Advanced Web and Big Data Technology

CST3130

Stock Trading Application

2023

By

Omohan Samuel Imoisili
M00846543

Description

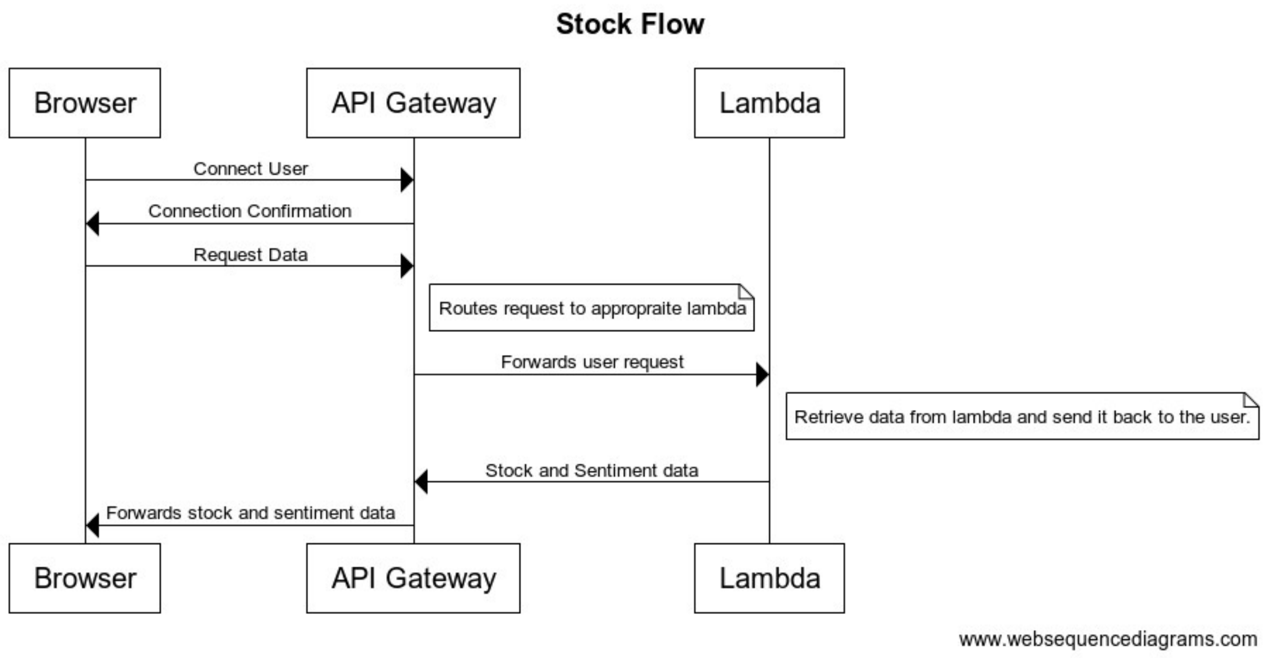
The focus of the project is to provide users with a visual representation of stock data, along with sentiment analysis based on news headlines. To achieve this, I utilized the free stock data provided by Alpha Advantage, which includes information such as stock prices, trading volumes, and other relevant financial metrics. This data is then presented to users in a user-friendly format that allows them to easily compare and analyse different stocks.

