# Assignment 5.1 - Multi-Agent System

Pranjal Mahajan (002375449)

Srushti Patil (002345025)

Ram Srikar Putcha (002304724)

# **Introduction**

With the growing complexity of data analysis and decision-making, leveraging AI-powered agents for research has become essential. This project builds an advanced multi-agent research assistant integrating structured and unstructured data sources. Using LangGraph, Pinecone, Snowflake, and real-time web search APIs, the system enables efficient and comprehensive research on NVIDIA’s financial performance and market trends. By combining Retrieval-Augmented Generation (RAG) and structured queries, the assistant delivers well-rounded insights through a user-friendly Streamlit + FastAPI interface.

# **Problem Statement**

Researching and analyzing financial data across multiple sources is often fragmented and time-consuming. Traditional approaches struggle to integrate structured financial metrics, historical reports, and real-time market trends into a single, cohesive analysis. Without automation, users face challenges in filtering relevant data, drawing insights, and generating meaningful reports. This project aims to address these issues by building an intelligent, multi-agent system that consolidates structured, unstructured, and real-time data into an interactive, AI-powered research assistant.

# **Project Goals**

* Implement three specialized agents—a Snowflake Agent for structured data, a RAG Agent for contextual retrieval, and a Web Search Agent for real-time insights.
* Automate the retrieval and consolidation of NVIDIA’s financial reports, valuation metrics, and industry trends.
* Utilize Pinecone’s metadata filtering to enable research filtering by Year and Quarter.
* Integrate AI-generated summaries, data-driven visuals, and real-time insights into well-structured research reports.
* Develop a Streamlit + FastAPI application to allow seamless user interaction, query execution, and visualization.
* Containerize the system using Docker for streamlined deployment and accessibility.

# **Proof of Concept**

This is to validate the feasibility of a multi-agent research assistant that integrates structured financial data, unstructured research reports, and real-time web search results using LangGraph, Pinecone, Snowflake, and FastAPI.

### System Architecture Overview

The project consists of three key agents working in tandem:

#### Snowflake Agent (Structured Data)

* Connects to Snowflake and queries NVIDIA’s valuation measures from Yahoo Finance.
* Generates structured text summaries and visualizations for key metrics (e.g., P/E ratio, market cap).

Example Query:

SELECT year, quarter, pe\_ratio, market\_cap FROM nvidia\_valuation WHERE year = 2024;

#### 

#### RAG Agent (Unstructured Data)

* Uses Pinecone as a vector database to store and retrieve NVIDIA’s quarterly reports.
* Implements metadata filtering for retrieval based on Year and Quarter.

Example Query:

query = "NVIDIA's revenue performance in Q4 2023"

results = pinecone\_index.query(query, filter={"year": 2023, "quarter": "Q4"})

#### C. Web Search Agent (Real-Time Data)

* Fetches real-time industry insights using APIs like SerpAPI, Tavily, or Bing Search API.

Example API call:

response = requests.get("https://api.serpapi.com/search?q=NVIDIA+stock+news")

