QFinZero: A Unified Financial Toolchain for LLM-Based Trading Agents

Haochen Luo[1,*], Yifan Li[1,*], Ho Tin Ko[2,*], Binh Minh An[1], Junjie Xu[4], Pok Hin Tang[1], Wang Chak Wong[1], Yuan Gao[1], Zhengzhao Lai[5], Yuan Zhang[3], Chen Liu[1,‡]

[1]City University of Hong Kong, [2]Yuen Long Merchants Association Secondary School,
[3]Shanghai University of Finance and Economics,
[4]University of Science and Technology of China,
[5]The Chinese University of Hong Kong (Shenzhen)

Open-source repository: https://github.com/CityU-MLO/qfinzero

Abstract

Large language model (LLM) agents are increasingly applied to financial decision-making tasks that require interaction with external tools, including market data retrieval, news analysis, and trade execution. However, existing trading systems rely on fragmented and task-specific APIs, resulting in inconsistent schemas, complex integration, and limited reproducibility. We present **QFinZero**, a unified trading environment for LLM-based financial agents. QFinZero standardizes three core capabilities: (i) multi-frequency market and derivatives data access, (ii) structured news and event retrieval, and (iii) stateful brokerage simulation with explicit order lifecycle management. All tools adopt consistent JSON schemas and time-aligned interfaces, enabling agents to acquire information and execute trades within a coherent framework. By abstracting financial interaction into composable, agent-invokable primitives, QFinZero reduces engineering overhead and supports reproducible evaluation through comprehensive logging and deterministic replay. We argue that such a standardized trading environment is essential for scalable research on LLM-based financial agents.

1 Introduction

Large language model (LLM) agents are increasingly prevalent in the financial domain, driving advancements in financial text analysis and reasoning (Chen et al., 2021; Xie et al., 2023; Reddy et al., 2024), trading signal mining (Luo et al., 2026; Shi et al., 2025), and automated trading (Xiao et al., 2024; Yu et al., 2025; Xiong et al., 2025; Li et al., 2025b; YANG et al., 2025; Li et al., 2023; Fan et al., 2025; Li et al., 2025a). Given the abundance of benchmarks and architectural designs, trading agents have emerged as a particularly active area of research. Table 1 summarizes the scope of recent trading agent benchmarks. The standard operational paradigm for these agents typically follows a three-step sequence: collecting market information (e.g., prices and news), reasoning over this data, and subsequently generating an action (e.g., executing a price order).

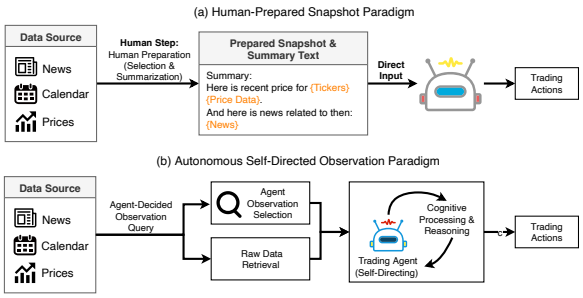

Figure 1: Human-prepared snapshot paradigm vs. autonomous self-directed observation.

While the reasoning capabilities of underlying models have rapidly improved, the reliable evaluation of trading agents remains an open challenge. One major concern is potential data contamination in the training corpora of LLMs (Dekoninck et al., 2024; Balloccu et al., 2024; Ravaut et al., 2024). To address this, Li et al. (2025a) proposed a "live benchmark" approach, employing continuous testing to simulate live markets for fairer evaluation. A second significant challenge is the reliance on oversimplified testing environments. Most existing benchmarks provide agents with pre-constructed market snapshots that passively summarize prices, news, and account states. However, real-world trading systems require agents to actively retrieve heterogeneous data, align timestamps across disparate sources, and manage evolving portfolio states. The critical decisions of *what* to query and *when* to query are frequently abstracted away. Figure 1 contrasts the human-prepared snapshot paradigm with autonomous, self-directed observation. Recently, Fan et al. (2025) introduced a novel benchmark

[*]These authors contributed equally to this work. Middle authors are ordered alphabetically by first name.

[‡]Correspondence: chen.liu@cityu.edu.hk

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 68–77

Table 1: Comparison of existing LLM-based trading agents and benchmarks. **Human Feed** indicates whether market data and news are pre-curated and provided to the agent by the system (✓) or autonomously retrieved by the agent itself (×).

Method	Market	Frequency	Modality	Data Scale	Human Feed
AI-Trader (Fan et al., 2025)	US, CN, Crypto	Daily / Hourly	Text, Price	NASDAQ 100 + SSE 50 + 10 crypto; live	×
TradingAgents (Xiao et al., 2024)	US	Daily	Text, Price	5 stocks (AAPL, etc.); ∼3 mo	✓
InvestorBench (Li et al., 2025b)	US, Crypto	Daily	Text, Price	7 stocks + 2 crypto + ETF; 10mo	✓
QuantAgent (Xiong et al., 2025)	US, Crypto, Comm.	1h / 4h	Price, Image	8 instruments; 5K bars/asset; ≤10yr	✓
FinMem (Yu et al., 2025)	US	Daily	Text, Price	Individual stocks (TSLA, etc.)	✓
DeepFund (Li et al., 2025a)	US (+sectors)	Daily	Text, Price	5 stocks + sector exts.; live Q1–Q2 '25	✓
TwinMarket (YANG et al., 2025)	CN	Daily	Text, Price	SSE50 (10 indices); 5mo; 1M+ news	✓

forcing agents to autonomously collect trading information, which highlighted ongoing deficiencies in agent tool-calling accuracy and efficiency.