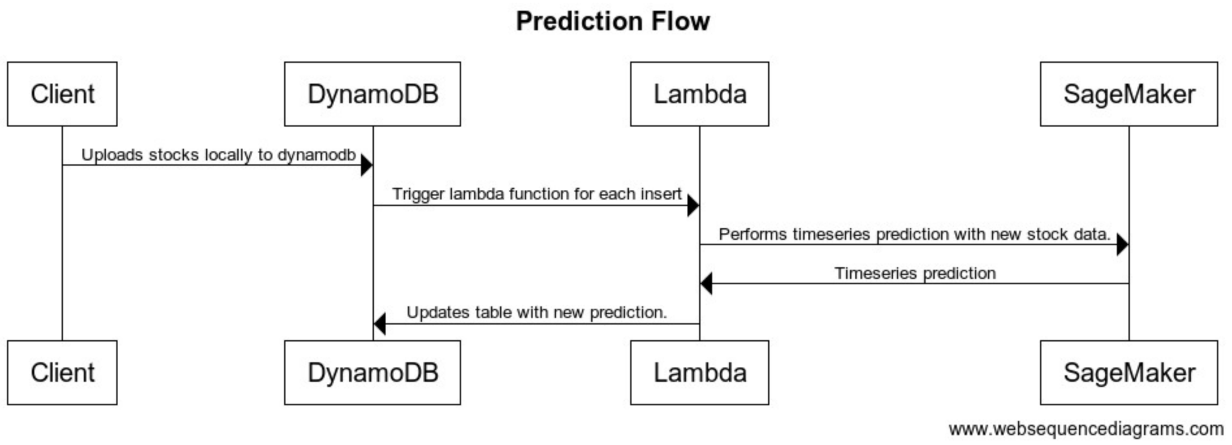
In addition to the stock data, the website also provides sentiment analysis for each stock. This is achieved by collecting news headlines from various sources using the News API, which is a popular news aggregator. The headlines are then processed using AWS Comprehend, which is a natural language processing tool that can determine the sentiment of a given text. The sentiment analysis is then displayed alongside the stock data, allowing users to see how news sentiment is impacting the stock price.

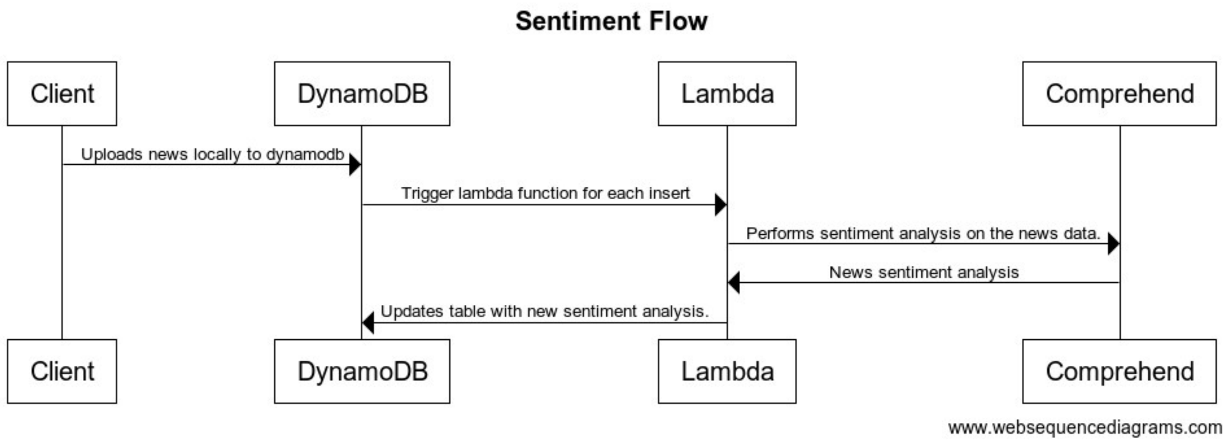
To ensure that the website provides real-time updates, I used web sockets to feed data from the DynamoDB database to the user interface. This means that users can see the latest stock data and sentiment analysis without having to refresh the page manually. The use of web sockets also allows for a more seamless user experience, as users can see changes to the stock data and sentiment analysis in real-time.

Overall, the project aims to provide users with a comprehensive view of stock data and sentiment analysis in an easy-to-use and accessible format. By combining stock data and sentiment analysis, users can make more informed decisions when it comes to investing in the stock market.

Architecture


Stock Flow
Browser
API Gateway
Lambda
Connect User
Connection Confirmation
Request Data
Routes request to appropraite lambda
Forwards user request
Retrieve data from lambda and send it back to the user.
Stock and Sentiment data
Forwards stock and sentiment data
Browser
API Gateway
Lambda
www.websequencediagrams.com


Prediction Flow
Client
DynamoDB
Lambda
SageMaker
Uploads stocks locally to dynamodb
Trigger lambda function for each insert
Performs timeseries prediction with new stock data.
Timeseries prediction
Updates table with new prediction.
www.websequencediagrams.com


Sentiment Flow
Client
DynamoDB
Lambda
Comprehend
Uploads news locally to dynamodb
Trigger lambda function for each insert
Performs sentiment analysis on the news data.
News sentiment analysis
Updates table with new sentiment analysis.
www.websequencediagrams.com

WebSockets

In the stock visualization and prediction website project, WebSockets were used to create a real-time communication channel between the server and the client's browser.

