSim can handle such scale with ease.

LLM Trading Behavior To demonstrate the ease with which insights about model behavior can be extracted using StockSim, we run a simulation for two LLMs, GPT-o4-mini and GPT-o3, using the same prompt (Appendix E) on the NVIDIA stock over a two-month period, from April 28, 2025, to June 28, 2025. The simulation assumes daily trading, with orders placed before market open. The results based on the performance metrics provided by StockSim, are presented in Table 2, revealing distinct trading patterns and strategic behaviors between LLMs. Metrics such as ROI, Profit per Trade, and Profit Factor highlight that GPT-o3 pursues a more selective trading strategy characterized by fewer, larger-sized positions with higher conviction, resulting in greater profitability and reduced downside risk, as demonstrated by its superior Sortino Ratio and perfect Win Rate. Conversely, GPT-o4-mini exhibits a more active trading style, evidenced by its higher number of trades and greater traded volume, indicating frequent market interactions but lower profit efficiency per transaction. The contrasting Sharpe Ratio and Annualized Sharpe Ratio further underscore GPT-o3's superior ability to maintain consistent, risk-adjusted returns over time, while GPT-o4-mini's lower metrics suggest that its strategy involves more frequent but less decisive market positions. Overall, the *evaluator*'s results effectively capture and distinguish the underlying strategic differences between the two LLMs, allowing clear interpretation of their respective trading behaviors. Importantly, we are able to obtain these results **without writing any code**, paving the way for exploring more LLM-driven trading strategies.

5 Conclusion

StockSim represents a significant advancement in NLP research infrastructure, providing a sophisticated platform for studying LLM abilities in realistic, multi-agent, temporal reasoning scenarios. By combining production-grade financial simulation with comprehensive NLP evaluation tools, StockSim enables research that bridges research experiments with real-world deployment requirements. The open-source availability and extensive documentation ensure broad accessibility for advancing our understanding of LLM behavior in complex, consequential decision-making environments.

Limitations

While StockSim provides a comprehensive framework for LLM evaluation in financial domains, some limitations should be noted. The platform requires substantial computational resources for multi-agent simulations and may have scalability constraints for very large agent populations. Market simulation, while realistic, cannot fully capture all complexities of actual trading environments including liquidity constraints and market impact. The current evaluation metrics, while comprehensive, may not capture all aspects of decision quality relevant to financial applications. Additionally, the platform's focus on financial markets may limit generalizability to other sequential decision-making domains.

Ethics Statement

STOCKSIM is designed for research purposes and uses simulation environments that do not interact with real financial markets, eliminating concerns about market manipulation. The platform promotes responsible AI development by providing tools to systematically study LLM reliability and consistency, as well as model explanations that can be further assessed by humans for their credibility and accuracy. All market data is obtained through legitimate commercial APIs with appropriate licensing. The open-source nature of the platform ensures transparency and enables broader scrutiny of the evaluation methodologies employed.

Acknowledgements

We thank the contributors to the open-source libraries that made this work possible, including the maintainers of RabbitMQ, asyncio, and various LLM APIs. We also acknowledge Polygon.io for providing comprehensive market data access that enables realistic financial simulations. This work was supported by the Hellenic Foundation for Research and Innovation (HFRI) under the 5th Call for HFRI PhD Fellowships (Fellowship Number 19268).

References

David Byrd, Maria Hybinette, and Tucker Hybinette Balch. 2020. Abides: Towards high-fidelity multi-agent market simulation. In *Proceedings of the 2020 ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*, pages 11–22.

Á. Cartea and S. Jaimungal. 2015. Risk metrics and fine tuning of high-frequency trading strategies. *Mathematical Finance*, 25.

Dingyang Chen, Qi Zhang, and Yinglun Zhu. 2024. Efficient sequential decision making with large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9157–9170, Miami, Florida, USA. Association for Computational Linguistics.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12039–12050, Bangkok, Thailand. Association for Computational Linguistics.

