COEN6313 Project: NexoSphere by Group G15

Nishant Kumar Barua
Electrical Computer Engineering
Concordia University
40267821
barua.nishant97@gmail.com

Ismael Barzani
Electrical Computer Engineering
Concordia University
40188139
ismaelmergasori@gmail.com

Zihan Ma
Electrical Computer Engineering
Concordia University
40270361
mazihan567@gmail.com

James Agbonhese
Electrical Computer Engineering
Concordia University
40280162
jamesagbonz@gmail.com

***Abstract*—This project offers a comprehensive solution for real-time financial insights. The Sentiment AI Analysis Tool uses advanced language models to extract nuanced sentiment from various media. By delivering accurate, up-to-date data and sentiment analysis while the Data Aggregation API consolidates stock market data from multiple sources, the platform empowers users to make informed decisions and optimize investment strategies, promoting greater transparency and stability in today's data-driven financial landscape.**

***Index Terms*—Sentiment analysis, Data aggregation, Stock market ticker, Summarization, Cloud Solution.**

I. Introduction

A. Goal

This project develops a system that integrates real-time stock market data with sentiment analysis using the Google T5 model[1]. By combining time-series data from sources like Yahoo Finance and leveraging a robust database for efficient storage, it provides a unified platform to support informed financial decision-making.

B. Objectives

- Retrieve stock-related news using APIs like Financial News API, summarizing content with the Google T5 model from Hugging Face.
- Perform sentiment analysis on summarized news content to assess market sentiment trends and insights.
- Fetch real-time time-series stock data from APIs like Yahoo Finance and EODHD[2], aggregating[3][4] it into a unified dataset for analysis.
- Provide a user-facing API to deliver stock data, sentiment analysis, and personalized recommendations based on stored user preferences.

C. Problem Statement

This project addresses inefficiencies in the financial industry by integrating real-time stock data aggregation with AI-driven sentiment analysis using Google T5. By consolidating diverse data sources, the platform provides actionable insights, helping users identify emerging trends and make informed decisions, transforming financial decision-making in a fast-paced market.

D. Assumptions

- Financial news APIs will deliver timely, relevant stock-related news for sentiment analysis using Google T5 via Hugging Face.
- APIs like Yahoo Finance and EODHD will provide reliable, real-time stock market data for time series analysis.
- Redis caching layer will improve system performance by managing high-frequency queries.
- The system deployment on the AWS cloud platform ensure scalability and high availability.

E. Methodology

We followed an agile, iterative approach to design, develop, and deploy our real-time sentiment analysis and stock data aggregation solution.

- **Data Collection:** We gathered real-time financial news from APIs like Yahoo Finance and EODHD, using time-series queries to ensure data accuracy.
- **Sentiment Analysis and Summarization:** We used the Google T5 model to analyze, summarize, and classify sentiment in financial news, providing insights into trends, risks, and opportunities.
- **NoSQL Database and Caching:** MongoDB hosted on an online cluster stored user preferences, while Redis cached data to enhance retrieval and storage efficiency.
- **API Development:** We developed RESTful APIs with Flask, exposed on SwaggerHub, to provide stock data, sentiment analysis, and personalized insights, integrated with MongoDB and Redis for improved performance.
- **Cloud Deployment and CI/CD:** The system was deployed on AWS for scalability, with GitHub Actions and Terraform automating the CI/CD pipeline for reliable deployment.

F. Schedule

The table presents the iterative overview of the project development life-cycle, from inception to completion, in a Gantt Chart below.

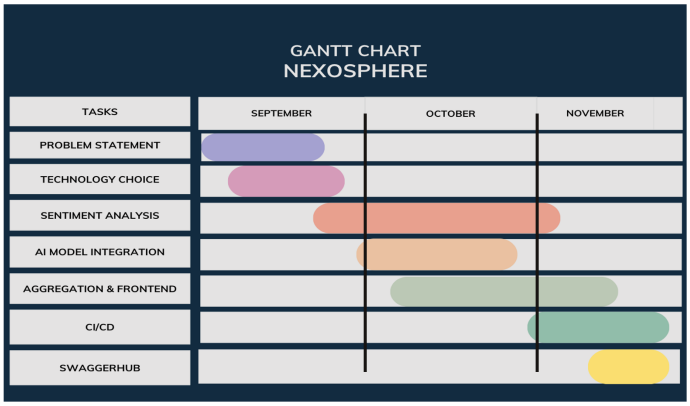

Fig. 1. Schedule Gantt Chart.