### 

### Data Flow & Integration

#### User Input:

#### The user queries financial insights via the Streamlit UI.

#### Example: *“Analyze NVIDIA’s financial performance in Q4 2023.”*

#### Data Retrieval & Processing:

#### Snowflake Agent fetches structured financial metrics.

#### RAG Agent retrieves quarterly report data using Pinecone.

#### Web Search Agent extracts current industry trends via APIs.

#### Response Generation:

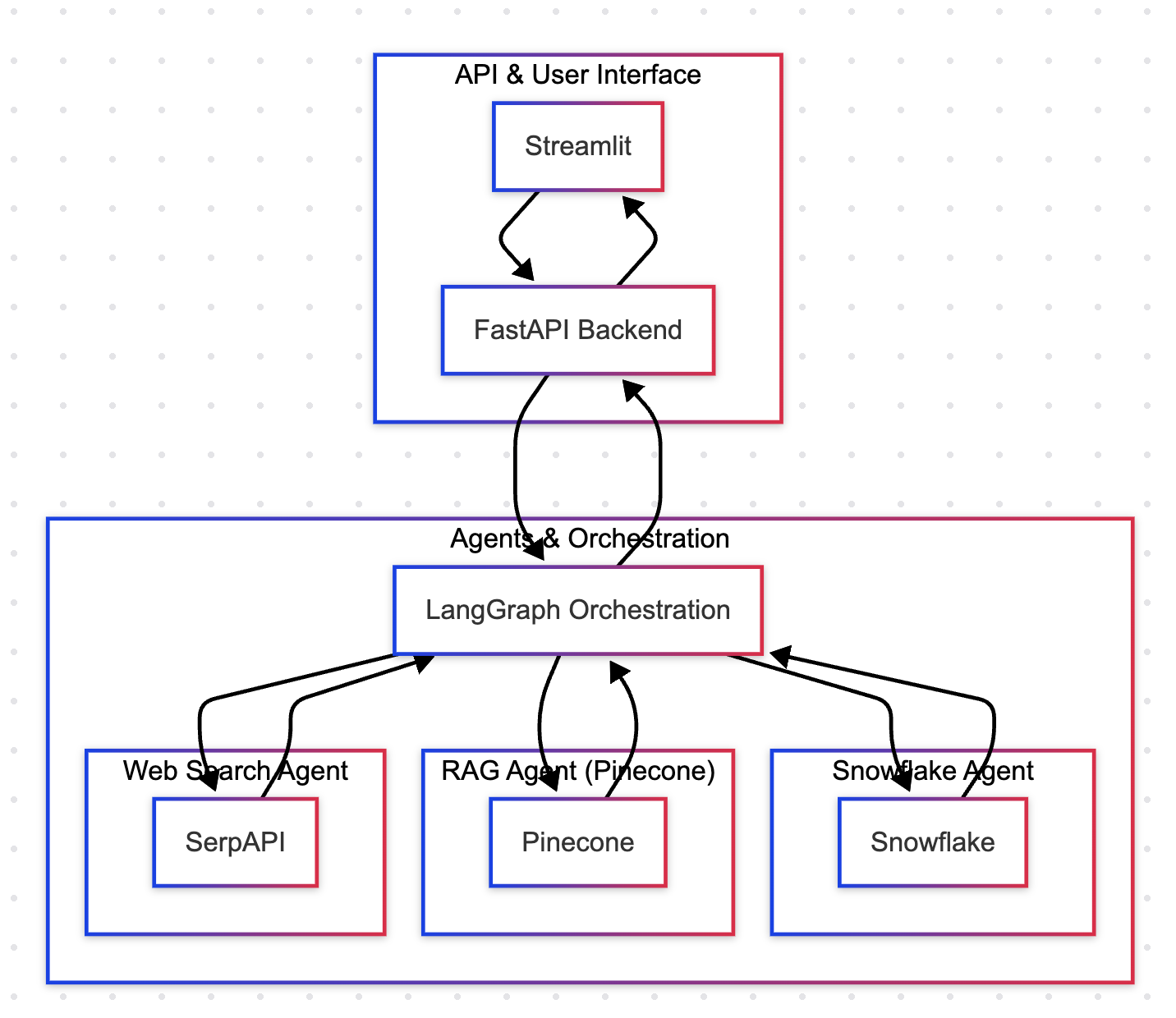
* A LangGraph-based orchestration combines all agent responses.
* The system generates a structured research report with summaries, charts, and real-time insights.

1. User Output (Streamlit UI):

A research report with:

* Financial charts (Snowflake)
* Contextual insights (RAG)
* Latest industry trends (Web Search)

# **Architecture Diagram**



The diagram represents the architecture of the Multi-Agent Financial Research Assistant, which consists of three major layers:

### 1. API & User Interface Layer

This layer handles user interaction and API communication.

* Streamlit: Provides a web-based user interface where users can input queries and receive responses.
* FastAPI Backend: Acts as a middleware, handling API requests from Streamlit and forwarding them to the LangGraph Orchestration layer.

Flow:

1. The user inputs a query in Streamlit.
2. The query is sent to the FastAPI backend, which forwards it to the orchestration layer.
3. Once the response is generated, FastAPI sends it back to Streamlit for display.

### 2. Agents & Orchestration Layer

This is the core processing layer where different AI agents retrieve and analyze data.

* LangGraph Orchestration:  
  1. Manages the workflow of three specialized agents.
  2. Ensures parallel execution and response aggregation.
* Three Specialized Agents:  
  1. Web Search Agent (SerpAPI)  
     + Fetches real-time financial news and market trends from online sources.
  2. RAG Agent (Pinecone)  
     + Retrieves unstructured textual data (e.g., quarterly reports) from a vector database (Pinecone).
     + Uses metadata filtering to ensure relevant document retrieval.
  3. Snowflake Agent  
     + Queries structured financial metrics from the Snowflake database.
     + Retrieves numerical data such as market capitalization, revenue, and P/E ratio.

