NexoSphere

Stock Market Sentiment Analysis and Data Retrieval Tool

COEN 6313 PROJECT PRESENTATION

GROUP - #G15

Ismael Barzani (40188139)

Zihan Ma (40270361)

James Agbonhese (40280162)

Nishant Kumar Barua (40267821)

UNIVERSITÉ Concordia UNIVERSITY

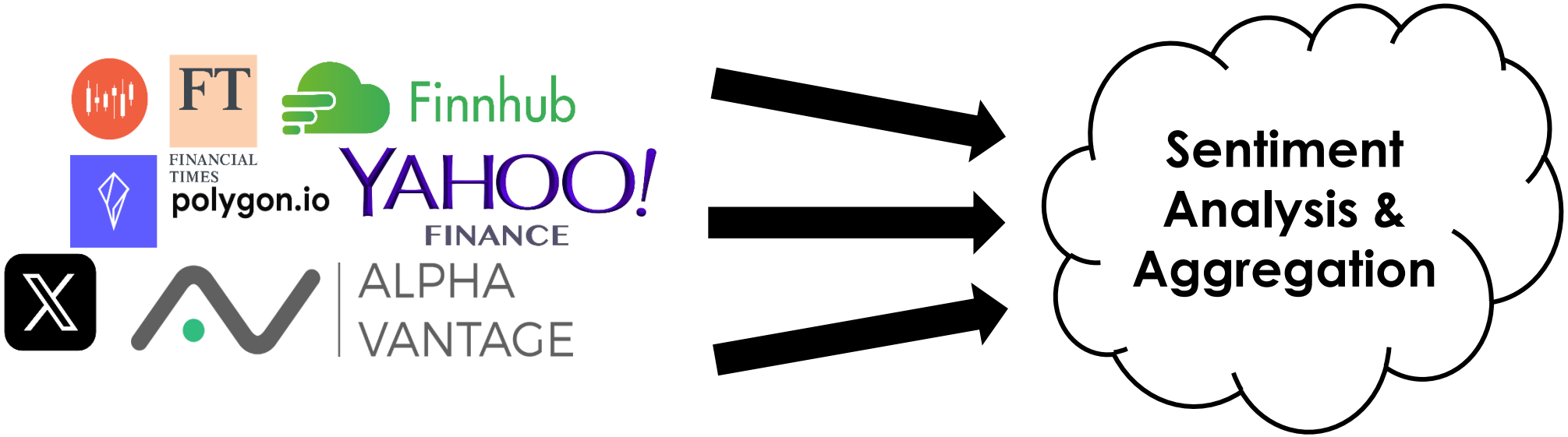
OUTLINE

1) Functionalities
2) Architecture Design
3) Technical Choices
4) Data Model
5) Implementation
6) Project Schedule
7) Quality Attributes
8) Strengths & Limitations
9) Observations & Lessons Learnt
10) Member Contributions

PROBLEM STATEMENT

→ Financial market participants struggle to efficiently access and analyze fragmented stock market data.

→ Investors lack tools to quickly gauge market sentiment from vast amounts of financial news.



MAIN FUNCTIONs OF OUR SERVICE

Stock Market Data Retrieval and Sentiment Analysis

- Provision of Data Retrieval and Retrieval Service
- Provision of a Comprehensive Sentimental AI Analysis Tool

Stock Data Retrieval

- Retrieving ***real-time*** stock market data from ***multiple*** sources or ***public API endpoints.***
- Combining these retrieved datasets into one object that **encompasses** all the data.