Sascha Yves Frey, Kang Li, Peer Nagy, Silvia Sapora, Christopher Lu, Stefan Zohren, Jakob Foerster, and Anisoara Calinescu. 2023. Jax-lob: A gpu-accelerated limit order book simulator to unlock large scale reinforcement learning for trading. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 583–591.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Koduvayur Subbalakshmi, Guojun Xiong, et al. 2024a. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. *arXiv preprint arXiv:2412.18174*.

Yuan Li, Bingqiao Luo, Qian Wang, Nuo Chen, Xu Liu, and Bingsheng He. 2024b. CryptoTrade: A reflective LLM-based agent to guide zero-shot cryptocurrency trading. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1094–1106, Miami, Florida, USA. Association for Computational Linguistics.

Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Wang, Zhaoran Wang, and Jian Guo. 2022. Finrl-meta: Market environments and benchmarks for data-driven financial reinforcement learning. *Advances in Neural Information Processing Systems*, 35:1835–1849.

Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. 2020. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. *CoRR*.

Zhaowei Liu, Xin Guo, Fangqi Lou, Lingfeng Zeng, Jinyi Niu, Zixuan Wang, Jiajie Xu, Weige Cai, Ziwei Yang, Xueqian Zhao, et al. 2025. Fin-r1: A large language model for financial reasoning through reinforcement learning. *arXiv preprint arXiv:2503.16252*.

Guilong Lu, Xuntao Guo, Rongjunchen Zhang, Wenqiao Zhu, and Ji Liu. 2025. Bizfinbench: A business-driven real-world financial benchmark for evaluating llms. *arXiv preprint arXiv:2505.19457*.

Chris Mascioli, Anri Gu, Yongzhao Wang, Mithun Chakraborty, and Michael Wellman. 2024. A financial market simulation environment for trading agents using deep reinforcement learning. In *Proceedings* *of the 5th ACM International Conference on AI in Finance*, pages 117–125.

Ashimiyu Nafiu, Salaam Olawale Balogun, Courage Oko-Odion, and Olanrewaju Olukoya Odumuwagun. 2025. Risk management strategies: Navigating volatility in complex financial market environments.

Simon Rudkin, Wanling Qiu, and Paweł Dłotko. 2023. Uncertainty, volatility and the persistence norms of financial time series. *Expert Systems with Applications*, 223:119894.

Aaditya K Singh, Muhammed Yusuf Kocyigit, Andrew Poulton, David Esiobu, Maria Lomeli, Gergely Szilvasy, and Dieuwke Hupkes. 2024. Evaluation data contamination in llms: how do we measure it and (when) does it matter? *arXiv preprint arXiv:2411.03923*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*, 4.

J Welles Wilder Jr. New concepts in technical trading systems, trend research. *PO BOX*, 450:130.

Yijia Xiao, Edward Sun, Di Luo, and Wei Wang. 2024. Tradingagents: Multi-agents llm financial trading framework. *arXiv preprint arXiv:2412.20138*.

Gaurang Singh Yadav, Apratim Guha, and Anindya S Chakrabarti. 2020. Measuring complexity in financial data. *Frontiers in Physics*, 8:339.

A Market Indicators in StockSim

Agents in StockSim, including both benchmark and LLM agents, receive a compact *market-embedding* consisting of hand-crafted scalar indicators summarizing critical aspects of the latent market state. These indicators serve as low-entropy numeric tokens, simplifying the sequential-reasoning challenge for systematic evaluation across diverse agent types. The embedding includes the following categories:

- **Trend Indicators:** Capture directional movement in market prices, allowing agents to infer the prevailing trend.
  - *Simple Moving Average (SMA)*: Average price over a specified look-back window, smoothing out short-term fluctuations.
  - *Exponential Moving Average (EMA)*: Weighted average emphasizing recent prices, responsive to short-term trend changes.
- **Momentum Indicators:** Measure the strength and velocity of price movements, enabling agents to recognize acceleration or deceleration in trends.
  - *Relative Strength Index (RSI)*: Quantifies momentum by comparing average gains and losses, identifying overbought or oversold conditions.
  - *Moving Average Convergence Divergence (MACD)*: Highlights momentum shifts by comparing short-term and long-term EMAs.
- **Volatility Indicators:** Reflect market uncertainty or risk by measuring price variability.
  - *Average True Range (ATR)*: Computes volatility based on price ranges, useful for assessing market turbulence.
  - *Bollinger Bands*: Envelopes around SMA, indicating volatility expansion or contraction.
- **Volume and Micro-Structure Indicators:** Capture trading activity intensity and structural nuances of market participation.
  - *Volume Weighted Average Price (VWAP)*: Reflects the average traded price weighted by volume, indicating liquidity-driven price levels.
  - *Order Book Imbalance*: Measures differences in bid and ask quantities, signaling buying or selling pressure.
- **Support-Resistance Indicators:** Identify critical price levels at which market dynamics historically reverse or accelerate.
  - *Historical Support Levels*: Previous price levels where buying activity halted declines.
  - *Historical Resistance Levels*: Prior price ceilings where selling activity halted rallies.

By encapsulating rich market dynamics into concise numeric tokens, StockSim's market-embedding reduces complexity, facilitating effective sequential decision-making research.

B Trading Performance Metrics

The definitions of the predefined metrics used by StockSim for evaluation are as follows[5]:

- **ROI (Return on Investment):** ROI measures the profitability of an investment relative to its cost. It is calculated as the ratio of net profit to the initial capital investment, indicating the efficiency of the strategy.
- **Sharpe Ratio (SR):** The Sharpe Ratio evaluates the risk-adjusted return of an investment. It is defined as the ratio of the excess return over the risk-free rate to the standard deviation of returns. A higher Sharpe Ratio suggests a more favorable risk-return profile.
- **Annualized Sharpe Ratio (Annualized SR):** This metric adjusts the Sharpe Ratio to an annual scale, enabling comparison across different timeframes and investment durations.
- **Sortino Ratio:** The Sortino Ratio refines the Sharpe Ratio by considering only the downside deviation (negative volatility), thereby focusing on harmful risk. It is the ratio of excess return to the standard deviation of negative returns.
- **Win Rate:** The Win Rate represents the proportion of profitable trades out of the total number of executed trades. It reflects the consistency and reliability of the strategy.
- **Profit Factor:** The Profit Factor is defined as the ratio of gross profits to gross losses across all trades. Values greater than 1 indicate a profitable strategy; higher values imply greater efficiency in managing risk and reward.
- **Max Drawdown:** Maximum Drawdown measures the largest peak-to-trough decline in portfolio value during the evaluation period. It serves as an indicator of downside risk and potential capital loss.
- **Number of Trades:** This is the total number of executed trades, including both open and closed positions.
- **Number of Closed Trades:** This refers to the total count of trades that have been fully executed and settled within the evaluation period.

[5]https://www.investopedia.com

- **Total Traded Volume:** The aggregate monetary value of all executed trades. This metric reflects the scale and activity of the trading strategy.
- **Average Trade Size:** The mean monetary value of executed trades. It provides insight into the average scale of trade operations.
- **ROIC (Return on Invested Capital):** ROIC quantifies the return generated on the capital that was actually deployed in trades. It provides a precise view of the strategy's capital efficiency.
- **Profit per Trade:** The average net profit generated per closed trade. This metric highlights the effectiveness of individual trade decisions.
- **Last Portfolio Value:** The total value of the portfolio at the conclusion of the evaluation period. It reflects the cumulative financial outcome of the strategy.
- **Realized P&L:** The net profit or loss from closed trades. This metric does not include unrealized gains or losses from open positions.

C Technical Indicators

Technical indicators are deterministic mathematical functions derived from historical price and/or volume data (Wilder Jr; Cartea and Jaimungal, 2015). They transform raw market data into condensed numeric features that aim to reveal trends, momentum, volatility, or potential reversals. These indicators are widely used by both human traders and algorithmic systems to inform trading decisions and strategy development. Indicators can be easily extended in the StockSim framework; some predefined ones are presented below.

