Multi-Agent Investment Analyst

An AI-Powered Financial Advisor

Muskan Sharma, Heenal Patel, Sanjeev Ratnani, Tej Shah, Harsh Shah

---

Abstract:

This proposal envisions a multi-agent AI system to assist both individual and institutional investors in navigating complex financial markets. The system combines specialized agents (Data, Analyst, Thesis, Verification) that communicate via prompts and shared memory. It leverages open-source large language models (e.g., Meta's LLaMA 2 and GPT4All) and finance-tailored embeddings (e.g., FinBERT) to retrieve and interpret diverse financial data.

Such a modular agent-based architecture has shown improved accuracy and explainability in investment decision systems. Open-source tools like LangChain and AutoGen will orchestrate the agents and enable retrieval-augmented workflows. Optional components (e.g. Whisper for voice Q&A) can further broaden accessibility. This ambitious yet feasible capstone project aims to create an adaptive advisor that generates data-driven investment theses, simulations, and alerts, helping users make informed decisions.

Use Case: Individual and Institutional Investors

Individuals and institutions alike face information overload: thousands of daily news items, earnings calls, and filings. A multi-agent advisor can personalize analysis for these users. For retail investors, the system can translate raw data into simple insights (e.g. "buy" or "sell" theses) and alerts. For institutions or wealth managers, it can produce detailed summaries and risk assessments. In traditional finance (TradFi), quantitative funds and "robo-advisors" already rely on AI for tasks from portfolio optimization to trade execution. Our tool extends this to real-time data curation: the Data Agent continuously ingests live news (via RSS), market prices (Yahoo Finance), and SEC filings, tailoring output to each user's portfolio or interests. In essence, the system acts as a 24/7 digital analyst, identifying opportunities or warning of threats for a variety of investor profiles.

Motivation

Financial decision-making is hindered by the sheer volume and diversity of data (news, reports, charts). Manual analysis is slow and error-prone; even static models quickly become outdated. An AI-powered advisor addresses these challenges by continuously integrating new information. Unlike one-shot generators, our multi-agent design allows iterative refinement and checks: for example, the Verification Agent ensures that a bullish thesis is consistent with the latest risk metrics.

AI/ML is already transforming trading and portfolio management, but most systems are either narrow (e.g. a single algorithm) or rely on proprietary APIs. By contrast, this project uses open models

(LLaMA 2, GPT4All) and fine-tuned finance models (FinBERT) to democratize advanced analysis. The modular, explainable pipeline (inspired by recent multi-agent frameworks) builds user trust: each agent's reasoning can be inspected or translated into natural language.

Model in Use: Open LLMs + Finetuned Embeddings

The core language models will be open-source LLMs such as:

- Meta's LLaMA 2 (7B–70B parameters)
- Nomic's GPT4All (which runs LLMs locally on laptops)

These provide flexible NLP capabilities without proprietary restrictions.

To tailor the system to finance, we will use domain-specific embeddings and models. For example:

- FinBERT (a BERT model fine-tuned on financial text) to capture sentiment and terminology from news and reports
- Specialized Sentence-Transformer embeddings (e.g. FinLang Investopedia embeddings) to map financial concepts into vector space
- Retrieval of relevant data by FAISS or Chroma as vector stores
- LangChain (or AutoGen) to tie these together into a retrieval-augmented generation (RAG) pipeline

All components chosen (LLaMA/GPT4All, FAISS/Chroma, Pandas/Numpy, LangChain, etc.) are open-source and have documentation suitable for student implementation.

Key Features

- **Modular Multi-Agent Pipeline:** Each agent handles a focused task and communicates via prompts. The Data Agent fetches and vectorizes information, the Analyst Agent performs calculations, the Thesis Agent constructs narrative cases, and the Verification Agent cross-checks results. This modular architecture makes development manageable and mimics how professional teams operate. In multi-agent AI systems, simple units combine to solve complex goals.

- **Real-Time and Batch Data Retrieval:** The Data Agent ingests data both in real-time (live price feeds, breaking news via RSS or APIs) and in batch (historical CSVs from Yahoo Finance, archives of filings). It vectorizes text (using finance-optimized embeddings) for similarity search. Agentic RAG design allows the system to adaptively query APIs or databases depending on the query context. For example, upon an earnings announcement, the Data Agent might fetch the latest 10-Q (via SEC API) and recent news headlines.