Flow:

1. The FastAPI backend sends a user query to LangGraph Orchestration.
2. LangGraph sends the query to all three agents in parallel execution.
3. Each agent retrieves relevant data from its respective source.
4. The aggregated response is sent back to FastAPI for final formatting and display in Streamlit.

### 

### 3. Data Retrieval & Response Flow

* The Web Search Agent, RAG Agent, and Snowflake Agent work independently but communicate via the LangGraph orchestrator.
* After processing, LangGraph compiles the responses from all three sources to generate a comprehensive financial research report.
* The final insights are displayed on Streamlit, ensuring a user-friendly and interactive experience.

# **Walkthrough of the Assignment**

## **Step 1: Clone the Repository**

1. Open a terminal or command prompt.

Run the following command to clone the repository:  
git clone <repository\_url>

1. cd Assignment\_4

## **Step 2: Create a Virtual Environment**

Create and activate a virtual environment:  
python -m venv venv

1. source venv/bin/activate # On Windows: venv\Scripts\activate

## **Step 3: Install Dependencies**

1. Install the required dependencies:  
   pip install -r requirements.txt

## **Step 4: Configure Environment Variables**

1. Create a .env file in the root directory.

Add required credentials such as API keys for AWS, LLM, etc.

AWS\_ACCESS\_KEY=<your\_access\_key>

AWS\_SECRET\_KEY=<your\_secret\_key>

## **Step 5: Run the Backend Server**

1. Start the FastAPI backend server:  
   uvicorn backend.app:app --reload
2. The API will be available at <http://127.0.0.1:8000>.

## **Step 6: Run the Frontend Dashboard**

1. Start the Streamlit dashboard:  
   streamlit run frontend/dashboard.py
2. Open the displayed local URL to access the dashboard.

### **Step 7: Use the Application**

1. Choose a data source for financial analysis:

* Web Search Agent (SerpAPI) for real-time financial news.
* RAG Agent (Pinecone) for retrieving historical reports.
* Snowflake Agent for structured financial data

1. Input a company name, stock ticker, or financial topic to retrieve relevant insights.
2. Choose how financial data should be retrieved:

* Web Search (SerpAPI) for latest news.
* Pinecone Vector Search for document-based retrieval.
* Snowflake SQL Query for structured datasets.

1. Select how to refine and process retrieved financial data:

* Entity Extraction to highlight key financial figures.
* Summarization to condense earnings reports.
* Sentiment Analysis to assess market sentiment.

1. Specify a time range (e.g., last quarter, last year) to refine financial insights.
2. Enter a financial-related question (e.g., *"What were Tesla’s Q4 earnings?"*) and receive an AI-generated response.

## **Step 8: Verify API Endpoints**

Use Postman or a browser to test API endpoints.

1. Example API request for Retrieving Extracted Text from Redis:  
   curl -X GET "http://127.0.0.1:8000/get\_extracted\_text/sample.md"
2. Example API request for Listing Stored Markdown Files in S3:  
   curl -X GET "http://127.0.0.1:8000/select\_pdfcontent/"
3. Example API request for Asking a Question on Retrieved Financial Insights:  
   curl -X POST "http://127.0.0.1:8000/ask\_question/" -H "Content-Type: application/json" -d '{"data\_id": "report123", "llm": "gemini", "question": "What are the key insights?"}'
4. Example API request for Summarizing a PDF using an LLM Model:  
   curl -X POST "http://127.0.0.1:8000/summarize/" -F "pdf\_name=sample.md" -F "llm=gpt-4"
5. Example API request for Searching Financial Data via Web Search Agent (SerpAPI):  
   curl -X GET "http://127.0.0.1:8000/search\_web/" -H "Content-Type: application/json" -d '{"query": "latest stock market trends"}'
6. Example API request for Fetching the Result of an AI Task (Summarization or Q&A):  
   curl -X GET "http://127.0.0.1:8000/get\_result/task-abc123"
7. Example API request for Uploading a PDF:  
   curl -X POST "http://127.0.0.1:8000/upload\_pdf/" -F "file=@sample.pdf"