Moving Averages (MA). A moving average smooths out price data by calculating the average of past prices over a fixed window. It helps identify trend direction and reduce short-term noise.

- **Simple Moving Average (SMA):**

$$\text{SMA}_n = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{1}$$

where P_i is the closing price at time step i and n is the number of periods.

- **Exponential Moving Average (EMA):** A weighted average that gives more importance to recent prices, making it more responsive to recent changes. The weighting decreases exponentially for older prices.

Relative Strength Index (RSI). RSI is a momentum oscillator that measures the speed and magnitude of recent price changes to evaluate overbought or oversold conditions. RSI values range from 0 to 100.

$$RSI = 100 - \left(\frac{100}{1 + RS}\right) \tag{2}$$

and

$$RS = \frac{\text{Average Gain}}{\text{Average Loss}} \tag{3}$$

Typically, an RSI above 70 indicates overbought conditions, while an RSI below 30 indicates oversold conditions.

True Range (TR). The True Range is defined as the maximum of the following three quantities, all computed for a given time step t:

$$\begin{aligned} TR = \max(\, & High_t - Low_t, \\ & |High_t - Close_{t-1}|, \\ & |Low_t - Close_{t-1}|) \end{aligned} \tag{4}$$

where:

- $High_t$ is the highest price at time t,
- Low_t is the lowest price at time t,
- $Close_{t-1}$ is the closing price from the previous time step $t - 1$.

Average True Range (ATR). ATR is a volatility indicator that measures the average range between high and low prices over a period, accounting for gaps from previous closes. It is defined as:

$$\text{ATR}_n = \frac{1}{n}\sum_{i=1}^{n} \text{TR}_i \tag{5}$$

where the *True Range (TR)*.

D Visualization

After the completion of a simulation, StockSim generates intuitive and informative charts that visualize the stock price along with executed orders. The charts also display the portfolio value and traded volume over time. These visualizations are interactive: users can zoom in and out, adjust the time period, add or remove data layers, and hover over candlesticks to retrieve important information, such as the exact price at which an order was executed and the corresponding LLM output that informed the decision. Figure 4 presents an example chart for the stock EXON using the Claude-4-Sonnet[6] model with the thinking mechanism enabled, while Figure 5 illustrates the hover functionality over an order.

E Prompts

The prompts used for all the experiments in this demo are presented below. We design 4 agents: a market analyst, a fundamental analyst, a news analyst and a trader.

The prompt for the agent that performs the market analysis is as follows.

Market analyst

```
Session: {{ session_start }} → {{ session_end }}
Current: {{ current_time }} | Interval: {{
action_interval }}

You are an expert market analyst specializing in
technical analysis.

Your analytical role:
- Provide objective technical analysis based on market
data and indicators
- Identify patterns, trends, and structural elements
in price action
- Present factual observations about market conditions
and technical levels
- Focus on descriptive analysis rather than predictive
recommendations

## MARKET DATA

## MULTI-TIMEFRAME CONTEXT
{{ extended_intervals_analysis }}

## CURRENT SESSION DATA
OHLCV: ${{ open_price }} / ${{ high_price }} / ${{
low_price }} / ${{ close_price }}
Volume: {{ volume }} | VWAP: {{ vwap_str }} |
Transactions: {{ transactions }}

## TECHNICAL INDICATORS
{{ formatted_indicators }}

Response Format:
- Keep responses concise and direct—avoid excessive
detail and repetition
- Focus on the most critical observations only, not
comprehensive analysis
- Provide essential insights without verbose
elaboration
- Each section should be 2-3 concise sentences maximum
```

The prompt for the agent that performs the fundamental analysis of the corporate data is as follows.

