Middlesex University London

School of Science and Technology

Project Report for Coursework 2 - Data Visualization (CST3130 - Advanced Web Development with Big Data)

Module Leader: Dr. David Gomez

Student Name: Madudili Freeman Chidera

Student ID: M00904768

Application Name: Forexify

Link to video demonstration: https://www.mediafire.com/folder/8si5xe1121n3c/coursework-2

1. Introduction

Forexify is a data visualization website that provides users with real-time and historical forex and news data for various currencies and currency pairs. The website is hosted on AWS S3 and utilizes several AWS services such as DynamoDB, Lambda functions, and Sagemaker Endpoint to source, process and store the data. Forexify's front-end interface is designed to help users make sense of the data by visualizing it using chartjs.

In summary, Forexify is a comprehensive data visualization website that provides users with up-to-date insights on forex and news data for various currencies and currency pairs. The website's architecture ensures that users have access to real-time and historical data, while the use of AWS services such as DynamoDB and Lambda functions ensures that the website is efficient and scalable. The frontend of the website, hosted on AWS S3, provides users with an intuitive and user-friendly interface that enables them to visualize the data easily.

2. How It Works

Forexify sources its data from the AlphaVantage API, which provides real-time and historical forex and news data for currencies such as USD, CAD, and currency pairs such as USDJPY, USDMXN, among others. The data is then saved in AWS DynamoDB, a NoSQL database that is fast, flexible, and scalable.

The website utilizes several Lambda functions that are triggered by changes in the DynamoDB table. One of the Lambda functions generates news sentiment from news data by analyzing it using AWS Comprehend, a Natural Language Processing (NLP) technique. The sentiment results are saved in a separate table, which is then utilized by the front-end interface to display the sentiment trend over time.

Another Lambda function is triggered when there is new forex data, which generates forex data predictions by querying an AWS Sagemaker Endpoint that has pre-trained models for various currency pairs. The results of the predictions are saved in a new table and used to display the predicted trends over time in the front-end interface.

To enable bi-directional communication between the client and server, the website utilizes a Websocket API Gateway. The Gateway sends new data to the client when there's an update and vice versa.

The front-end interface of the website is hosted on an AWS S3 bucket for public access. The interface utilizes chartjs to visualize the various data types such as News Sentiments, Forex Data, Forex Data Predictions, Synthetic Data, and Synthetic Data Predictions. The data is plotted against various timestamps, providing users with an insightful analysis of forex and news data.

3. Technologies Used

The technologies utilized in the application process are enumerated below, grouped by the aforementioned components:

AWS Services:

- AWS S3: used to host the frontend website for public access
- AWS DynamoDB: used as the primary database to store and retrieve real-time and historical forex and news data
- AWS Lambda: used to run serverless functions that are triggered by changes in the DynamoDB table
- AWS Sagemaker: used to train machine learning models for predicting forex trends
- AWS API Gateway: used to create a Websocket API Gateway that enables bi-directional communication between the client and server
- AWS Comprehend: used to perform natural language processing (NLP) on news data to generate sentiment scores

Programming Languages:

- JavaScript: used for the majority of the application's codebase, including the front-end interface and the Lambda functions
- TypeScript: used to retrieve data from AlphaVantage and upload it to DynamoDB

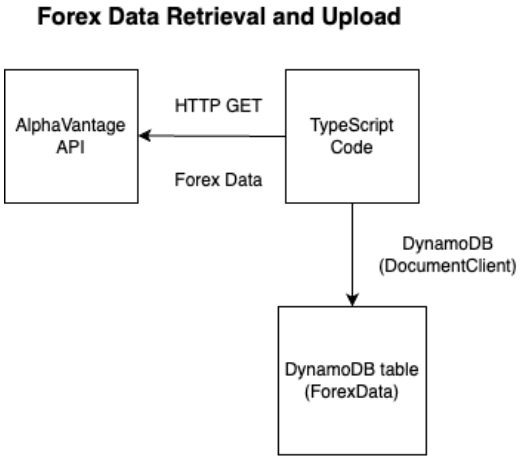
Libraries and Frameworks:

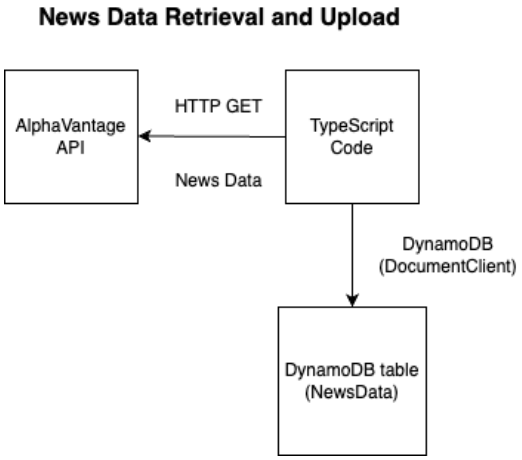
- Chartjs: used to visualize forex and news data on the front-end interface
- TailwindCSS: used for designing the website's user interface
- Vue.js: used as a progressive JavaScript framework for building the user interface components
- Moment: used as a lightweight JavaScript library for parsing, validating, manipulating, and formatting dates and times in the user interface
- Node.js: used to execute JavaScript code outside of the browser
- AWS SDK for JavaScript: used to interact with various AWS services such as S3, DynamoDB, and Lambda functions

4. Application Breakdown

4.1 Data Retrieval and Upload from AlphaVantage API to DynamoDB

To retrieve Forex and News data from the AlphaVantage API and upload it to DynamoDB, the application makes use of TypeScript code running locally on your system. This code retrieves Forex and News data from the AlphaVantage API by making HTTP GET requests, and then uploads this data to their respective tables (ForexData and NewsData) in DynamoDB using the AWS DynamoDB Document Client. The process is illustrated in the diagram below.





4.2 DynamoDB Schema

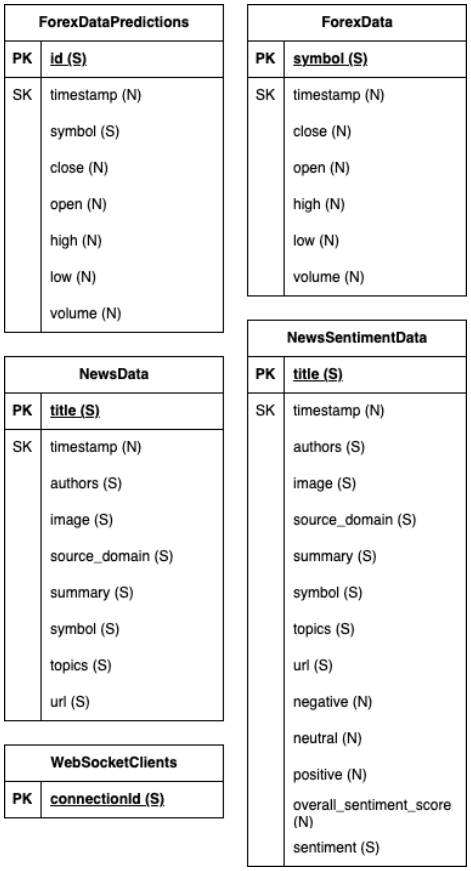
DynamoDB is a fully managed NoSQL database service provided by AWS. In this project, DynamoDB is used to store various types of data, including Forex data,

Forex data predictions, News data, News sentiment data, and WebSocket client data.

The ForexData and NewsData tables store the raw data retrieved from the AlphaVantage API, while the ForexDataPredictions table stores the predicted values generated by the Sagemaker model. The NewsSentimentData table stores the sentiment analysis results generated by the Lambda function using AWS Comprehend.

The WebSocketClients table is used to keep track of the clients connected to the WebSocket API Gateway, allowing bi-directional communication between the client and server. By storing this information in DynamoDB, the server can maintain a list of connected clients and send updates to them when new data is available.

Illustrated below is an entity diagram of my dynamoDB database:



4.3 Lambda Functions

For the server-side logic, AWS Lambda functions are used to process and analyze data. There are also WebSocket functions to handle real-time communication with the clients.