From the server's perspective, using WebSockets allowed for a more efficient and scalable way to handle real-time data updates. When a user connects to the website, a WebSocket connection is established between their browser and the server. This connection remains open as long as the user is on the website, allowing the server to push data updates to the client in real-time. This eliminates the need for the client to constantly make HTTP requests to the server for updated data, which can be inefficient and slow down the website.

From the browser's perspective, WebSockets are a powerful technology that enables real-time communication with the server. When a user connects to the website, their browser sends a WebSocket handshake request to the server. If the server accepts the request, a WebSocket connection is established between the client and server, and the client's browser can start sending and receiving real-time data updates.

AWS Lambda

In the stock visualization and prediction website project, we utilized AWS Lambda to perform several functions. Lambda is a serverless compute service that allows developers to run code without the need to manage servers.

One of the primary functions of Lambda in the project was to respond to save triggers on DynamoDB. When new data was inserted into the DynamoDB database, Lambda was triggered to perform analysis and predictions on the newly inserted data. This allowed us to ensure that the predictions were up to date with the latest data and provided real-time analysis for the users.

Additionally, Lambda was also used as a REST endpoint to display sentiment. Sentiment analysis was performed on news headlines using AWS Comprehend, and the results were stored in DynamoDB. When a user made a request for sentiment data, Lambda was triggered to retrieve the data from DynamoDB and display it as a response.

Lambda was also utilized as a WebSocket endpoint to handle incoming WebSocket requests from API Gateway. We had three different Lambda functions to handle the different stages of the WebSocket connection: one function for handling disconnections, another for handling connections, and the last for processing user requests. When a user connected to the WebSocket, the Lambda function was triggered to store the user's information in DynamoDB. When a user disconnected, the Lambda function removed the user's information from DynamoDB.

AWS CloudWatch

In the stock visualization and prediction website project, we utilized Amazon CloudWatch to aid in debugging and provide a centralized logging solution for various AWS services. CloudWatch is a monitoring and observability service that provides metrics, logs, and alarms for AWS resources and applications.

We used CloudWatch to debug issues that arose during the development process by analysing the logs generated by the Lambda functions, S3, SageMaker, and API Gateway. This helped us identify and troubleshoot any issues that occurred, allowing us to quickly address and resolve them.

CloudWatch also provided a centralized logging solution for the project, allowing us to store all the logs generated by the various AWS services in a single location. This made it easier to search, analyse, and troubleshoot issues across the entire application stack.

AWS S3

As part of the stock visualization and prediction website project, we utilized Amazon S3 (Simple Storage Service) to store and host static data and files. S3 is a highly scalable and durable object storage service that provides easy and secure storage of data in the cloud.

We used S3 to store historical stock data required for training the SageMaker model. This data was used to train the machine learning model to predict future stock prices accurately. S3 provided a cost-effective and efficient solution for storing large amounts of historical data required for training the machine learning model.

In addition, we also used S3 to host static files required for the frontend application. These included HTML, CSS, and JavaScript files, which were necessary to build and display the website's user interface.

AWS API Gateway

In the stock visualization and prediction website project, we utilized Amazon Web Services (AWS) API Gateway to manage the routing of traffic to the Lambda function. The API Gateway is a fully managed service that makes it easy to create, deploy, and manage RESTful and WebSocket APIs.

We used API Gateway to forward both HTTPS traffic and WebSocket traffic to the Lambda function. The WebSocket traffic was further handled by three separate Lambda functions. The first function was responsible for handling disconnections, the second function was responsible for handling connections, and the third function was responsible for processing user requests.

Upon establishing a connection, the user's details were stored in DynamoDB, which was used for future broadcasts. When a user disconnected, their details were removed from DynamoDB to keep the database updated with the current status of connected users.

