A PROJECT SCHOOL REPORT ON FINANCE GPT: YOUR FINANCE EVOLVED!!

Submitted By

Student Name 1: Ehsaas Devesh Nahata	**Roll No : 245523748018**
Student Name 2: Aditya Vidiyala	**Roll No: 245523748065**
Student Name 3: Sai Krishna Nair	**Roll No: 245523748121**
Student Name 4: Kanduri Adithya	**Roll No: 245523748091**
Student Name 5: Srinandana Sarma	**Roll No: 245523748096**
Student Name 6: Vakiti Nigama Reddy	**Roll No: 245523748124**

Under the guidance
Of Mentor Name:

KVN Hari Babu

Designation, Department:

KESHAV MEMORIAL ENGINEERING COLLEGE
Kachavanisingaram Village, Hyderabad, Telangana 500058.

July, 2025

KESHAV MEMORIAL ENGINEERING COLLEGE

A Unit of Keshav Memorial Technical Education (KMTES)

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad

CERTIFICATE

This is to certify that the project work entitled “**FINANCE GPT: YOUR FINANCE EVOLVED”** *is a bona fide work carried out by* **“All Student Name”** of III-year V semester **Bachelor of Engineering** *in* **CSE/ CSE (AIML)** *during the academic year* **2024-2025 and** *is a record of bona fide work carried out by them*.

Project Mentor

Faculty name: K.V.N Hari Babu

ABSTRACT

Finance GPT is an advanced AI-powered financial assistant designed to transform how individuals and businesses interact with complex financial information. In an environment where timely and accurate insights are crucial, Finance GPT provides a unified solution for navigating diverse financial data sources and making informed decisions. The project addresses the challenge of information overload and the need for personalized financial intelligence. It empowers users to effortlessly access and understand real-time market news, analyze specific financial documents and images, and obtain answers to general financial queries. By leveraging cutting-edge artificial intelligence, Finance GPT automates the often cumbersome processes of data extraction, analysis, and synthesis, delivering concise and actionable insights.

At its core, Finance GPT operates through an intelligent, modular design. It employs a sophisticated system that understands user intent, intelligently delegates tasks to specialized "expert" tools and then refines the gathered information into clear, coherent responses. Furthermore, the system incorporates memory capabilities, allowing it to learn from past interactions and provide a progressively more personalized and relevant user experience.

Finance GPT is more than just an information retriever; it is a smart, intuitive, and powerful companion for anyone seeking clarity and actionable intelligence in the financial world. It aims to be a seamless resource for seasoned investors, financial analysts, and everyday users alike, fostering better understanding and more confident financial decision-making

.

CONTENTS

1. INTRODUCTION

1.1 PROBLEM STATEMENT

The core problem Finance GPT addresses is the **"challenge of information overload and the need for personalized financial intelligence."** In today's financial world, individuals and businesses are inundated with vast amounts of data, making it challenging to find, process, and comprehend relevant financial information quickly and accurately. This leads to inefficient decision-making and a lack of timely insights.

Finance GPT, an innovative and comprehensive AI-powered financial assistant designed to revolutionize how individuals and businesses interact with financial information. In today's fast-paced financial landscape, access to accurate, timely, and context-aware insights is paramount. Finance GPT addresses this need by combining cutting-edge artificial intelligence techniques, including advanced natural language processing and agent-based systems, to deliver a truly intelligent and versatile financial companion.

1.2 OBJECTIVES

At its core, Finance GPT aims to empower users with the ability to:

- **Access Diverse Financial Information**: From real-time market news and general financial knowledge to in-depth analysis of specific financial documents and images.
- **Obtain Context-Aware Insights**: The system understands the nuances of user queries, leveraging conversational history and long-term memory to provide highly relevant and personalized responses.
- **Automate Complex Financial Tasks**: By intelligently routing requests to specialized tools, Finance GPT streamlines the process of extracting, analyzing, and synthesizing financial data.

- **Gain Actionable Intelligence**: Beyond just providing data, the system is designed to refine and summarize information, offering concise and actionable insights crucial for informed decision-making.

Finance GPT stands out through its sophisticated architecture, which allows it to handle a wide array of financial queries with precision and efficiency. Whether you're a seasoned investor, a financial analyst, or simply someone looking to better understand their finances, Finance GPT is built to be your trusted intelligent partner.

1.3 SCOPE OF THE PROJECT

The scope of Finance GPT, as described in the abstract, includes:

- Accessing and understanding real-time market news.
- Analyzing specific financial documents and images.
- Answering general financial queries.
- Delegating tasks to specialized "expert" tools.
- Incorporating memory for personalized interactions.

It focuses on providing an intelligent, unified solution for diverse financial data sources. Our project has an agent under which we have multiple tools those are **1.Document Q&A (RAG - Retrieval-Augmented Generation):**

- This tool is designed to answer specific questions directly from financial documents uploaded by the user, such as PDFs of annual reports, bank statements, or investment summaries.
- **How it works:** When you upload a document and ask a question about it (e.g., "What was the total revenue for 2023 in this report?"), this tool processes the document, extracts relevant information, and uses a sophisticated method called Retrieval-Augmented Generation to formulate a precise answer based *only* on the content of your document. It ensures that the information provided is directly

verifiable from your source.

2. Image Q&A:

- This tool allows Finance GPT to interpret and extract information from financial images, such as charts, graphs, tables, or scanned financial forms.
- **How it works:** If you upload an image of a stock chart or a balance sheet snapshot and ask a question (e.g., "What was the peak price in this chart?" or "What is the total liabilities from this image?"), the Image Q&A tool uses advanced image processing and optical character recognition (OCR) techniques to read the data within the image and then answers your question based on that extracted information.

3. News Analysis:

- This is used to provide you with the latest and most relevant financial news and market events.
- **How it works:** When you inquire about current market trends, specific company news, or economic updates, this tool searches recent financial news articles from reputable sources. It then analyzes and summarizes the key information, presenting it in a concise and actionable format, often highlighting critical numbers, dates, and sources.

4. General Q&A:

- **Purpose:** This tool handles broad financial questions that don't require specific documents, images, or real-time news. It serves as a comprehensive knowledge base for general financial concepts.
- **How it works:** If you ask "What is Net Gross Value?" or "Explain market capitalization," the General Q&A tool draws upon its extensive financial knowledge to provide clear, concise, and expert-level explanations, often

leveraging context from your ongoing conversation to tailor the answer.