Stock News Sentiment Analysis

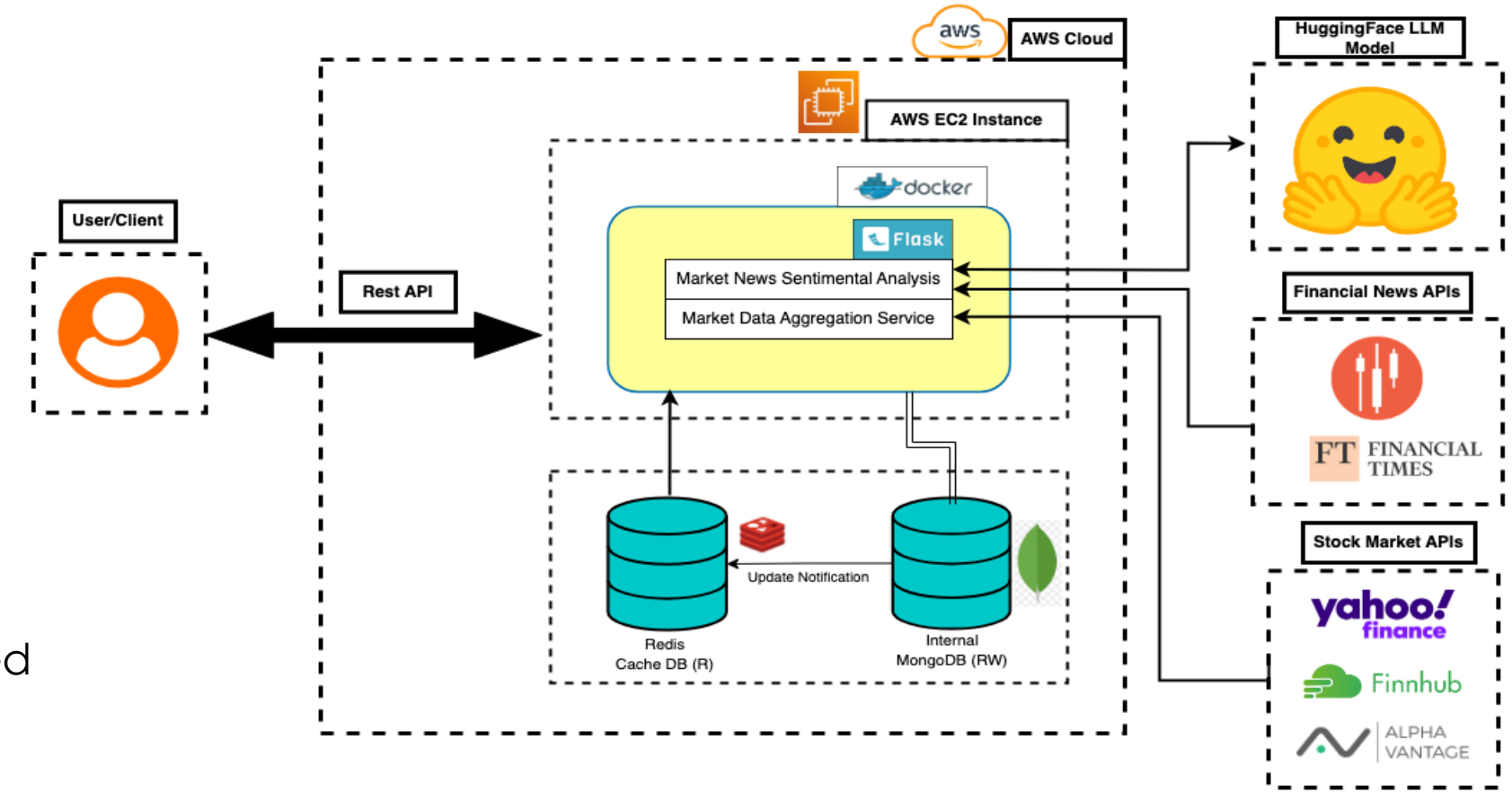
- Retrieving ***real-time stock news and reports*** from Financial News API.
- Performing ***sentiment analysis*** on the summarized content, using pre-trained AI model.

TECH STACK

- **Cloud Platform:** Amazon Web Service (AWS) - EC2
- **NoSQL Database:** MongoDB, Redis
- **AI Model Hosting:** Hugging Face
- **CI/CD & VCS:** GitHub, Terraform, GitHub Actions, Docker
- **Programming Language:** Python 3.10

ARCHITECTURE DESIGN

- **Monolithic Architecture**
  - Consolidated Deployment
- **RESTful API:**
  - Statelessness
  - Unified Interface
  - Client-server based