G. Data Sources and Technologies

Data Sources

- **Yahoo Finance**: Provides real-time stock market data through its API.
- **EODHD Stock Market and Financial News API**: Supplies historical stock data, financial news, and real-time stock updates.

AI Models

- **Google T5 Small Model**: A pre-trained model from Hugging Face used for summarizing financial news.
- **ProsusAI FinBERT**[5]: A sentiment analysis model tailored to financial data, which provides insights into positive, negative, or neutral sentiment in stock-related news.

Technology Stack

- **Amazon Web Services (AWS - EC2)**: Cloud platform hosting the application on scalable virtual machines.
- **MongoDB**: NoSQL database for managing unstructured data, offering scalability and flexibility.
- **Redis**: In-memory data store used for caching and optimizing data retrieval.
- **Hugging Face**: AI model hosting platform for deploying and serving machine learning models.
- **GitHub**: Version control system to manage code and collaboration across teams.
- **Terraform**: Infrastructure as Code tool for automating resource provisioning on AWS.
- **GitHub Actions**: CI/CD tool for automating deployment pipelines and workflows.
- **Docker**: Containerization platform ensuring consistent and portable application deployment.
- **Python 3.10**: Programming language used for building backend logic and AI integration.

Collaboration Tools and Project Management

- **Discord**: Used for team communication, real-time discussions, and project management.
- **Microsoft SharePoint**: Facilitated document sharing, version control and centralized access to resources.
- **Overleaf**: Used for collaborative editing of the final report and ensuring seamless documentation with LaTeX.

II. Project Description

A. Architectural Design

Our application follows a **monolithic architecture** design, integrating sentiment analysis and stock data aggregation services within a single Flask web app. It uses Free Stock Market APIs to fetch real-time data and **Hugging Face AI models** for sentiment analysis. **MongoDB** stores structured data, while **Redis** caches high-frequency results for improved performance. The solution is deployed on **AWS Cloud** for scalability and reliability, with the entire application containerized using **Docker** for seamless deployment and portability. This architecture enables efficient, real-time financial insights through integrated data aggregation and sentiment analysis.

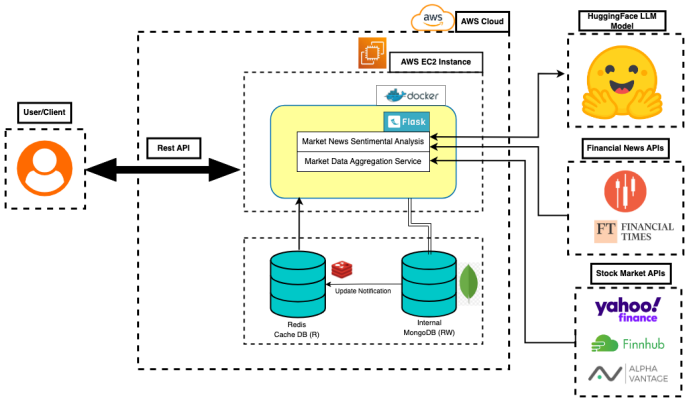

Fig. 2. Architecture Diagram.

B. Technical Implementation Details

Sentiment Analysis

Our sentiment analysis service, built with Python Flask, uses MongoDB for NoSQL storage and APScheduler for periodic news fetching and processing. A free-tier Redis online cache stores sentiment results for popular stocks, optimizing data retrieval and enhancing efficiency.

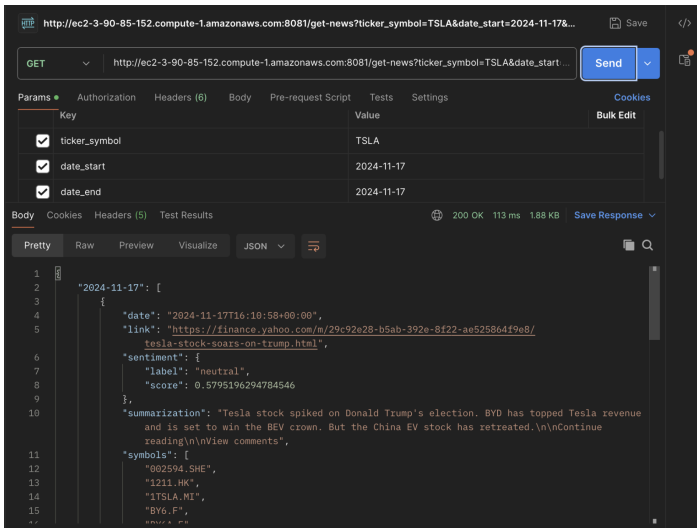

Fig. 3. Endpoint testing on Postman.