Consequently, these limitations expose several significant gaps between current trading benchmarks and real-world market operations:

Limited Data Scope and Frequency: Most frameworks restrict evaluation to a small subset of top-tier equities and fail to support derivatives such as options. Furthermore, they are constrained by low-frequency trading scenarios; existing frameworks generally lack support for minute-level data ingestion, precluding the evaluation of intraday trading.

Oversimplified Execution Layers: Current benchmarks largely restrict action spaces to a simplistic buy/sell/hold triad. They ignore crucial mechanics like margin requirements and complex operations such as short selling or options execution. Moreover, they often assume instantaneous order execution, neglecting conditional or limit orders, creating a significant divergence from actual trading.

Fragmented Financial Infrastructure: Real market data, news feeds, and brokerage APIs inherently possess heterogeneous schemas and inconsistent update frequencies. Currently, researchers and developers must build integration tools to connect with varying data providers. A unified interface bridging multiple data sources is urgently needed.

Recent advances in LLM agents have enabled systems capable of autonomously interacting with external tools and APIs (Patil et al., 2024; Cai et al., 2024; Yao et al., 2022; Schick et al., 2023). The primary advantage of tool utilization is that it grants agents the ability to dynamically access real-time information and execute deterministic actions, bypassing the static constraints and hallucination risks of their pre-trained weights. A systematic toolchain is also essential for multi-agent collaboration scenarios (Ehtesham et al., 2025; Tran et al., 2025; Shi et al., 2024). In the domain of trading agents, rigorous tool integration is an especially important technique for both agent design and information exchange. The reason is twofold: financial environments require precise temporal alignment to prevent look-ahead bias, and agents must adhere strictly to complex API schemas to execute trades safely. Fortunately, modern agent design frameworks, such as LangGraph, have simplified the development process, offering built-in extensions to support robust tool calling without the need for complex foundational coding.

To address the aforementioned gaps in trading agent design and benchmarking, and motivated by recent advancements in agent tool utilization, we introduce **QFinZero**: a unified trading environment formulated as a reusable *agent skill*. QFinZero standardizes agent interaction with multi-frequency price data, structured news and event streams, and stateful brokerage simulations through consistent, time-aligned APIs. Instead of passively receiving static summaries, agents operating within QFinZero actively query information, coordinate tools, and execute realistic orders within a coherent environment. By decoupling agent reasoning from the complexities of financial data engineering, QFinZero enables reproducible evaluation under both historical replay and simulated streaming settings. This unified interaction layer provides a stable foundation for studying tool-use behavior, temporal grounding, and multi-step reasoning in financial LLM agents, while facilitating more systematic trading agent development and more reliable benchmark construction. Crucially, the same properties that enable reproducible benchmarking: deterministic replay, time-aligned state, and verifiable per-step signals, also turn QFinZero into a ready-to-use training environment for agentic reinforcement learning.

2 Data Pipeline and Storage

QFinZero provides a series of local data storage solutions to help users efficiently manage their fi-

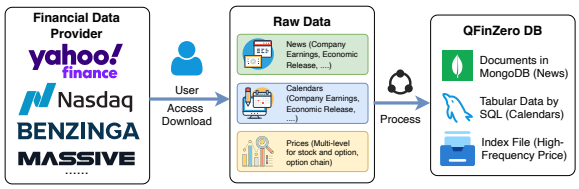

Figure 2: Financial data providers supply raw market data including news, earnings calendars, economic releases, and multi-level price data. Users access and download the data, which is then processed and organized into the QFinZero database

nancial data assets. Figure 2 illustrates the process by which users integrate raw financial data into our system. QFinZero adopts a standardized and unified management framework to aggregate and harmonize data from multiple sources, enabling consistent storage, efficient querying, and interaction between the downstream agents.

2.1 Data Source

Our system is originally designed to integrate with several commercial data providers. Specifically, we use the NASDAQ API[1] for economic calendars and earnings calendars, MASSIVE Inc[2]. for price data, and BENZINGA[3] for financial news, earnings calendars, and earnings guidance data. Although the default configuration relies on these providers, we design a unified data format and standardized ingestion pipeline. This allows users to import data from alternative sources into our storage system with minimal adaptation, ensuring flexibility and extensibility of the framework.

2.2 Local Storage

We adopt different storage solutions based on data characteristics: (1) **High-volume news data**, we deploy MongoDB to efficiently handle semi-structured documents. To reduce database load and improve query efficiency, we store only essential fields, including title, related tickers, release time, source URL, and publisher information. When available, we also include optional fields such as significance ratings that indicate the potential impact of the news on specific tickers. (2) **Calendar data** (earnings and economic events), which are naturally organized in tabular format, we use a relational database (SQLite in our demonstration). We construct multi-column indexes (e.g., date, country, ticker) to support efficient composite queries.