### **Common Issues and Troubleshooting**

1. Streamlit Dashboard Not Loading: Ensure that Streamlit is installed correctly and the script is running without errors. Try restarting the application and manually opening the provided URL.
2. FastAPI Server Not Responding: Verify that FastAPI is running at http://127.0.0.1:8000 and ensure there are no port conflicts. Restart the server if needed.
3. LangGraph Orchestration Failing: Check if all required agents (SerpAPI, Pinecone, Snowflake) are properly initialized and that API keys are correctly configured.
4. Web Search Agent (SerpAPI) Not Returning Results: Ensure the SerpAPI key is valid and that API rate limits have not been exceeded. Debug by printing API responses.
5. RAG Agent (Pinecone) Not Retrieving Data: Confirm that embeddings are correctly indexed and that queries match stored vector formats. Verify API keys and database connectivity.
6. Snowflake Agent Not Fetching Data: Ensure the correct Snowflake credentials and database configurations are provided. Debug SQL queries for syntax errors or incorrect filters.
7. Summarization Output is Incorrect or Empty: Check if the LLM model is properly loaded, API keys are valid, and request payloads are correctly structured.
8. Hybrid Search Not Filtering Properly: Verify that data is correctly indexed and that query filters match the expected format. Debug by printing search parameters.
9. User Queries Not Processing: Ensure that requests are reaching LangGraph Orchestration and that agents are correctly invoked. Check FastAPI logs for request failures.
10. Slow Response Times: Identify bottlenecks by logging execution times for each agent. Optimize query processing and reduce redundant API calls where possible.

# **Application Workflow**

### The application workflow consists of five key stages, detailing how user queries are processed, data is retrieved, and insights are generated.

### 1. User Input & Query Submission

* The user accesses the Streamlit web interface.
* Inputs a financial research query (e.g., *"Analyze NVIDIA’s financial performance in Q4 2023"*).
* The query is sent to the FastAPI backend via an API request.

### 2. Query Processing & Orchestration

* FastAPI forwards the query to LangGraph Orchestration, which manages multiple AI agents.
* LangGraph parses the query, determines which agents need to be invoked, and initiates parallel execution.

### 3. Multi-Agent Data Retrieval (Parallel Execution)

LangGraph distributes the query among three specialized agents:

Web Search Agent (SerpAPI)

* Searches real-time financial news, market trends, and expert insights.
* Retrieves articles, blog posts, and stock-related updates.

RAG Agent (Pinecone)

* Conducts a vector-based search for unstructured financial documents.
* Retrieves earnings reports, investor briefings, and analyst commentaries from Pinecone’s vector database.

Snowflake Agent

* Queries structured financial data from Snowflake.
* Extracts key metrics such as quarterly revenue, market cap, and financial ratios.

Each agent returns relevant information to LangGraph for processing.

### 4. Response Aggregation & Summarization

* LangGraph aggregates data from all three agents.
* Uses an LLM-based summarization model (GPT-4 or similar) to:  
  + Extract key insights.
  + Summarize findings in a concise and structured format.
  + Remove redundant or irrelevant data.

### 5. Response Delivery & User Interaction

* The final report is formatted and sent back to the FastAPI backend.
* FastAPI delivers the response to Streamlit, where it is displayed in an interactive UI.
* Users can:  
  + Refine their query for more specific insights.
  + View structured data, extracted reports, and real-time news.
  + Download or export findings for further analysis.

# **References**

* [Pinecone Hybrid Search with Metadata](https://colab.research.google.com/github/pinecone-io/examples/blob/master/learn/generation/langchain/langgraph/01-gpt-4o-research-agent.ipynb)
* [LangGraph Multi-Agent Example](https://langchain-ai.github.io/langgraph/tutorials/customer-support/customer-support/#flights)
* [FastAPI Documentation](https://fastapi.tiangolo.com/)
* [Streamlit Documentation](https://docs.streamlit.io/)
* [Snowflake Documentation](https://docs.snowflake.com/en/user-guide/snowflake-cortex/cortex-agents)
* [SerpAPI](https://serpapi.com/)
* [Pinecone Documentation](https://docs.pinecone.io/guides/get-started/overview)

# **Disclosures**

WE ATTEST THAT WE HAVEN’T USED ANY OTHER STUDENTS’ WORK IN OUR

ASSIGNMENT AND ABIDE BY THE POLICIES LISTED IN THE STUDENT HANDBOOK

| Pranjal Mahajan(002375449) | 33.33% |
| --- | --- |
| Srushti Patil (002345025) | 33.33% |
| Ram Putcha (002304724 | 33.33% |

AiUseDisclousure :

We have utilized AI to assist us in the following areas:

* Understanding how to deploy FastAPI for rendering and the necessary deployment steps.
* Deploying a Streamlit frontend application to Streamlit Community Cloud.
* Connecting frontend and backend deployments to work seamlessly together
* Learning how to implement RAG pipelines
* Understanding how Pinecone and ChromaDB stores data and accessing it