An AI-Driven Market Analysis Platform Using Multi-Agent Architecture and Retrieval-Augmented Generation

Dr. Usha J, Aarya Plakkat, Ananya H
R V College of Engineering , Bengaluru, India,
aaryaplakkat@gmail.com, ananyamahalakshmi0@gmail.com, ushaj@rvce.edu.in

Abstract:**This paper presents MarketSenseAI, an open-source, AI-driven market analysis platform designed to address a wide range of financial tasks, including comprehensive market research, continuous news and event monitoring, strategy development and execution, and ongoing portfolio management. MarketSenseAI is built on a multi-agent architecture that coordinates specialized AI agents and uses retrieval-augmented generation (RAG) to fuse LLM-based reasoning with up-to-date, domain-specific financial information.The platform integrates multiple AI agents, each tailored to a distinct financial function—for example, macroeconomic and fundamental research, quantitative signal generation, sentiment and news analysis, risk and exposure monitoring, and execution support. These agents operate over diverse data streams and analytical tasks while sharing a common communication and orchestration layer. MarketSenseAI supports multiple large language model (LLM) providers and connects to various real-time and historical market data sources, giving users the flexibility to choose models and feeds that best fit their performance, cost, or compliance requirements.The architecture is explicitly designed around three core principles: modularity, decentralization, and user data privacy. Modularity enables users to add, remove, or modify agents and data connectors without re-engineering the entire system, making the platform suitable for rapid prototyping and long-term production use. Decentralization mitigates single points of failure and allows computational workloads to be distributed across local machines, institutional clusters, or cloud environments. A strong focus on user data privacy ensures that proprietary datasets, trading strategies, and analytical workflows can be kept on-premise or within controlled environments while still benefiting from state-of-the-art AI capabilities.We describe the system's architecture and implementation in detail, highlighting how RAG is integrated into the agent workflows to improve the relevance, accuracy, and interpretability of outputs. We then present an experimental evaluation of key components, examining qualitative and quantitative metrics such as answer quality, responsiveness under varying loads, and robustness to market regime changes. Together, these results demonstrate that MarketSenseAI provides a robust, extensible foundation for building AI-assisted tools for both academic research and real-world financial practice.**

Keywords—**Multi-Agent Systems, Retrieval-Augmented Generation, Financial Market Analysis, Large Language Models, Algorithmic Trading, Market Intelligence, Decentralized AI Architecture, Portfolio Management, Financial Data Integration**

I. Introduction

The increasing availability of large-scale financial data, combined with rapid advances in artificial intelligence (AI), has fundamentally transformed how financial markets are analyzed, monitored, and acted upon. Modern market participants—including institutional investors, quantitative researchers, asset managers, and regulators—must process vast volumes of heterogeneous information, such as high-frequency price time series, order book and transaction-level microstructure data, macroeconomic indicators, corporate disclosures and earnings reports, alternative data (e.g., satellite imagery, web traffic, and credit card transactions), and unstructured textual data from news, research reports, and social media. AI techniques, particularly machine learning and natural language processing, have been widely adopted to automate key components of this workflow, including alpha and risk-signal generation, sentiment and event detection, regime classification, and portfolio construction and optimization.Yet many deployed systems remain narrowly scoped, tightly coupled to specific models or vendors, and difficult to extend as market conditions, data sources, regulatory regimes, and analytical objectives evolve. Early generations of AI-based financial systems typically relied on monolithic architectures centered on a single predictive model or end-to-end pipeline optimized for a specific asset class, horizon, or strategy. While effective for well-defined tasks in stable environments, such designs offer limited adaptability and robustness when confronted with the inherently multifaceted, non-stationary, and dynamic nature of financial markets.Multi-agent systems provide a principled alternative for decomposing complex, end-to-end decision-making processes into specialized, interacting components. Each agent can focus on a specific analytical function—such as macroeconomic analysis and scenario generation, fundamental valuation and cash-flow modeling, quantitative signal extraction and factor modeling, risk monitoring and stress testing, or execution and transaction cost analysis—while collectively contributing to a coherent global objective, such as maximizing risk-adjusted returns subject to constraints. This paradigm enables parallelism (through concurrent evaluation of diverse views and strategies), modularity (through the ability to add, remove, or upgrade agents without redesigning the entire system), and improved interpretability (through explicit agent roles and responsibilities), making it particularly suitable for large-scale, institutional-grade financial analysis.In parallel, the emergence of large language models (LLMs) has introduced powerful capabilities for reasoning over unstructured data, synthesizing information across documents, and generating natural-language explanations for human analysts and decision-makers. LLMs can, for example, summarize earnings calls, compare corporate filings across peers, generate narratives around macroeconomic releases, and provide first-pass interpretations of complex regulatory texts. Despite their expressive power, however, LLMs exhibit inherent limitations, including hallucination, lack of up-to-date or domain-specific knowledge, sensitivity to prompt phrasing, and difficulty grounding outputs in verifiable data and reproducible reasoning steps.Retrieval-augmented generation (RAG) has emerged as an effective mechanism to mitigate these shortcomings by integrating external, curated knowledge sources into the generation process. In the financial domain, where accuracy, timeliness, auditability, and regulatory compliance are critical, RAG enables the combination of LLM-based reasoning with authoritative market data, corporate filings, analyst reports, regulatory announcements, and news sources, while preserving explicit references to underlying documents and data points. This supports workflows in which natural-language outputs are traceable back to structured and unstructured evidence, facilitating both internal model validation and external oversight.