By using AWS API Gateway to manage the routing of traffic to the Lambda function, we were able to create a scalable, efficient, and secure system for handling both HTTPS and WebSocket traffic. The Lambda functions provided the necessary functionality to handle WebSocket connections and disconnections while ensuring that the user's details were accurately stored and updated in the DynamoDB database. This allowed for a smooth and reliable user experience, with minimal downtime and interruption.

AWS SageMaker

In the stock visualization and prediction website project, we utilized Amazon Web Services (AWS) SageMaker to perform time series prediction on the stock data. SageMaker is a cloud-based machine learning platform that offers a wide range of tools for data scientists and developers to build, train, and deploy machine learning models. In our case, we used it to develop a model that can predict future stock prices based on historical data.

When using AWS SageMaker, creating a model involves several steps. First, we split the stock data into a training set and a test set. We then use the training set to train the model, which involves feeding the historical data into the model and adjusting the model's parameters to minimize errors.

Once the training is complete, we use the trained model to create an endpoint. This endpoint is essentially a web service that allows us to make predictions based on the trained model. The endpoint takes in new data, processes it using the trained model, and returns predictions for future stock prices.

In our case, we utilized this endpoint to generate predictions for future stock prices based on the historical data stored in the database. These predictions were then stored in a separate database that only contains prediction data, allowing us to easily track and analyse the accuracy of the predictions over time.

Overall, utilizing AWS SageMaker and time series prediction involved several steps, including creating a model, training the model, creating an endpoint, and generating predictions based on new data. By adding this layer of analysis to the website, users can make more informed decisions about the future performance of a given stock, ultimately leading to more successful investments.

AWS DynamoDB

In this project, DynamoDB was used to store historical stock data, sentiment analysis results derived from news headlines, and user information. The stock data was obtained for free from Alpha Advantage and stored in DynamoDB for use in training the SageMaker model. The sentiment analysis results were derived from the news headlines obtained from News API, and the processed results were also stored in

DynamoDB. User information, including the user's WebSocket connection ID, was also stored in DynamoDB for future broadcast messages.

Below is a table containing the DynamoDB table structure and their names.

Table Name	**Sort Key**	**Partition Key**
clients	-	id (string)
news	ticker (string)	timestamp (number)
predictions	ticker (string)	timestamp (number)
sentiments	ticker (string)	timestamp (number)
stocks	ticker (string)	timestamp (number)

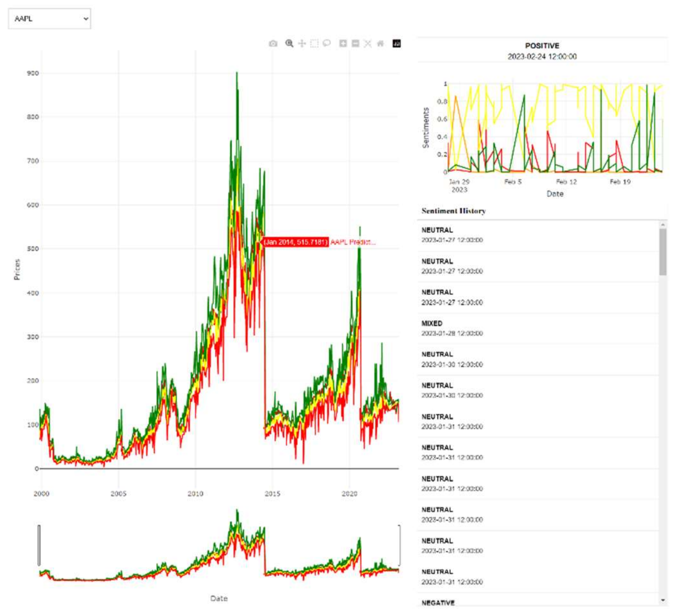
Links

Web Application: https://cst3130-cw2-mdx.s3.amazonaws.com/index.html

Synthetic Prediction Graph: https://chart-studio.plotly.com/~si468/2/synthetic-m00846543/#/

Synthetic Prediction API: https://lvi1t6ndd7.execute-api.us-east-1.amazonaws.com/default/synthetic-prediction