The ”NewsServices”class fetches and processes news using the ”get news of date”method, retrieving sentiment analysis and summaries. If data exists in MongoDB, it’s retrieved; otherwise, it triggers a sequence to fetch and process news.

- **News Fetching**: The application pulls raw news content from the EODHD public API.
- **Text Summarization**: This content is summarized using the HuggingFace T5 model.
- **Sentiment Analysis**: The summaries are analyzed for sentiment using the HuggingFace FinBERT model.
- **Storage**: Results are stored in MongoDB.
- **Response**: Processed data is returned to the user.

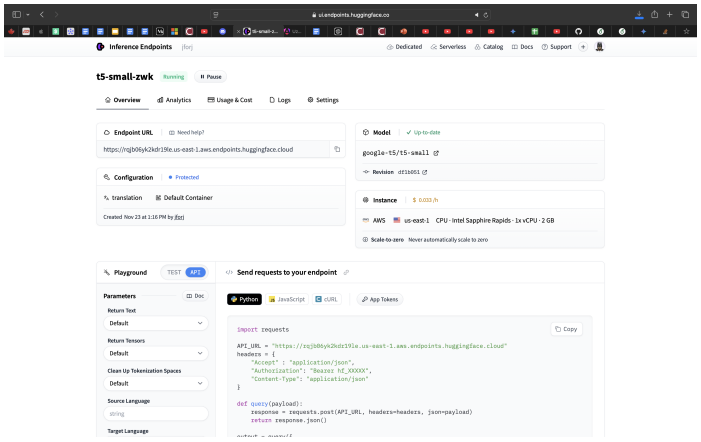

Fig. 4. T5 Model Running.

Additionally, the ”CacheNewsServices”class runs hourly to fetch and process news using ”NewsServices.”It stores results for five popular stocks from the past five days in MongoDB and Redis, automatically purging older data.

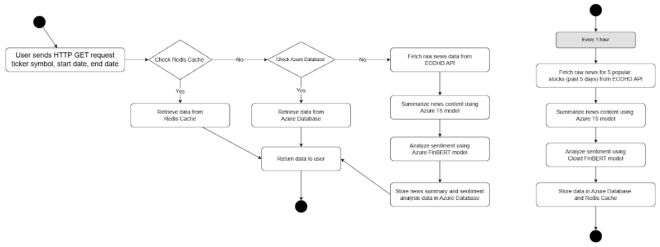

Fig. 5. Activity Diagram.

User Story V1:

- Title: Retrieve News Sentiment Analysis and Summary for Specific Stock
- Story: As a financial analyst, I want to get summarized news and sentiment analysis for a stock within a date range to make informed decisions.
- Acceptance Criteria: The API returns sentiment analysis and summary for the requested stock, covering all news between the specified dates.

User Story V2:

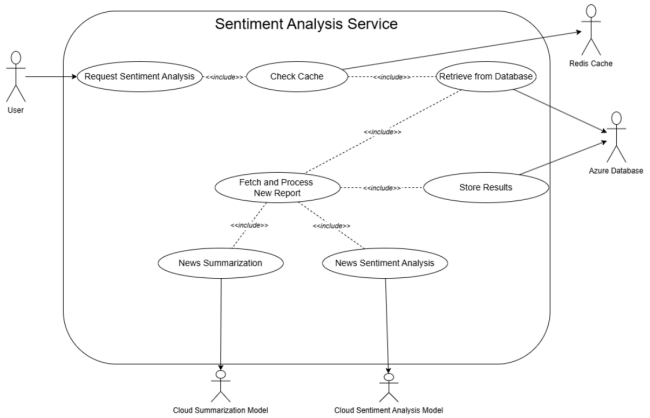

Fig. 6. Use Case Diagram.