IMPLEMENTATION - Sentiment Analysis Service

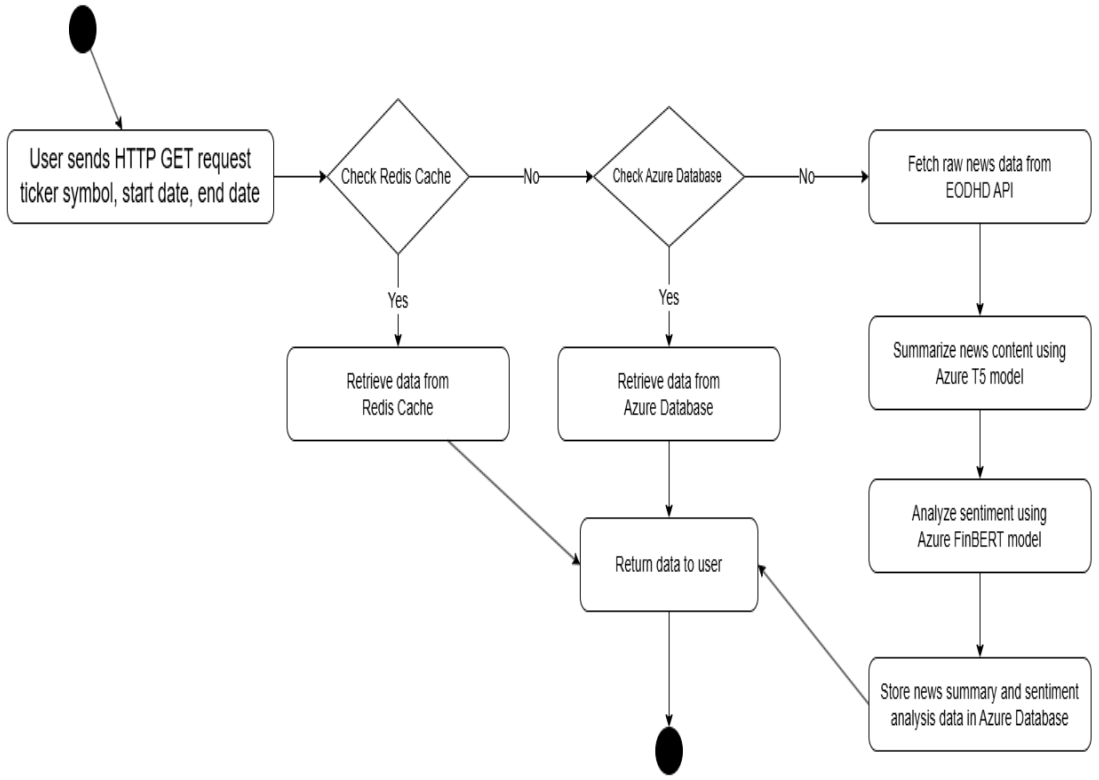
Service: Provide summary and sentiment analysis to financial news

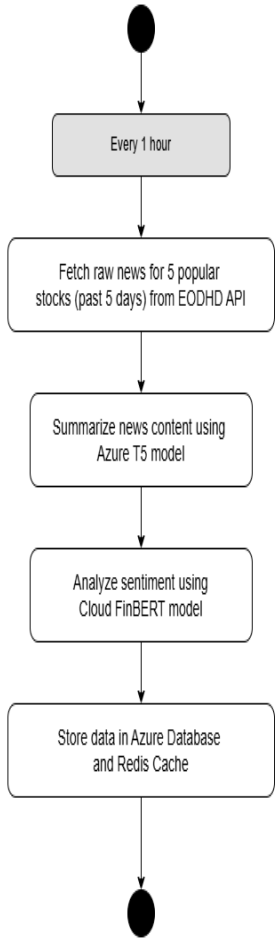
Input: HTTP GET request, specifying ticker symbol, start_date and end_date
Output: summarized news and sentiment analysis for the stock over the given date range

News Processing Pipeline:
- News fetched from public API
- Summarized by Cloud T5 model
- Sentiment analyzed by Cloud FinBERT model

Storage and Caching:
- Processed data stored in (MongoDB)
- Frequently accessed data stored in Redis Cloud