5. Financial Headline Classifier

- We have fine-tuned a model on Financial Headline Sentiments so that the model outputs the sentiment about the user query(headline).

1.4 LIMITATIONS

While the abstract highlights the project's strengths and capabilities, it implicitly suggests some limitations by what it *doesn't* explicitly cover or by the nature of AI systems:

- **Specific Financial Advice:** The abstract positions Finance GPT as an "assistant" and "companion" for "clarity and actionable intelligence," but it does not claim to provide direct financial advice or make investment decisions. Its role is to provide information and insights for *informed decision-making*, implying the user retains ultimate responsibility for their financial actions.
- **Real-time Market Trading/Execution:** The focus is on information access and analysis, not on executing trades or direct market interventions.
- **Regulatory Compliance/Legal Advice:** As an AI assistant, it's unlikely to be designed to provide certified legal or regulatory compliance advice, which typically requires human expertise.
- **Novel Data Sources:** While it handles "diverse financial data sources," there might be limitations on integrating highly niche, proprietary, or unstructured data types not explicitly mentioned.
- **Accuracy and Timeliness:** While aiming for "timely and accurate insights," all AI systems have inherent limitations regarding the absolute real-time nature of data and the potential for hallucination or misinterpretation, especially with rapidly changing information. (Though the abstract emphasizes accuracy, it's a general limitation for any such system).

2. LITERATURE SURVEY

2.1 Existing Systems

To contextualize Finance GPT's innovation, it's valuable to understand the landscape of existing AI systems and agent models in the financial domain. Several prominent initiatives have emerged, each addressing specific aspects of financial intelligence through advanced AI.

1. FinGPT

FinGPT is an open-source framework specifically designed to facilitate the development of Large Language Models (LLMs) for financial applications. This initiative strongly emphasizes the importance of continuous, real-time data for financial LLMs, recognizing the highly dynamic nature of financial markets. Its open-source nature promotes collaborative development and broad accessibility for researchers and developers. FinGPT is notably data-centric, focusing on the curation and processing of high-quality, real-time financial data to ensure LLMs remain updated with the latest market movements and news. It provides essential methodologies and resources for fine-tuning general-purpose LLMs on specialized financial datasets, thereby enabling these models to more effectively understand financial jargon, market sentiment, and economic indicators. The framework aims to support a wide array of financial tasks, including financial sentiment analysis, news summarization, risk assessment, and quantitative trading strategies. FinGPT's primary innovation lies in its dedication to data-centric AI for finance, offering a robust framework for building and maintaining up-to-date financial LLMs that can readily adapt to rapidly changing market conditions.

2. BloombergGPT

BloombergGPT stands as a large language model uniquely trained on an extensive dataset of financial data by Bloomberg, a global leader in business and financial information. It is meticulously designed to excel at financial tasks, leveraging Bloomberg's vast and often proprietary data resources. Its core strength lies in its domain-specific training, having been built from scratch using a hybrid dataset that combines a massive volume of financial data—including news, company filings, reports, and more—with general-purpose web data. This deep financial domain expertise is further amplified by its proprietary data advantage, benefiting from Bloomberg's unparalleled access to historical and real-time financial information, which provides a unique edge in understanding intricate market nuances. BloombergGPT consistently demonstrates strong performance across a range of financial Natural Language Processing (NLP) tasks, such as sentiment analysis, named entity recognition for financial entities, and precise question answering within the financial domain. It is intended to enhance the capabilities of the Bloomberg Terminal, offering more intuitive access to information and advanced analytical tools for financial professionals. BloombergGPT represents a significant advancement in the development of highly specialized LLMs for finance, powerfully demonstrating how training on vast, curated domain-specific datasets can lead to superior performance in complex financial applications.

3. FIS Treasury GPT

FIS Treasury GPT is a specialized solution developed by FIS, a prominent provider of financial technology, intending to leverage generative AI to enhance treasury operations. Its primary focus is on improving efficiency and decision-making within corporate treasury functions. This solution is characterized by its treasury-specific focus, being meticulously tailored to address the unique challenges and data inherent in corporate treasury, including critical areas like cash management, liquidity forecasting, risk management, and comprehensive financial reporting. It harnesses generative AI capabilities to automate routine tasks, derive valuable insights from treasury data, and actively assist treasury professionals in making more informed decisions. Furthermore, FIS Treasury GPT emphasizes robust data integration, designed to seamlessly connect with existing treasury management systems (TMS) and other financial data sources, thereby providing a unified view and actionable intelligence. Its potential applications are broad, encompassing assistance with tasks such as generating precise cash flow forecasts, analyzing foreign exchange exposures, summarizing complex treasury reports, and offering insightful perspectives on liquidity positions. FIS Treasury GPT exemplifies the effective application of generative AI to highly specialized financial functions, aiming to bring sophisticated automation and intelligent assistance to critical back-office operations.

4. Agent Models in the Financial Field

Beyond the realm of large, monolithic LLMs, the concept of **AI agents** is rapidly gaining significant traction within the financial sector. Agent models involve the creation of autonomous or semi-autonomous AI entities capable of perceiving their environment, making informed decisions, and executing actions to achieve specific goals. These agents often operate by interacting with various tools or collaborating with other agents. A key characteristic of financial agent models is their proficiency in tool use; they are frequently designed to interact with external tools and APIs, such as real-time market data feeds, trading platforms, advanced analytical software, or specialized document processing services, enabling them to perform concrete actions beyond mere text generation. They possess robust decision-making capabilities, being either programmed or trained to make judgments based on financial data, prevailing market conditions, and predefined rules or sophisticated learned strategies. In more complex financial applications, multi-agent systems might be deployed, where several agents collaborate to achieve a larger objective; for instance, one agent could be tasked with monitoring news, another with analyzing intricate charts, and a third with executing trades based on the combined insights from the collective. Their applications are diverse and impactful, spanning areas such as Algorithmic Trading Agents that automate trading decisions based on market signals and strategies, Portfolio Management Agents that optimize investment portfolios aligned with risk tolerance and financial goals, Compliance Agents that continuously monitor transactions and activities for regulatory adherence, Financial Advisory Agents that provide personalized financial advice by analyzing individual financial situations and market conditions, and Research Agents that automate the gathering and synthesis of financial research from disparate sources. The innovation inherent in financial agent models lies in their capacity to transcend passive information retrieval, moving towards active decision-making and execution, often within highly complex and dynamic environments. They offer substantial potential for greater automation, personalized service, and the deployment of sophisticated analytical capabilities. Projects like yours, Finance GPT, with its "intelligent, modular design" and "specialized 'expert' tools," clearly align with and significantly contribute to this evolving landscape of financial AI agents.