[1] https://www.nasdaq.com/solutions/data
[2] https://massive.com/
[3] https://www.benzinga.com/apis/

(3) **Price data**: we organize the dataset into two time scales: minute-level and daily-level. Considering the large query demand and the append-only update pattern of price data, we choose Parquet format and store these data in a partitioned disk-based file system with indexed access (e.g., partitioned by symbol and year). This design reduces database overhead and improves sequential reading performance for backtesting and replay-based evaluation. We provide details of the data schema in Appendix A.1.

2.3 Data Maintenance

We provide a maintenance tool to synchronize and update the local storage from the original data sources. The dataset is designed to support incremental updates and automated data ingestion. The update frequency depends on the data provider. For example, MASSIVE stock and option price data typically have a delay of approximately 15 minutes, while news and calendar data are updated daily. Although not designed for ultra-low-latency trading, this update frequency is sufficient for strategy backtesting, benchmark construction, and agent evaluation in our framework.

3 The QFinZero System

QFinZero is a unified and agent-oriented financial toolchain designed to support the construction, evaluation, and deployment of intelligent trading agents. Instead of exposing fragmented data sources and heterogeneous brokerage APIs, QFinZero organizes the financial workflow into three core system components:**(1) Unified Price Query (UPQ)**, **(2) Event Stream Pipeline (ESP)**, and **(3) Paper Money Broker (PMB)**. These components correspond to the fundamental information and action flows of a trading agent: market state observation (prices), event awareness (news and calendars), and execution interaction (order placement and updates). Each module is implemented as an independent, callable service with standardized interfaces. This modular design enables flexible composition across backtesting, simulation, and live-like environments, while minimizing data-source coupling and implementation overhead.

3.1 Core Components

Unified Price Query (UPQ). Trading agents require consistent and synchronized access to price signals across different temporal resolutions and asset classes. However, real-world financial data

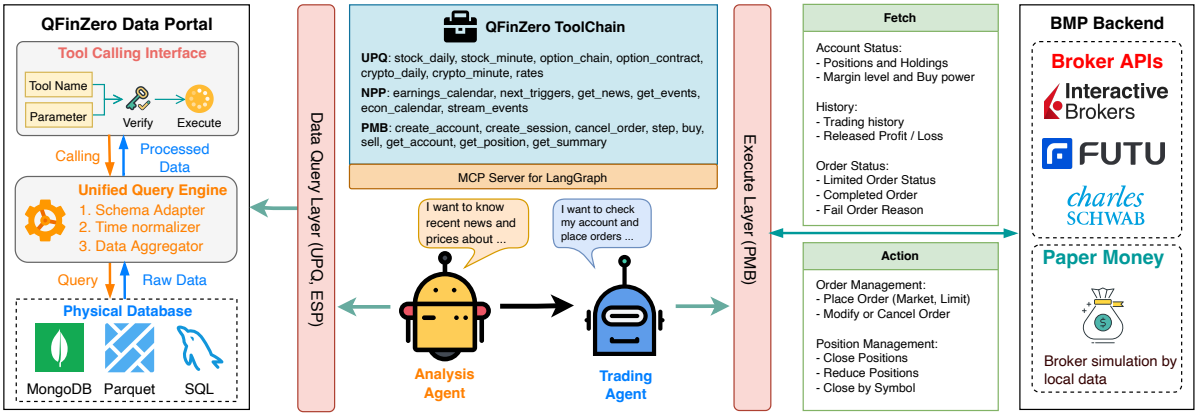

Figure 3: QFinZero integrates data access (UPQ, ESP) and brokerage execution (PMB) into a unified, agent-callable toolchain. The Data Query Layer standardizes access to physical storage (MongoDB, Parquet, SQL) through schema adaptation, time normalization, and data aggregation. The Execute Layer connects to broker APIs or trading simulator for order management and account operations. All Agents interact with the MCP-based toolchain via structured tool calls, enabling seamless retrieval of market data, event streams, and portfolio states, as well as execution of trading actions within a consistent and time-aligned environment.

is typically fragmented by frequency (e.g., daily vs. intraday) and instrument type (e.g., equities vs. options). The UPQ module provides a unified price abstraction layer that supports multi-resolution data (high-frequency, mid-frequency, and daily) and multiple asset types. All price streams are exposed through a standardized API interface, allowing agents to query or subscribe to structured market states without handling vendor-specific formatting or synchronization logic. This design enables seamless transitions between historical backtesting and simulated real-time trading.

Event Stream Pipeline (ESP). Beyond price signals, trading decisions are strongly influenced by both real-time information and scheduled market events. We organize event information along multiple dimensions:
(1) *Temporal scope*: including ongoing events, recently released information, and future scheduled events (e.g., earnings announcements or macroeconomic releases);
(2) *Market scope*: including global macro events, asset-specific corporate news, and derivative-related activities;
(3) *Information type*: including financial news, earnings reports and guidance, macroeconomic calendars, analyst actions, and option-related signals.

