

Multi-Agent AI System for Company Financial Risk Evaluation



32933 Research Project



The Best Research Project Team

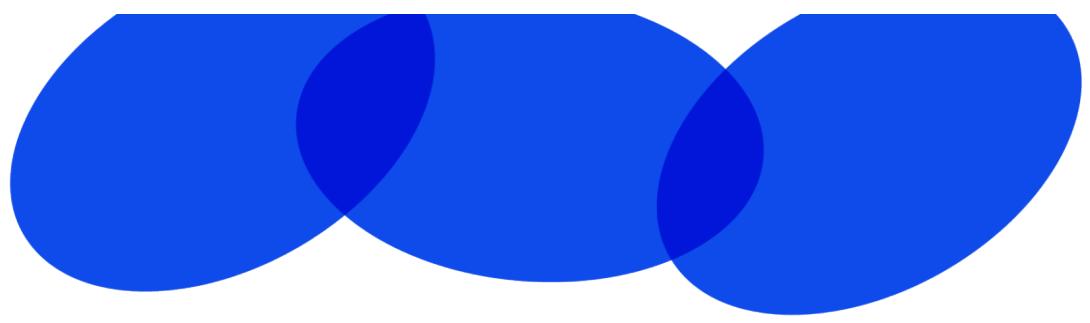
Ye Yint Thu 25422371

Rianto Utomo 25163306

Yingrong Zhang 25428842

George Luther 13528904

Project GitHub Link: <https://github.com/GeorgeLuther25/multi-agent-finance-risk-analysis>



32933

Research Project

Table of Contents

Table of Contents.....	1
Table of Figures	3
1. Introduction.....	4
Product GitHub Link.....	4
2. Literature review.....	4
A. Agentic AI.....	4
B. Existing Products	5
C. Gaps in the existing system.....	6
3. Requirements of the prototype system.....	7
3.1 Functional Requirements	7
- Sentiment Agent.....	7
- Valuation Agent.....	7
- Fundamental Agent.....	7
- Agent Collaboration.....	7
- Agent Debate	7
- Display results on GUI.....	7
- Download report PDF file.....	7
3.2 Non-functional Requirements.....	7
4. Database Design.....	7
5. System Architecture.....	8
5.1 Frontend/UI.....	9
- Stock Ticker	9
- Time Period.....	9
- Interval	9
- Risk Horizon (Days):	9
5.2 Backend.....	9

-	Agent.....	10
-	Tools	10
-	Model	10
5.3	External 3 rd party API.....	11
5.4	Component and Library	11
-	RAG Database	11
-	Tracing and Monitoring tools.....	11
6.	High-level Program Code Structure.....	11
7.	Implementation	13
7.1	Agent Collaboration Workflow	13
-	Data Agent	13
-	Sentiment Agent.....	13
-	Valuation Agent.....	15
-	Fundamental Agent.....	17
-	Risk Agent	18
-	Writer Agent	18
7.2	Agent Debate Model Workflow.....	19
8.	Results & UI	23
9.	Testing against requirements	30
9.1	Agent collaboration, debate & GUI	30
9.2	Testing the accuracy of the recommendation (Integration Test)	31
10.	Discussion.....	33
11.	Remarks	34
12.	Conclusion & Future Work.....	35
13.	Declaration of GenAI Usage.....	35
14.	References.....	36
15.	Appendix.....	36
	Team Individual Contribution Split.....	36

Table of Figures

Figure 1: Agent definition.....	5
Figure 2: FinGPT Framework.....	5
Figure 3: Database ChromaDB Structure	8
Figure 4: Our Product Architecture Overview.....	8
Figure 5: Backend.....	9
Figure 6: Agents.....	10
Figure 7: Tools.....	10
Figure 8: External 3rd Party API	11
Figure 9: Components and Libs.....	11
Figure 10: High-Level Code Structure	12
Figure 11: Collaboration Agent Model and Debate Agent Model	13
Figure 12: Sample Output of the stock price history data retrieving Function.....	13
Figure 13: Sample output of the news data retrieving function.....	13
Figure 14: System Prompt of Sentiment Agent	14
Figure 15: System Prompt of Sentiment Agent	15
Figure 16: System Prompt of Valuation Agent.....	16
Figure 17: User Prompt of Valuation Agent.....	16
Figure 18: System Prompt of Fundamental Agent	17
Figure 19: User Prompt of Fundamental Agent.....	18
Figure 20: Orchestration of Debate Model	19
Figure 21: debate conversation flowchart.....	22
Figure 22: Debate Model Output of Final Recommendation	23
Figure 23: UI of TSLA Stock input.....	24
Figure 24: UI for AAPL stock input.....	25
Figure 25: UI for GOOGL Stock input.....	26
Figure 26: Debate agent transcript for ticker symbol TSLA	28
Figure 27: Complete LangGraph architecture in the product	29
Figure 28: Collaboration Workflow.....	30
Figure 29: Integration Test Case Output Log (1).....	32
Figure 30: Integration Test Case Output Log (2).....	32
Figure 31: Failed Test Cases Log	32
Figure 32: Command for running integration test alone.....	32

1. Introduction

Companies sometimes have underlying risks that can be analysed from stock price information and their annual reports. Furthermore, the stock price could be volatile, and key risk analyses from companies could generate essential information and provide a buy, hold, or sell recommendation.

With the rapid growth of AI Agent technology, the results of analysing companies' risks would be more accurate. The LangChain and LangGraph framework is implemented in the product. The agent collaboration workflow and debate workflow are implemented and tested using the OpenAI model (GPT-4o), Ollama model (Llama3.2:3B) and the following: fundamental agent, sentiment agent, valuation agent, risk agent, data agent, and writer agent. These agents collaborated, through the collaboration workflow and the debate workflow, to provide the company's risk report and provide the information of buy, hold or sell.

In our product, multi-AI agents can generate a PDF risk report for the company based on users' input. And in the report, it will first recommend whether to buy, hold, or sell this company's stock. Afterwards, it provides a detailed risk analysis, recording financial performance, balance sheet strength, profitability, debt, valuation, and long-term business potential. Tests were conducted using data from different companies, and the risk information in the PDF file were credible.

Product GitHub Link

<https://github.com/GeorgeLuther25/multi-agent-finance-risk-analysis>

2. Literature review

The development of Natural Language Processing (NLP) has undergone significant development in recent years. Rule-based approaches and statistical models, such as the N-gram and the Hidden Markov Model, began the development of NLP. Furthermore, word embedding models such as Word2Vec (Mikolov et al., 2013) and GloVe introduce an efficient approach to represent word semantics in vector form. After that, a huge development occurred with the introduction of the Transformer Architecture (Vaswani et al., 2017), which became the basis for models such as BERT (Devlin et al., 2019) and GPT. Nowadays, modern models such as GPT-4/5, Claude, and Gemini are able to understand context and produce text with excellent quality through large-scale pretraining with a huge number of parameters.

A. Agentic AI

With the emergence of various modern models, it has also increased expectations for further utilization of the model that has been built, one of which is to embed the LLM model in an entity called an agent. The definition of an agent, according to (Russell et al., 1995) is "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors". In the context of NLP, agents can have the ability to process text from the

environment, plan to achieve goals, and execute tools to produce more comprehensive action responses.

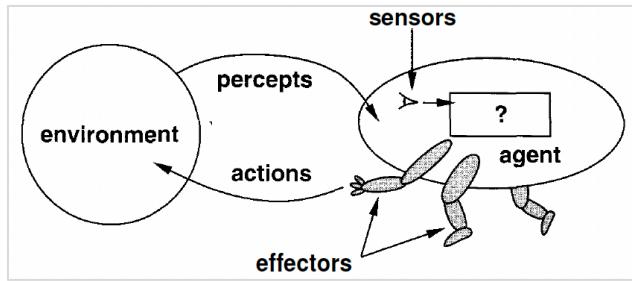


Figure 1:Agent definition

Not only in the form of a single agent, the development of multi-agent AI also promises significant things. One of the benefits is to overcome the limitations of a single agent by enabling modular development, allowing collaboration between agents with diverse specialties, and contextual learning according to the implementation domain. For example, in the financial and investment domains, multiple agents can argue, coordinate, and criticize each other's arguments to produce more accurate and balanced recommendations for the analysis of a stock and produce balanced investment recommendations.

B. Existing Products

Some of the existing products that utilize an agent-based approach include ChatGPT from OpenAI. ChatGPT is a GPT-model-based system built to perform natural language processing across domains. The advantage of this system is its ability to understand the context of the conversation, utilize the tools it has, and produce natural text that is quite accurate. However, because the system is a general domain, its performance can be considered less than optimal compared to models that are specifically built on a specific domain. In the financial domain, there is an existing model developed by Bloomberg, namely BloombergGPT (Wu et al., 2023). This LLM can analyze financial texts such as financial statements, market news, and stock analysis. However, this model is trained on proprietary financial data, so that although it has advantages in the financial context, it is closed source. To democratize the LLM model, one of the open-source products was developed by (Liu et al., 2023), namely FinGPT.

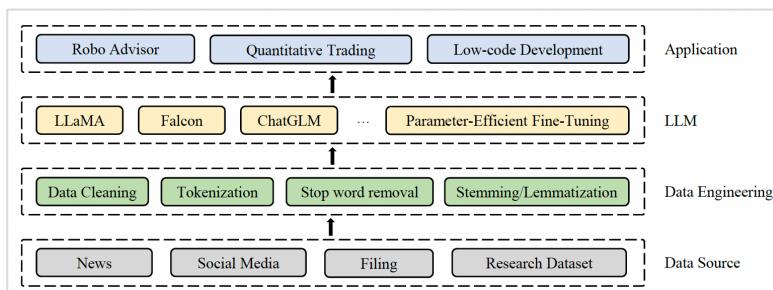


Figure 2: FinGPT Framework

FinGPT is built as a framework with the ability to handle various tasks in the financial domain. FinGPT comprises several layers— Data Source, Data Engineering, LLM, and Application —making it a pipeline that can be extended for specialisation and more efficient use in the financial domain.