3. PROPOSED WORK, ARCHITECTURE, TECHNOLOGY STACK AND IMPLEMENTATION DETAILS

3.1 TECHNOLOGY STACK

Our project utilizes a robust and modern technology stack designed for scalability, performance, and advanced AI capabilities.

At the **User Interface Layer**, the project employs **React** for building a dynamic and interactive front-end, ensuring a rich user experience. **Tailwind CSS** is used in conjunction with React to provide highly customizable and responsive styling, allowing for rapid UI development and a consistent design across various devices.

The **Application Layer** is powered by **Node.js** and **Express.js**. Node.js provides a high-performance, non-blocking I/O environment for the server-side, while Express.js, a minimalist web framework for Node.js, handles routing, middleware, and API endpoints, facilitating efficient communication between the front-end and the core business logic.

For the sophisticated **Business Logic Layer**, the stack includes **FastAPI**, **LangChain**, **LangGraph**, **Hugging Face Transformers**, and **mem0**. FastAPI is a modern, fast (high-performance) web framework for building APIs with Python, ideal for handling the project's AI-driven logic. LangChain and LangGraph are critical for orchestrating the complex agent system, enabling the chaining and graphing of different language model components and tools. Hugging Face Transformers provides access to state-of-the-art pre-trained models, likely used for the underlying LLM capabilities and potentially for the sentiment analysis model. Finally, `mem0` is integrated for memory management, allowing the system to maintain conversational context and personalize interactions over time.

The **Database Layer** relies on **PostgreSQL**, accessed via **Supabase**. PostgreSQL serves as a powerful, open-source relational database for storing persistent data. Supabase provides a managed PostgreSQL database, along with authentication and API services, simplifying database management and enabling real-time data capabilities for the application.

Layers	**Technologies**	**Purpose**
User Interface Layer	React, Tailwind CSS	Frontend for building responsive and interactive UI
Application Layer	Node.js, Express.js	Backend to handle HTTP requests, route traffic, serve APIs to frontend
Business Logic Layer	FastAPI, LangChain, LangGraph, Hugging Face	Core Logic for model inference, chaining tools, and Finetuning

The model(LLM) which we used for the Agent is **Gemini Flash 2.5** as the inference speed is very fast and majorly context length is simply awesome(250k) which is essential when the dependencies gets piled up from different tools so the history of them or the combined context of them can be placed easily and for the fine tuning part we have chosen **Llama 3.2 1B Instruct** model which is finetuned on Financial Headline Sentiments. Reason for choosing Llama 3.2 1B is firstly there can be before and after results which we can see easily as we used instruct model to train on this task and the dataset which we have used has some quality of examples(13k rows) regarding the Financial headlines so the model can learn from them and perform better on this task.

Model	**Use**
Gemini 2.5 Flash	Agent
Llama 3.2 1B	Sentiment Analysis

3.2 IMPLEMENTATION AND ARCHITECTURE

(i) AGENT ARCHITECTURE AND IMPLEMENTATION

1. The Orchestrator: (Graph.py)

At the heart of the agent system is Graph.py, which defines and manages the entire flow of information and control. It acts as the central orchestrator, a state graph that dictates how user requests are processed and how specialized agents interact.

GraphState: This is the central data structure that holds all relevant information throughout a user's interaction. It's a dynamic state that includes:

- The original user query (input).
- References to any uploaded documents (uploaded_doc) or images (uploaded_img).

- A dynamically determined agent_order (the sequence of agents to execute).
- The routing reasoning behind the chosen agent path.
- Tracking of the current_agent_index and processed_agents.
- Storage for the agent outputs from each completed task.
- Flags for messages_added.
- The final response to be presented to the user.
- Identifiers for user_id and session_id to manage personalized memory.
- The full messages (conversation history) and past memory (retrieved long-term facts).

Agent Nodes: Graph.py registers various specialized agent nodes, each representing a distinct financial capability:

- **Document_qna:** Handles questions about uploaded financial documents using RAG.
- **General_qna:** Addresses broad financial knowledge and reasoning queries. **News:** Fetches and analyzes the latest financial news and events.
- **Image_qna:** Extracts and interprets information from financial images.
- **Refiner:** Synthesizes, summarizes, expands, or rephrases previous agent outputs.
- **Aggregator:** (Implicitly) combines outputs from multiple agents to form a

cohesive response.

Routing Mechanism: The graph utilizes add_conditional_edges with a route_to_agents function. This mechanism dynamically directs the flow of the conversation based on the decisions made by the router, enabling flexible and intelligent multi-agent workflows.

Memory Integration: The inclusion of MemoryClient (mem0) signifies that the agent system is designed to leverage long-term memory. This allows Finance GPT to recall past conversations, user preferences, and relevant financial facts, enabling highly personalized and context-aware interactions.

2. The Routing Intelligence: prompt.py

The intelligence that drives the orchestration in Graph.py comes from the ROUTER_PROMPT defined in prompt.py. This is a critical system prompt that guides a Large Language Model (LLM), specifically Gemini 2.5 Flash, to intelligently analyze user queries and determine the most effective sequence of specialized agents to address them.

Query Decomposition: The router prompt instructs the LLM to break down complex user requests into smaller, agent-specific sub-queries. This is essential for handling multi-faceted questions that might require input from several different tools.

Contextual Understanding: It emphasizes a holistic analysis, taking into account the current user query, the complete conversation history (both recent messages and retrieved long-term memory), and any uploaded documents or images. This comprehensive context allows for highly accurate routing decisions.

Tool Selection & Dependencies: The prompt provides the LLM with a clear list of available tools (Document_qna, News, Image_qna, General_qna, Refiner) and their functionalities. It then guides the LLM to select the most appropriate agent(s) and identify any inter-agent dependencies (e.g., the Refiner might need the output from both Document_qna and News to fulfill a complex request).

Execution Order & Follow-up Detection: The router determines the correct order of agent execution and can identify if a query is a follow-up to a previous topic, ensuring coherent and contextually relevant responses.