IMPLEMENTATION - Sentiment Analysis Service





Activity Diagrams for Sentiment Analsysis workflows

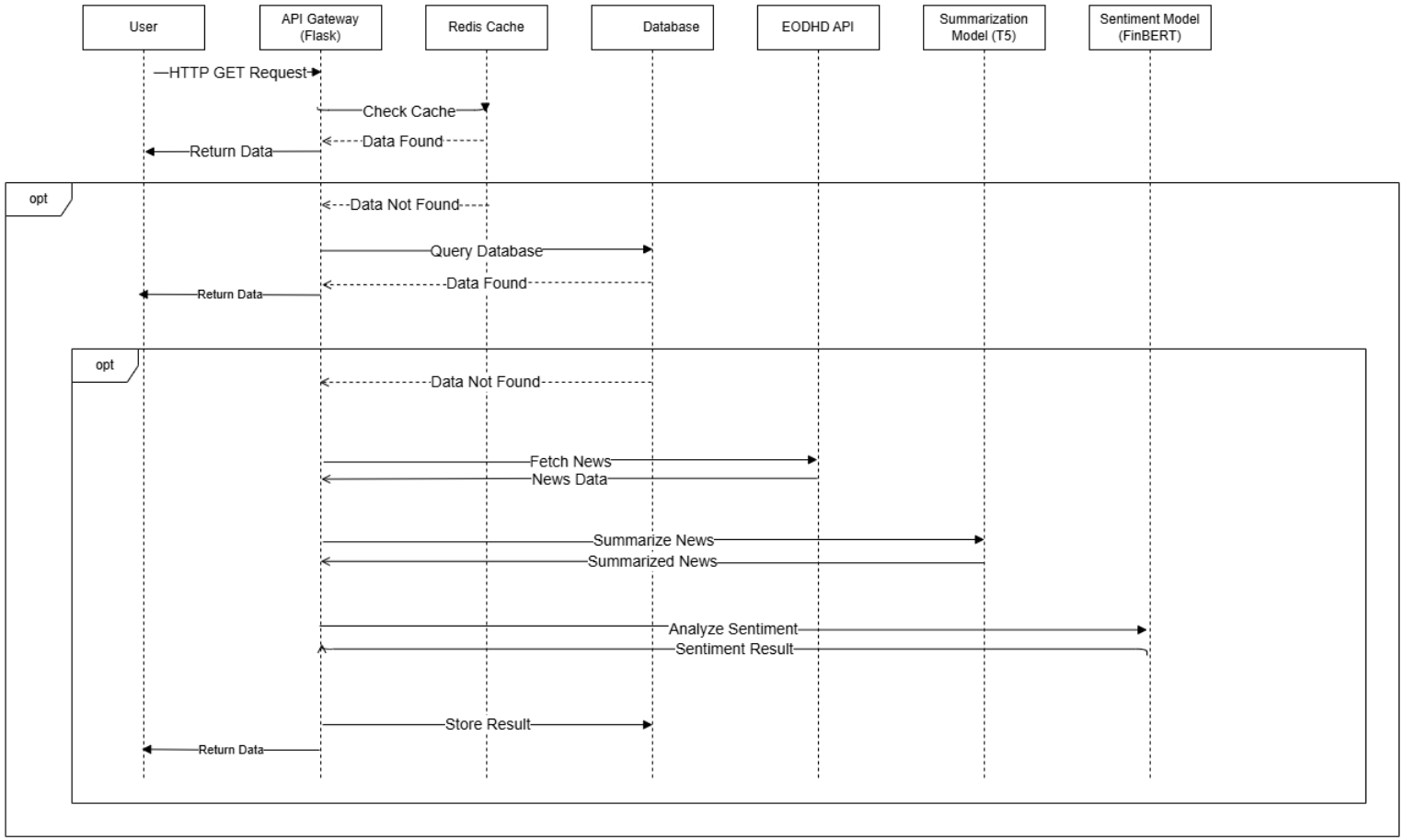
Data lookup and processing workflow:

1. API gateway receives request
2. Lookup Redis Cache
3. Lookup MongoDB
4. News Fetching and Processing:
5. Fetches raw financial news from EODHD API
6. Summarizes and sentiment analyzes news
7. Store results to MongoDB, return result.