C. Gaps in the existing system

There are several gaps found based on the study of the existing system, including:

- Single LLM utilisation: Most products use only one model/agent, limiting model specialisation to having only one analysis perspective. This can lead to bias and a lack of objectivity in conclusions. Product development with multi-agent support becomes a bridge to sharpen analysis in specific domains using various necessary perspectives.
- Data Limitation and Proprietary Model: Products with proprietary models like BloombergGPT are limited to training data that is private in nature. This is because financial data is generally sensitive and rarely available to the public model.
- Multi-Agent Debate System: There are fewer options for products on the market that combine a multi-agent-based debate mechanism with the LangGraph framework, and are open-source. This kind of system has great potential in improving the transparency and accuracy of financial analysis with orchestration and tuning.

Based on these gaps, the products developed in this project aim to pave the way for the development of multi-agent products in the financial domain using LangGraph framework. One of the papers that is used as a reference for multi-agent development in this project is AlphaAgents: Large Language Model-based Multi-Agents for Equity Portfolio Constructions (Zhao et al., 2025), developed by BlackRock, Inc., and is proprietary. Taking the idea of the approach used by AlphaAgent, we developed this project with several adjustments to the approach, including:

- Adjustment of the orchestration design of the agent collaboration model, AlphaAgents use a group chat agent as the leader in the agent collaboration model. However, in this project, the agent collaboration structure is arranged sequentially to simplify the process and ease of changing the composition of the agents.
- The agent orchestration of AlphaAgents uses the AutoGen framework, while the agent orchestration on this project uses the LangGraph framework, both of which support multi-agent workflows. However, LangGraph was chosen because of its better ecosystem support.
- Changes in the design of the debate model. The process of routing and handoff debates in this project is carried out by one debate manager using conditional routing to maintain a more controlled debate process. There is also a maximum debate round parameter to overcome the possibility of continuous iterations in finding consensus.

3. Requirements of the prototype system

3.1 Functional Requirements

The functional requirements below are necessary during prototype system development.

- **Sentiment Agent:** The sentiment agent must be able to provide correct and trustworthy insights on market impact and financial implications.
- **Valuation Agent:** The valuation agent must be able to compute valuation metrics and give insights into valuation analysis.
- **Fundamental Agent:** The fundamental agent must be able to retrieve relevant documents from the vector database and perform analysis on them.
- **Agent Collaboration:** The program must be able to run all the agents in a collaboration workflow, each giving their individual insights based on their specialisations.
- **Agent Debate:** The program must be able to hold debate rounds, where a debate manager supervises the rounds and each debate agent gives their opinion.
- **Display results on GUI:** The results and insights obtained from multi-agent collaboration and debate workflows must be displayed on the GUI appropriately.
- **Download report PDF file:** The final results obtained after debate rounds must be downloadable by users in the form of one PDF file.

3.2 Non-functional Requirements

The non-functional requirements below can improve the system's quality if implemented during prototype system development.

- Execution time must be below 3 minutes
- The program must pass at least 50% of the integration test cases

4. Database Design

As the project uses Retrieval Augmented Generation (RAG) to acquire context for Large Language Models (LLMs), a vector database was implemented using ChromaDB. It is an open-source database widely used in RAG/ AI-integrated systems for storing and searching embeddings. It is easy to use, developer-friendly and handles millions of embeddings efficiently. The database includes rich metadata schemas that capture essential financial document attributes such as ticker symbols, filing types, fiscal year and month ranges. The metadata-driven approach enables sophisticated filtering, and data retrieval can be performed by month and year. To ensure data gets retrieved on every execution, documents from at least three months ago are retrieved. It was also designed to handle two different file collections — one for OpenAI Embeddings and one for Ollama Embeddings. Recursive Character Text Splitter was used during document ingestion to maintain document context through overlapping chunks. Currently, the 10Q SEC filing documents from Apple, Google and Tesla

are saved in a folder, which are ingested into ChromaDB on first program execution. The data is then retrieved by the fundamental agent.

id	page_label	key	string_value	int_value	float_val	bool_val
1		producer	EDGRpdf Service w/ EO.Pdf 22.0.40.0			
2		ingestion_date	2025-11-04T16:52:16.946793			
3		creator	EDGAR Filing HTML Converter			
4		total_pages	NULL	29		
5		filing_year	NULL	2023		
6		source	./data/filings/AAPL-10Q-Q3-4-6-2023.pdf			
7		subject	Form 10-Q filed on 2023-08-04 for the period endin...			
8		filing_start_month	NULL	4		
9		ticker	AAPL			
10		filing_type	10Q			
11		page	NULL	0		
12		title	0000320193-23-000077			
13		chroma:document	UNITED STATES SECURITIES AND EXCHANGE COM...			
14		author	EDGAR Online, a division of Donnelley Financial Solut...			
15		filing_end_month	NULL	6		
16		creationdate	2023-08-04T06:02:09-04:00			
17		moddate	2023-08-04T06:02:23-04:00			
18		keywords	0000320193-23-000077; 10-Q			
19	2	filing_start_month	NULL	4		
20	2	title	0000320193-23-000077			
21						

Figure 3: Database ChromaDB Structure

5. System Architecture

Here is the architecture of the product we built:

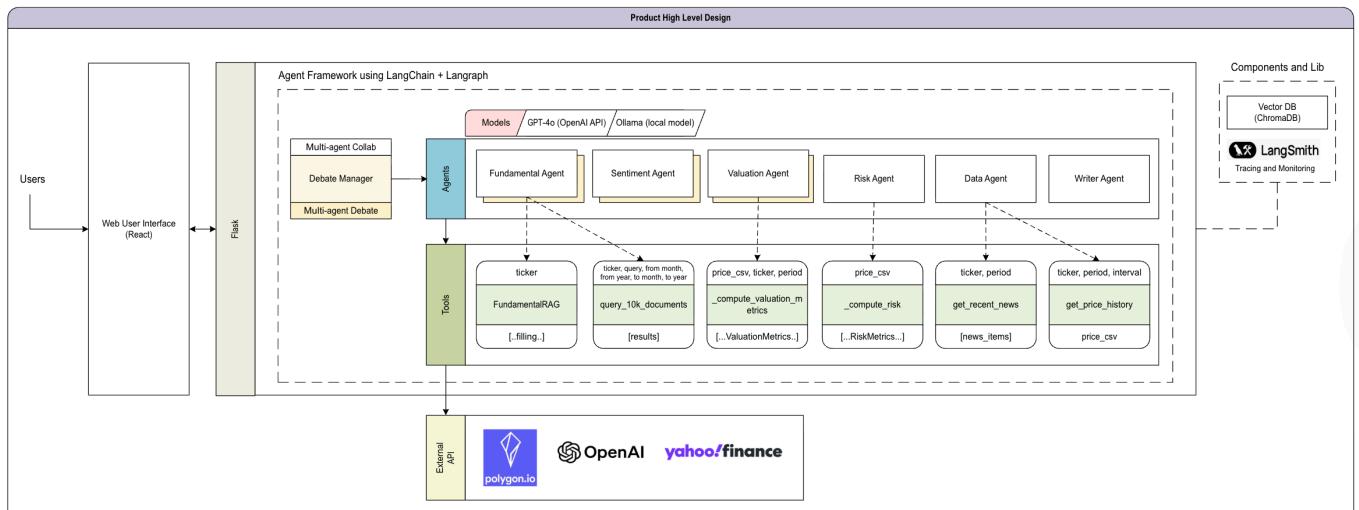


Figure 4: Our Product Architecture Overview

The architectural drawing above consists of several main parts, namely:

5.1 Frontend/UI

From the user's perspective, the product is developed as a web-based application. The Web User interface is designed to provide the flexibility that users need regarding the analysis needs that can be done on the application, including inputs in the form of stock codes (tickers), time periods, intervals, and risk horizons, each of which has the following functions:

- **Stock Ticker:** The ticker symbol of the stock being assessed (i.e., GOOGL, AAPL)
- **Time Period:** The amount of time we look back in the past to gather data for the agents to use (i.e., 3 months).
- **Interval:** How granular we'd like our time period to be, where if we choose a lower interval, we get more datapoints on the stock price (i.e., 1-day).
- **Risk Horizon (Days):** The risk of holding the stock for the specified days, where holding the stock for a longer period is generally safer.

For output, the application will display the results of the analysis on the web page, as well as in the form of a downloadable PDF file. The results of the test on some sample inputs will be described in the Results section. As for the technical side of application development, the entire frontend is built using ReactUI.

5.2 Backend

As a major part of the core logic of the application, backend development is carried out using the Python programming language to facilitate compatibility with the models and frameworks used. To bridge the process flow starting from user input on the frontend with the logic process on the backend side, an internal service API is created on the backend side using a library flask to provide access to the main functions of the application, including the '/api/analyze' function, the system availability check function '/api/health', and the configuration of the LLM model used 'api/models'.

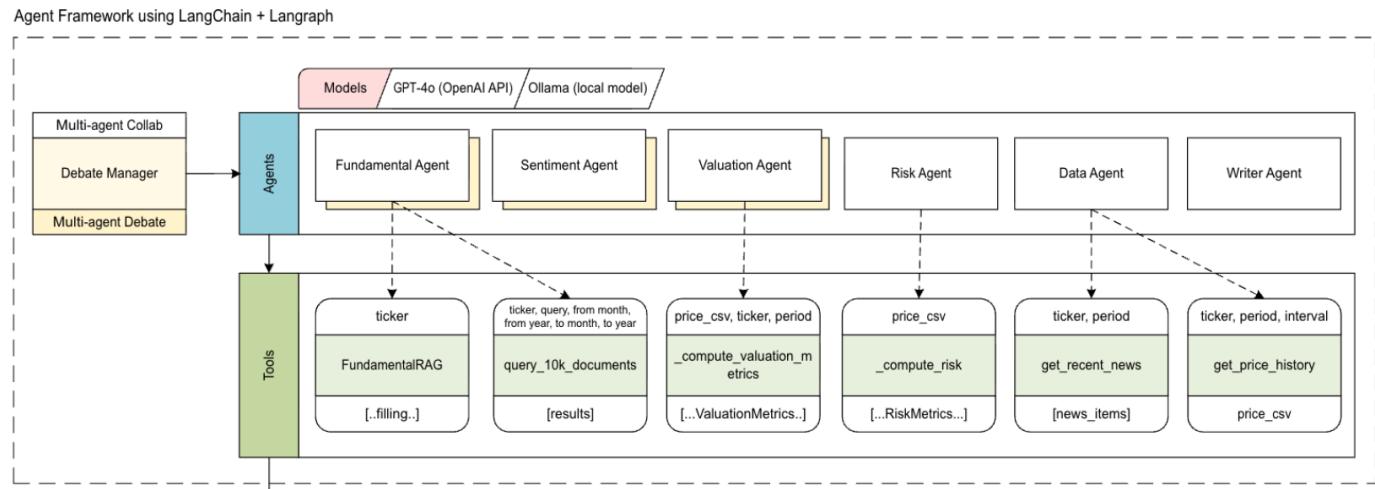


Figure 5: Backend

Furthermore, in the core part of the backend side, there are important components, including:

- **Agent**

In this section, there are several types of agents that specialise in different prompts and capabilities. There are two types of multi-agent models that are implemented, namely the collaboration model and the debate model. Each of these types of multi-agent models uses a different pattern of relationships between agents. The collaboration model uses a chain-based relationship pattern, while the debate model uses a one-to-many model (debate led by a debate manager). Specialist agents that are part of the collaboration model are fundamental agents, sentiment agents, valuation agents, risk agents, data agents, and writer agents. As for the debate model, it is a debate manager, a fundamental debate agent, a sentiment debate agent, a valuation debate agent, and a writer agent.