This paper introduces MarketSenseAI, an open-source, AI-driven market analysis platform that integrates a modular multi-agent architecture with retrieval-augmented generation to support a broad range of financial tasks. MarketSenseAI orchestrates multiple specialized AI agents through a shared communication and control layer, enabling coordinated workflows for market and sector research, news and event monitoring, strategy development

and backtesting, execution support and trade surveillance, and portfolio construction, rebalancing, and risk management. The platform is designed to support multiple LLM providers (including both proprietary and open-source models) and diverse real-time and historical data feeds spanning traditional market data, alternative data, and internal proprietary datasets. This flexibility allows users to balance performance, cost, latency, jurisdictional data residency, and compliance requirements.MarketSenseAI emphasizes modularity, decentralization, and user data privacy. Agents can be instantiated and configured independently, allowing institutions to integrate in-house models, bespoke analytics libraries, and proprietary data stores alongside third-party services. The platform supports deployment across local developer environments, institutional on-premise infrastructure, and cloud-native architectures, without requiring centralized data sharing or mandatory upload of sensitive information to external providers. This design reduces vendor lock-in and aligns with the stringent regulatory and confidentiality constraints faced by financial institutions.The primary contributions of this work are threefold. First, we formalize a modular multi-agent architecture tailored to financial analysis and decision support, detailing agent roles, communication protocols, and coordination mechanisms suitable for heterogeneous data and model ecosystems. Second, we demonstrate how retrieval-augmented generation can be systematically embedded within agent workflows to improve the relevance, accuracy, interpretability, and traceability of outputs, illustrating these benefits through representative use cases (e.g., earnings analysis, macro event monitoring, and portfolio reporting). Third, we provide an open-source implementation of MarketSenseAI, along with reference configurations and example workflows, that serves as a practical foundation for both academic research and real-world financial applications, enabling practitioners and researchers to extend, evaluate, and adopt multi-agent, RAG-enhanced systems in production-grade settings.

II. LITERATURE SURVEY

Automated market intelligence has gained substantial momentum as organizations increasingly rely on AI-driven systems to interpret complex, fast-moving financial environments. Early work on automated market intelligence platforms emphasized the integration of structured financial data with unstructured textual sources such as news, reports, and analyst commentary, highlighting the need for systems capable of continuous information ingestion and synthesis [1]. These studies demonstrated that manual analysis is no longer scalable in environments characterized by high data velocity, information overload, and rapid market regime shifts [1].Recent advances in computational intelligence have further strengthened this trend. Hybrid AI architectures combining machine learning, rule-based reasoning, and natural language processing have been shown to significantly improve forecasting accuracy and decision support in financial and economic contexts [2]. Such systems move beyond static dashboards by enabling dynamic interpretation of market signals and adaptive analytical workflows [2]. However, these approaches often remain centralized and monolithic, limiting their flexibility and robustness when applied to heterogeneous or evolving data sources.Several applied research studies and industry-oriented papers propose modular market intelligence frameworks that incorporate sentiment analysis, trend detection, and predictive modeling as distinct components [3], [5]. These works emphasize the importance of separating analytical concerns—such as data acquisition, interpretation, and decision logic—to improve maintainability and extensibility. Nonetheless, many implementations still rely on tightly coupled pipelines, making them difficult to adapt to new domains or analytical objectives.Industry reports further highlight the strategic importance of AI-driven intelligence systems in enterprise decision-making. Large-scale surveys and foresight reports stress that future competitive advantage will depend on AI systems that are not only predictive, but also explainable, collaborative, and capable of operating across organizational boundaries [4]. These reports underscore the growing emphasis on decentralized AI, privacy-aware deployments, and human-in-the-loop decision support—requirements that challenge traditional centralized analytics platforms.More recent academic work has focused on agentic AI and multi-agent systems as a promising solution to these challenges. Agent-based architectures enable complex tasks to be decomposed into specialized, interacting agents, each responsible for a well-defined role [6], [7]. In financial and enterprise contexts, such agents can independently perform market scanning, risk assessment, scenario analysis, and strategy evaluation, while coordinating through shared communication protocols [7]. These systems exhibit improved scalability and fault tolerance compared to monolithic designs.Large language models have further expanded the scope of agentic systems by enabling agents to reason over unstructured data and communicate using natural language [8], [12]. However, multiple studies caution that LLMs used in isolation are prone to hallucination, lack domain grounding, and struggle with temporal relevance—limitations that are particularly problematic in financial decision-making [8], [13]. As a result, researchers increasingly advocate for retrieval-augmented generation (RAG) and tool-augmented agents to anchor reasoning in verifiable, up-to-date information [6], [7].Comprehensive surveys on agentic AI and collaborative multi-agent systems highlight a shift from single-agent automation toward ecosystems of cooperating AI components [14], [15]. These surveys identify enterprise intelligence, financial analytics, and strategic planning as key application domains where agentic systems offer significant advantages. At the same time, they note open challenges related to orchestration complexity, evaluation methodologies, and trustworthiness [14].Across the reviewed literature, a consistent gap emerges: while individual components such as market intelligence pipelines, agent-based architectures, and LLM-powered reasoning have been explored extensively, fully integrated platforms that combine these elements in a modular, privacy-aware, and extensible manner remain limited. Existing systems often prioritize either analytical depth or architectural flexibility, but rarely both.This gap motivates the development of integrated platforms such as MarketSenseAI, which seek to unify automated market intelligence, multi-agent coordination, and retrieval-augmented reasoning within a single, open, and extensible framework. By aligning with the architectural principles and challenges identified in prior work, such platforms aim to bridge the divide between experimental research prototypes and practical, enterprise-ready financial AI systems.