Data Caching workflow:

1. Triggered by scheduler hourly.
2. Read popular stock symbols from MongoDB
3. Fetches and Processes news for last 5 days
4. Stores results to MongoDB
5. Caches results to Redis Cloud, delete outdated cache

IMPLEMENTATION - Sentiment Analysis Service



Sentiment Analysis Sequence Diagram

Core Classes:

NewsService:

1. Handles database interactions.
2. Handles API calls
3. Handles data processing

CacheNewsService:

1. Caching & Reading data
2. Managing outdated cache

Data Management:

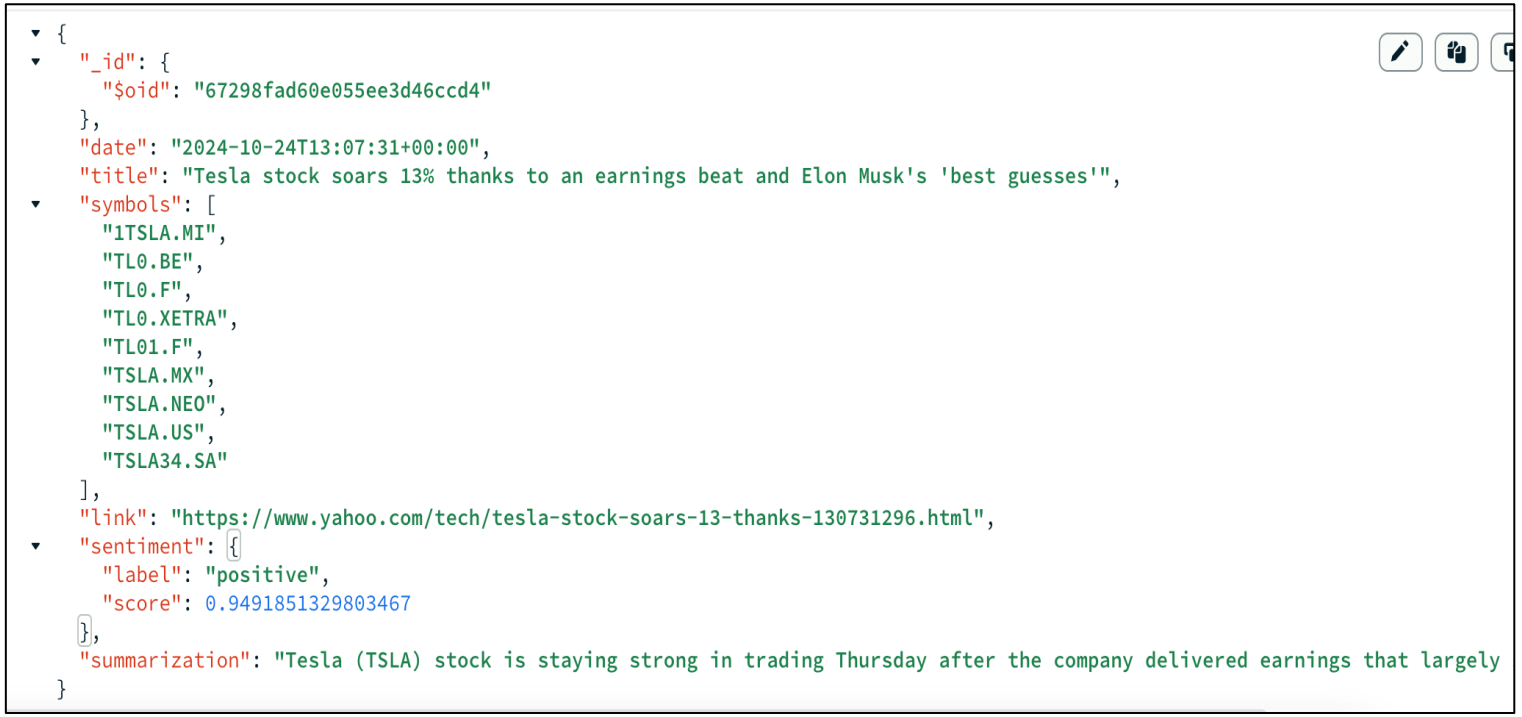
MongoDB:

1. Stores news Labels, summary, sentiment analysis, news URL.
2. Stores list of popular stocks for caching

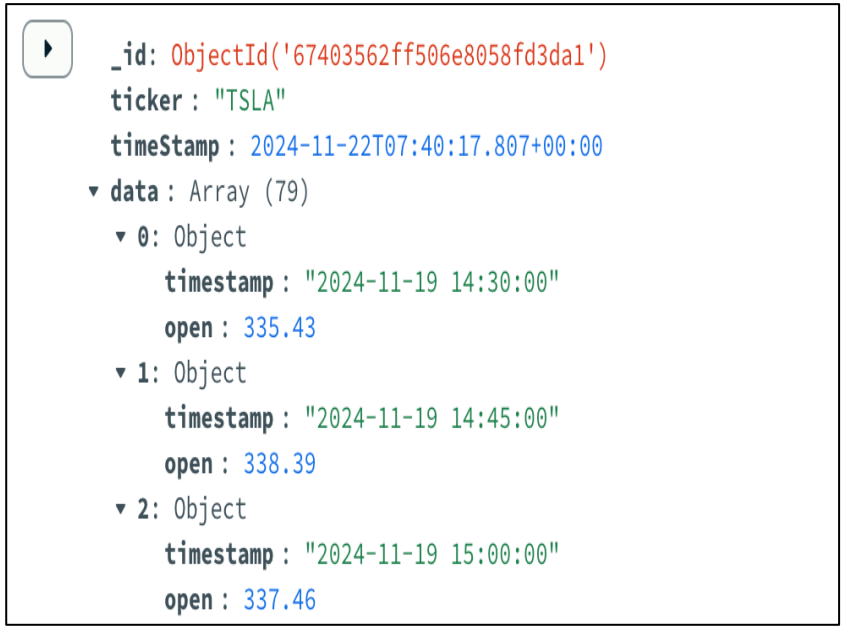
Redis Cache:

1. Stores data in JSON
2. Categorized by date

DATA MODELS



Sentiment Analysis Data Model



Ticker Data Model

Université Concordia University

IMPLEMENTATION – *Swagger Hub (Extra Features)*

Sentiment Analysis Data Model

NewsData {

_id*	string	The id of the request
date*	string	The date of the news
title*	string	The title of the news
symbols*	string	The stock ticker symbols list
link*	string	The source of the news
sentiment*	{ description:	The sentiment analysis of the news
	label	string
	score	number
	}	
summarization	string	The summary of the news

}

RESTful API Listing of the Routes

sentiment Sentiment operations

- GET `/get-news` Fetch news for a given stock ticker symbol and date range

tickers Ticker operations

- POST `/user/createTicker` Create a new ticker for a user
- DELETE `/user/deleteTicker` Delete a ticker for a user
- GET `/user/getTickers` Fetch tickers for a given user ID
- PUT `/user/updateTicker` Update a ticker for a user

stock_prices Stock Price operations

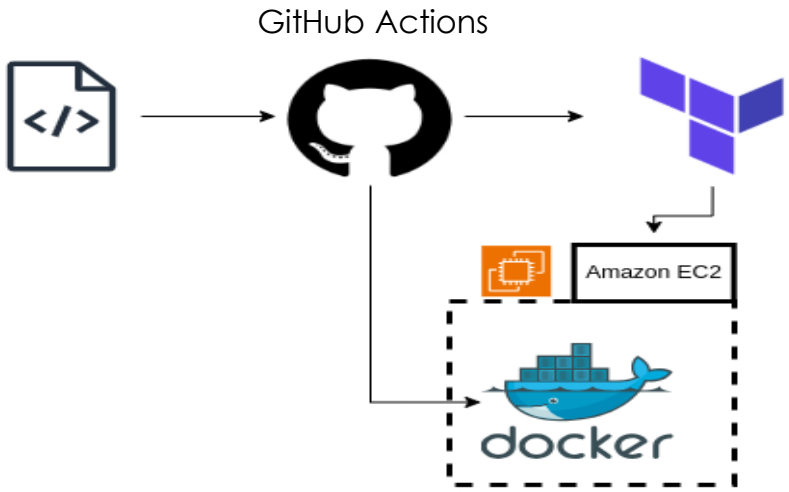
- GET `/stockPrice/{ticker}` Fetch stock prices for a given ticker symbol