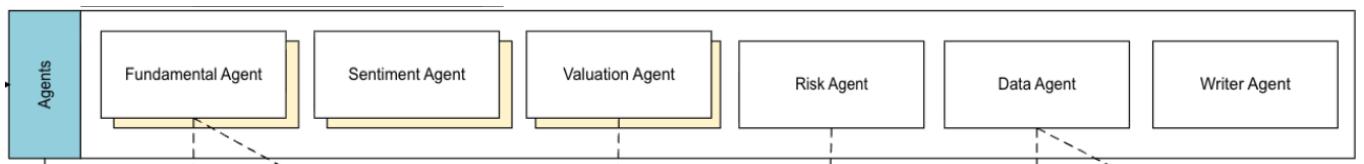


Figure 6: Agents

- **Tools**

Several tools are defined to be used by agents who need them in the analysis process, including FundamentalRAG to process data fillings, query_10k_document to find information from published financial statements (which have been processed by FundamentalRAG), and _compute_valuation_metrics to produce valuation analysis of a stock. _compute_risk used to produce several risk measurement metrics, get_recent_news to find information related to the news of a stock, and get_price_history as a tool used to find historical data in the form of stock price data from time to time.

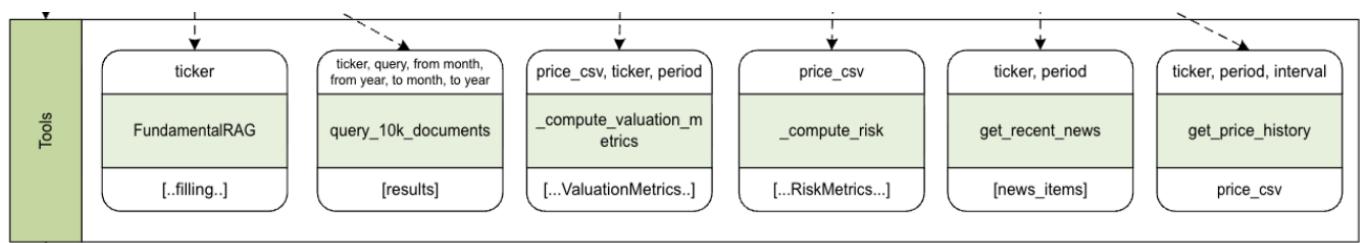


Figure 7: Tools

- **Model**

In this product, the use of the model can be customized. During product development, we used several models, including **OpenAI's GPT-4o** model and Ollama's local model, **Llama3.2:3B**. Based on the results of the tests conducted, the GPT-4o model has the best compatibility among other models.

5.3 External 3rd party API

In order to support the defined tools, several external 3rd party APIs are used, namely: yfinance is used to obtain information related to stock price movements. Polygon.io is used to find the latest news information related to a stock, as well as the OpenAI API to access the GPT-4o model.



Figure 8: External 3rd Party API

5.4 Component and Library

Some of the supporting components of the built product are:

- **RAG Database**

This component is used to store data fillings that have been converted into embedding in vector form, as well as to facilitate the retrieval process to complete the LLM context. We use the ChromaDB database in the implementation of this product.

- **Tracing and Monitoring tools.**

A monitoring and tracing tool from LangChain, LangSmith, is used to observe agent performance and debugging. LangSmith is very useful for seeing the full conversation agent at the collaboration stage and debate stage.

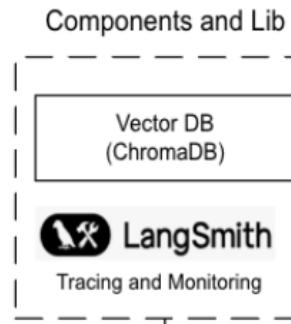


Figure 9: Components and Libs

6. High-level Program Code Structure

As illustrated in the architecture diagram (high level product design), in general, applications consist of a frontend/UI, internal service API (flask), and core program (agents). The use of internal service APIs makes the application flexible and scalable. UI development can be done loosely with product logic, for example, in the future the UI will be developed in mobile form, or the product will be deployed in containers such as docker, Kubernetes or cloud-based platforms such as AWS, Azure, etc.

The following diagram illustrates the source code structure of the built project.

multi-agent-finance-risk-analysis (source code dependency)

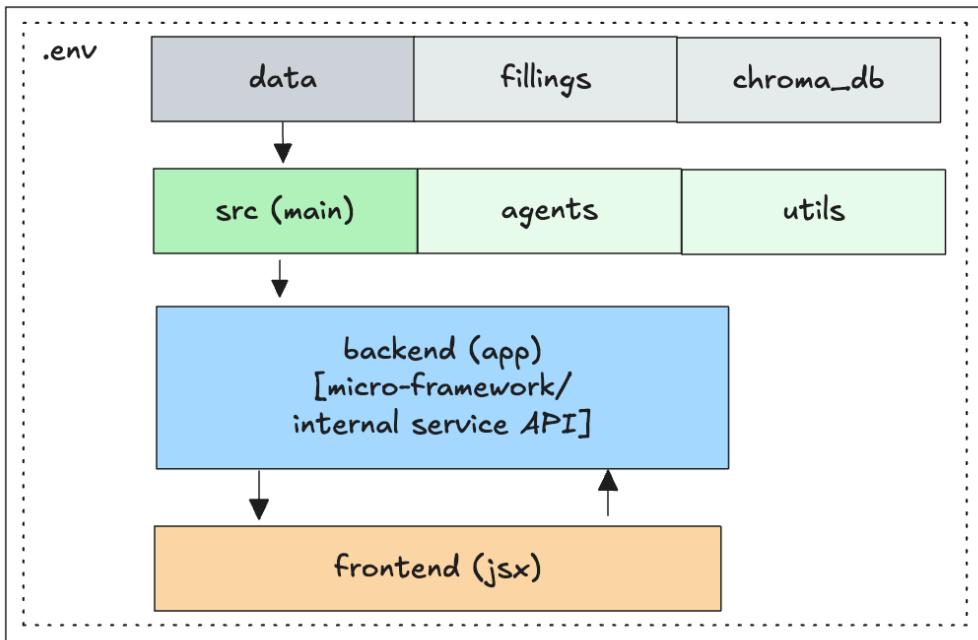


Figure 10: High-Level Code Structure

The explanation of each part of the source code is:

- **Data** is a place to store fillings before being processed into a database vector, and `chroma_db` is the location where data is stored, including metadata from the ChromaDB database used.
- **Src** is the main source code where agents are defined from prompting to workflow. The `utils` folder contains schema definitions (input/output structure), tools, and RAG functions.
- **The backend**, where the service route is defined to connect the frontend with the application's core logic.
- **The frontend**, where the application user interface is built using React.

In addition to the source code, there is an `.env` file for project configuration parameters. Some of the configurations required include `POLYGON_API_KEY` which is the API key to access the polygon service, `OPENAI_API_KEY` which is the API key to access the OpenAI model, `LANGCHAIN_API_KEY` which is the API key to connect the program with the LangSmith service, `OLLAMA_MODEL` to define the type of local model to be used, and `MODEL_PROVIDER` to configure the method of selecting the type of model to be used. Two model providers cannot be used at once - either OpenAI model or Ollama model must be chosen.

7. Implementation

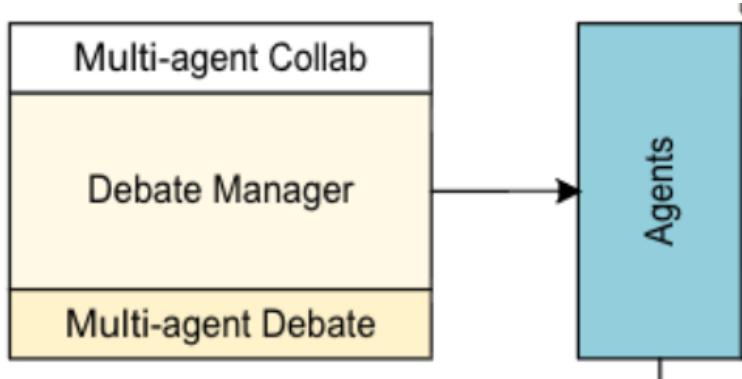


Figure 11: Collaboration Agent Model and Debate Agent Model

7.1 Agent Collaboration Workflow

The agent collaboration workflow was implemented as shown below. Each agent is further explained below.