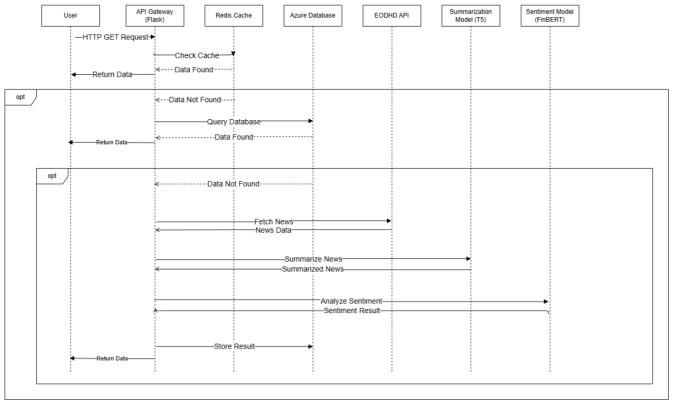

Fig. 7. Sequence Diagram.

- Title: Integrate Sentiment Analysis API into Application
- Story: As an app developer, I want to integrate the sentiment analysis API to provide users insights into financial news sentiment.
- Acceptance Criteria: The API returns sentiment analysis and summary for the requested stock, covering news within the specified date range.

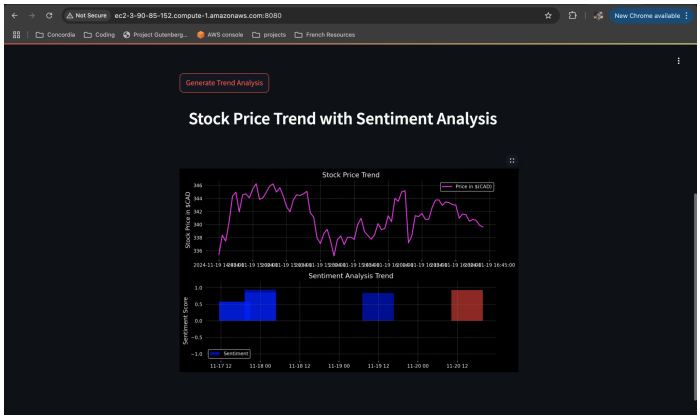

Fig. 8. Sample Use Case Front-end.

Sample Front End Use Case We developed a sample front-end that takes a date range and stock ticker symbol, generates

stock price fluctuations, analyzes sentiment news within the timeframe, and displays the data on a graph, helping analyze the current trend and status of the selected stock.

C. Extra Features

- **Expose APIs to SwaggerHub:** We have exposed our APIs to swaggerhub, link already shared above. This provides a centralized platform for API documentation, enabling easy collaboration, clear communication of API functionality, and simplified integration for developers across different teams.

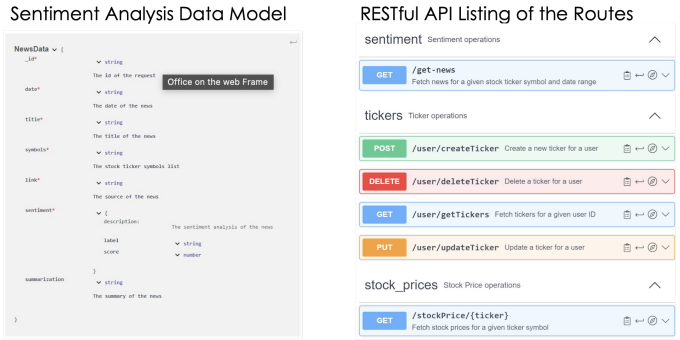

Fig. 9. Sentiment Analysis Data Model and SwaggerHub Doc.

- **CI/CD Pipeline:** CI/CD pipeline developed using GitHub Actions, Terraform and Docker to automate and streamline the deployment of our Flask app[6] on AWS. The pipeline has two main stages:
  - Provisioning Infrastructure on AWS: Terraform automates the creation of AWS resources, such as an EC2 instance, ensuring the environment is reproducible and easily managed as infrastructure as code (IaaC).
  - Deploying the Dockerized Flask Application: We build and deploy a Docker container for our Flask app to the EC2 instance, ensuring consistency across environments and isolating dependencies for faster, reliable deployments.

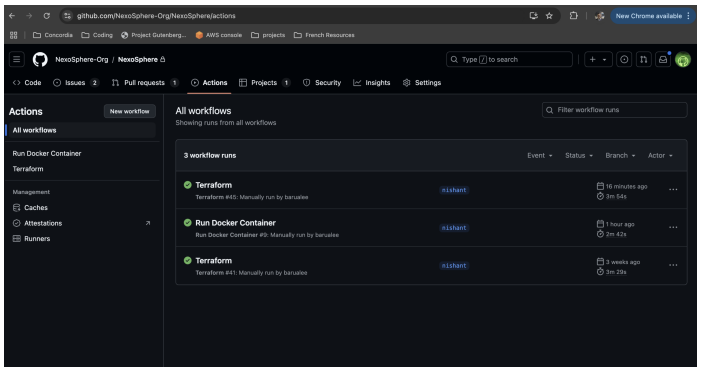

Fig. 10. CI/CD Pipeline in Action.