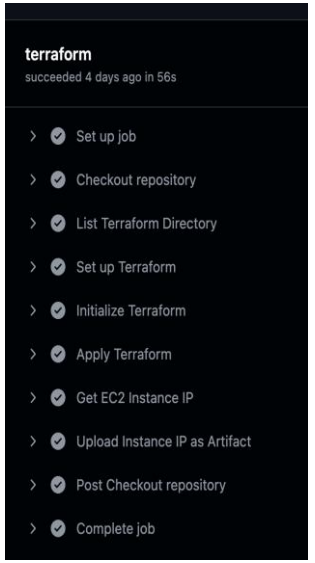
UNIVERSITÉ Concordia UNIVERSITY

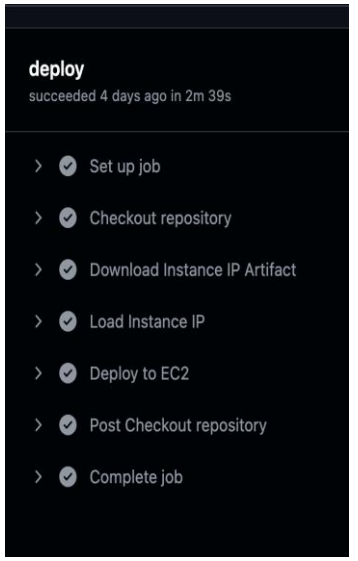
IMPLEMENTATION – CI/CD Deployment (Extra Features)

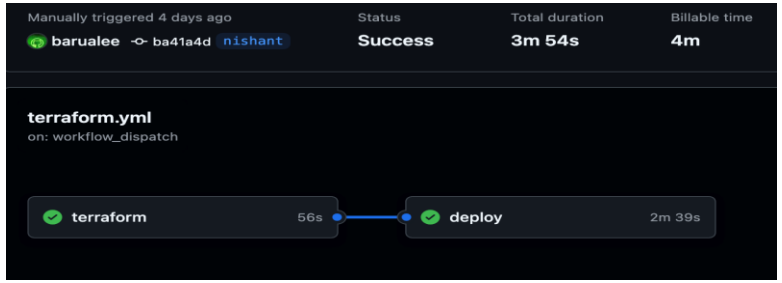
To Implement CI/CD, we have used the following workflow using GitHub Actions:

- Checkout source code
- Terraform to instantiate our EC2 cloud instance on AWS
- Deploy and run the docker container









QUALITY ATTRIBUTES

- **Availability:** AWS's global network of data centres ensures high availability and low-latency access to users across different regions. Similarly, MongoDB ensures high availability through automatic replication and failover, distributing data across multiple nodes to prevent a single point of failure.
- **Performance:** We are leveraging Redis as a caching solution on top of our MongoDB solution to store our analysis results, which improves performance.
- **Scalability:** AWS EC2 offers both vertical and horizontal scaling, enabling rapid adjustment of resources to match varying workload demands. MongoDB supports horizontal scaling through sharding, distributing data across multiple servers as it grows. Docker containers can be easily scaled using orchestration tools like Kubernetes, allowing applications to scale seamlessly based on demand.