- Data Agent

The data agent performs the retrieval of news data and stock prices using different functions without calling LLM. The stock price history data are obtained using the YFinance API, and the news data are obtained using the Polygon API, both of which are free API services. The sample output of the stock price history data retrieving function is shown below.

```
1 Date,Adj Close,Close,High,Low,Open,Volume
2 2025-08-07,196.34439086914062,196.52000427246094,197.5399932861328,194.3300018310547,197.05999755859375,26321800
```

Figure 12: Sample Output of the stock price history data retrieving Function

The sample output of the news data retrieving function (content truncated) is shown below. Six latest relevant news were obtained through the API.

```
1 {
2   "date": "2025-11-07",
3   "headline": "My Advice? Don't Get Distracted by Oracle Stock's Latest Slump.",
4   "content": "It's been a wild couple of months for Oracle's (ORCL 2.60%) stock investors. Oracle stock surged 36% on Sept. 10 in response to a blockbuster cloud deal with OpenAI and a five-year ...\\n\\nKey topics: cloud computing, AI, enterprise services, stock investment",
5   "sentiment": "neutral",
6   "source": "Polygon.io (The Motley Fool)",
7   "author": "Daniel Foelber",
8   "url": "https://www.fool.com/investing/2025/11/07/buy-dip-oracle-growth-stock-sell-off/?source=iedfolrf0000001",
9   "ticker": "GOOGL",
10  "extracted_full_content": True,
11 }
```

Figure 13: Sample output of the news data retrieving function

- Sentiment Agent

Sentiment agent performs the sentiment analysis of the stock price data using an LLM. The system prompt is shown below. The agent is asked to summarise, criticise, refine and conclude with an overall sentiment analysis with every execution. A few key factors, such as market impact & relevance, sentiment indicators, financial implications and credibility of the source, are also considered, and the required elements to include in the final output are also stated.

```
7 SENTIMENT_SYSTEM = """As a sentiment equity analyst your primary responsibility  
8 is to analyze the financial news, analyst ratings and disclosures related to  
9 the underlying security, and analyze its implication and sentiment for  
10 investors or stakeholders. You analyze financial news using reflection-enhanced  
11 prompting.  
12  
13 Your task is to:  
14 1. SUMMARIZE: First, provide a concise summary of each news item  
15 2. CRITIQUE: Evaluate the quality and relevance of your summary  
16 3. REFINE: Improve your analysis based on the critique  
17 4. CONCLUDE: Provide an overall sentiment analysis and investment recommendation  
18  
19 For each news item, consider:  
20 - Market impact and relevance  
21 - Sentiment indicators (positive, negative, neutral language)  
22 - Financial implications  
23 - Credibility of the source  
24  
25 Your final output should include:  
26 - Overall sentiment (bullish/bearish/neutral)  
27 - Confidence score (0.0-1.0)  
28 - Investment recommendation with reasoning  
29 - Key insights from the news analysis  
30  
31 Use reflection to ensure your analysis is thorough and well-reasoned."""
```

Figure 14: System Prompt of Sentiment Agent

The user prompt is shown below. The news data is inserted into the prompt and the agent is asked to perform analysis for the next set number of horizon days. The prompt also asks the agent to add json data format at the end of the report, which is then parsed and stored in State object that gets passed to other agents.

```

171 | Analyze the following news items for {state.ticker} using reflection-enhanced prompting:
172 |
173 | NEWS DATA:
174 | {news_content}
175 |
176 | PROCESS:
177 | 1. SUMMARIZE: First, provide a concise summary of each news item and its potential market impact.
178 | 2. CRITIQUE: Evaluate your summary – is it comprehensive? Are you missing key insights? What biases
179 | might be present?
180 | 3. REFINE: Based on your critique, improve and refine your analysis.
181 | 4. CONCLUDE: Provide your final assessment.
182 |
183 | Your response should also include:
184 | - Overall sentiment analysis (bullish/bearish/neutral)
185 | - Confidence level (0.0 to 1.0)
186 | - Key insights and reasoning
187 | - Investment recommendation with clear rationale
188 |
189 | Provide a concise summary along with an informed recommendation on whether to invest in this stock for
190 | the next {state.horizon_days} days.
191 |
192 | IMPORTANT: Also include the structured json data like the example below, with its values replaced by
193 | your analyzed results, and the exact text STRUCTURED DATA.
194 | STRUCTURED DATA
195 | ````json
196 | {
197 |     "ticker": "...",
198 |     "news_items_analyzed": number of news items here,
199 |     "overall_sentiment": "...",
200 |     "confidence_score": "...",
201 |     "investment_recommendation": "...",
202 |     "key_insights": list of key insights of each news item,
203 |     "methodology": "LLM-based reflection-enhanced summarization"
204 | }
205 | ````
```

Figure 15: System Prompt of Sentiment Agent

- Valuation Agent

A valuation agent uses an LLM with a tool that uses a function to calculate valuation metrics from stock price history and perform analysis on that data. The system prompt is shown below. The agent is prompted to focus its analysis on price trend, volatility regime, risk-return evaluation, investment implications, key patterns, and inflection points.

```

29 VALUATION_SYSTEM = """As a valuation equity analyst, your primary
30 responsibility is to analyze the valuation trends of a given asset or portfolio
31 over an extended time horizon. To complete the task, you must analyze the
32 historical valuation data of the asset or portfolio provided, identify trends
33 and patterns in valuation metrics over time, and interpret the implications of
34 these trends for investors or stakeholders. You also have access to a tool that
35 can help you compute valuation metrics. Your analysis should be based solely on
36 the information that you retrieve using this tool. You can interact with this
37 tool using natural language queries. The tool will understand your requests and
38 return relevant outputs.
39
40 Focus your analysis on:
41 1. Price trend analysis (upward, downward, sideways movement)
42 2. Volatility regime assessment (low, medium, high volatility periods)
43 3. Risk-return profile evaluation
44 4. Investment implications and outlook
45 5. Key patterns and inflection points in the data
46
47 Provide clear, actionable insights based on the computational metrics provided.
48 """

```

Figure 16: System Prompt of Valuation Agent

The user prompt is shown below. The agent is prompted to provide valuation analysis for the given ticker symbol, stock price history, and period over the next given horizontal days.

```

2 analysis_prompt: str = f"""
3 Compute the valuation metrics for {state.ticker}, price_csv:{state.market.
price_csv}, period:{state.period}.
4 Provide enhanced trend analysis and investment implications for the next {state.
horizon_days} days based on these metrics.
5 """

```

Figure 17: User Prompt of Valuation Agent

The tool for computing valuation metrics provides the following:

- Cumulative return: the total percentage change in stock price from the beginning to the end of the analysis period. It measures the overall gain or loss without considering time. Calculated as $(\text{end_price} - \text{start_price}) / \text{start_price}$
- Annualised return: the compound annual growth rate that normalises returns across different time periods. It uses the formula $((1 + \text{cumulative_return})^{(252/n)} - 1$, where 252 represents trading days per year. This allows meaningful comparison between investments held for different durations - a 10% return over 6 months is very different from 10% over 2 years when annualised.
- Daily volatility: the standard deviation of daily price returns, measuring day-to-day price fluctuation intensity and shows how much daily returns deviate from their average.
- Annualised volatility: daily volatility scaled up to represent annual risk levels using the square root of time rule: $\text{daily_volatility} \times \sqrt{252}$. This standardizes risk measurement

across different time horizons, making it easier to compare volatility between stocks or assess against annual risk tolerance thresholds.

- Price trend: categorical classification of overall price direction during the analysis period (Upward: more than 5% increase, Downward: more than 5% decrease, Sideways: within ±5%).
- Volatility regime: risk categorisation system based on annualised volatility levels (Low: <15%, Medium: 15~30%, High: >30%).
- Valuation insights: Description of observation derived from other calculated values.
- Trend analysis: summary combining price direction and volatility regime into a coherent statement.
- Risk assessment: Final rule-based evaluation that describes the overall investment attractiveness based on the calculated values.

- Fundamental Agent

The fundamental agent is used for analysing the 10-K/10-Q SEC filing reports of the given company and providing its insights. The system prompt is shown below. The agent is prompted to base its analysis solely on the SEC filings it retrieved from the ChromaDB vector database.

```
40 FUNDAMENTAL_SYSTEM = """As a fundamental financial equity analyst your primary
41 responsibility is to analyze the most recent 10-K/10-Q SEC filings report
42 provided for a company.
43 You have access to a powerful tool that can help you extract relevant
44 information
45 from the 10-K/10-Q SEC filings. Your analysis should be based solely on the
46 information that you
47 retrieve using this tool. You can interact with this tool using natural language
48 queries. The tool will understand your requests and return relevant text
snippets| You, 2 weeks ago via PR #8 • refactor: mv util files to utils/; c
and data points from the 10K document. Keep checking if you have answered the
users' question to avoid looping.
48 """
```

Figure 18: System Prompt of Fundamental Agent

The user prompt is shown below. The agent uses LLM and tool call functionality to retrieve the relevant 10-K/10-Q SEC filing documents from the vector database according to the given ticker symbol, time period and the following query string - financial metrics, business segments, risk factors, competitive position, growth prospects, investment thesis, concerns and risks. Then the data was used by the LLM as context to perform analysis, with the structured json data included at the end.

```

287      f"""
288      Please analyze {state.ticker} using the 10-K/10-Q documents from month:
289      {from_month}, year:{from_year} to month:{to_month}, year:{to_year}.
290      When calling the query_10k_documents tool, pass your queries as a comma-separated
291      string like this: "financial metrics, business segments, risk factors, competitive
292      position, growth prospects, investment thesis, concerns and risks"
293      IMPORTANT: Also include the structured json data like the example below, with its
294      values replaced by your analyzed results, and the exact text STRUCTURED DATA.
295      Ensure the response is valid JSON and keep numeric scores between 0 and 10.
296      STRUCTURED DATA
297      ```json
298      {
299          "executive_summary": "...",
300          "key_financial_metrics": {"metric": "value", ...: "..."}, ...
301          "business_highlights": [...],
302          "risk_factors": [...],
303          "competitive_position": "...",
304          "growth_prospects": "...",
305          "financial_health_score": financial health score here,
306          "investment_thesis": "...",
307          "concerns_and_risks": [...]
308      }
309      ```
310      """

```

Figure 19: User Prompt of Fundamental Agent

- Risk Agent

The risk agent performs risk analysis based on the given stock price history using a function that calculates risk metrics without calling LLMs. The function for computing risk analysis calculates the following:

- Annualized volatility: daily volatility scaled up to represent annual risk levels using the square root of time rule: $\text{daily_volatility} \times \sqrt{252}$.
- Maximum drawdown: the largest peak-to-trough decline in portfolio value, representing the worst possible loss an investor would have experienced. Uses the equation $((\text{close} / \text{np.maximum.accumulate}(\text{close})) - 1).\text{min}()$
- Daily Value at Risk 95%: the maximum expected daily loss with 95% confidence, assuming normal distribution of returns. Uses the equation $-(\mu + \sigma * 1.645)$
- Sharpe-like Ratio: risk-adjusted return measure comparing mean return to volatility. Uses the equation $\mu / (\sigma * \text{np.sqrt}(252))$. Higher values indicate better risk-adjusted performance.