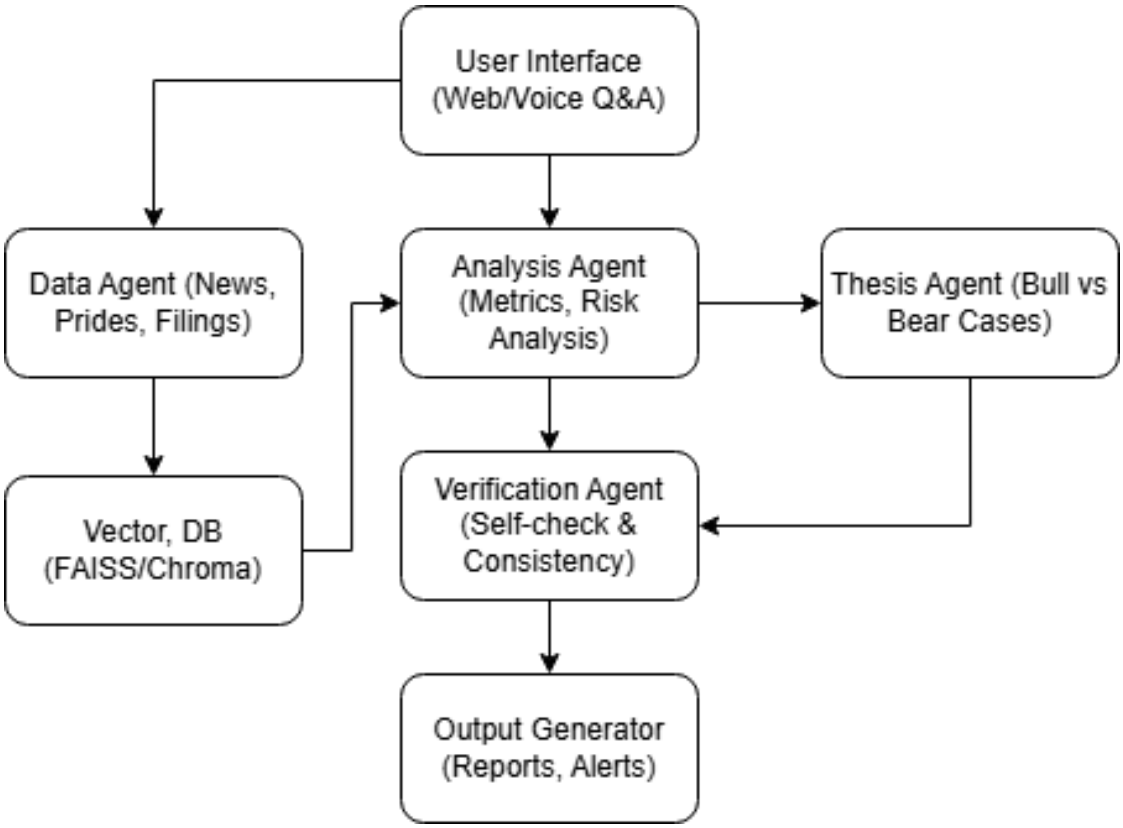
- **Investment Thesis Generation and Debate:** The Thesis Agent uses the Analyst's outputs (valuation, risk metrics) to form arguments for or against an investment. It generates structured "bull" vs. "bear" cases in natural language. The multi-agent design even supports internal debate: one agent can play devil's advocate to challenge assumptions, enhancing robustness. This reflects

research showing multi-agent LLM frameworks (with distinct expert agents) can yield more accurate and explainable finance decisions.

- **Scenario Simulation Engine:** The system can simulate hypothetical market scenarios. For example, “What if the Fed raises rates by 0.5%?” Agents can run stress-test simulations: adjusting earnings forecasts or risk parameters to model outcomes. This helps users explore “what-if” cases. The engine would combine simple quantitative models (e.g. Monte Carlo simulations, CVaR calculators) with narrative explanations from LLMs.

- **Alert and Insight Generation:** Using thresholds and anomaly detection, the system generates timely alerts (e.g. “Company X’s volatility just doubled, reevaluating exposure”). It also produces periodic insight summaries (daily or weekly briefs). Natural language outputs and optional voice Q&A (via Whisper for speech input/transcription) make the insights accessible. Crucially, a Verification Agent ensures alerts are justified (e.g. checking that a detected anomaly isn’t a data glitch), supporting the transparency needed for financial advice.

Architecture Diagram



Workflow Overview

1. **User Query:** The investor (or frontend interface) poses a question (e.g. “Should I buy stock XYZ?”).

2. **Data Agent (Retrieval):** Triggers pipelines to gather relevant data: recent news (vectorized via FinBERT embeddings), latest financial filings (SEC API JSON), stock prices (Yahoo Finance), etc. Results are cached in shared memory.

3. **Analyst Agent:** Processes the data: computes valuation metrics (P/E ratios, DCF analysis), risk measures (volatility, beta, correlations), and key performance indicators. Produces structured summaries of financial health.