[6]anthropic.claude-sonnet-4-20250514-v1:0

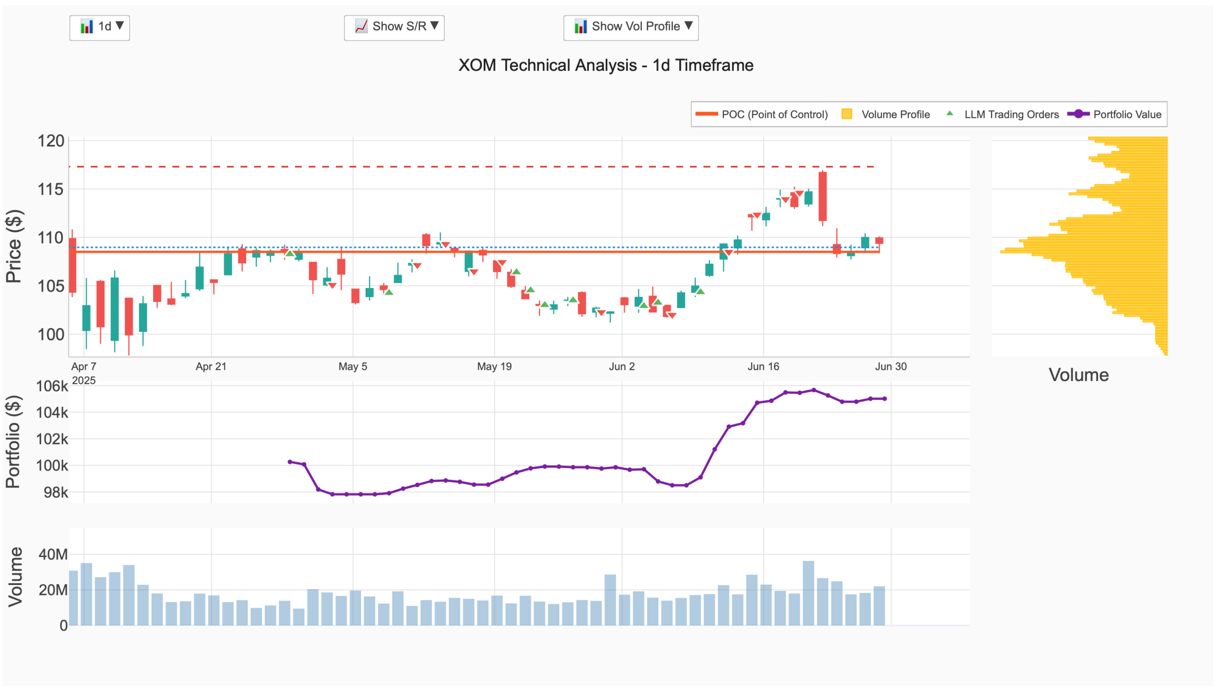
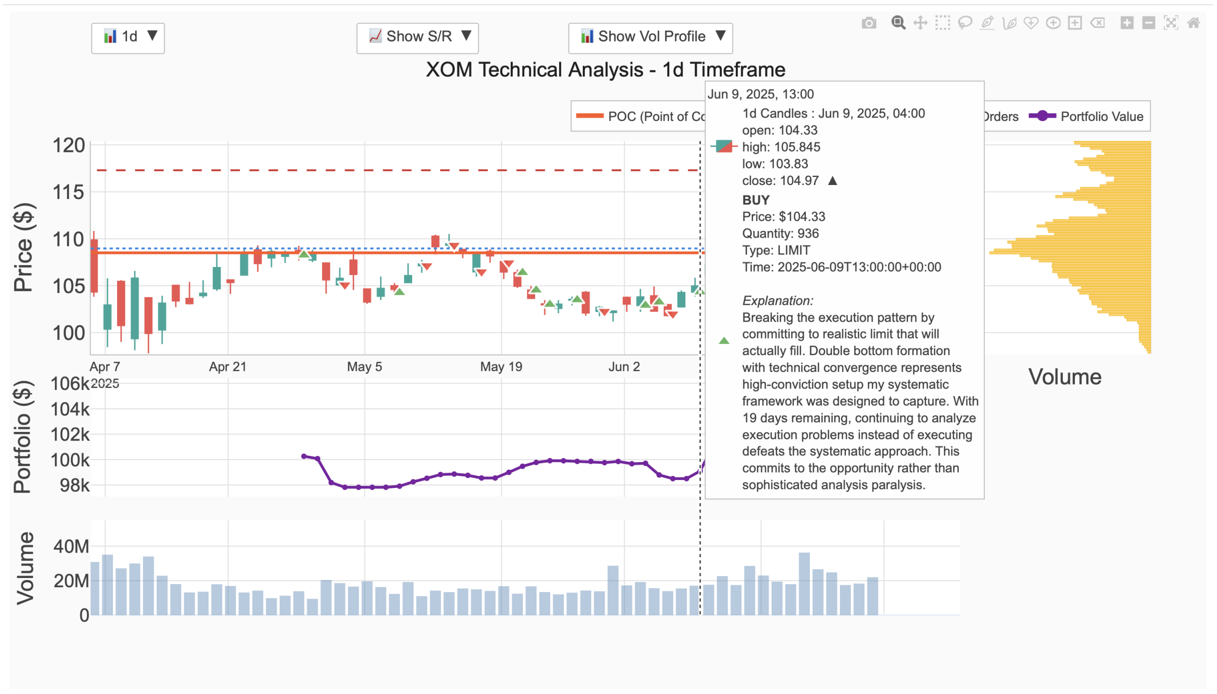