- Writer Agent

The writer agent combines all the data from other agents and parses them into json data (without using LLMs) which is then used in the debate workflow.

7.2 Agent Debate Model Workflow

One of the main advantages of this product is the existence of an agent orchestration model through a debate mechanism. Here is the orchestration of the debate model that was created.

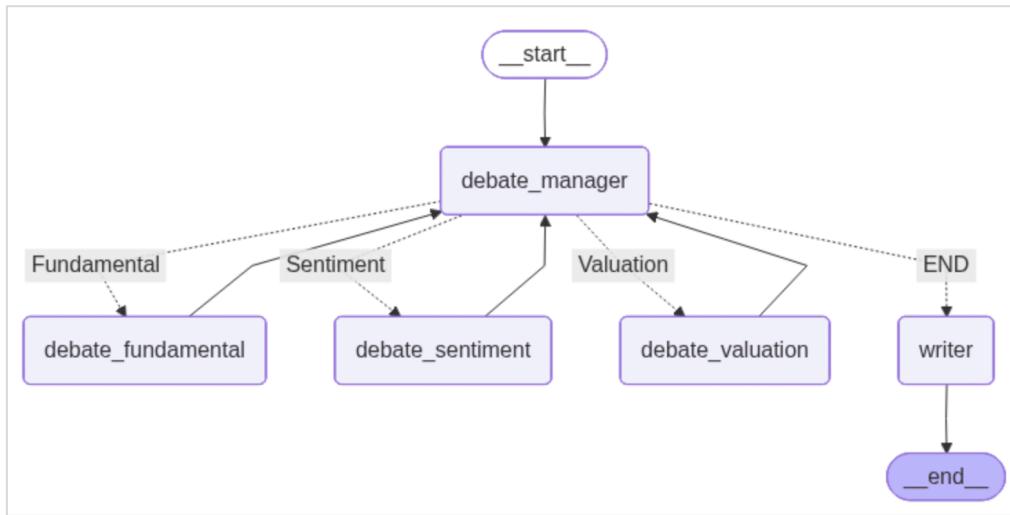


Figure 20: Orchestration of Debate Model

The implementation of this debate model uses a multi-agent approach where each agent has a specific role to evaluate a topic from a different perspective. The process starts with the `__start__` node that initializes the `debate_manager` primary agent. This agent is in charge of regulating the course of the debate by arranging the turns of three sub-agents: `debate_fundamental`, `debate_sentiment`, and `debate_valuation`. Each agent will be allowed to argue based on their respective analysis focus using the arguments generated by other agents. Each time a round is run (one round in which all agents have given their arguments), the results will be concluded by the `debate manager` to determine whether the next round of debate will be run or the debate process will be stopped because consensus has been reached or if the number of rounds has reached a maximum (5 rounds). From the results of the debate, the `writer` agent will then be executed to add the consensus result to the final report and send it to the node `__end__`.

The prompts implemented in the debate manager are shown in the table below. There are three main prompts, namely the SYSTEM prompt, the route prompt used to determine whether the debate is resumed or stopped, and the finalize prompt to smooth the final consensus.

Debate Manager (SYSTEM)
You are the Debate Manager coordinating three agents: Fundamental, Sentiment, and Valuation.
Your task:
<ul style="list-style-type: none">- Carefully read the specialized agents arguments.- Analyze agreements, disagreements, and the overall tone.- Identify key evidence and logic from each.- Synthesize these viewpoints into **one concise, reasoned conclusion**.- The conclusion must be objective, actionable, and justified.
Your output is final conclusion and must follow this concern:
<ul style="list-style-type: none">- Highlight points of agreement or conflict.- Note which arguments are stronger or better supported.- Provide a single, coherent conclusion that integrates all perspectives.- If uncertainty remains, explain it clearly.- Be balanced, analytical, and clear about judgement to invest in the stock within the next {state.horizon_days} days.

<p>Output format:</p> <ul style="list-style-type: none"> - Output is concise summary - Based on your summary, give a recommendation to 'buy', 'hold', or 'sell'.
Debate Manager (route)
<p>Based on both Consensus Summaries, do this action:</p> <ul style="list-style-type: none"> - Compare both Consensus Summaries, and get the final recommendation - Say in this format 'First: {your first recommendation}, Second: {your second recommendation}, Action: {DEBATE or END}' - If both Consensus Summaries have similar recommendations, just fill Action with 'END', else than that say 'DEBATE'.
Debate Manager (finalize)
<p>Refine the Final Consensus into a clear, concise summary suitable for a report.</p> <ul style="list-style-type: none"> - Make it brief and precise. - Do not mention agents or the debate process. - Output only the final polished summary, with no headings or extra text.
Final Consensus: "{new_state.debate.consensus_summary}"

Table 1: Debate Manager System Prompt

As for each specialist agent, the following prompt is defined as a SYSTEM prompt:

Fundamental (debate)
<p>You are the Fundamental Analysis Agent. Specialization: Focus on financial performance, balance sheet strength, profitability, debt, valuation, and long-term business potential. Report: \"\"\"{state.fundamental.executive_summary}\"\\""</p>
Sentiment (debate)
<p>You are the Sentiment Analysis Agent. Specialization: Analyze the tone, emotional language, and implied investor sentiment in a report. Identify whether the sentiment is optimistic, neutral, or negative, and explain why. Report: \"\"\"{state.sentiment.summary}\"\\""</p>
Valuation (debate)
<p>You are the Valuation Analysis Agent. Specialization: Analyze the valuation trends of a given asset or portfolio over an extended time horizon. To complete the task, you must analyze the historical valuation data of the asset or portfolio provided, identify trends and patterns in valuation metrics over time, and interpret the implications of these trends for investors or stakeholders.</p> <p>Focus your analysis on:</p> <ol style="list-style-type: none"> 1. Price trend analysis (upward, downward, sideways movement) 2. Volatility regime assessment (low, medium, high volatility periods) 3. Risk-return profile evaluation 4. Investment implications and outlook 5. Key patterns and inflection points in the data <p>Report: \"\"\"{state.valuation.trend_analysis}\"\\""</p>

Table 2: Each specialist Agent System Prompt

In the first turn, each specialist agent will be given the following prompt to produce an initial argument in the form of an investment recommendation.

Initial analysis prompt (1 st turn):
Task:
- Make investment recommendation analysis for {state.ticker} within the next {state.horizon_days} days, based ONLY your specialization.\n- Based on your analysis, give a recommendation to 'buy', 'hold', or 'sell'.

Figure 22: Each specialist Agent Initial System Prompt

Then, on the second turn and so on, each agent will be given a prompt to justify their argument by considering the arguments of the other agents.

Debate prompt (2 nd turn and next)
You will given other Agents judgements, then:
- Challenge the judgment from your specialisation, don't put a heading for this section.\n- Make investment recommendation analysis for {state.ticker} based ONLY your specialization.\n- Based on your analysis, give a recommendation to 'buy', 'hold', or 'sell'.\n- If the other Agent provides strong evidence that challenges your recommendation, you may revise your recommendation accordingly.

Table 3: Each specialist Agent 2nd round System Prompt

Here is a flowchart that illustrates the debate conversation.

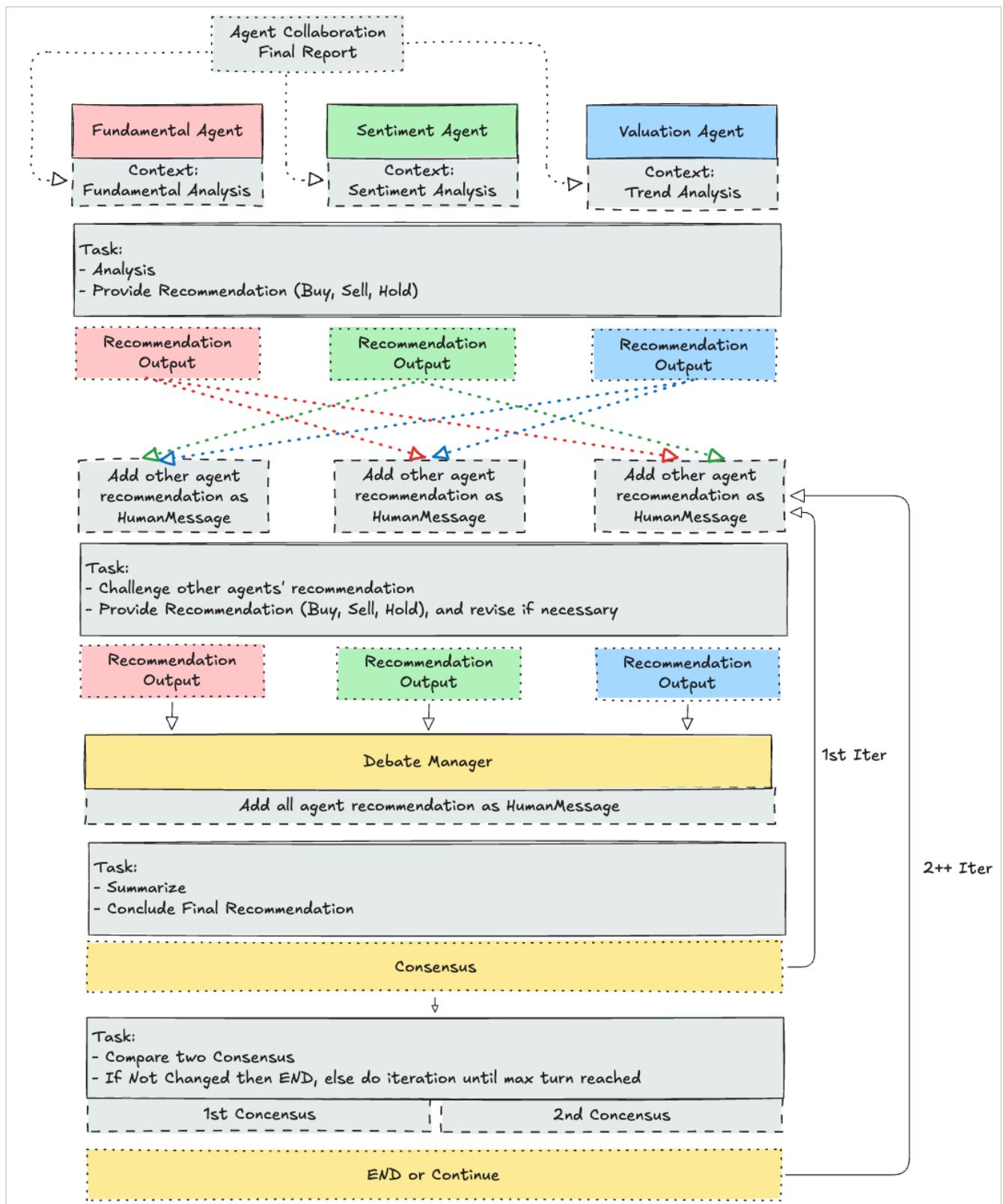


Figure 21: debate conversation flowchart

This debate model will produce an output in the form of an investment final recommendation section which is placed as the final part of the overall analysis report.