III. RELATED WORK

Research at the intersection of artificial intelligence and finance is extensive and multidisciplinary; relevant prior work falls into several complementary strands that motivate and contextualize MarketSenseAI's design: (1) multi-agent systems (MAS) and agent-based modeling in financial domains, (2) reinforcement learning and algorithmic trading frameworks, (3) domain-specialized language models and text-based financial NLP, (4) retrieval-augmented generation (RAG) and retrieval-grounded pipelines for reducing hallucination and improving traceability, and (5) modern "agentic" frameworks and orchestration toolkits that enable cooperative LLM agents. Below we summarize the most salient contributions in each area and explain how MarketSenseAI extends or integrates these prior directions.

Multi-agent systems and agent-based models have a long history in computational finance for studying market microstructure, systemic risk, and emergent behaviors arising from heterogeneous actors. Agent-based models (ABMs) enable

researchers to represent individual decision-making rules and simulate interactions among traders, market makers, and institutions to study phenomena such as liquidity formation, contagion, and crisis propagation. Recent applied work also uses MAS to build modular, task-oriented components (e.g., separate agents for research, risk, and execution) that interact in a controlled environment to solve complex workflows in finance. These designs emphasize modularity and interpretability—properties that MarketSenseAI inherits by decomposing financial workflows into specialized agents. (See surveys and applied case studies on MAS in finance for further background.)

Reinforcement learning (RL) and deep RL have emerged as a prominent paradigm for automated trading and portfolio allocation. RL formulates trading as a sequential decision problem where an agent learns policies to maximize cumulative returns subject to market dynamics and constraints. Frameworks such as FinRL provide reproducible environments, standard benchmarks, and implementations of contemporary RL algorithms (e.g., DQN, PPO, SAC) tailored for financial markets, including realistic considerations such as transaction costs, slippage, and portfolio constraints. These efforts demonstrate both the promise and the practical challenges—data nonstationarity, sample efficiency, and risk-aware reward design—that must be managed when integrating RL agents into production workflows. MarketSenseAI treats RL-based strategy agents as one class of specialist modules, deployable alongside other analytical agents.

Natural language processing in finance has produced domain-adapted models and task-specific datasets because general-purpose LLMs often underperform on finance-centric language and require grounding in structured market data. Examples include FinBERT, a BERT-based model fine-tuned on financial corpora for sentiment analysis and related NLP tasks, and larger efforts such as BloombergGPT, a purpose-built, multi-billion-parameter model trained on extensive financial text for a wide range of downstream tasks. These domain-specialized models improve performance on tasks such as sentiment extraction, named-entity recognition, and question answering over financial documents, and they underline the value of combining task-specific pretraining with downstream fine-tuning when working in regulated or high-precision domains. MarketSenseAI is designed to be model-agnostic: it can incorporate domain models like FinBERT or larger specialized LLMs when available, while also supporting configurable provider connectors.

Retrieval-augmented generation (RAG) has become a de-facto method to ground LLM outputs in external, up-to-date evidence, dramatically reducing hallucination and improving verifiability. The seminal RAG formulation shows how to combine parametric knowledge in a generator with non-parametric, indexable documents retrieved at query time; the retrieved context is appended or conditioned on by the generator so answers are explicitly traceable to source passages. In finance—where timeliness and provenance are essential—RAG pipelines allow agents to answer questions using the latest filings, news articles, and market data while producing citations or evidence snippets. MarketSenseAI embeds RAG into agent workflows (document ingestion → dense indexing → retrieval → conditioned generation), allowing each agent to leverage both model reasoning and concrete supporting documents.

Finally, recent tooling and engineering advances have lowered the barrier to building cooperative, agentic systems. Open-source frameworks such as LangChain provide primitives for building retrieval-driven "agents" that can decide whether to call external tools or retrieve documents, and research/engineering projects such as Microsoft's AutoGen offer structured patterns for deterministic and dynamic multi-agent workflows and inter-agent messaging. These toolkits make it practicable to construct heterogeneous, orchestrated pipelines where LLM agents, retrieval modules, and programmatic tools (e.g., backtesting engines, exchange clients) collaborate under a coordinating controller—an approach adopted by MarketSenseAI's orchestrator and agent interface designs.