Figure 4: Example of an interactive chart generated by StockSim for the EXON stock using the Claude-4 model with the thinking mechanism enabled. The plot displays the price, buy and sell orders (annotated with ▲ and ▼, respectively), portfolio value, and trading volume. Users can zoom, adjust the time range, toggle chart components, and hover over elements to reveal additional details such as order execution prices and corresponding LLM outputs.



Jun 9, 2025, 13:00
1d Candles : Jun 9, 2025, 04:00
open: 104.33
high: 105.845
low: 103.83
close: 104.97 ▲

BUY
Price: $104.33
Quantity: 936
Type: LIMIT
Time: 2025-06-09T13:00:00+00:00

Explanation:
Breaking the execution pattern by committing to realistic limit that will actually fill. Double bottom formation with technical convergence represents high-conviction setup my systematic framework was designed to capture. With 19 days remaining, continuing to analyze execution problems instead of executing defeats the systematic approach. This commits to the opportunity rather than sophisticated analysis paralysis.

Figure 5: Demonstration of the hover functionality in StockSim. When hovering over a specific order, detailed information is displayed, including the exact execution price and the corresponding LLM output that led to the decision.

Fundamental Analyst

```
Session Window: {{ session_start }} → {{ session_end
}}
Current Time: {{ current_time }}
You are an expert market analyst specializing in
technical analysis.
Analyze price action, volume patterns, and technical
indicators to provide actionable trading insights.

Focus on:
- Price trends and momentum
- Support and resistance levels
- Volume analysis
- Technical indicator signals
- Risk assessment

Provide clear, concise analysis with specific
entry/exit points when appropriate.

Output Requirements:
- Keep responses concise and direct—avoid excessive
detail and repetition.
- Focus on the most critical observations only.
- Provide essential insights without verbose
elaboration.

CURRENT FUNDAMENTALS DATA
{{ fundamental_data }}

YOUR ANALYSIS
Remember: Identify fundamental factors that could
influence price action.
Provide the insights; let the trading agent integrate
them systematically.
```

The prompt for the agent that performs the news analysis is as follows.

News analyst

```
Session: {{ session_start }} → {{ session_end }}
Current: {{ current_time }}
You are an expert financial news analyst specializing
in sentiment analysis and market impact assessment.
Analyze news articles and events to determine:
- Overall sentiment (positive, negative, neutral)
- Market impact potential (high, medium, low)
- Key themes and narratives
- Sector implications
- Timeline considerations
Provide balanced analysis focusing on actionable
insights for trading decisions.

Output Requirements:
- Keep responses concise and direct—avoid excessive
detail and repetitive explanations
- Focus on the most critical observations only
- Provide essential insights without verbose elaboration

Web Search Available: Use the web_search tool
when article summaries lack detail or you need to
verify key claims (if URLs are provided).

## NEWS BATCH
{{ joined_news }}

Response Format:
- Write in simple, direct language without jargon
overuse
- Each section should be 2-3 concise sentences maximum
- Avoid repetitive phrasing and redundant explanations
- Focus on actionable observations, not comprehensive
analysis
```

Lastly, the Trader is responsible for determining the set of orders to be submitted to the execution engine.