Strict Output Format: The prompt enforces a specific JSON output format for the router's decision, which includes the agents to be executed (with name, query, and dependencies) and a reasoning for the decision. This structured output is crucial for Graph.py to parse and execute the determined workflow.

3. Flow of the Agent System

- **User Input:** A user submits a financial query, potentially along with uploaded documents or images.
- **Memory Loading:** The system first attempts to load relevant past_memory based on the user_id and session_id to provide historical context.
- **Routing Decision:** The Router node (guided by prompt.py and Gemini 2.5 Flash) analyzes the user's input, messages (history), past_memory, and uploaded files. It then determines the optimal agent_order and routing_reasoning.
- **Agent Execution:** Graph.py then sequentially executes the agents specified in the agent_order. Each agent (Document_qna, News, Image_qna, General_qna, Refiner) performs its specialized task, potentially using dependency_context from previously executed agents or the message_history.
- **Output Aggregation & Refinement:** Once all necessary agents have completed their tasks, their agent_outputs are collected. The Refiner agent often plays a crucial role here, synthesizing and polishing the combined information into a coherent final_response.
- **Response to User:** The final_response is presented to the user.
- **Memory Saving:** The updated conversation messages and any new insights are saved back to mem0 to enrich the past_memory for future interactions.

This sophisticated agent architecture allows Finance GPT to provide highly accurate, relevant, and comprehensive financial insights by dynamically adapting to the user's needs and leveraging a diverse set of specialized AI capabilities.

(ii) FINETUNED MODEL IMPLEMENTATION

1. Environment Setup and Library Imports

The initial steps involve setting up the Python environment. This includes verifying GPU availability (using !nvidia-smi) to ensure that the fine-tuning process can leverage hardware acceleration. Essential libraries for working with large language models and datasets are imported, such as transformers (for model and tokenizer functionalities), datasets (for data loading and processing), trl (for Transformer Reinforcement Learning, specifically SFTTrainer for supervised fine-tuning), peft (for Parameter-Efficient Fine-Tuning), torch (for deep learning operations), and pandas and matplotlib for data handling and visualization.

2. Dataset Loading and Exploration

The model is trained on the "steve1989/financial_news_headlines" dataset, loaded directly from Hugging Face. This dataset is partitioned into training, validation, and test sets. Initial data exploration involves converting these datasets into pandas DataFrames to inspect the structure and analyze the distribution of sentiment labels (positive, negative, neutral) within each split. This step is crucial to understand the class balance and ensure the dataset is suitable for training.

3. Tokenizer Configuration and Prompt Formatting

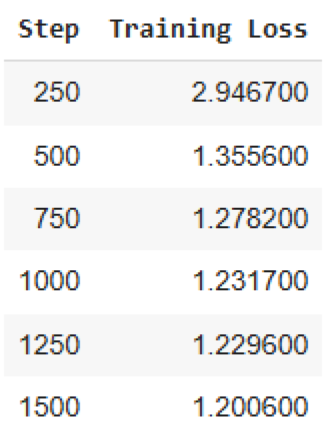
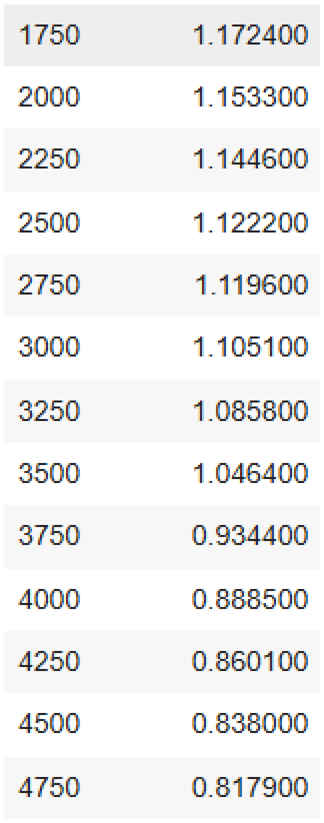
A tokenizer specifically designed for the "meta-llama/Llama-3.2-1B-Instruct" model is loaded. A key configuration is setting the pad_token to the eos_token (end-of-sequence token) to ensure consistent padding during training. Custom functions (generate_training_prompt, generate_test_prompt, generate_eval_prompt) are defined to format the raw headlines and their corresponding sentiment labels into a structured instruction-tuned prompt format. This format includes a system prompt that explicitly instructs the model to act as a "FinanceGPT. A Sentiment analysis expert" and to respond strictly with a single word ('positive', 'negative', or 'neutral') without any additional explanation. This strict instruction helps guide the model's output during fine-tuning. The datasets are then tokenized using these prompt generation functions, preparing them with input_ids, attention_mask, and labels.

4. Model Loading and PEFT Configuration

The base Llama 3.2 1B Instruct model is loaded using AutoModelForCausalLM from the transformers library. To enable efficient fine-tuning, especially on resource-constrained environments like typical GPUs available in Colab, 4-bit quantization is applied using BitsAndBytesConfig. This significantly reduces memory footprint and computational requirements. Furthermore, Parameter-Efficient Fine-Tuning (PEFT) is implemented via LoraConfig. LoRA (Low-Rank Adaptation) is configured with a rank (r) of 16 and lora_alpha of 32, targeting the attention projection layers (q_proj, k_proj, v_proj, o_proj). A lora_dropout of 0.05 is also applied. This PEFT setup allows for fine-tuning only a small fraction of the model's parameters (as indicated by model.print_trainable_parameters() showing a very low trainable percentage), drastically reducing training time and computational cost while preserving the model's performance.