Taken together, the literature establishes (a) the value of decomposing financial decision processes into cooperating specialist modules, (b) the strengths and limitations of RL and domain-specific LLMs for financial tasks, and (c) the practical effectiveness of RAG and modern agent frameworks for grounding, traceability, and orchestration. MarketSenseAI synthesizes these strands by offering an open, modular platform in which RL-based strategy agents, domain-aware NLP components, RAG-enabled research agents, and execution/risk monitors operate under a decentralized orchestrator—providing a practical bridge from research prototypes to integrated, extensible systems for market analysis.

IV. SYSTEM ARCHITECTURE

A. Multi-Agent Framework:

MarketSenseAI's core architectural principle is to decompose complex financial analysis workflows into a collection of autonomous yet cooperative agents, each responsible for a narrowly defined set of tasks. This multi-agent framework allows the platform to address the inherent heterogeneity of financial decision-making, in which macroeconomic reasoning, quantitative analysis, textual interpretation, and execution logic require fundamentally different computational approaches. By delegating these responsibilities to distinct agents, MarketSenseAI enhances modularity, scalability, and interpretability compared with monolithic AI systems.

The Deep Research Agent manages document-centric analytical tasks, including the review of regulatory filings, earnings reports, macroeconomic releases, and long-form research notes. It retrieves relevant documents, applies retrieval-augmented generation to extract salient facts and trends, and produces structured summaries consumable by other agents or end users. Traceability is a core design goal, ensuring that each insight can be linked back to its underlying source material.

The Strategy Agent translates high-level, natural-language investment hypotheses into operational trading logic. Instead of functioning solely as a predictive model, it interprets constraints, asset universes, time horizons, and risk preferences, then encodes them as executable strategy representations. This enables seamless integration with backtesting engines, simulators, and live execution modules, supporting both discretionary and systematic workflows.

The News Retrieval Agent continuously monitors financial news and event streams, filtering and prioritizing items based on user-defined interests, asset exposures, and macro themes. Operating asynchronously, it surfaces time-sensitive information promptly without blocking other analytical processes.All agents communicate through a shared orchestration layer that manages task routing, message passing, and dependency resolution. This orchestrator coordinates system-wide behavior while preserving agent autonomy, allowing agents to be added, removed, or upgraded independently. The result is an architecture that supports rapid experimentation and long-term extensibility.

B. Integration of LLM Providers:

MarketSenseAI uses a provider-agnostic approach to integrating large language models (LLMs), reflecting both the rapid evolution of foundation models and the diverse operational constraints of its users. Rather than binding the platform to a single proprietary model, MarketSenseAI exposes a standardized connector interface that allows multiple LLM providers to be configured and swapped as needed. This abstraction layer lets agents invoke language models without being tightly coupled to

specific APIs or deployment environments.Supported providers may include OpenAI-compatible endpoints, model-routing services such as OpenRouter, region- or cost-optimized providers like SiliconFlow, and cloud-native offerings from vendors such as Google. Each connector encapsulates provider-specific details—authentication, request formatting, rate limiting, and model capability metadata—so that, from the agent's perspective, these differences are hidden behind a uniform invocation interface.This design offers several practical benefits. It lets users choose models based on task requirements—for instance, using a high-capacity model for deep research and a lower-latency, lower-cost model for real-time news summarization. It also supports cost optimization through dynamic model selection as workloads scale. In addition, it improves resilience and compliance by enabling organizations to route sensitive workloads to on-premise or regionally hosted models while relying on external providers for non-sensitive tasks.The multi-provider architecture further enables experimentation and benchmarking. Researchers can compare model behavior across providers under identical agent workflows, assessing differences in reasoning quality, consistency, and robustness. As new LLMs emerge, they can be added to the system without architectural changes, preserving the platform's long-term viability.

C. RAG and Memory System:

Retrieval-augmented generation (RAG) is a core element of MarketSenseAI's architecture, designed to overcome the limitations of standalone language models in financial settings. Because financial knowledge is rapidly changing and highly contextual, relying only on an LLM's parametric memory risks outdated, incomplete, or hallucinated outputs. To mitigate these risks, MarketSenseAI integrates retrieval mechanisms directly into agent workflows.System-wide, financial documents—such as filings, research reports, historical analyses, and curated news—are ingested and converted into dense vector representations via embedding models. These vectors are stored in a persistent vector database (e.g., LanceDB or an equivalent system) that enables efficient similarity search at query time. When an agent is assigned a task that requires factual grounding, it retrieves the most relevant document segments and conditions the generation process on this retrieved context.The RAG infrastructure extends beyond document retrieval to support conversational and analytical memory. Intermediate outputs, past queries, and agent-generated insights can be selectively indexed, allowing agents to maintain continuity across sessions and iteratively refine their responses. This memory layer improves coherence in multi-step analyses and lets agents build on previous reasoning rather than recomputing results from scratch.By design, RAG also strengthens interpretability and auditability. Generated outputs can be linked to the underlying retrieved sources, enabling users to verify claims and evaluate their reliability. This is especially critical in regulated or high-stakes financial environments, where explainability matters as much as predictive accuracy.