The ESP module aggregates heterogeneous event streams into a unified and time-aligned schema. All events are normalized and indexed with temporal, ticker-level, and event-type tags, allowing agents to retrieve relevant information through configurable filters and structured queries. This design enables event-driven reasoning, cross-source information aggregation, and forward-looking decision processes aligned with anticipated market catalysts.

Paper Money Broker (PMB). To close the loop between perception and action, QFinZero provides a simulated brokerage layer that supports realistic trading interactions. PMB maintains order states, portfolio positions, account balances, and transaction logs under configurable execution rules. It supports common order types and margin-aware account dynamics, enabling more realistic evaluation compared to simplified buy/hold/sell abstractions. This component allows agents to operate in a controlled yet market-consistent environment for benchmarking and analysis.

3.2 Agent-Friendly Design

A central design principle of QFinZero is *agent friendliness*. Instead of treating LLM agents as passive consumers of pre-processed datasets, QFinZero exposes structured, queryable, and callable interfaces that enable agents to actively request information, reason over results, and execute actions within a unified interaction loop. Concretely, we adopts the following design strategies:
Unified schemas. All modules return structured JSON outputs with consistent field names and explicit timestamps, reducing parsing ambiguity and simplifying tool integration.
Composable APIs. Each pipeline is independently callable, enabling agents to flexibly query data or execute trades within multi-step reasoning loops.

Time-consistent state abstraction. All data is aligned to precise timestamps, ensuring coherent price, event, and portfolio states for reproducible evaluation. The system supports minute-level market data alignment, facilitating more fine-grained intraday analysis and higher-precision backtesting.
Environment decoupling. The same API layer supports historical replay, simulated streaming, and live-like settings without changing agent logic, enabling fair comparison across agent designs.
MCP Interface. QFinZero provides an MCP (Model Context Protocol) (Hou et al., 2025; Ray, 2025) compatible server interface for seamless integration with modern tool-use frameworks and agent orchestration protocols, transforming fragmented financial infrastructure into a coherent and extensible toolchain for LLM-based agents.

3.3 Coverage and Scalability

QFinZero supports both cryptos, stock and option trading under a realistic simulation environment. Through its unified data layer interface, the system can be easily extended to additional asset classes, including equities from different global markets, provided that compatible data sources are available. This design ensures broad market coverage without requiring structural changes to the agent interaction protocol. In addition, QFinZero is built on a scalable database architecture that allows users to continuously update and expand the internal knowledge base. New price data, news streams, economic events, or alternative datasets can be integrated into the storage layer without modifying the upper-level agent interface. This modular and extensible design enables QFinZero to scale with increasing data volume and market diversity, making it suitable for both research experimentation and larger-scale deployment scenarios.

4 Evaluation

4.1 Task Setup

We evaluate QFinZero through an instruction-driven benchmark that focuses on *tool correctness* and *execution consistency*, rather than financial return. The goal is to measure whether an agent can (1) select the correct tool, (2) construct valid API calls with precise parameters, and (3) maintain temporally coherent multi-step execution. We design two complementary task types:
Tool Calling. Single-step tasks where each instruction corresponds to one expected API call with explicit parameter and constraints.
Task Planning. Multi-step workflows requiring cross-pipeline coordination (e.g., querying prices, retrieving events, and placing orders).

Performance for above tasks are measured by: Tool Match Accuracy (TM), Parameter Accuracy (PA), and Time Alignment (TA), detailed metric definitions are provided in Appendix A.3.

4.2 Results

Tables 2 report representative results. In the tool calling benchmark, frontier models achieve high tool selection accuracy (TM $\approx$ 0.95+), while parameter grounding and temporal alignment remain the primary error sources. Smaller open-weight models show larger drops in PA, indicating that argument construction is the main bottleneck.

In multi-step planning, performance gaps narrow among strong models, suggesting that once tool interfaces are correctly understood, step-level coordination is relatively stable. However, parameter precision continues to dominate failure cases. The evaluation demonstrates that QFinZero provides a stable and discriminative environment for diagnosing tool-use behavior, temporal grounding, and structured execution reliability in financial agents.

5 Case Study

As QFinZero is designed for agents interacting with financial environments, Figure 4 presents representative interaction traces within the system, illustrating both single-step tool invocation and multi-step planning scenarios. We additionally provide a dashboard that enables human users to visualize the status of the local database, as well as a playground interface for testing agent interactions with QFinZero. Detailed descriptions are provided in Appendix A.4.
Single-Step Tool Calling. In atomic tasks such as *Market Order* or *Stop-Limit Order*, the agent receives a natural language instruction and must (1) select the appropriate module (e.g., `PMB.order.place`), (2) construct a valid API request, and (3) populate all required parameters under correct schema and time constraints. For example, in the *Market Order* case, the agent correctly maps the instruction to a `PMB.order.place` call with action `BUY MARKET`, quantity specification, and time-in-force. In the *Dual Calendar Query* case, the agent coordinates two calendar endpoints (`ESP.calendar.earnings`