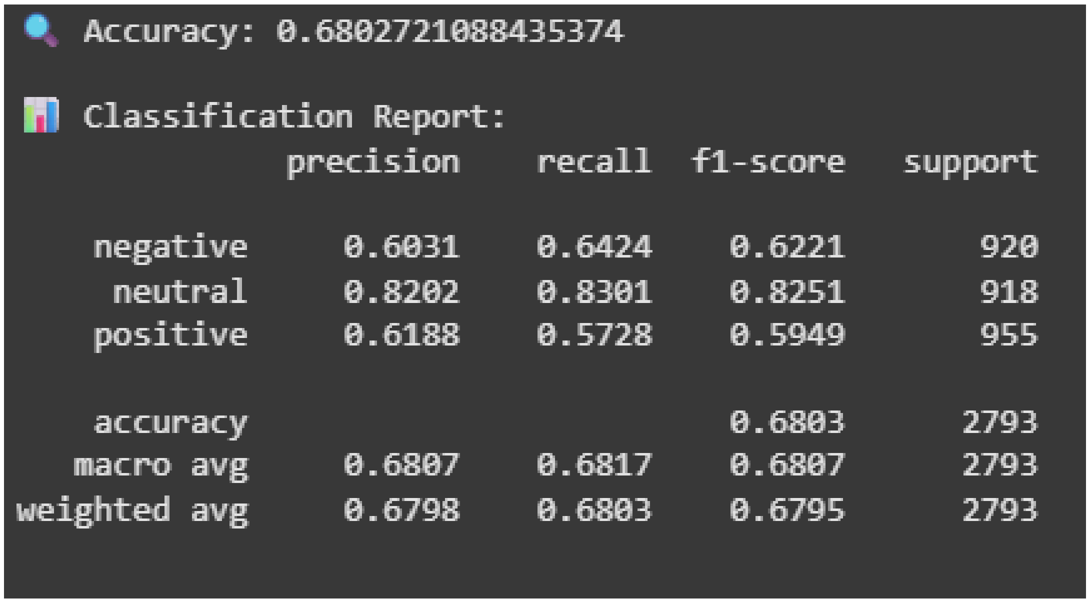
5. Data Collator and Training Setup

A DataCollatorForLanguageModeling is used to prepare batches of data for training, ensuring proper padding and handling of sequences. The notebook also sets up logging for the training process using TensorBoard, allowing for real-time monitoring of metrics and progress during the fine-tuning phase. While the actual training execution (SFTTrainer) is not fully detailed in the provided snippet, the setup clearly indicates that the model is ready for supervised fine-tuning on the prepared financial sentiment dataset.

Step	Training Loss
250	2.946700
500	1.355600
750	1.278200
1000	1.231700
1250	1.229600
1500	1.200600
1750	1.172400
2000	1.153300
2250	1.144600
2500	1.122200
2750	1.119600
3000	1.105100
3250	1.085800
3500	1.046400
3750	0.934400
4000	0.888500
4250	0.860100
4500	0.838000
4750	0.817900

```
TrainOutput(global_step=4887, training_loss=1.1753569139106304, metrics={'train_runtime':
3339.9846, 'train_samples_per_second': 11.705, 'train_steps_per_second': 1.463, 'total_flos':
1.1727797728208486e+17, 'train_loss': 1.1753569139106304})
```



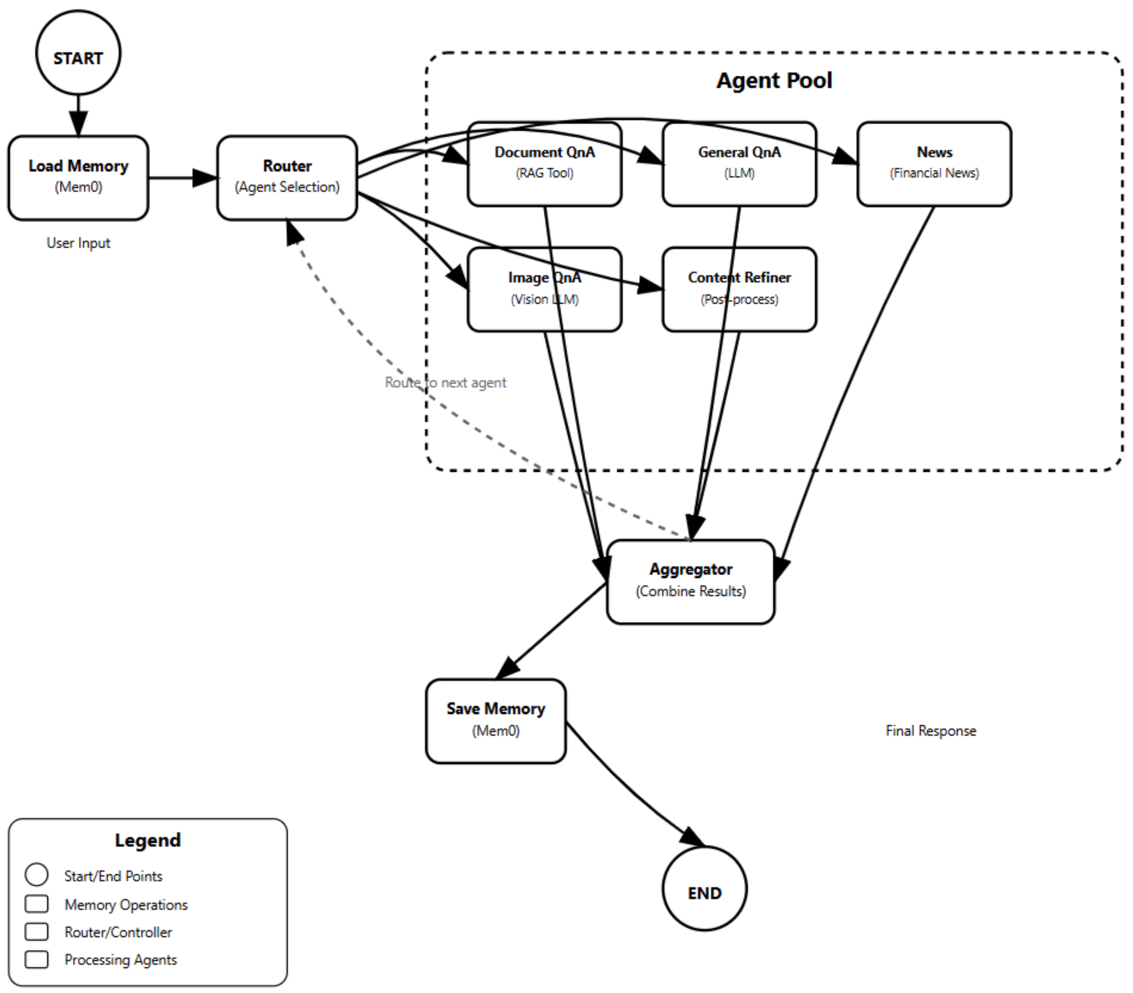
```
🔍 Accuracy: 0.6802721088435374

📊 Classification Report:
              precision    recall  f1-score   support

    negative     0.6031    0.6424    0.6221       920
     neutral     0.8202    0.8301    0.8251       918
    positive     0.6188    0.5728    0.5949       955

    accuracy                         0.6803      2793
   macro avg     0.6807    0.6817    0.6807      2793
weighted avg     0.6798    0.6803    0.6795      2793
```

Multi-Agent System Architecture



Front-End and Back-End Implementation:-

Front-End

The front end of our project is developed using React, a powerful JavaScript library for building user interfaces. React, maintained by Meta, is well-suited for developing fast, interactive, and scalable web applications, making it an excellent choice for our AI-driven finance platform.