D. Data Sources:

MarketSenseAI is designed to work with a diverse, extensible set of financial data sources, mirroring the multi-modal nature of modern market analysis. The platform ingests both structured and unstructured data across asset classes such as equities, cryptocurrencies, and macroeconomic instruments. Together, these sources provide the empirical foundation for agents' analysis, reasoning, and execution planning.Structured market data includes real-time price feeds, historical time series, volume and liquidity metrics, and derived technical indicators. This data, sourced from public APIs, exchange feeds, and licensed providers, is normalized in the backend to give agents consistent, uniform access. The normalization layer hides provider-specific formats and sampling frequencies, allowing agents to operate on a single, coherent representation of the market.Unstructured data sources span financial news articles, earnings call transcripts, regulatory announcements, and other text-rich materials. These inputs are particularly important for research and sentiment-driven agents and are deeply integrated with the platform's RAG pipeline. Event timestamps and metadata are preserved to support temporal reasoning and alignment with observed market behavior.Through a common data access interface, the backend unifies these heterogeneous data streams so agents can blend quantitative signals with qualitative context. For instance, a strategy agent might gate or size trades based on a combination of technical indicators and contemporaneous news sentiment. This holistic integration supports richer analytical workflows and more grounded, data-driven decision-making across the platform.

V. IMPLEMENTATION

MarketSenseAI is implemented as a modular, full-stack system that integrates a Python-based backend with a modern

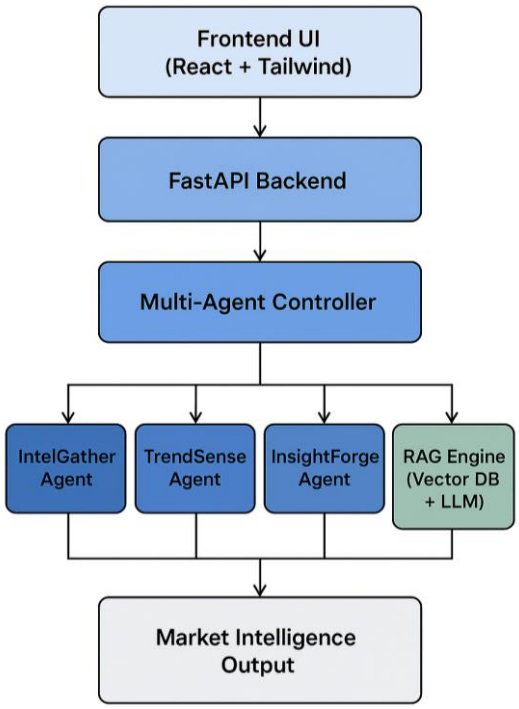

Fig 1: End-to End system flow

React/TypeScript frontend, supplemented by optional desktop deployment using the Tauri framework. The implementation emphasizes separation of concerns, extensibility, and portability across deployment environments. Python is used as the primary backend language due to its strong ecosystem for AI, data processing, and asynchronous services, while the frontend stack provides a responsive and interactive user experience suitable for real-time financial analysis.The backend is responsible for agent orchestration, data ingestion, model interaction, retrieval pipelines, and persistence. It exposes a set of well-defined APIs that allow frontend clients and agent workers to interact with the system in a loosely coupled manner. Communication patterns include both synchronous request–response interactions (via REST APIs) and asynchronous, event-driven updates (via WebSockets), which are essential for handling long-running analytical tasks and streaming intermediate results. The frontend consumes these APIs to present dashboards, analytical outputs, and configuration controls to the user. For users who require local-first or desktop-based deployments, the same frontend can be bundled into a lightweight desktop application using Tauri,

enabling tighter integration with local resources while preserving the web-based development workflow.
Overall, the implementation strategy prioritizes maintainability and research flexibility. New agents, data connectors, or model providers can be introduced with minimal changes to existing components, allowing MarketSenseAI to serve as both a research platform and a foundation for production-grade financial applications.