Here are the Lambda functions and their implementations:

- getForexDataPredictions: This function retrieves the latest Forex data from the ForexData DynamoDB table and uses the AWS SageMaker endpoint to make prediction trends. The predictions are stored in the ForexDataPredictions DynamoDB table.
- analyzeNewsSentiment: Analyzes news data using AWS Comprehend and stores the sentiment results in the NewsSentimentData DynamoDB table.
- broadcastForexData: Retrieves the latest forex data from the ForexData DynamoDB table and broadcasts it to all connected WebSocket clients.
- broadcastNewsSentiment: Retrieves the latest news sentiment data from the NewsSentimentData DynamoDB table and broadcasts it to all connected WebSocket clients.
-

And here are the Lambda-WebSocket functions and their implementations:

- wsConnect: Handles new WebSocket connections by adding the client to the WebsocketClients DynamoDB table.
- wsDisconnect: Handles WebSocket disconnections by removing the client from the WebsocketClients DynamoDB table.
- wsDefault: Handles WebSocket messages that are not explicitly handled by other WebSocket functions.
- wsGetForexData: Retrieves the latest forex data from the ForexData DynamoDB table and sends it to the newly connected client.
- wsGetForexDataPredictions: Retrieves the latest forex data predictions from the ForexDataPredictions DynamoDB table and sends it to the newly connected client.
- wsGetNewsSentiment: Retrieves the latest news sentiment data from the NewsSentimentData DynamoDB table and sends it to the newly connected client.

4.4 WebSocket API Gateway

WebSocket API Gateway allows two-way communication between the client (front-end) and server (Lambda functions) in real-time. In this application, we use WebSocket API Gateway to establish and maintain a persistent connection between the client and server.

The WebSocket API Gateway serves as an entry point for the WebSocket protocol in the AWS infrastructure. It manages connections with clients and routes incoming messages to the corresponding Lambda functions.

To explain the routes, the WebSocket API Gateway routes incoming messages based on the route key in the incoming message. In this application, we have six routes: $connect, $disconnect, $default, getForexData, getForexDataPredictions, and getNewsSentiment.

- $connect: This is called when a client connects to the WebSocket API Gateway. It triggers the Lambda function wsConnect to handle the connection and store the client's connection details.
- $disconnect: This is called when a client disconnects from the WebSocket API Gateway. It triggers the Lambda function wsDisconnect to handle the disconnection and remove the client's connection details.
- $default: This is called when the WebSocket API Gateway receives a message with an unrecognized route key. It triggers the Lambda function wsDefault to handle the message.
- getForexData: This is called when a client requests for the latest Forex data. It triggers the Lambda function wsGetForexData to retrieve the latest Forex data from the DynamoDB table and send it to the client.
- getForexDataPredictions: This is called when a client requests for the latest Forex data predictions. It triggers the Lambda function wsGetForexDataPredictions to retrieve the latest Forex data predictions from the DynamoDB table and send it to the client.
- getNewsSentiment: This is called when a client requests for the latest news sentiment data. It triggers the Lambda function wsGetNewsSentiment to retrieve the latest news sentiment data from the DynamoDB table and send it to the client.

4.5 AWS Sagemaker

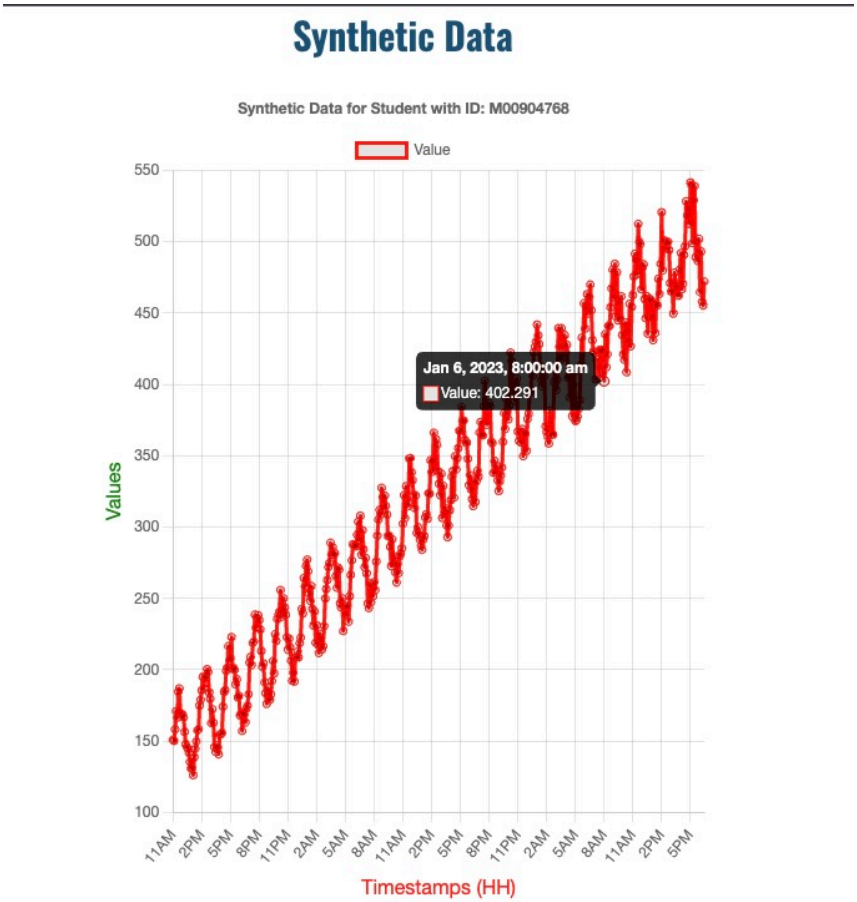
AWS Sagemaker is a fully-managed service that enables developers and data scientists to build, train, and deploy machine learning models at scale. In this project, AWS Sagemaker is used for training machine learning models for making predictions on Forex data.