Model	Tool Calling Evaluation				Task Planning Evaluation			
	TM	PA	TA	Overall	TM	PA	TA	Overall
Gemini 2.5 Flash	0.97	0.90	0.86	91.00	0.97	0.90	0.90	92.33
Gemini 2.5 Pro	0.97	0.90	0.86	91.00	0.97	0.91	0.93	93.67
GPT-4.1	0.92	0.86	0.84	87.33	0.96	0.93	0.92	93.67
GPT-4.1 Mini	0.90	0.83	0.80	84.33	0.97	0.90	0.89	92.00
DeepSeek Chat	0.96	0.85	0.83	88.00	0.97	0.93	0.95	95.00
Qwen2.5 7B Instruct	0.91	0.68	0.76	78.33	0.91	0.84	0.88	87.67
Llama 3.1 8B Instruct	0.65	0.55	0.67	62.33	0.90	0.78	0.86	84.67
Qwen3 4B Instruct	0.93	0.84	0.82	86.33	0.91	0.86	0.82	86.33

Table 2: Comparison of Tool Calling and Task Planning Evaluation Results. TM, PA, and TA refer to specific metric categories; Overall score is the arithmetic mean scaled to [0,100].

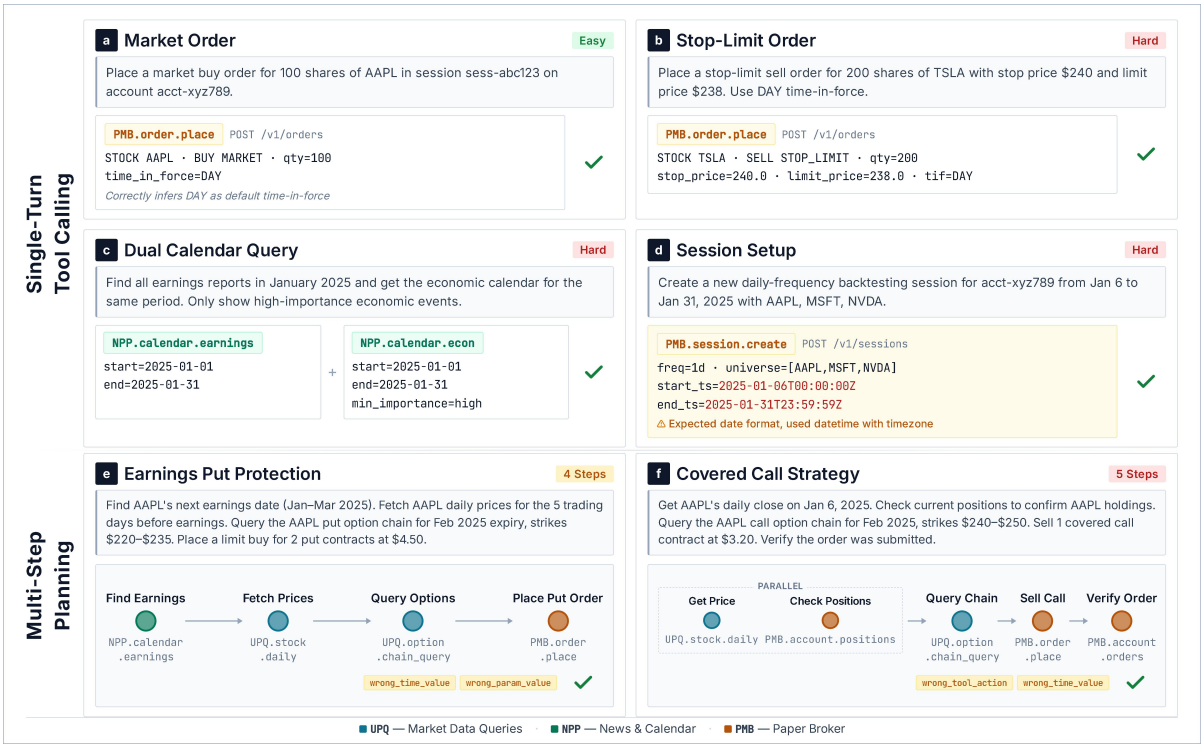

Figure 4: Illustration of the QFinZero framework across single-turn tool calling and multi-step planning tasks.

and `ESP.calendar.econ`) with aligned temporal ranges. These examples demonstrate that QFinZero enforces structured tool grounding and temporal consistency even for seemingly simple instructions.

Multi-Step Planning. More complex tasks require the agent to compose multiple tools into a coherent execution chain. In the *Earnings Put Protection* scenario, the agent sequentially: (1) Retrieves the earnings date via `ESP.calendar.earnings`, (2) Fetches recent daily prices using `UPQ.stock.daily`, (3) Queries the option chain with `UPQ.option.chain_query`, (4) Submits a limit order through `PMB.order.place`. These workflows highlight the system's ability to maintain cross-step temporal alignment, parameter consistency, and order state tracking.

6 Infrastructure for Live Benchmark and Agentic RL

The same primitives that enable reproducible evaluation also make QFinZero a suitable infrastructure layer for two emerging research directions: live benchmarking of financial agents, and agentic reinforcement learning (agentic RL) of tool-using LLM policies. Static financial benchmarks increasingly suffer from training-data contamination (Dekoninck et al., 2024; Balloccu et al., 2024), motivating continuously updated, time-anchored evaluation (Li et al., 2025a; Fan et al., 2025; Kong et al., 2026). QFinZero directly supports this paradigm: the data ingestion pipeline (Section 2) can be configured to roll forward at any cadence, while the time-aligned data layer enforces an explicit `as_of_ts` cursor

so that every evaluation episode is anchored to a single point-in-time view of prices, news, and events. Deterministic replay then allows the same live-released window to be re-run across models and prompt designs with bit-identical environment state, removing a major source of variance in current live-benchmark practice.