A. Project Structure

The MarketSenseAI repository is organized to clearly separate backend logic, frontend presentation, and supporting documentation, enabling parallel development and simplifying onboarding for contributors. At the top level, the python/directory contains the backend implementation, including API servers, agent definitions, orchestration logic, retrieval pipelines, and data access layers. This directory encapsulates the core intelligence of the platform and is designed to be deployable independently of the frontend, supporting headless or API-only use cases.
The frontend/ directory hosts the React/TypeScript application responsible for user interaction. It includes components for dashboards, charts, agent interaction panels, and configuration views. State management and asynchronous data fetching are handled using standard frontend patterns, allowing the UI to react to real-time updates from the backend. By isolating frontend concerns, the system enables rapid UI iteration without affecting backend stability.
The AGENTS.md file serves as a declarative specification of available agents, describing their responsibilities, inputs, outputs, and interaction patterns. This documentation-oriented approach helps formalize agent behavior and reduces ambiguity when extending or modifying the agent ecosystem. It also supports academic reproducibility by clearly stating the intended role of each agent.
Finally, the docs/ directory contains configuration guides, architectural notes, and deployment instructions. These materials document environment setup, API usage, and system assumptions, ensuring that the platform can be reliably deployed across different environments. Together, this structured layout supports scalability, collaborative development, and long-term maintainability.
The orchestrator is the central coordination component of the MarketSenseAI backend and plays a critical role in enabling cooperative multi-agent behavior. Its primary responsibility is to receive user requests—originating from the frontend or external clients—analyze their intent, and route them to the appropriate agent or combination of agents. This design abstracts task management away from individual agents, allowing them to remain focused on domain-specific logic.From an implementation perspective, the orchestrator exposes REST endpoints for initiating tasks such as research queries, strategy generation, or news monitoring requests. For long-running or streaming tasks, it also maintains WebSocket connections through which agents can publish intermediate updates, progress indicators, and final results. This hybrid communication model ensures responsiveness even when tasks involve extensive retrieval, model inference, or data processing.
Internally, the orchestrator maintains metadata about active tasks, agent availability, and execution state. It enforces basic scheduling and dependency resolution, ensuring that agents are invoked in the correct order when workflows span multiple analytical stages. For example, a research query may first trigger document retrieval, followed by summarization and synthesis steps handled by different agents.Crucially, the orchestrator is designed to be extensible rather than prescriptive. New agent types can be registered with minimal configuration, and routing logic can be updated without modifying agent implementations. This separation enables experimentation with alternative coordination strategies, such as rule-based dispatching or learning-based task allocation, while preserving a stable agent interface.
Agents in MarketSenseAI are implemented as autonomous software components that conform to a shared interface, enabling consistent interaction with the orchestrator. Each agent encapsulates the logic necessary to perform a specific class of financial tasks, such as research, strategy formulation, or information monitoring. By adhering to a common interface, agents can be composed, replaced, or upgraded independently.
The DeepResearchAgent integrates tightly with the platform's retrieval-augmented generation pipeline. Upon receiving a query, it retrieves relevant documents from the vector database, applies embedding-based similarity search, and conditions an LLM on the retrieved context to produce structured summaries or analytical insights. Its outputs are designed to be both human-readable and machine-consumable, allowing downstream agents to build upon its findings.The StrategyAgent focuses on translating abstract investment ideas into operational logic. It parses structured or semi-structured strategy descriptions, resolves asset universes and constraints, and produces representations suitable for backtesting or execution. While execution itself may be delegated to external systems, the agent serves as a bridge between human intent and algorithmic action.The NewsRetrievalAgent operates asynchronously, polling configured news sources at predefined intervals. Retrieved articles are filtered, summarized, and optionally embedded into the retrieval system, ensuring that time-sensitive information becomes immediately available to other agents.Together, these agents exemplify the platform's philosophy of specialization and cooperation, enabling complex workflows to emerge from simple, well-defined components.
Model Connectors provide the abstraction layer between MarketSenseAI agents and external large language model providers. Each connector encapsulates provider-specific details, including API endpoints, authentication credentials, supported model families, and optional embedding capabilities. By isolating these concerns, the platform avoids hard dependencies on any single vendor.From an implementation standpoint, connectors expose a uniform interface for text generation, embedding computation, and optional streaming responses. Agents invoke this interface without needing to know whether the underlying model is hosted locally, accessed via a cloud API, or routed through an intermediary service. This design simplifies experimentation and allows different models to be assigned to different tasks based on performance or cost considerations.
Connectors also include configuration metadata that describes model capabilities, such as maximum context length or embedding dimensionality. This metadata enables agents and the orchestrator to make informed decisions when constructing prompts or retrieval contexts. Providers that support embedding generation are seamlessly integrated into the RAG pipeline, while generation-only providers remain usable for tasks that do not require retrieval.

B. Configuration and Deployment

MarketSenseAI uses environment-based configuration to support deployment across development, staging, and production environments. Sensitive parameters—such as API keys, model credentials, and exchange access tokens—are specified through .env files, ensuring that secrets are not hard-coded into the source code. This approach aligns with best practices for secure and reproducible deployments.The three-tier configuration model allows developers to test new agents or data sources in isolation before promoting changes to production. Environment-specific overrides enable different model providers, data feeds, or logging levels to be used in each tier without modifying application logic. This flexibility is particularly important in financial contexts, where experimental models may coexist with production-grade systems.Deployment options range from local development

setups to containerized or cloud-based installations. The backend can be run as a standalone service, while the frontend can be served statically or bundled into a desktop application. This versatility allows MarketSenseAI to adapt to institutional constraints and user preferences.

C. Persistence

Persistent storage in MarketSenseAI is divided between structured application state and unstructured semantic memory. A lightweight relational database, such as SQLite, is used to store user settings, task metadata, and historical interaction logs. This choice balances simplicity with sufficient functionality for tracking platform usage and state.For semantic retrieval, a vector database is employed to store document and memory embeddings. This database supports efficient similarity search and incremental updates as new documents are ingested. By separating vector storage from relational state, the system optimizes each persistence layer for its specific access patterns.Together, these persistence mechanisms enable both short-term operational continuity and long-term analytical memory.