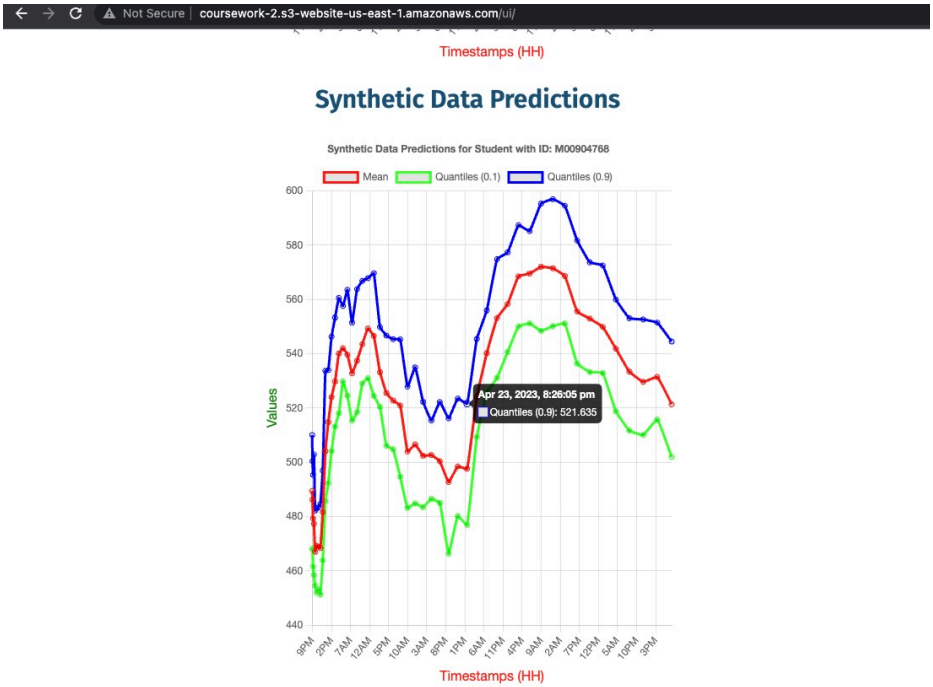
The training jobs are created using the Amazon SageMaker Dashboard. The training job involves training a deep learning model using the existing ForexData in

DynamoDB. The model is then trained on this synthetic data to generate predictions. This is same process utilized for the synthetic data.

Once the training is complete, the model is deployed as an endpoint on SageMaker. The endpoint can be accessed using the API Gateway URL from the lambda function **getForexDataPredictions**. The predictions made by the model are then sent back to the frontend for display.

Illustrated below are the data visualizations for the synthetic data and predictions from the Sagemaker model:





4.6 Frontend

The Frontend component of the system is responsible for providing an intuitive user interface for interacting with the system's data and predictions. The Frontend is hosted on Amazon S3 and is accessible to the public via the following URL: http://coursework-2.s3-website-us-east-1.amazonaws.com/ui/.