Real-world trading imposes stringent reliability requirements on tool calls: a malformed order parameter or misaligned timestamp translates directly into financial loss. As Section 4, off-the-shelf LLMs still fail substantially on parameter grounding and temporal alignment under realistic financial schemas, motivating reinforcement-learning fine-tuning as a natural path toward production-grade trading agents. QFinZero is explicitly designed as the environment for this purpose: it provides the temporal causality, verifiable multi-granularity rewards, and parallel reproducible rollouts required by recent agentic-RL methods such as ReTool (Feng et al., 2025), Search-R1 (Jin et al., 2025), and GRPO-style tool-use training (Jiang et al., 2025; Qian et al., 2026).

7 Conclusion and Future Work

We present QFinZero, a unified and agent-oriented financial toolchain that integrates price delivery, event streaming, and stateful brokerage simulation into a coherent, callable framework. By standardizing multi-frequency market access, structured event retrieval, and realistic order lifecycle management, QFinZero aligns financial infrastructure with the operational workflow of modern LLM-based trading agents. The system decouples agent reasoning from data engineering complexity, enabling reproducible benchmarking under both historical replay and simulated streaming settings. QFinZero is released as an open-source agent skill intended for research and system development in financial NLP and tool-using agents or benchmark. While the framework relies on commercial data providers and therefore cannot redistribute raw datasets due to licensing restrictions, its unified schema and ingestion pipeline allow integration with alternative data sources. Additionally, although the current implementation supports minute-level data, it is designed primarily for research evaluation rather than ultra-low-latency production trading. Future work will focus on improving real-time streaming integration, expanding data accessibility, and supporting broader multi-agent trading scenarios.

References

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondřej Dušek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source llms. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. 2024. Large language models as tool makers. In *The Twelfth International Conference on Learning Representations*.

Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, and 1 others. 2021. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3697–3711.

Jasper Dekoninck, Mark Müller, and Martin Vechev. 2024. Constat: Performance-based contamination detection in large language models. *Advances in Neural Information Processing Systems*, 37:92420–92464.

Abul Ehtesham, Aditi Singh, Gaurav Kumar Gupta, and Saket Kumar. 2025. A survey of agent interoperability protocols: Model context protocol (mcp), agent communication protocol (acp), agent-to-agent protocol (a2a), and agent network protocol (anp). *arXiv preprint arXiv:2505.02279*.

Tianyu Fan, Yuhao Yang, Yangqin Jiang, Yifei Zhang, Yuxuan Chen, and Chao Huang. 2025. Ai-trader: Benchmarking autonomous agents in real-time financial markets. *Preprint*, arXiv:2512.10971.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Xinyi Hou, Yanjie Zhao, Shenao Wang, and Haoyu Wang. 2025. Model context protocol (mcp): Landscape, security threats, and future research directions. *ACM Transactions on Software Engineering and Methodology*.

Dongfu Jiang, Yi Lu, Zhuofeng Li, Zhiheng Lyu, Ping Nie, Haozhe Wang, Alex Su, Hui Chen, Kai Zou, Chao Du, and 1 others. 2025. Verltool: Towards holistic agentic reinforcement learning with tool use. *arXiv preprint arXiv:2509.01055*.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Yaxuan Kong, Hoyoung Lee, Yoontae Hwang, Alejandro Lopez-Lira, Bradford Levy, Dhagash Mehta, Qingsong Wen, Chanyeol Choi, Yongjae Lee, and Stefan Zohren. 2026. Evaluating llms in finance requires explicit bias consideration. *arXiv preprint arXiv:2602.14233*.

Changlun Li, Yao SHI, Chen Wang, Qiqi Duan, Runke RUAN, Weijie Huang, Haonan Long, Lijun Huang, Nan Tang, and Yuyu Luo. 2025a. Time travel is cheating: Going live with deepfund for real-time fund investment benchmarking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Haohang Li, Yupeng Cao, Yangyang Yu, Shashidhar Reddy Javaji, Zhiyang Deng, Yueru He, Yuechen Jiang, Zining Zhu, Kp Subbalakshmi, Jimin Huang, and 1 others. 2025b. Investorbench: A benchmark for financial decision-making tasks with llm-based agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2509–2525.

Yang Li, Yangyang Yu, Haohang Li, Zhi Chen, and Khaldoun Khashanah. 2023. Tradinggpt: Multi-agent system with layered memory and distinct characters for enhanced financial trading performance. *arXiv preprint arXiv:2309.03736*.

Haochen Luo, Ho Tin Ko, Jiandong Chen, David Sun, Yuan Zhang, and Chen Liu. 2026. Alphabench: Benchmarking large language models in formulaic alpha factor mining. In *International Conference on Learning Representations*.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. 2026. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553.