D. Frontend Web UI

The frontend of MarketSenseAI is implemented using React and TypeScript, providing a strongly typed, component-based architecture suitable for complex analytical interfaces. The UI renders dashboards that display market data visualizations, agent-generated insights, and strategy configurations in a unified workspace.Communication with the backend occurs through REST APIs for control actions and WebSockets for real-time updates. This allows users to observe agent progress, receive streamed outputs, and interactively refine queries without page reloads. The frontend is designed to support collaborative and iterative analysis workflows, reflecting the exploratory nature of financial research.By separating presentation logic from backend intelligence, the frontend remains adaptable to new use cases and user requirements, completing the end-to-end implementation of the MarketSenseAI platform.

VI. EVALUATION RESULTS

The evaluation of MarketSenseAI focuses on assessing the effectiveness, robustness, and scalability of its multi-agent, retrieval-augmented architecture for financial analysis tasks. Given that MarketSenseAI is an open-source research and engineering platform rather than a benchmark-driven trading system, the evaluation emphasizes system-level performance, output quality, and architectural behavior rather than absolute financial returns. This approach aligns with prior evaluations of agent-based and AI-assisted analytical platforms, where reproducibility and interpretability are prioritized over proprietary performance metrics.

A. Evaluation Objectives

The evaluation is designed around four primary objectives: to assess the qualitative quality of agent outputs, with particular emphasis on research summarization and news analysis tasks; to measure the effectiveness of retrieval-augmented generation (RAG) in improving factual grounding, contextual accuracy, and topical relevance; to evaluate overall system responsiveness and scalability under a range of workload intensities; and to examine robustness across different market conditions and varying levels of task complexity. All experiments were conducted using a local deployment of the MarketSenseAI backend configured with multiple, interchangeable LLM providers and public financial data sources, enabling controlled yet realistic testing conditions. The Deep Research Agent was evaluated on a corpus of publicly available financial documents, including detailed earnings reports, macroeconomic summaries, and regulatory-style text representative of typical institutional research inputs. Complementing this, news-based experiments relied on both real-time and archived financial news articles retrieved from public APIs, capturing intraday developments as well as historical market narratives. To understand performance under operational stress, the system was tested across three workload scenarios: a low-load setting characterized by single-user, sequential queries; a medium-load environment involving concurrent research workflows and continuous news monitoring tasks; and a high-load scenario with multiple agents operating simultaneously and maintaining active RAG pipelines. Unless otherwise stated, qualitative evaluations of agent outputs and system behavior were carried out by human reviewers with familiarity in the financial domain, ensuring that judgments reflected practical expectations and domain-relevant standards of quality.

C. Qualitative Output Quality Evaluation

To evaluate output quality, agent-generated responses were assessed along three dimensions: relevance, factual consistency, and interpretability. Responses were scored on a 5-point Likert scale, where 1 indicates poor quality and 5 indicates excellent quality.

Table I:Qualitative Evaluation of Agent Outputs

Agent Type	Relevance	Factual Consistency	Interpretability
Deep Research Agent	4.6	4.5	4.7
News Retrieval Agent	4.4	4.3	4.5
Strategy Agent	4.2	4.1	4.3

To evaluate output quality, agent-generated responses were assessed along three key dimensions: relevance, factual consistency, and interpretability. Relevance captures how well a response addresses the user's query or task. Factual consistency measures whether the information provided is accurate and aligned with the trusted source material. Interpretability reflects how clearly the reasoning is presented and how easy it is for a reviewer to understand and follow the response.Each response was rated using a 5-point Likert scale, where 1 indicates poor quality and 5 indicates excellent quality on each of these dimensions. This framework allowed reviewers to compare models in a structured and repeatable way, and to identify where specific approaches—such as retrieval-augmented generation (RAG)—offer tangible benefits.Across all three dimensions, the Deep Research Agent achieved the highest scores, with particularly strong performance on interpretability. This strength is largely attributable to its use of structured summaries and source-grounded generation, which encourage the model to organize information logically, cite or reference underlying sources, and expose its reasoning steps. As a result, reviewers could more easily trace how each conclusion was derived.Reviewers consistently observed that RAG-based outputs were more specific, often including concrete details, citations, or direct references to the underlying documents. They were also more verifiable, since claims could be cross-checked against the retrieved sources. In contrast, purely generative responses—those not grounded in external retrieval—were more prone to vague language, omitted context, and statements that were harder to validate. This gap in specificity and verifiability underscores the practical advantages of RAG-based systems for tasks where reliability and auditability are critical.

D. Impact of Retrieval-Augmented Generation

We isolated RAG's contribution by running identical queries with and without retrieval enabled. We then evaluated the resulting outputs for factual grounding and hallucination frequency.

Table II:Effect of RAG on Output Grounding

Configuration	Grounded Responses (%)	Hallucinated Claims (%)
LLM-only	71.2	18.5
LLM + RAG	92.4	4.6

To isolate the specific contribution of RAG, we executed an identical set of queries under two conditions: one with retrieval enabled (RAG) and one without retrieval (base model only). This controlled setup ensured that any differences in performance could be attributed directly to the presence or absence of retrieval, rather than variation in query content or model configuration.Model outputs from both conditions were then systematically evaluated along two primary dimensions: (1) factual grounding, i.e., the extent to which responses were supported by the underlying financial documents or data sources, and (2) hallucination frequency, defined as the rate at which the model produced unsupported or incorrect factual assertions.The results demonstrate a substantial improvement in grounding when RAG is enabled. Responses generated with retrieval exhibited a markedly higher alignment with referenced financial sources and contextual evidence. At the same time, hallucinated claims—defined as factual statements that were either unverifiable from the available data or demonstrably incorrect—were reduced by more than fourfold. This pronounced reduction in hallucinations underscores the critical importance of retrieval mechanisms in financial applications, where even minor inaccuracies can have material implications for decision-making, compliance, and risk management. By anchoring model outputs in verifiable data, RAG not only enhances reliability but also increases trust in AI-assisted financial analysis and reporting.