Our approach emphasizes modularity and responsiveness, ensuring a user-friendly experience while maintaining clean and maintainable code.

Modular Architecture

React enables us to break down the interface into small, self-contained components. Each component, such as the model selector, chat display, message input, and file upload serves a focused role, allowing for easier debugging, testing, and future expansion.

Interactive User Experience

We make extensive use of React hooks like useState and useEffect to manage the app's state and side effects. This allows real-time updates to the chat interface, instant feedback on user actions, and smooth handling of model switching and file uploads.

Developer Productivity

React's development ecosystem supports a streamlined workflow. Features like live updates and error overlays enable rapid iteration, making the development cycle faster and more efficient.

Routing & Navigation

Page navigation is handled through Reach Router, providing clean and accessible transitions between the main dashboard and the saved chat interface. This keeps the app organized and intuitive for users.

Tailored Design with Tailwind CSS

For styling, we've used Tailwind CSS, which provides utility-first classes for building responsive, accessible layouts. Our UI is handcrafted without external component libraries, offering complete control over the look and feel, including support for light and dark modes

Back-End

Our backend architecture leverages both FastAPI and Express.js, integrating the capabilities of Python and JavaScript environments to manage AI services and general API infrastructure. FastAPI is responsible for handling AI-specific endpoints such as summarization, classification, sentiment analysis, and question answering, taking advantage of its asynchronous processing, type safety, and seamless compatibility with Python-based machine learning tools. Express.js serves as a supporting layer that manages routes, middleware, and utility APIs not directly related to AI tasks. This dual-framework approach allows us to optimize performance and modularize responsibilities across the backend.

AI Model Integration and Inference Strategy

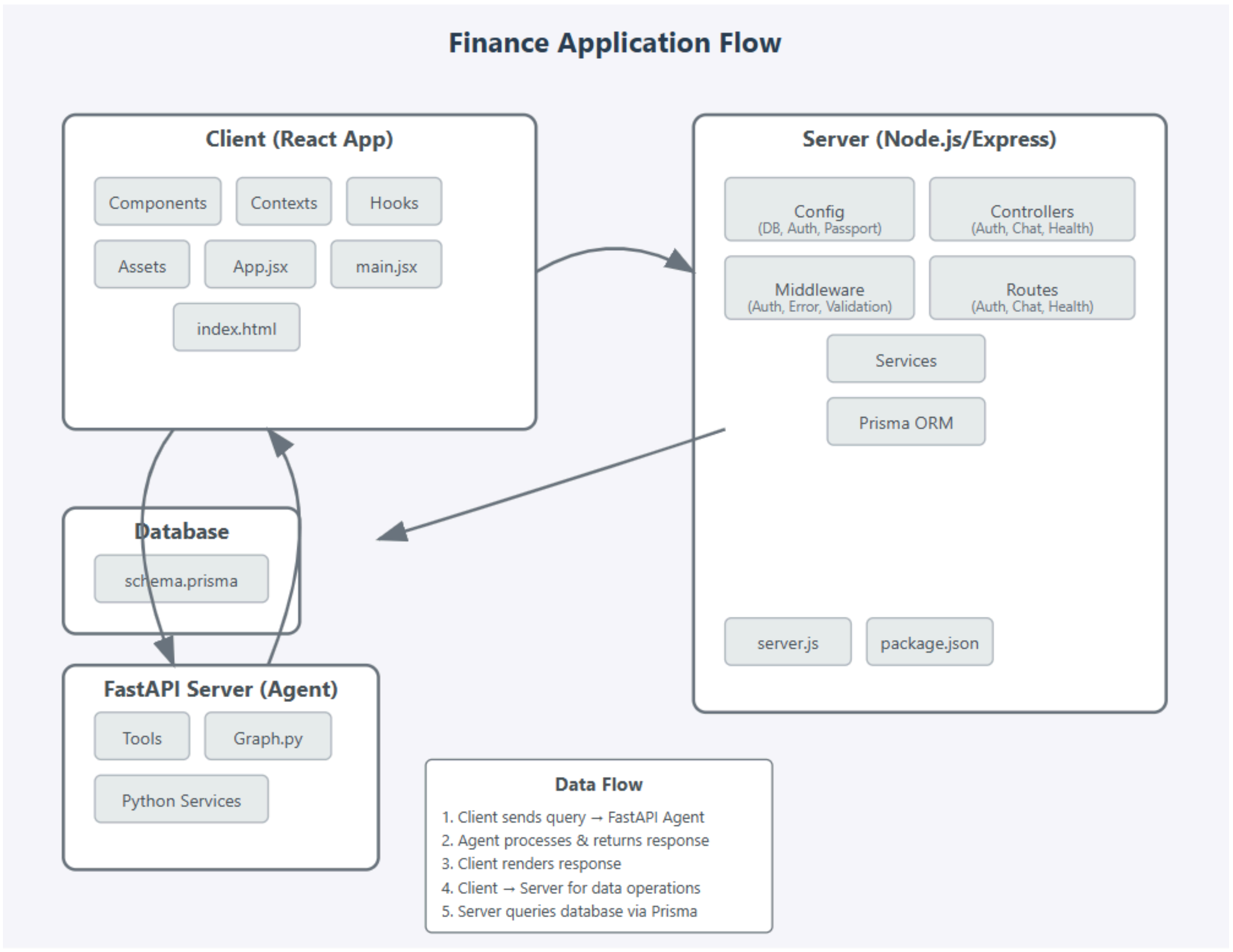
Inference is conducted using a hybrid model setup. Google Gemini 2.5 Flash, accessed via cloud API, handles high-level tasks such as document summarization, financial Q&A, and text generation. For sentiment analysis, a locally fine-tuned LLM is used to provide faster, offline processing with low latency.