Project URLs

- Team Github Repository
- Application FrontEnd
- swaggerhub API
- Stockprice api
- sentiment API

D. Team Contribution

Task List	40188139 (Ismael B.)	40270361 (Zihan M.)	40280162 (James A.)	40267821 (Nishant B.)	Total
Problem Description	25%	20%	30%	25%	100%
Tech Stack & Review	25%	20%	30%	25%	100%
Data Model	20%	20%	30%	30%	100%
Sentiment Analysis Service	10%	65%	15%	10%	100%
AI Model Integration	10%	50%	30%	10%	100%
Data Aggregation Service	60%	10%	15%	15%	100%
Frontend	25%	10%	25%	40%	100%
CI/CD Deployment	35%	5%	10%	50%	100%
Reporting & Documentation	15%	25%	40%	20%	100%

Fig. 11. Team Contribution Table.

III. Conclusion

A. Observations and Lessons Learnt

- **Data Acquisition and Quality**: Unavailability of free/affordable real-time data sources that are free of request and access limits.
- **Model Complexity and Performance**:Choosing the right model architecture and hyper-parameters can be complex.
- **CI/CD Deployment**:Automated deployment using CI/CD, utilize GitHub secrets to securely store environment variables and ssh keys.
- **AI Model**:Consciously terminate the launched Hugging Face AI models to avoid incurring extra charges.
- **Lack of User Authentication**:All incoming requests without user authentication, potentially leading to unauthorized access and system overload.
- **Data Processing Delay**:Sentiment analysis for less popular stocks can result in longer processing times.

B. Quality Attributes

- **Availability**: AWS's global network ensures high availability and low-latency access, while MongoDB provides automatic replication and failover, distributing data across nodes to prevent single points of failure.
- **Performance**: Redis caches sentiment analysis results, improving performance by reducing database load and enhancing data retrieval speed alongside MongoDB.
- **Scalability**: AWS EC2 supports vertical and horizontal scaling, while MongoDB uses sharding to distribute data across servers, allowing seamless scaling to handle growing workloads.

C. Strengths and Limitations

Strengths

- **Flexible Data Models**:MongoDB's document-based model allows for dynamic, flexible schema changes, making it ideal for applications dealing with semi-structured or evolving data.

- **High-Performance Caching**:Redis operates as an in-memory database, providing fast data access by storing information in memory; highly effective for caching.
- **Pre-Trained AI Models**:Hugging Face offers pre-trained AI models, improving development time and eliminating the need for training from scratch.
- **Collaboration and Automation**:GitHub provides version control and collaboration features. GitHub Actions automates CI/CD pipelines, enabling seamless workflows for building and deploying applications. Docker offers portability and consistency.

Potential Limitations

- **Memory Constraint**:Redis, being in-memory, is limited by available RAM and managing growing data can be challenging without proper management policies.
- **Deployment Costs**:Hosting large AI models on Hugging Face are costly, particularly for real-time applications.
- **Latency**:Hugging Face models experience latency in real-time, under high-volume traffic or models requiring high compute power.

REFERÊNCIAS

[1] Huggingface. *google-t5*. 2024. URL: https://huggingface.co/google-t5/t5-small.

[2] eodhd.com. *Financial News Feed and Stock News Sentiment data API*. 2024. URL: https://eodhd.com/financial-apis/stock-market-financial-news-api.

[3] Investopedia.com. *Aggregation: Meaning, Importance, Effects*. 2022. URL: https://www.investopedia.com/terms/a/aggregation.asp.

[4] Segment.com. *Guide to Data Aggregation for Financial Services*. 2024. URL: https://segment.com/data-hub/data-aggregation/financial-services/.

[5] HuggingFace. *ProsusAI/finbert*. 2024. URL: https : / / huggingface.co/ProsusAI/finbert.

[6] Pallets. *Project Layout*. 2010. URL: https : / / flask . palletsprojects.com/en/stable/tutorial/layout/.