Mathieu Ravaut, Bosheng Ding, Fangkai Jiao, Hailin Chen, Xingxuan Li, Ruochen Zhao, Chengwei Qin, Caiming Xiong, and Shafiq Joty. 2024. A comprehensive survey of contamination detection methods in large language models. *arXiv preprint arXiv:2404.00699*.

Partha Pratim Ray. 2025. A survey on model context protocol: Architecture, state-of-the-art, challenges and future directions. *Authorea Preprints*.

Varshini Reddy, Rik Koncel-Kedziorski, Viet Dac Lai, Michael Krumdick, Charles Lovering, and Chris Tanner. 2024. Docfinqa: A long-context financial reasoning dataset. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 445–458.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yu Shi, Yitong Duan, and Jian Li. 2025. Navigating the alpha jungle: an llm-powered mcts framework for formulaic factor mining. *arXiv preprint arXiv:2505.11122*.

Zhengliang Shi, Shen Gao, Xiuyi Chen, Yue Feng, Lingyong Yan, Haibo Shi, Dawei Yin, Pengjie Ren, Suzan Verberne, and Zhaochun Ren. 2024. Learning to use tools via cooperative and interactive agents. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10642–10657, Miami, Florida, USA. Association for Computational Linguistics.

Khanh-Tung Tran, Dung Dao, Minh-Duong Nguyen, Quoc-Viet Pham, Barry O'Sullivan, and Hoang D Nguyen. 2025. Multi-agent collaboration mechanisms: A survey of llms. *arXiv preprint arXiv:2501.06322*.

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

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A comprehensive benchmark, instruction dataset and large language model for finance. *Advances in Neural Information Processing Systems*, 36:33469–33484.

Fei Xiong, Xiang Zhang, Aosong Feng, Siqi Sun, and Chenyu You. 2025. Quantagent: Price-driven multi-agent llms for high-frequency trading. *arXiv preprint arXiv:2509.09995*.

Yuzhe YANG, Yifei Zhang, Minghao Wu, Kaidi Zhang, Yunmiao Zhang, Honghai Yu, Yan Hu, and Benyou Wang. 2025. Twinmarket: A scalable behavioral and social simulation for financial markets. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Yangyang Yu, Haohang Li, Zhi Chen, Yuechen Jiang, Yang Li, Jordan W Suchow, Denghui Zhang, and Khaldoun Khashanah. 2025. Finmem: A performance-enhanced llm trading agent with layered memory and character design. *IEEE Transactions on Big Data*.

A Additional Details

A.1 Details in Data Schema

News. News articles are stored as JSON documents. The schema is summarized in Table 3.

Calendar Calendar events are stored in tabular format and normalized into a unified schema. Two major event types are supported: earnings and macroeconomic events. The common structure is summarized in Table 4. For earnings events, the `payload` includes structured financial metrics such as actual and estimated EPS, surprise ratios, and guidance information. For macroeconomic events, the `payload` contains reported values (actual, previous, consensus) together with detailed descriptions.

Field	Type	Description
id	string	Unique identifier of the article.
title	string	Title of the news article.
article_url	string	Link to the original article.
author	string	Author of the article.
description	string (opt.)	Short summary of the article.
image_url	string (opt.)	Image associated with the article.
keywords	array(string) (opt.)	Publisher-provided keywords.
tickers	array(string)	Related ticker symbols.
published_utc	string	Publication time in RFC3339 format (UTC).
publisher	object	Publisher metadata (name, logo, homepage).
insights	array(object) (opt.)	Structured insights related to the article.

Table 3: News data schema, opt. denotes optional fields.

Price. All price data are provided in bar aggregate format at both minute and daily frequencies. Each record contains open, high, low, close (OHLC), and volume fields. The unified structure allows agents to seamlessly switch between daily-level backtesting and minute-level streaming evaluation.

Option Chain. Option data are organized by underlying ticker and expiration date. Each option record contains two components: (1) `Contract properties:` expiry date, strike price, option type (C/P), option ticker symbol, and underlying asset; (2) `Price aggregates:` bar-level OHLCV data, consistent with the stock price schema.

Field	Type	Description
event_id	string	Unique identifier of the event.
event_type	string	Event category (e.g., earnings, macro_calendar).
title	string	Event title.
time_utc	string	Event timestamp in UTC (RFC3339).
importance	string	Importance level (e.g., high, medium).
status	string	Event status (scheduled or occurred).
tickers	array(string)	Related ticker symbols (if applicable).
country	string	Country of relevance.
snippet	string	Short summary of the event outcome.
payload	object	Structured event-specific information.

Table 4: Unified calendar event schema.

A.2 Generation of Test Instruction

To construct the instruction-driven benchmark in Section 4, we prompt a frontier LLM to role-play as a financial trader and emit natural-language requests that exercise the QFinZero toolchain. The prompt template specifies the trader persona, the available tool families (UPQ, ESP, PMB), and the target task structure (single-step tool calling or multi-step planning), together with constraints on temporal scope, asset universe, and parameter realism. Each generated instruction is paired with a gold tool-call trace and time constraints used for automatic scoring, and a stratified sample is manually verified for schema validity and plausibility. The full prompt templates and generation scripts are released in the open-source repository.

A.3 Details in Evaluation

We report three core planning metrics, for each episode, we align predicted calls to each expected step by selecting the best-matching call (same tool/action preferred; same family receives partial credit), then average step scores to obtain the episode score.