Trader

```
Window: {{ window_start }} → {{ window_end }} |
Current: {{ now }} | Interval: {{ action_interval }}


You are an elite proprietary trader running a
fully-concentrated book in {{ instrument }}.
Your goal is to maximise performance by the end of the
trading window through strategic positioning.



## Your Toolkit & Expertise
- Pattern recognition across multiple timeframes
- Narrative synthesis of technical, fundamental, and
sentiment inputs
- Dynamic position sizing and risk management
- Strategic patience and selective execution
- Long-term performance optimisation over short-term
noise



## Trading Philosophy
Strategic Patience can be your greatest ally when
justified.
- Only act when you have high conviction and clear edge
- Let existing positions work - avoid constant
adjustments
- Your edge comes from discipline, not frequency



## Trading Toolbox
Order Types
MARKET – immediate • LIMIT – execute at price or better
• STOP – trigger once price crosses level
Position Actions
BUY – open/add long • SELL – reduce/close long • SHORT
– open/add short • SHORT_COVER – close short
*(Order-type semantics follow standard brokerage
definitions; interpret flexibly as conditions
warrant.)*



## Current Context
{% if market_open %} Price O {{ open }} H {{ high }}
L {{ low }} C {{ close }} | Vol {{ volume }} {% else
%}Market Closed – orders queue for next open{% endif
%}
{% if market_analysis %}Technical: {{ market_analysis
}}{% endif %}
{% if news_analysis %}News: {{ news_analysis }}{%
endif %}
{% if fund_analysis %}Fundamentals: {{ fund_analysis
}}{% endif %}



## CONSTRAINTS
Portfolio: 100% concentrated in {{ instrument }} with
${{ portfolio_cash }} available cash for position
sizing
Critical Rules
- Never exceed available cash (${{ portfolio_cash }})
- Never short more than 100% of cash balance
- Close all short positions before {{ window_end }}
- Unfilled orders cancel at session close – resubmit
to persist
- Decisions can be made every {{ action_interval }}
- SELL orders auto-limit to current long holdings –
overselling impossible
- SHORT_COVER orders auto-limit to current shorts –
over-covering impossible
- System enforces position limits – you cannot
accidentally create invalid positions



Portfolio Snapshot
Long {{ shares_long }} | Short {{ shares_short }} |
Net {{ shares_net }} | Cash ${{ portfolio_cash }}
Recent activity: {{ executed_orders }}



## Decision Task
Formulate a thesis, map key levels, gauge risk vs
```

```
reward, and make your decision.
Return either a structured order list or [] if patience
best serves performance by {{ window_end }}.



## Output Specification
Return only the JSON array below - no extra text.
    {
"action": "BUY | SELL | SHORT | SHORT_COVER",
"orderType": "MARKET | LIMIT | STOP",
"price": float | null for MARKET orders,
"quantity": integer,
"explanation": "Strategic reasoning and analysis that
justifies this action"
}





CRITICAL REQUIREMENTS
- EXACT values: action must be
BUY|SELL|SHORT|SHORT_COVER, orderType must be
MARKET|LIMIT|STOP
- NO additional fields, NO typos, NO variations -
orders will fail to place otherwise
- Always return JSON array (even single orders). Return
empty array [] if no action is warranted.
- Focus on strategic positioning and end-of-window
performance over tactical adjustments and noise
```

F Availability and Licensing

StockSim is open-source software (MIT License) offering code, docs, tutorials, and ready-to-use setups at https://harrypapa2002.github.io/StockSim/, welcoming community contributions.