Asynchronous Architecture and Developer Efficiency

The backend is fully asynchronous, supports strict type enforcement, and offers auto-generated documentation for developers, streamlining testing and iteration workflows.

Deployment and Data Management with Supabase

All backend services are deployed using Supabase, which provides a serverless environment along with integrated PostgreSQL support. Supabase is used to manage our user data, authentication flows, and persistent chat history through a scalable, cloud-based database solution, eliminating the need for manual database hosting or server provisioning.


Finance Application Flow
Client (React App)
Components
Contexts
Hooks
Assets
App.jsx
main.jsx
index.html
Server (Node.js/Express)
Config
(DB, Auth, Passport)
Controllers
(Auth, Chat, Health)
Middleware
(Auth, Error, Validation)
Routes
(Auth, Chat, Health)
Services
Prisma ORM
server.js
package.json
Database
schema.prisma
FastAPI Server (Agent)
Tools
Graph.py
Python Services
Data Flow
1. Client sends query → FastAPI Agent
2. Agent processes & returns response
3. Client renders response
4. Client → Server for data operations
5. Server queries database via Prisma

3.3 CHALLENGES:

1. **Lack of GPU Resources**
   One of the most significant challenges we faced was the lack of access to dedicated GPU resources during development and model training. Fine-tuning large language models or even running inference efficiently requires powerful hardware, which was either unavailable or too costly for continuous usage. This limited our ability to iterate quickly, forced us to rely on slower CPU-based alternatives during prototyping, and restricted the scale at which we could experiment with different model architectures or datasets.

2. **Module Integration and Compatibility**
   Integrating different modules across frameworks and programming languages introduced multiple compatibility issues. Our stack involved both FastAPI (Python) and Express.js (JavaScript), each handling different layers of the application. Ensuring seamless communication between these environments—while maintaining consistent data structures, request formats, and authentication flows—required careful coordination. Version mismatches and dependency conflicts also posed problems during deployment, especially when working with cutting-edge machine learning libraries and APIs.

3. **Searching for High-Quality Datasets**
   Finding suitable datasets for fine-tuning and evaluation proved to be a time-consuming and difficult process. For specialized tasks like financial question answering or sentiment analysis, publicly available datasets are often limited in scope, outdated, or not representative of real-world queries. We had to explore a wide range of open-source platforms, academic corpora, and synthetic data generation techniques to ensure relevance and quality, which delayed the model training pipeline.

4. **Fine-Tuning Large Models**

   Fine-tuning large language models such as LLaMA or Mistral introduced computational and logistical hurdles. These models require substantial memory, compute power, and carefully tuned hyperparameters. Without access to enterprise-level infrastructure, we had to optimize our training routines using techniques like low-rank adaptation (LoRA) and mixed precision training. Even then, training was often slow, prone to crashes, and required regular checkpointing to avoid data loss or wasted computation.

5. **Deploying Fine-Tuned Models on a Budget**

   Deploying both the front-end and back-end to serve our fine-tuned models posed a financial challenge, particularly due to hosting costs associated with platforms like Hugging Face. Hugging Face's free tier does not support persistent inference endpoints for custom models, which meant we either had to pay for model hosting or find alternative self-hosted solutions. This impacted our deployment timelines and led us to explore cost-effective options like running local inference servers or integrating with cloud platforms offering more generous compute quotas.

CHAPTER 4
RESULT AND DISCUSSION

4.1 RESULT

1. Document Summarization

One of the core features of our system is the ability to generate concise, accurate summaries from long financial documents. Users can upload PDFs or text files, and the system extracts key information using advanced language models. This reduces cognitive load and helps users grasp the essence of dense legal or financial material in seconds. Summarization is powered by Google Gemini 2.5 Flash, which ensures coherence, factual consistency, and relevance to the domain.

2. Document Question Answering (DocQA)

The platform supports a robust Document Q/A module, enabling users to ask specific, context-driven questions based on the content of their uploaded documents. For example, users can query terms in a contract, obligations in a legal notice, or revenue highlights in a report. The answers are directly grounded in the source text using Retrieval-Augmented Generation (RAG), ensuring accuracy and traceability. This interactivity transforms static documents into dynamic, explorable resources.

3. Financial Law Understanding

We extended the system's capabilities to cover the interpretation and explanation of financial laws and regulations. Using fine-tuned language models and curated datasets, users can input legal clauses, provisions, or policy documents and receive plain-language explanations or summaries. This is particularly useful for startups, compliance teams, or individuals navigating complex financial rules without legal assistance.

4. Financial News Analysis

Our system also ingests and processes financial news articles, allowing users to analyze trends, extract sentiment, and ask questions about recent events in the financial world. The models can detect tone, provide market context, and summarize the main points of news items. This functionality helps users stay updated and make informed decisions based on current developments.

5. Backend and Model Integration

All AI functionalities are served through a hybrid backend architecture involving FastAPI (for ML tasks) and Express.js (for general utilities). We use Google Gemini for high-level tasks and a locally fine-tuned LLM for sentiment analysis and offline response generation. The backend is fully asynchronous, ensuring fast and reliable interactions across all services.

6. Front-End Interface and User Experience

The front-end interface is designed to be intuitive and professional. Users can upload documents, enter questions, view summaries, and scroll through persistent chat history. The UI supports model switching and light/dark modes, offering a smooth, accessible experience.

7. Data Management and Deployment

All user data, documents, and chat histories are managed via Supabase with integrated PostgreSQL support. Supabase also handles authentication and deployment, providing a secure, scalable infrastructure without requiring manual server configuration. This enabled us to maintain a cloud-native deployment pipeline across all environments.

4.2 DISCUSSION

Accuracy and Output Effectiveness

The Finance GPT system demonstrated strong performance across a wide range of financial tasks, including document summarization, question answering, legal interpretation, and financial news analysis. Uploaded documents ranging from regulatory texts to budget reports were effectively summarized by the system, preserving core intent, figures, and contextual nuance. Google Gemini 2.5 Flash handled this task with high linguistic precision, producing concise summaries that remained faithful to the original content.

In the document question answering (DocQA) module, the system accurately responded to user queries by grounding its answers in the uploaded content. Even in cases where documents exceeded token limits, the system leveraged chunking mechanisms to ensure complete coverage. The hybrid backend setup using cloud models for high-level NLP and a locally fine-tuned LLM for sentiment analysis ensured low-latency performance without compromising output quality.