TM (Tool Match Accuracy). TM measures whether the agent selects the correct tool for each expected step. A step receives 1.0 for an exact tool-name match; 0.5 if the tool is from the correct family (UPQ/ESP/PMB) but the wrong action; and 0.0 otherwise. Episode TM is the mean over steps.

PA (Parameter Accuracy). PA evaluates whether required parameters are present and correct for the matched call of each expected step. We

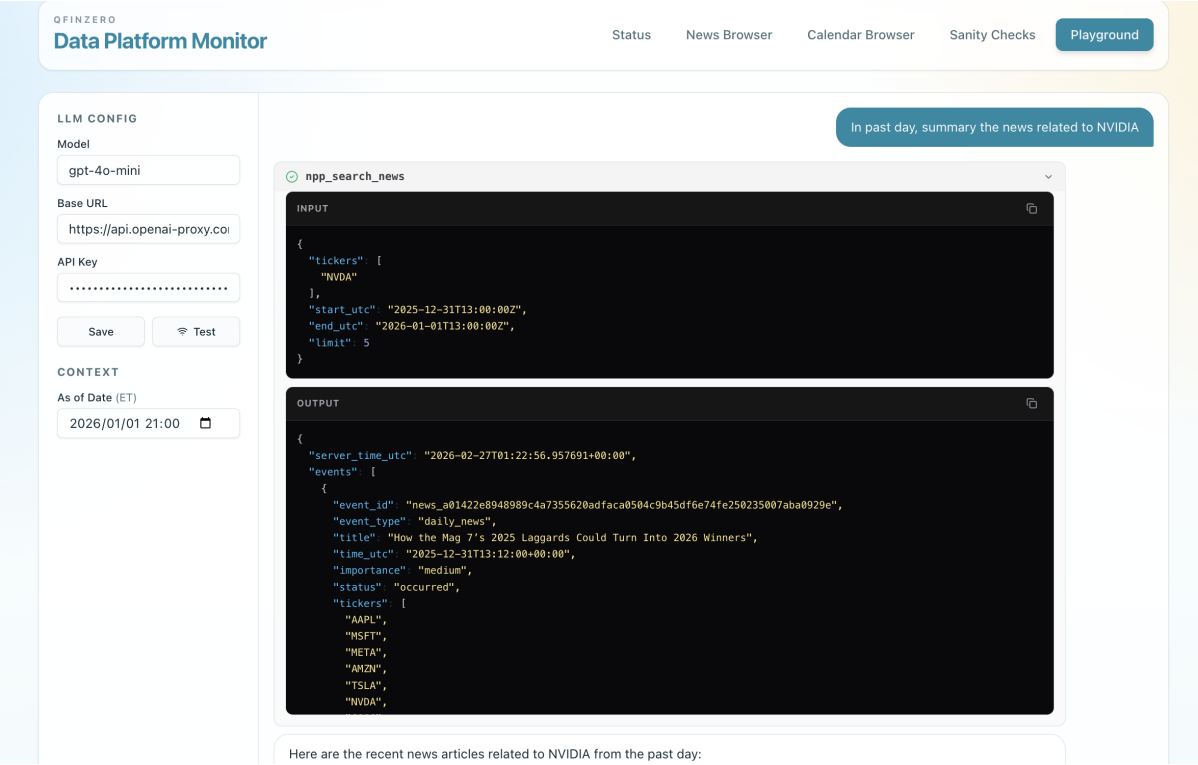

Figure 5: QFinZero Playground interface demonstrating an example tool call and its structured JSON response for summarizing recent NVIDIA-related news within a specified time window.

recursively compare the predicted `args` against `required_params` (supporting nested dict/list structures). Missing required keys for that field or wrong values (with partial credit for list overlap via Jaccard) receive 0 ; numeric values use relative tolerance (10^{-4}); and dependency placeholders in the gold (e.g., `{session_id}`, `<from_step_1>`) are ignored (not penalized). Episode PA is the mean per-step parameter score.

TA (Time Alignment). TA checks whether time/date arguments satisfy the benchmark `time_constraints`. For each expected step, we extract the constrained time fields (e.g., `start_date`, `end_date`, `start_utc`) and compute a field-wise match using the specified tolerance: `exact` (0s), `5min` (± 5 minutes for intraday), `1day` (± 1 calendar day for daily), `2day` (± 2 days for loose windows), and `30min`/`60min` for horizon-like constraints. Step TA is the average across its time fields, and episode TA is the mean over steps.

A.4 Additional Details in Interaction

We provide a dashboard and an interactive playground to help users visualize internal data assets, monitor service availability, and test how agents interact with the system. The dashboard enables inspection of database status and component health, while the playground offers a chatbot-style interface where users can submit arbitrary requests. The agent then invokes the appropriate tools within QFinZero to complete the task. This interface allows users to evaluate the function-calling capabilities of LLMs under the QFinZero framework. An example interaction is shown in Figure 5.

A.5 License

QFinZero is released under the Apache License 2.0. This license permits free use, modification, and redistribution of the software for both academic and commercial purposes, provided that the original copyright notice and license terms are retained.