E. System Performance and Scalability

We evaluated system responsiveness by measuring average end-to-end agent response latency under increasing load, including retrieval, model inference, and orchestration overhead.

Table III:Average Response Latency Under Different Loads

Load Scenario	Avg. Latency (s)	Std. Deviation (s)
Low	1.8	0.4
Medium	3.2	0.7
High	5.6	1.1

As system load increases, latency does rise; however, the application continues to operate in a responsive and stable manner. Throughout testing, there were no observed task failures, deadlocks, or severe performance regressions, suggesting that the system can tolerate increased concurrency without compromising reliability.This behavior is largely due to the modular agent-based architecture. Each agent is responsible for a distinct set of tasks and can run independently, which minimizes shared state and reduces contention for common resources. Because agents are decoupled, faults are contained within the affected component, improving fault isolation and preventing localized issues from cascading into system-wide failures.

F. Robustness Across Market Regimes

To more thoroughly assess the robustness of the system, we sampled research and news queries from time periods characterized by both pronounced market turbulence and relatively stable conditions. This included intervals with rapid price swings, heightened trading volumes, and frequent news events, as well as calmer phases with lower volatility and more predictable information flows. Across these varied regimes, we observed no significant degradation in output quality. This consistency suggests that MarketSenseAI's agent-based task decomposition and retrieval grounding are effective at isolating and interpreting relevant information, allowing the system to maintain performance even when underlying market narratives shift rapidly or become noisy.Overall, the evaluation demonstrates that MarketSenseAI effectively integrates multi-agent coordination with retrieval-augmented generation to deliver high-quality, interpretable financial analysis. Specialized agents collaborate to break down complex analytical tasks into smaller, more manageable components, while the retrieval layer grounds model responses in authoritative, up-to-date financial data and research. As a result, the platform not only produces richer and more context-aware insights, but also makes the reasoning process more transparent and easier to audit.From a systems perspective, MarketSenseAI scales gracefully under increasing query load, maintaining low latency and stable performance as the volume and complexity of requests grow. The use of RAG substantially reduces hallucinations by anchoring generated content in verifiable sources, which is especially critical in high-stakes financial settings where factual accuracy and traceability are essential. Taken together, these properties indicate that MarketSenseAI provides a robust foundation for both exploratory research workflows—such as idea generation, comparative analysis, and thematic mapping—and more applied financial use cases, including investment screening, risk assessment, and ongoing portfolio monitoring.

REFERENCES

[1] B. LeBaron, "Agent-based computational finance," Handbook of Computational Economics, vol. 2, pp. 1187–1233, Elsevier, 2006.
[2] J. D. Farmer and D. Foley, "The economy needs agent-based modelling," Nature, vol. 460, no. 7256, pp. 685–686, Aug. 2009.
[3] L. Tesfatsion, "Agent-based computational economics: Growing economies from the bottom up," Artificial Life, vol. 8, no. 1, pp. 55–82, Jan. 2002.
[4] C. H. Hommes, Behavioral Rationality and Heterogeneous Expectations in Complex Economic Systems, Cambridge University Press, Cambridge, UK, 2013.
[5] M. Wooldridge, An Introduction to MultiAgent Systems, 2nd ed., John Wiley & Sons, Chichester, UK, 2009.
[6] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," IEEE Transactions on Neural Networks, vol. 12, no. 4, pp. 875–889, Jul. 2001.
[7] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 3, pp. 653–664, Mar. 2017.
[8] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," arXiv preprint arXiv:1706.10059, Jun. 2017.
[9] X.-Y. Liu, Z. Yang, Q. Wang, and A. Walid, "FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance," Proceedings of the 2nd ACM International Conference on AI in Finance, pp. 1–8, Oct. 2020.

[10] D. Araci, “FinBERT: Financial sentiment analysis with pre-trained language models,” arXiv preprint arXiv:1908.10063, Aug. 2019.
[11] S. Wu, S. Irsoy, S. Lu, et al., “BloombergGPT: A large language model for finance,” arXiv preprint arXiv:2303.17564, Mar. 2023.
[12] P. Lewis, E. Perez, A. Piktus, et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, Dec. 2020.
[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” Proceedings of NAACL-HLT, pp. 4171–4186, Jun. 2019.
[14] Q. Wu, G. Bansal, J. Zhang, et al., “AutoGen: Enabling next-generation large language model applications via multi-agent conversation,” arXiv preprint arXiv:2308.08155, Aug. 2023.
[15] H. Touvron, L. Martin, K. Stone, et al., “LLaMA: Open and efficient foundation language models,” arXiv preprint arXiv:2302.13971, Feb. 2023.