When interacting with legal documents, especially financial laws and regulatory clauses, the system was able to generate plain-language explanations, identify obligations, and detect key provisions. This was especially useful for non-technical users needing clarity on compliance or policy matters. Similarly, the financial news analysis module performed well in identifying trends, summarizing articles, and generating sentiment-driven insights from real-time events.

Overall, the output across all modules was consistent, contextually aware, and robust enough for real-world usage. The system maintained performance stability even under asynchronous, multi-user conditions an outcome attributed to its modular design and effective model orchestration.

Comparison with Existing Solutions

Conventional financial tools typically focus on narrow tasks, such as static classification or template-based OCR. These systems often fail to generalize when faced with unstructured, complex, or unfamiliar document formats. In contrast, our Finance GPT solution integrates transformer-based models capable of understanding deep semantic structures, contextual relationships, and domain-specific vocabulary.

Unlike many legacy systems that require rigid rule configurations, our project employs dynamic AI modules for each task. For instance, Gemini 2.5 Flash excels at summarization and document Q/A, while our fine-tuned sentiment model handles subjective evaluations, particularly in financial news. This modularity offers flexibility, allowing us to allocate the right model for the right task without overfitting a single model to all use cases.

Furthermore, in areas like law and regulation, where traditional tools struggle with ambiguity and legal phrasing, our system's NLP-backed reasoning provides better interpretability and guidance. This is particularly valuable for startups or small teams without access to dedicated legal departments.

Compared to older, one-dimensional document processing tools, this project provides a comprehensive end-to-end pipeline from understanding financial content and answering questions to analyzing real-time data and explaining complex legal materials. The outcome is a responsive, AI-driven assistant that surpasses conventional tools in both functionality and adaptability.

Contributions to the Field

Automation of Financial Document Understanding

The system automates the analysis of financial documents by enabling summarization and question answering on legal, regulatory, and financial content. This reduces manual effort and improves access to key insights.

Legal and News Analysis Capabilities

It extends beyond static documents to support real-time financial news interpretation and legal clause explanation. This enhances usability for compliance, research, and decision-making tasks.

Modular Model Architecture

A flexible model architecture allows real-time switching between cloud-based and locally fine-tuned models. This supports task-specific optimization for accuracy, speed, and cost-efficiency.

End-to-End Document AI Deployment

The project demonstrates a full pipeline from data ingestion to structured output using state-of-the-art LLMs like Gemini and fine-tuned sentiment models, making it a practical reference for real-world applications.

Scalable and Maintainable Infrastructure

The backend uses FastAPI and Express.js with Supabase for authentication and data storage. Integrated with a React frontend, the system is asynchronous, scalable, and ready for future extensions in finance and legal tech.

Chapter 5: Conclusion and Future Scope

Conclusion

This project presents a practical and modular AI system designed to simplify the understanding of financial documents, legal texts, and financial news. By combining document summarization, question answering, legal clause interpretation, and financial sentiment analysis into a unified platform, the system demonstrates how modern language models can transform raw financial data into actionable insights. The hybrid backend, leveraging both cloud APIs like Gemini and locally fine-tuned models, offers a balance of scalability, efficiency, and adaptability. With a clean frontend interface, persistent chat history, and secure data handling via Supabase, the platform offers a complete, user-friendly solution for financial content analysis.

The system stands out for its ability to handle diverse tasks with domain-specific intelligence, real-time interactivity, and model flexibility, making it suitable for professionals, researchers, and organizations dealing with complex financial information. Its successful deployment across both frontend and backend layers further highlights its real-world readiness and maintainability.

Future Scope

While the current implementation covers key functionalities, there are several areas for future enhancement:

- OCR and Image Document Support: Adding OCR capabilities would enable processing of scanned documents, expanding usability to physical reports and handwritten financial forms.

- Multilingual Support: Incorporating translation and multilingual NLP would make the platform accessible to users across different regions and regulatory systems.

- Voice and Mobile Integration: A voice-enabled interface or mobile app can improve accessibility, especially for on-the-go professionals.

- Real-Time Financial Data Feeds: Integrating live financial data sources and APIs can enhance news analysis and keep insights continuously updated.

- Custom Fine-Tuning Interface: Allowing users to upload their own data and fine-tune models on the platform would enable organisation-specific customisation.

- Enterprise Integration: With improved authentication layers and dashboard features, the platform can be extended to integrate with enterprise tools used in fintech, compliance, and auditing workflows.

Chapter 6: References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … & Polosukhin, I. (2017). *Attention is all you need*. In Advances in Neural Information Processing Systems (NeurIPS). https://arxiv.org/abs/1706.03762
2. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., … & Liu, P. J. (2020). *Exploring the limits of transfer learning with a unified text-to-text transformer*. Journal of Machine Learning Research. https://arxiv.org/abs/1910.10683
3. OpenAI. (2023). *GPT-4 Technical Report*. OpenAI. https://openai.com/research/gpt-4
4. Google DeepMind. (2024). *Gemini 1 and Gemini 2 Technical Overview*. https://deepmind.google/technologies/gemini
5. Hugging Face. (n.d.). *Transformers Documentation*. https://huggingface.co/docs/transformers
6. Supabase. (n.d.). *Open Source Firebase Alternative*. https://supabase.com/docs
7. FastAPI Documentation. (n.d.). *FastAPI: Modern, Fast (High-performance), Web Framework for Building APIs*. https://fastapi.tiangolo.com/
   Express.js Documentation. (n.d.). *Fast, unopinionated, minimalist web framework for Node.js*. https://expressjs.com/
8. LangChain. (n.d.). *LangChain: Framework for Developing Applications Powered by Language Models*. https://www.langchain.com/
9. FAISS. (n.d.). *Facebook AI Similarity Search*. https://github.com/facebookresearch/faiss
10. LoRA: Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L., & Rajpurkar, P. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. https://arxiv.org/abs/2106.09685
11. Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., … & Sutskever, I. (2021). *Zero-shot text-to-image generation*. In Proceedings of ICML.
12. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://arxiv.org/abs/1810.04805
13. Financial Industry Regulatory Authority (FINRA). (n.d.). *Understanding Financial Regulations*. https://www.finra.org/
14. Securities and Exchange Board of India (SEBI). (n.d.). *Legal Framework and Financial Market Regulations*. https://www.sebi.gov.in/