Investment Final Recommendation

Tesla's long-term growth potential is recognized due to its leadership in electric vehicles and tech innovation. However, short-term challenges, such as declining China-made vehicle sales and governance issues, necessitate caution. While the valuation suggests potential high returns for risk-tolerant investors, high volatility and mixed sentiment signals lead to a 'hold' recommendation. Existing investors should maintain their positions, monitoring market developments and financial performance. New investors might wait for a more favorable entry point. Risk-tolerant investors may pursue opportunities in current momentum but should manage risks effectively.

Figure 22: Debate Model Output of Final Recommendation

8. Results & UI

This section presents the results of the Multi-Agent Finance Risk Analysis System, an AI-driven product designed to automate stock risk evaluation through a collaborative, debate-oriented framework. The system integrates a Flask backend, LangGraph-based orchestration layer, Retrieval-Augmented Generation (RAG) module for factual grounding, and a React-based frontend for user interaction and report visualization. The product was tested locally across three representative stocks: GOOGL, AAPL, and TSLA, each using a 3-month lookback window, 1-day interval, and a 30-day risk horizon. These parameters were chosen to balance recency, volatility sensitivity, and runtime efficiency, aligning with realistic investor timeframes. The inputs to the system are defined below:

- **Stock Ticker:** The ticker symbol of the stock being assessed (i.e., GOOGL, AAPL)
- **Time Period:** The amount of time we look back in the past to gather data for the agents to use (i.e., 3 months).
- **Interval:** How granular we'd like our time period to be where if we choose a lower interval, we get more datapoints on the stock price (i.e., 1-day).
- **Risk Horizon (Days):** The risk of holding the stock for the specified days, where holding the stock for a longer period is generally safer.

The user initiates an analysis by entering the stock ticker symbol (e.g., GOOGL, AAPL, or TSLA), selecting a time period of three months, an interval of one day, and a risk horizon of thirty days before clicking "Analyze Stock Risk." This triggers the backend endpoint, which constructs and executes the LangGraph pipeline. Once their independent analyses are complete, the debate manager moderates a structured debate to resolve conflicting interpretations and produce a consensus recommendation.

Figures 23-25 showcase the results across the three case studies. For TSLA, the system reported a cumulative return of 43.9% and an annualized volatility of 46.7%, reflecting a

highly dynamic market phase with strong short-term growth but increased risk exposure. AAPL displayed a cumulative return of 33.08% and volatility of 26.1%, aligning with a stable upward trend and no major risk flags. GOOGL achieved a cumulative return of 42.57% with volatility of 28.7%, showing a strong bullish momentum driven by AI and cloud expansion. While these quantitative metrics capture financial dynamics, their value lies in the multi-agent reasoning pipeline that generated them, where data analysis, evidence retrieval, and structured debate converge to form interpretable insights.

Multi-Agent Finance Risk Analysis

Using Model: OpenAI gpt-4o

Analysis Parameters

Describe your request

e.g., "Give me AAPL stock for the last 2 years at intervals of 1 day with a 45 day risk horizon."

[Apply Prompt](#)

Stock Ticker	Time Period	Interval	Risk Horizon (Days)
TSLA	3 Months	1 Day	30

Analyze Stock Risk

[Download Report PDF](#)

CUMULATIVE RETURN

43.90%

ANNUALIZED VOLATILITY

46.7%

PRICE TREND

Upward

RISK FLAGS

1

INVESTMENT RECOMMENDATION

Tesla's long-term growth potential is recognized due to its leadership in electric vehicles and tech innovation. However, short-term challenges, such as declining China-made vehicle sales and governance issues, necessitate caution. While the valuation suggests potential high returns for risk-tolerant investors, high volatility and mixed sentiment signals lead to a 'hold' recommendation. Existing investors should maintain their positions, monitoring market developments and financial performance. New investors might wait for a more favorable entry point. Risk-tolerant investors may pursue opportunities in current momentum but should manage risks effectively.

Figure 23: UI of TSLA Stock input

Figure 23. UI showing input for TSLA stock for a 3-month time period, 1-day interval and 30 day risk horizon using the debate agent mode. A cumulative return of 43.90% is calculated along with an annualized volatility of 46.7% with an upward trend, there is one risk. Additionally, we see an investment recommendation with further detail about the system below this snippet.

Multi-Agent Finance Risk Analysis

Using Model: OpenAI gpt-4o

Analysis Parameters

Describe your request

e.g., "Give me AAPL stock for the last 2 years at intervals of 1 day with a 45 day risk horizon."

Apply Prompt

Stock Ticker	Time Period	Interval	Risk Horizon (Days)
AAPL	3 Months	1 Day	30

Analyze Stock Risk

Download Report PDF

CUMULATIVE RETURN

33.08%

ANNUALIZED VOLATILITY

26.1%

PRICE TREND

Upward

RISK FLAGS

0

INVESTMENT RECOMMENDATION

Apple's robust financial health is widely acknowledged, supporting its capacity to handle economic challenges and pursue growth. However, concerns about its high valuation suggest market expectations may be too optimistic. Competitive pressures, especially from AI-driven rivals, pose a significant risk, with Apple perceived as lagging in AI investments. Mixed sentiment is evident, highlighted by cautious investor actions like Warren Buffett's share sale. Despite these concerns, the strong upward price trend and medium volatility present an attractive risk-return profile. The overall recommendation is to hold, balancing the potential for gains against valuation and competitive risks. Investors should monitor Apple's AI progress and strategic moves to adjust their positions accordingly.

Figure 24: UI for AAPL stock input

Figure 24. UI showing input for AAPL stock for a 3-month time period, 1-day interval and 30 day risk horizon using the debate agent mode. A cumulative return of 33.08% is calculated along with an annualized volatility of 26.1% with an upward trend, there are no risk flags.

Multi-Agent Finance Risk Analysis

Using Model: OpenAI gpt-4o

Analysis Parameters

Describe your request

e.g., "Give me AAPL stock for the last 2 years at intervals of 1 day with a 45 day risk horizon."

Apply Prompt

Stock Ticker

GOOGL

Time Period

3 Months

Interval

1 Day

Risk Horizon (Days)

30

Analyze Stock Risk

Download Report PDF

CUMULATIVE RETURN

42.57%

ANNUALIZED VOLATILITY

28.7%

PRICE TREND

Upward

RISK FLAGS

0

INVESTMENT RECOMMENDATION

Alphabet (GOOGL) presents a compelling long-term growth opportunity due to its strong strategic positioning in AI and emerging technologies, supported by robust financial performance and a solid balance sheet. However, potential overvaluation and market volatility pose short-term risks. While positive market sentiment suggests optimism, caution is advised due to valuation concerns. Thus, the recommendation is to hold the stock, maintaining exposure to its growth potential while monitoring market conditions and valuation dynamics closely.

Figure 25: UI for GOOGL Stock input

Figure 25. UI showing input for GOOGL stock for a 3-month time period, 1-day interval and 30 day risk horizon using the debate agent mode. A cumulative return of 42.57% is calculated along with an annualized volatility of 28.7% with an upward trend, there are no risk flags.

The Valuation Agent examines how the stock price has moved over time, calculating returns and volatility from simple time-series data. The Fundamental Agent, which is supported by the RAG system, pulls real information from company filings and news releases to make sure its analysis is based on facts, not guesses. The Sentiment Agent studies how the news talks about the company, scoring whether the tone is positive or negative, while the Risk Agent checks how risky the stock might be using things like price drops and daily volatility. All of these agents feed their findings into one shared State, which LangGraph then brings together and checks for consistency. Once every agent has finished, the debate workflow is initiated and supervised by the debate manager. This debate runs in a turn-by-turn format, where each agent has to back up its view before the debate manager forms a final decision. For TSLA, this process ended with a “hold” recommendation at 70% confidence, balancing strong public sentiment with caution about market conditions. GOOGL’s agents agreed on a “buy” due to

consistent fundamentals, while AAPL’s agents settled on a “hold” based on mixed but steady growth signals. The debate transcript lets users see exactly how the AI reached its conclusion, turning what would normally be a black-box decision into a clear and explainable process.

Once all agents have produced their assessments, the debate manager initiates a multi-round debate if any conflicts arise. This process, illustrated in **Figure 4**, follows a round-robin structure in which each agent must provide reasoning, counterpoints, and supporting data before a final consensus is reached. For TSLA, this resulted in a 70% confidence “hold” recommendation, balancing positive sentiment with macroeconomic caution. Similarly, GOOGL produced a “buy” decision reflecting strong fundamentals, while AAPL received a “hold” recommendation based on moderate growth and sentiment convergence. The debate transcript allows the user to trace the entire reasoning process.

▼ Debate Transcript

Fundamental Agent

Round 1:

To make an investment recommendation for TSLA within the next 30 days, let's consider the key aspects of fundamental analysis: financial performance, balance sheet strength, profitability, debt, valuation, and long-term business potential.

1. Financial Performance:

- Tesla has shown strong revenue growth over recent years, driven by increased vehicle deliveries and expansion into new markets. However, short-term fluctuations can occur due to supply chain issues or changes in consumer demand.

2. Balance Sheet Strength:

- Tesla has historically maintained a solid balance sheet with significant cash reserves and manageable debt levels. This strength provides a buffer against economic downturns and supports continued investment in growth initiatives.

3. Profitability:

- Tesla's profitability has been improving, with the company achieving consistent quarterly profits. This trend is a positive indicator of operational efficiency and scalability. However, fluctuations in profit margins can occur due to factors like raw material costs or production challenges.

4. Debt:

- Tesla's debt levels are generally considered manageable, especially given its cash flow generation and growth prospects. The company has been able to finance its operations and expansion without taking on excessive debt, which is a positive sign.