Screenshots



POSITIVE
2023-02-24 12:00:00

Sentiment History

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-27 12:00:00

MIXED
2023-01-28 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

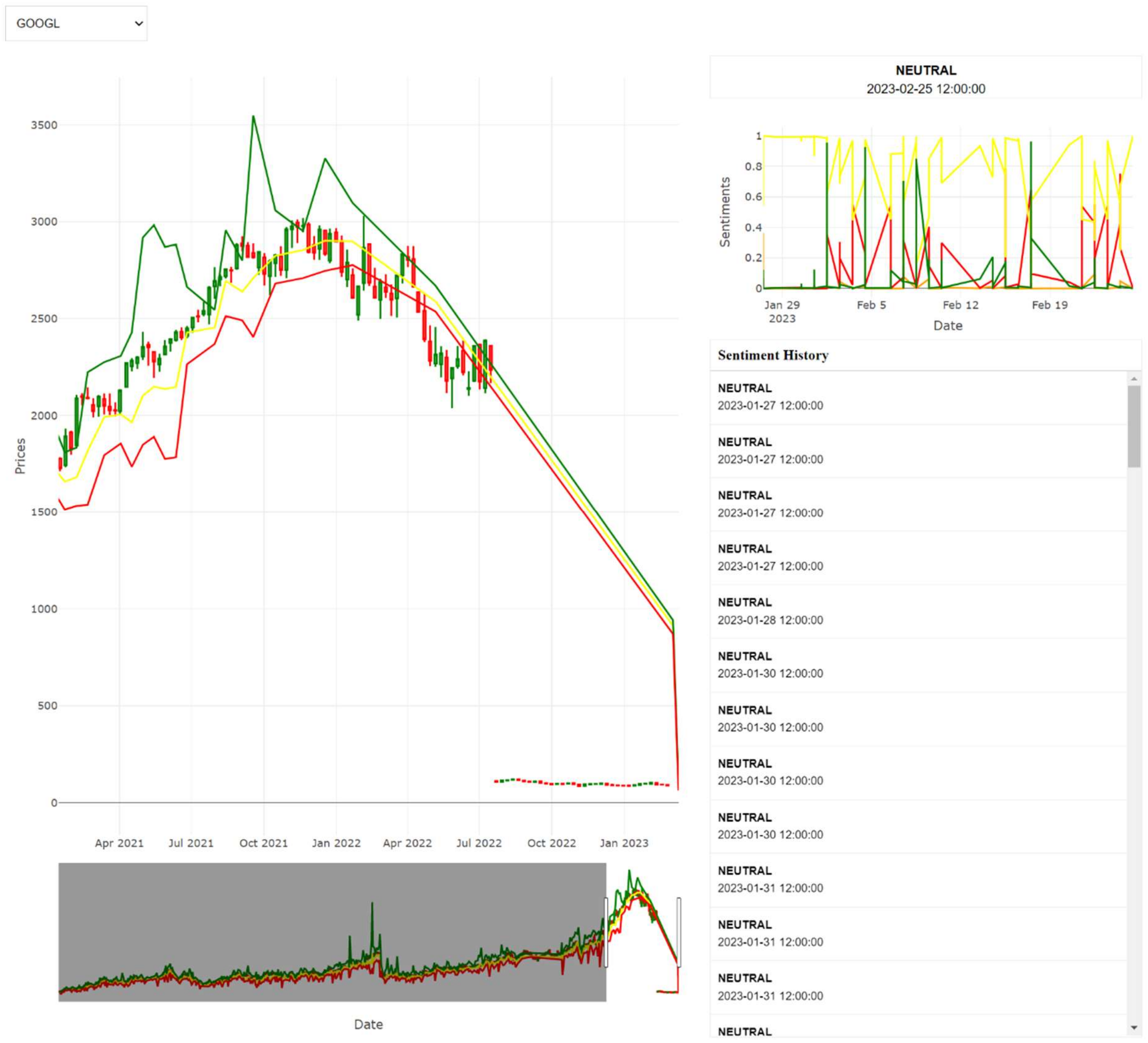
NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEGATIVE



GOOGL

NEUTRAL
2023-02-25 12:00:00

Sentiment History

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-27 12:00:00

NEUTRAL
2023-01-28 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-30 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL
2023-01-31 12:00:00

NEUTRAL