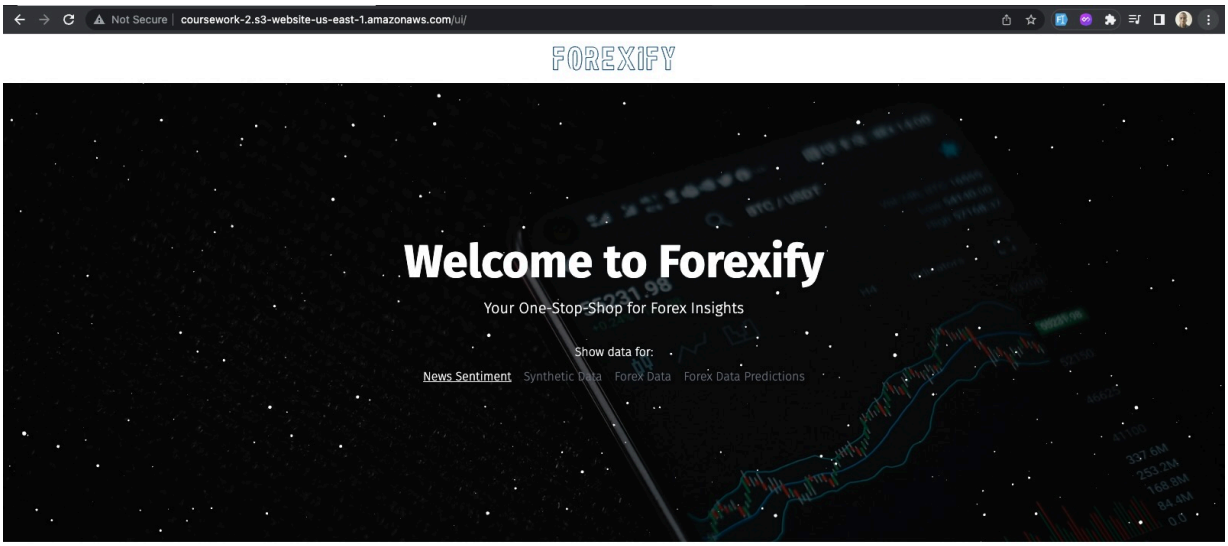
The Frontend is designed using Vue.js and TailwindCSS, which allowed for efficient and responsive development of the user interface. The frontend allows users to view the latest Forex data and predictions, as well as news sentiment analysis for different currencies. Users are subscribed automatically to receive real-time updates via WebSocket communication.

To enhance the user experience, the Frontend features interactive visualizations of the Forex data predictions and the sentiment analysis predictions. The visualizations were created using Chart.js and provide a clear and easy-to-understand representation of the system's predictions.

The Frontend component communicates with the backend components of the system via WebSocket communication. This allows for efficient and real-time updates of the data and predictions.

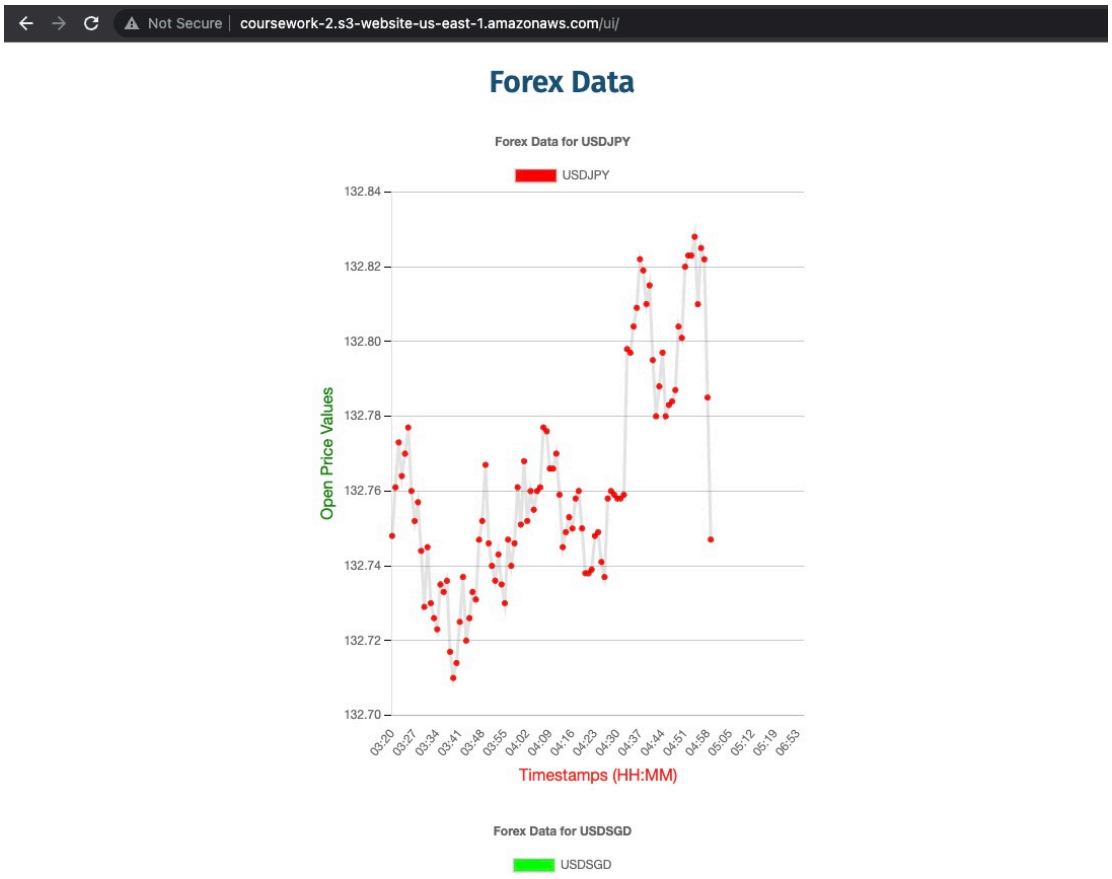
Below are screenshots of the landing page and other necessary site previews:

Landing Page with texts/buttons to toggle between Data Visualizations for News Sentiment, Synthetic Data, Forex Data and Forex Data Predictions

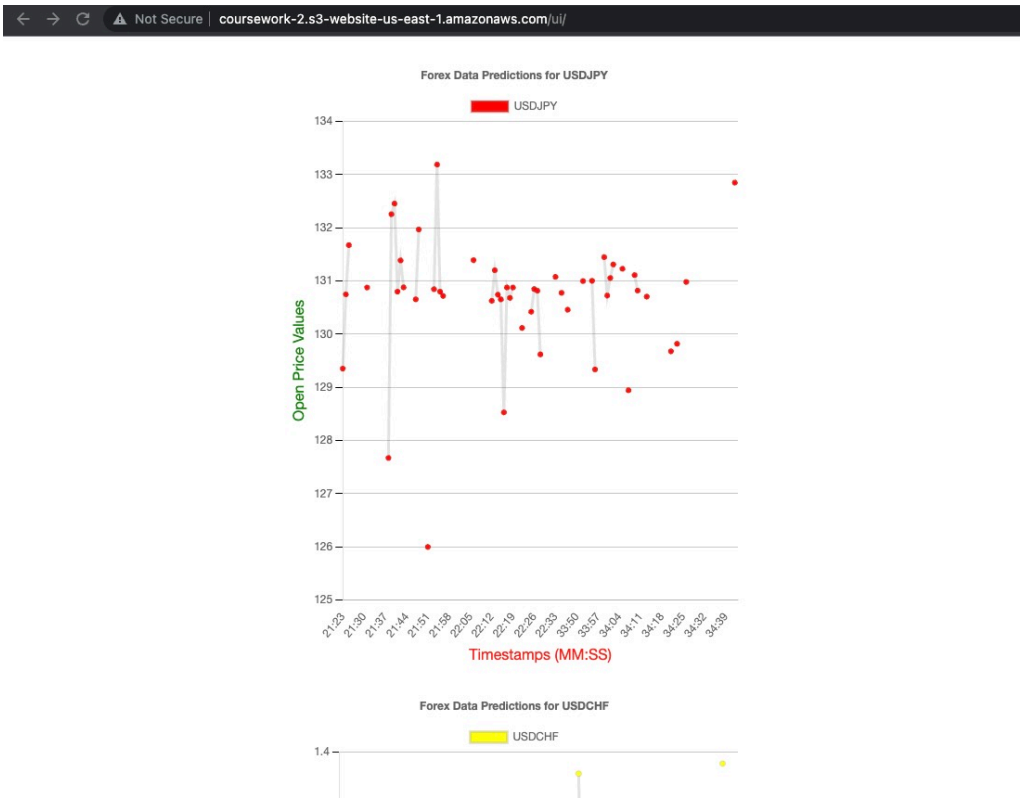


Data Visualization for News Sentiment Analysis Results

Data Visualization for Forex Data - USDJPY



Data Visualization for Forex Data Prediction - USDJPY



Overall, the Frontend component of the system provides an efficient and user-friendly interface for accessing the system's predictions and data. The use of Vue.js and TailwindCSS allowed for efficient development and the inclusion of interactive visualizations further enhances the user experience.