5. Valuation:

- Tesla's valuation has historically been high relative to traditional metrics like price-to-earnings (P/E) ratios. While this reflects investor optimism about future growth, it also means the stock could be more volatile and susceptible to market sentiment changes.

6. Long-term Business Potential:

- Tesla is positioned as a leader in the electric vehicle market, with significant investments in battery technology, autonomous driving, and energy solutions. The company's innovation and market position suggest strong long-term potential, but competition is intensifying.

Recommendation: Given Tesla's strong financial performance, manageable debt, and long-term growth potential, but considering the high valuation and potential short-term market volatility, a "hold" recommendation is appropriate for the next 30 days. This stance allows investors to maintain their position while monitoring market conditions and Tesla's ongoing execution of its growth strategy.

Round 2:

The Sentiment Agent suggests a 'hold' recommendation based on a neutral sentiment analysis, highlighting both short-term challenges in China and long-term optimism due to Tesla's position in AI and tech. Meanwhile, the Valuation Agent recommends a 'buy' for risk-tolerant investors, citing strong upward momentum and high volatility, which implies potential for high returns but also necessitates careful risk management.

From a fundamental analysis perspective, let's evaluate these positions:

1. **Financial Performance and Profitability:** Tesla has shown strong revenue growth over the years, driven by increased vehicle deliveries and expansion into new markets. However, profitability has been variable, with margins impacted by production costs and pricing strategies. The recent sales decline in China could affect revenue growth in the short term, though Tesla's ability to manage costs and maintain profitability in other regions will be crucial.
2. **Balance Sheet Strength:** Tesla's balance sheet is relatively strong, with significant cash reserves and manageable debt levels. This financial flexibility allows Tesla to invest in R&D and expand its production capabilities, which is essential for maintaining its competitive edge.

Figure 26: Debate agent transcript for ticker symbol TSLA

Figure 26. The debate agent transcript for ticker symbol TSLA, showing the first two rounds of the debate using a round-robin approach. This highlights the AI systems behaviour to the user.

Finally, **Figure 27** illustrates the complete LangGraph architecture used in this workflow, showing how each agent node connects through directed edges to the Leader Manager node.

This structure allows for concurrent processing, modular scalability, and targeted debate activation.

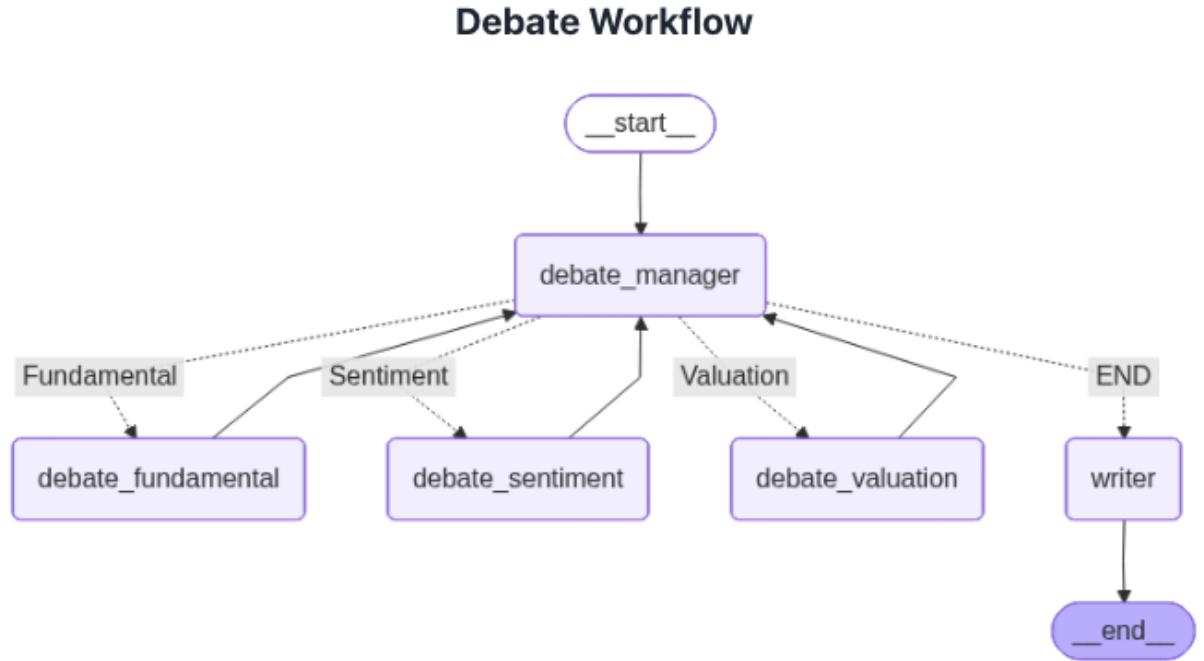


Figure 27: Complete LangGraph architecture in the product

Figure 27. The debate workflow, showing how each agent interacts with each other and the debate manager leading the debate.

Figure 28 highlights the Collaboration Workflow, a sequence to sequence agent orchestration, where there is no debate between agents, just information flow. The Collaboration Workflow is not used for the results in this section, where the Debate Workflow is the final product. Upon completion, the results and debates are compiled into a Markdown-to-PDF report, which is returned to the frontend for download. The report includes executive summaries, valuation overviews, sentiment analyses, debate transcripts, and key insights, providing users with a well-rounded UI for evaluating stocks and their potential.

Collaboration Workflow

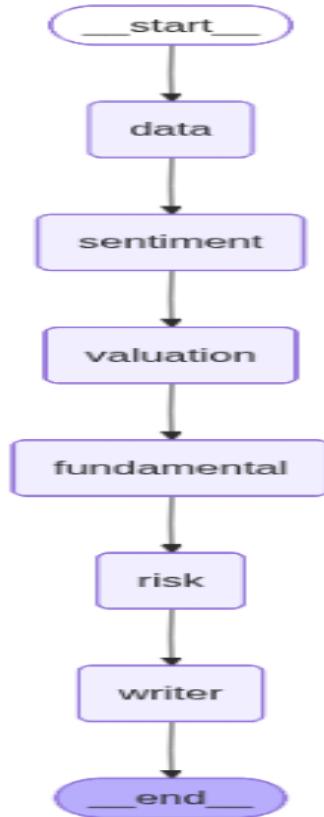


Figure 28: Collaboration Workflow

Figure 28. The collaboration workflow, showing each agent performing in sequence where there is no debate. This model uses LangChain instead of LangGraph for its architecture.

9. Testing against requirements

Each functional requirement was tested to make sure that they were working as expected. The app was executed using ticker symbol=GOOGL, period=3 months, interval=1 day, horizon days=30 from the GUI.

9.1 Agent collaboration, debate & GUI

The agent collaboration workflow and the agents in it correctly executed without any errors when used with the GPT-4o model, producing a State object which was confirmed by LangSmith tracking. All agents were able to follow the given prompt correctly, with sentiment agent summarizing and refining the news articles for shorter but rich insights, with conclusion on overall sentiment, recommendation and structured json output included. The valuation agent LLM correctly called the tool provided to it with the right parameters and correctly calculated the valuation metrics which is then used as context in the LLM to perform analysis. The agent also checked whether it should continue the execution process, call the tool and LLM again or not. All of these processes are automatically performed by `create_react_agent()` functionality. The fundamental agent, like valuation agent, also correctly

called the LLM, the provided tool with the right parameters and used its output as context in its analysis and correctly include structured json data in its output. It called the tool efficiently by passing parameters in a list of strings and calling it once instead of calling tool multiple times for each string. Like valuation agent, it also checked if it should continue and call the tool and LLM or not.

The agent debate workflow also ran correctly without any errors when used with the GPT-4o model. Each debate agent checked all the past messages and correctly provided investment recommendation analysis based on each of their specializations. The debate manager routed to the required debate agent in a round-robin approach, and gave a concise summary with recommendation after each round. It also correctly compared its concise summaries and determined whether the debate should continue or end. As the debate ends, it provided with a final concise summary in a short paragraph as well.

9.2 Testing the accuracy of the recommendation (Integration Test)

To test whether the final recommendation given by the program is accurate enough, a test script is developed using pytest library. The goal is to test by starting from the date from a few months back so that the stock price already available for the next 30 days after that date can be used for validation. A bash script is executed that executes the main program four times with different parameters, save them in src/tests/json folder, and then check the final recommendation with the expected result. The expected result is calculated by comparing the average close price of stock prices during the past month to the next 30 days. If the average close value becomes higher, the expected value is “BUY” or “HOLD”, “SELL” if otherwise. The parameters are set as shown below.

Ticker	Period	Inteval	Horizon Time	End Date
GOOGL	1 month	1 day	30 days	2025-4-1
GOOGL	1 month	1 day	30 days	2025-6-1
AAPL	1 month	1 day	30 days	2025-6-1
AAPL	1 month	1 day	30 days	2023-6-1
TSLA	1 month	1 day	30 days	2025-1-30

After running and testing each test case file, 3 of them passed and 2 of them failed as shown below.

```

integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_AAPL_1mo_2023-6-1.json] ┌ Fetching AAPL price data from 2023-05-01 to 2023-06-01
└ Fetching AAPL price data from 2023-06-01 to 2023-07-01
  Average Closing Price increased/did not change from 172.62272782759234 to 184.28333318801154 before & after 2023-6-1
  ✓ final_state_AAPL_1mo_2023-6-1.json: HOLD = expected: ['BUY', 'HOLD']
PASSED
integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_TSLA_1mo_2025-1-30.json] ┌ Fetching TSLA price data from 2024-12-30 to 2025-01-30
└ Fetching TSLA price data from 2025-01-30 to 2025-03-01
  Average Closing Price decreased from 405.8374969482422 to 348.96809459867933 before & after 2025-1-30
  ✓ final_state_TSLA_1mo_2025-1-30.json: SELL = expected: ['SELL']
PASSED
integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_AAPL_2025-6-1.json] ┌ Fetching AAPL price data from 2025-05-01 to 2025-06-01
└ Fetching AAPL price data from 2025-06-01 to 2025-07-01
  Average Closing Price decreased from 203.94714282807848 to 200.65699996948243 before & after 2025-6-1
FAILED
integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_GOOG_1mo_2025-6-1.json] ┌ Fetching GOOGL price data from 2025-05-01 to 2025-06-01
└ Fetching GOOGL price data from 2025-06-01 to 2025-07-01
  Average Closing Price increased/did not change from 164.37952386765252 to 172.55899963378906 before & after 2025-6-1
  ✓ final_state_GOOG_1mo_2025-6-1.json: HOLD = expected: ['BUY', 'HOLD']
PASSED
integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_GOOG_1mo_2025-4-1.json] ┌ Fetching GOOGL price data from 2025-03-01 to 2025-04-01
└ Fetching GOOGL price data from 2025-04-01 to 2025-05-01
  Average Closing Price decreased from 165.36095174153647 to 154.55809747605096 before & after 2025-4-1
FAILED