4. **Thesis Agent:** Examines the Analyst’s findings and crafts investment narratives: a bullish case (“growth prospects, undervaluation”) and a bearish case (“overvaluation, high debt”). Uses the LLM to ensure coherent, persuasive language.

5. **Verification Agent:** Reviews the reasoning chain: cross-checks figures, ensures consistency (e.g. revenue growth assumptions match given data). Flags contradictions or overly biased views. It may query the LLM to justify or revise conclusions.

6. **Output Generation:** The final advice is formatted as a report or alert. If voice interaction is enabled, Whisper transcribes the user’s voice query, and responses can be spoken back. Throughout, all agents communicate via LangChain/AutoGen tooling, enabling prompts and shared context.

Modules Overview

Module	Description	Tech Used
Data Agent	Ingests and preprocesses data streams. Retrieves news (RSS/APIs), financial filings (SEC EDGAR API), price history (Yahoo Finance CSVs), etc. Splits and embeds text for vector search.	Python (requests, Pandas), FinBERT/Sentence-Transformers for embeddings, FAISS or Chroma (vector DB), RSS/CSV parsers

Analyst Agent	Performs quantitative finance analysis. Calculates valuation metrics (P/E, EV/EBITDA, DCF models), risk metrics (volatility, VaR, correlations), and portfolio impact. Prepares numerical summaries.	Python (NumPy, Pandas), PyPortfolioOpt or custom DCF code, Numpy/Scipy for stats
Thesis Agent	Generates natural-language investment theses. Writes bull and bear cases using LLM prompts that incorporate Analyst data. May simulate agent debate or counterarguments.	LLaMA/GPT4All (LLM), LangChain for prompt chaining
Verification Agent	Validates and cross-checks analyses. Ensures consistency in reasoning (e.g., checks that a 50% growth claim matches historical data). Provides explainability (queries LLM to justify calculations).	LLaMA/GPT4All, custom validation rules, possibly simple rule-based checks
(Optional Voice Interface)	Transcribes and synthesizes audio for Q&A. Converts spoken queries to text and reads back answers.	OpenAI Whisper or Mozilla DeepSpeech (speech-to-text)

Potential Datasets

- **Yahoo Finance:** Historical price/volume data and key statistics (via yfinance Python library) for equities and indices.
- **SEC EDGAR:** Company filings (10-K, 10-Q, 8-K, etc.) via the SEC's JSON APIs. These provide official financial statements and disclosures.
- **News RSS Feeds:** Real-time news headlines and articles from sources like Reuters, Bloomberg, Wall Street Journal. Used for sentiment and event detection.
- **Financial News Datasets:** Curated datasets (e.g., Kaggle news sentiment) for model fine-tuning or testing.
- **Economic Indicators:** Time series from sources like FRED (Federal Reserve) for macro variables (interest rates, CPI, GDP).

- **Others**: Company profiles (e.g. Wikipedia or Refinitiv data), earnings calendar APIs (for event predictions).

Future Scope

Future extensions could include:

- **Multilingual Reports:** Using multilingual models or translation (like Whisper's translation capability) to serve non-English investors or global markets.

- **Regulatory Compliance Alerts:** Adding a module to scan for compliance issues (e.g. SEC investigation news, insider trading signals) and notify legal teams. Natural language processing on regulations could flag policy breaches.

- **Expanded Asset Classes:** Incorporating crypto, forex, or alternative assets. The data/analyst agents could be extended with market-specific models (e.g. DeFi risk metrics) as in emerging "DeFAI" systems.

- **Personalization and Learning:** Adapting advice to user preferences and historical performance. For example, using reinforcement learning so agents learn from feedback (storing which recommendations users followed).

- **Interactive Visualizations:** Integrating dynamic charts or dashboards (e.g. Plotly, D3) for richer output. Users could adjust assumptions on the fly (changing inputs to the scenario engine).

- **Enhanced Explainability:** Building an "explain" mode where each agent's rationale is logged and presented. This could tie into research on AI transparency to meet financial accountability standards.

References

- Microsoft AutoGen – *"Agent and Multi-Agent Applications"* (documentation)
- Ganesh, Jagadeesan – *"Agentic RAG with LangChain: Revolutionizing AI…"* (Medium, 2024)
- Liu, Jung-Hua – *"Multi-Agent AI Architecture for Personalized DeFi Investment Strategies"* (Medium, 2023)
- Facebook AI (Meta) – *LLaMA 2 GitHub* (open-source LLMs 7B–70B)
- ProsusAI – *FinBERT* (financial text sentiment model)
- FinLang – *Finance-Embeddings-Investopedia* (domain-specific sentence embeddings)
- OpenAI – *Introducing Whisper* (open-source speech recognition)
- LangChain (J. Ganesh Medium, 2024) – *LangChain framework documentation*