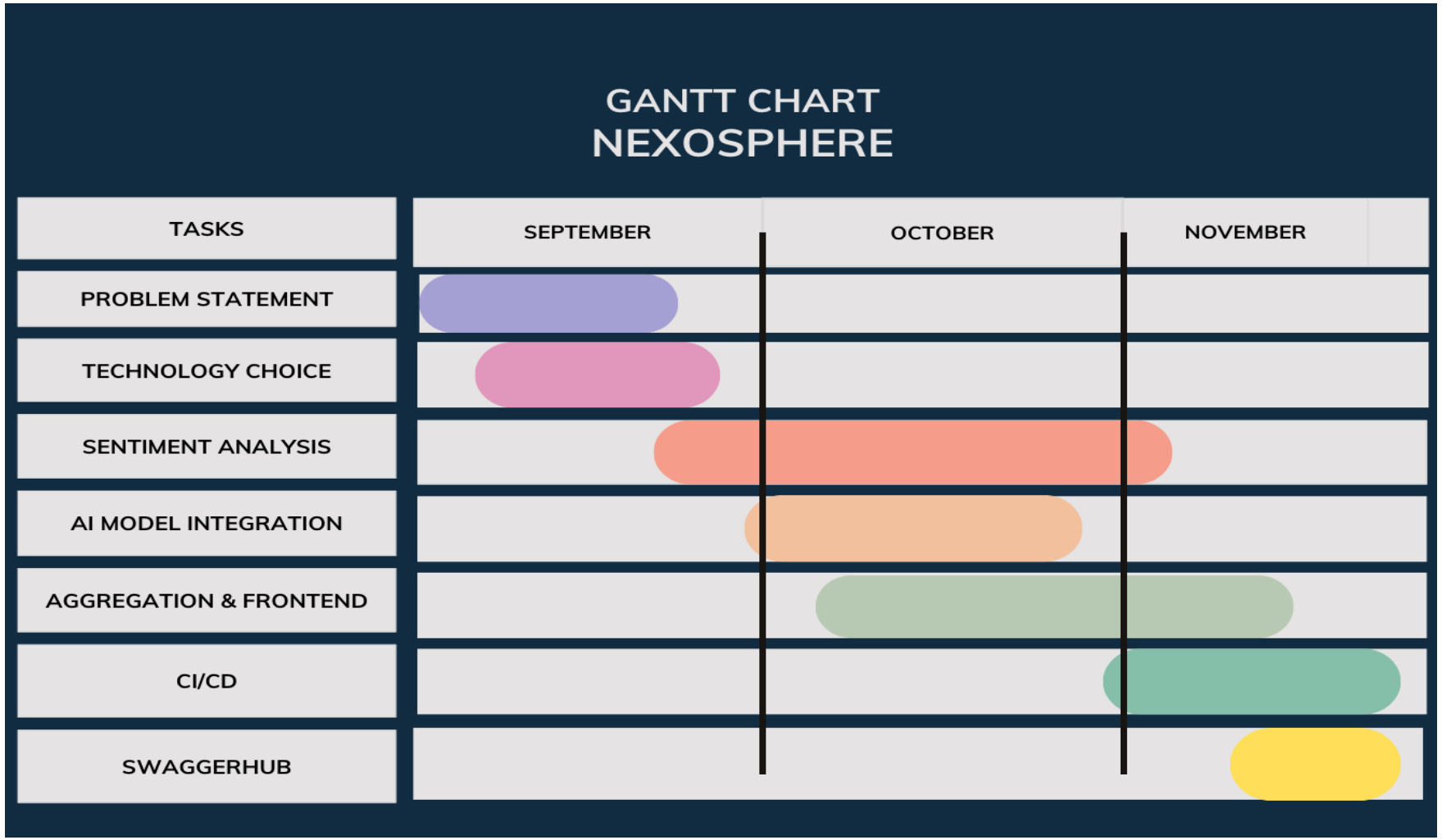
OBSERVATIONS & LESSONS LEARNT

- **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 hyperparameters 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**: Accessing sentiment analysis for less popular stocks can result in longer processing times.

STRENGTHS & LIMITATIONS

- **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.
- **Powerful Querying and Analytics**: MongoDB provides advanced querying and aggregation pipelines.
- **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.
- **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.

SCHEDULE & PROJECT MANAGEMENT



MEMBER CONTRIBUTIONS

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%

Links

- https://github.com/NexoSphere-Org/NexoSphere
- http://ec2-35-153-83-207.compute-1.amazonaws.com:8080/
- http://ec2-35-153-83-207.compute-1.amazonaws.com:8081/get-news?ticker_symbol=MSFT&date_start=2024-11-20&date_end=2024-11-25
- http://ec2-35-153-83-207.compute-1.amazonaws.com:8081/user/getTickers?userId=12347
- http://ec2-35-153-83-207.compute-1.amazonaws.com:8081/stockPrice/TSLA?start_date=2024-11-17&end_date=2024-11-21
- https://app.swaggerhub.com/apis/ISMAELRIDHA/nexosphere_api/1.0-oas3

THANK YOU

NexoSphere

UNIVERSITÉ Concordia UNIVERSITY