```

Figure 29: Integration Test Case Output Log (1)

```

=====
short test summary info =====
FAILED integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_AAPL_1mo_2025-6-1.json] - AssertionError: Invalid test case. Actual recommendation HOLD doesn't match t...
FAILED integration_test.py::test_agent_predicted_correct_recommendation[/Users/kan/important/uts_related/Research_Project/multi-agent-finance-risk-analysis/src/tests/json/final_state_GOOG_1mo_2025-4-1.json] - AssertionError: Invalid test case. Actual recommendation HOLD doesn't match t...
=====
2 failed, 3 passed in 3.17s =====
(multiagent_finance_risk_analysis) kan@Kans-MacBook-Pro multi-agent-finance-risk-analysis %

```

Figure 30: Integration Test Case Output Log (2)

Both failed test cases showed the message below, indicating that the expected value is to sell but the system recommended to hold.

```

E     AssertionError: Invalid test case. Actual recommendation HOLD doesn't match the expected value ['SELL']
E     assert 'HOLD' in ['SELL']

integration_test.py:62: AssertionError

```

Figure 31: Failed Test Cases Log

The output files are saved in src/tests/json folder, which can be tested again using the command:

```

2   cd src/tests
3   pytest integration_test.py -v -s

```

Figure 32: Command for running integration test alone

10. Discussion

A few key points were noticed during program executions and testings.

The implemented system offers some advantages over other products already in the market. Although online chatbot services like chatgpt.com are able to provide insights based on the prompt, stock price history, news and SEC filings (through its web search tool), it is only a single agent. Users also have to type in the prompts manually. Thus, there could be more hallucinations and the LLM confusing in its own chat history as the input data and chat history gets larger. The choice of model is also only within the GPT family. Our platform is designed to handle both GPT and Ollama models. It is also able to run the workflows without much user input and with several agents, each having its own specialization and handling their part of work. And the agents are given insights by other agents to assess, debate and reconsider their opinions, which is something that cannot be done in online chatbot services.

A few challenges were also encountered during project implementation. Although GPT-4o LLMs performed well everytime, Ollama LLMs used (Llama3.2:3b, Llala3.2:7b, Llama3.1) could not accurately follow the prompts given. In most cases, they would output structured json output with different keyword identifiers and they would also sometimes miss or incorrectly give the json key inside as well. Thus, the json parsing function after LLM execution would fail. A more detailed prompt did not solve the problem either. Although this could be mitigated with a more complex json parsing function, a better solution would be to test with a bigger Ollama model. And as input data size becomes bigger (e.g. more than 6 months), even clever models like GPT-4o stops calling the tools provided to them, which was not solved even with some modifications to prompts. Instead of passing such large data into LLM so it would pass it as a tool call parameter, an approach of LLM first calling tool to get these price data and then calling a tool for computation should be considered.

To discuss the overall system, this Multi-Agent Finance Risk Analysis System shows how AI frameworks such as LangGraph can move beyond theory into working, product-level systems. The transition from a LangChain-based sequential pipeline to a LangGraph-based setup was a key step in improving flexibility, speed, and interpretability. Key inspiration was taken from the paper by Zhao, et al. (2025). LangGraph allows each agent to operate as its own reasoning module, and allows using the latest and complex functions. This structure makes the system more dynamic, as agents can reason independently but also collaborate through the debate process. The architecture not only reduces latency but also enables more complex reasoning, where conditional debate logic can be triggered based on the agents' level of agreement.

The debate workflow is what makes the system interpretable and transparent. Rather than averaging answers or relying on a single model output, the debate manager coordinates structured discussions between the Fundamental, Sentiment, Valuation, and Risk agents. Each agent presents its analysis, confidence score, and supporting data from either numerical metrics or text retrieved through the RAG system. The debate manager then facilitates a debate by asking targeted questions and prompting agents to justify or challenge each other's findings. This creates a process that feels more like a rational conversation than a typical AI output. Users can even view the transcript of this debate, which provides a clear record of how each decision was reached and the evidence behind it.

To make sure all analyses are grounded in factual information, the system integrates a Retrieval-Augmented Generation (RAG) subsystem. The Fundamental Agent retrieves real SEC filings to back up its insights. This reduces hallucination and ensures that the AI uses

data directly tied to the company being analysed. For example, the recommendation for GOOGL was supported by retrieved text about AI and Cloud revenue growth, while AAPL’s “hold” rating reflected a balanced view of production updates and sales trends. This grounding in verified data ensures that recommendations are consistent and based on real-world evidence.

From a system design perspective, the Flask backend serves as the coordination layer between the frontend and the LangGraph workflow. It exposes three endpoints: one to confirm the system is running, one to check available model providers, and one to perform the analysis. The analysis route includes a watchdog mechanism that prevents long-running operations from hanging, ensuring that users always receive a response. Model providers are selected dynamically: if an OpenAI key is detected, the system defaults to GPT-4o, otherwise, it runs locally on Ollama (Llama3.2:3B). This flexibility makes the system adaptable to both online and offline environments.

Overall, the system balances advanced AI techniques with a clean, user-focused design. It incorporates concepts such as distributed reasoning, evidence-based debate (using RAG), and graph orchestration in LangGraph while keeping the output simple and interpretable. The combination of these ideas transforms the system from a research experiment into an application. Each analysis runs efficiently, is easy to interpret, and produces a professional-quality report that can be used for presentations or further review. The project demonstrates how modern AI frameworks can be built into usable tools that are both explainable and technically sound.

11. Remarks

The project demonstrates how multi-agent AI systems can be developed into practical, transparent products that produce explainable results. The main technical milestone was the evolution of the system from a simple LangChain pipeline into a LangGraph architecture capable of running multiple agents in parallel. This transition improved performance and made the reasoning process easier to understand and extend. The introduction of the debate mechanism, managed by the debate manager, also added structure and interpretability, ensuring that each agent’s output was discussed and validated before reaching a consensus.

The system’s integration of Retrieval-Augmented Generation (RAG) further improved its reliability by grounding agent insights in real company data rather than relying purely on model predictions. This made outputs more consistent and less prone to errors or fabricated information. The design is modular, allowing new agents to be added in the future—such as those focused on macroeconomic factors or sustainability—without needing to rebuild the pipeline.

From a product standpoint, the interface and overall workflow were designed to be clear and fast to use. Analyses typically complete in under a minute, even when processing multiple stocks sequentially. The frontend presents all results clearly, with transcripts showing how agents interacted and debated, along with the ability to download the full report in PDF format. The consistent formatting and responsiveness make the tool accessible to both technical and non-technical users.

In its entirety, this system applies AI research to a real-world application. It highlights how multi-agent orchestration, debate-driven reasoning, and evidence-based analysis can work

together in a single cohesive product. The Multi-Agent Finance Risk Analysis System demonstrates the potential of LangGraph to create explainable AI tools that perform complex reasoning in an interpretable way. Future improvements could include portfolio-level analysis, adjustable risk settings, or real-time data integration/ ingestion, all of which would move the system closer to a fully deployable product.

In summary, the project successfully combines strong AI engineering with practical design. It showcases how multiple agents, structured debate, and retrieval-based reasoning can produce fast, clear, and trustworthy outputs. The result is a complete, explainable AI system that demonstrates how complex decision-making can be turned into a user-friendly product suitable for professional or research use.

12. Conclusion & Future Work

In conclusion, the product we developed uses a multi-agent LangChain and LangGraph framework. It successfully performs a risk analysis of the searched company and provides buy, hold, or sell recommendations. Combining the collaboration model with the debate model, the Sentiment agent, the visualisation agent, the risk agent, the data agent, and the writer agent work together to generate the final result of recommendation and risk analysis of financial performance, balance sheet strength, profitability, debt, visualisation, and long-term business potential. The product generates a relatively accurate risk analysis and an acceptable recommendation of the stock.

For future work, some improvements can be made to the system. Although with the same specializations, the debate agents are currently using a different system prompt, which can be combined with the agents from the collaboration workflow so that they use the same LLM. The ChromaDB can use more complex retrieval functions for efficiency and be tested using different test cases. The agent collaboration and debate workflows can be improved for faster execution time, and each agent prompt can be further improved. More test cases can be added into integration tests and other metrics such as max drawdown can be used as conditions during testing. Other LLM models such as GPT-5 or other Ollama models can also be used to assess if the results improved. The system can also be improved to use memory and handle long chat sessions. The GUI can be further improved by only displaying the necessary points and with options to view the full data. The product could also be commercialised. The go-to-market strategy needs to be carefully designed, and the UI would need to be optimised for potential buyers.

13. Declaration of GenAI Usage

During the project, the GenAI tool is used to support it. Cursor is used to generate some parts of the code. ChatGPT is used to explain the details of literature papers and to debug code during implementation. Grammarly is used to correct grammar issues after the report is drafted.

14. References

- Devlin, J., Ming-Wei, C., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Ithaca.
- Liu, X.-Y., Wang, G., Yang, H., & Zha, D. (2023). Fingpt: Democratizing internet-scale data for financial large language models. *arXiv preprint arXiv:2307.10485*.
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27), 79-80.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Wu, S., Irsoy, O., Lu, S., Dabrowski, V., Dredze, M., Gehrmann, S., Kambadur, P., Rosenberg, D., & Mann, G. (2023). Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.
- Zhao, T., Lyu, J., Jones, S., Garber, H., Pasquali, S., & Mehta, D. (2025). AlphaAgents: Large Language Model based Multi-Agents for Equity Portfolio Constructions. *arXiv preprint arXiv:2508.11152*.

15. Appendix

Team Individual Contribution Split

Ye Yint Thu	25%
Rianto Utomo	25%
Yingrong Zhang	25%
George Luther	25%