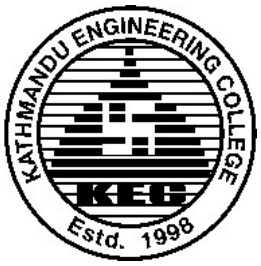
TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

KALIMATI, KATHMANDU



Minor Project Report on

Stock Market Prediction

[Code No: CT 654]

BY

Abhishek Man Shrestha - 078BCT009

Ayush Acharya - 078BCT022

Bishranta Regmi - 078BCT027

Dhiraj Sah - 078BCT030

TO

DEPARTMENT OF COMPUTER ENGINEERING

KATHMANDU, NEPAL

Magh, 2081

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

KALIMATI, KATHMANDU

Stock Market Prediction

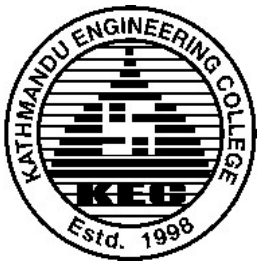
[Code No: CT 654]

PROJECT REPORT SUBMITTED TO

THE DEPARTMENT OF COMPUTER ENGINEERING

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE BACHELOR OF ENGINEERING



Abhishek Man Shrestha - 078BCT009

Ayush Acharya - 078BCT022

Bishranta Regmi - 078BCT027

Dhiraj Sah - 078BCT030

KATHMANDU, NEPAL

Falgun, 2081

TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

Department of Computer Engineering

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering a minor project work entitled “Stock Market Prediction" submitted by Abhishek Man Shrestha (KAT078BCT009), Ayush Acharya (KAT078BCT022), Bishranta Regmi (KAT078BCT027), Dhiraj Sah (KAT078BCT030) in partial fulfillment of the requirements for the degree of Bachelor of Engineering.

_____

Chaitya Shova Shakya

(External Examiner)

Kathford International College of Engineering and Management

_____

Er. Krista Byanju

(Project Coordinator)

Department of Computer Engineering

_____

Assoc.Prof. Sudeep Shakya

(Head of Department)

Department of Computer Engineering

ACKNOWLEDGEMENT

First of all, we would like to acknowledge our deep sense of gratitude to Institute of Engineering for including the minor project in our course. Similarly, we are very thankful to the Department of Computer Engineering, KEC for guiding us throughout this project.

We are also grateful to **Er. Krista Byanju**, our project coordinator for her recommendations regarding the development of our project. Also, we extend a special thanks to **Er. Sudeep Shakya**, Head of Computer Department, **Er. Kunjan Amatya**, Deputy Head of Computer Department as well as **Er. Nabin Neupane**, our year coordinator for all the help support and supervision provided to us so far. We can't thank the teachers of KEC enough for providing us with the invaluable supervision and necessary information for the successful completion of the project.

We would also like to thank all our friends for the assistance they have provided during this report. We hope all their help and support can result in a project that all of us can be proud of which can help us inherit different set of skills in the field of programming. Finally, we would like to give our sincere gratitude towards anyone who helped us directly or indirectly to finalize this report.

ABSTRACT

This project focuses on developing a stock market prediction system primarily based on historical data analysis. We have developed a model for the prediction of stock prices using historical data of NEPSE registered companies, collected from reliable sources (NepseAlpha and ShareSansar). For this, we utilized the Long Short-Term Memory (LSTM) model, a type of recurrent neural network specifically effective in analyzing sequential data. The collected historical data was preprocessed, normalized and divided into an 80% training set and 20% test set. The LSTM model is trained on this dataset, and the output is presented in both tabular and graphical formats. We have created a web application using Flask to visualize these predictions interactively. The webapp presents the predicted prices of the user chosen company along with the statistical parameters of the data and accuracy metrics of the model. User database is setup so that user can add maintain their profile and portfolio of investment. The user portfolio along with the prediction model was used to predict the overall portfolio of the user's investment after certain days. Finally, a sentiment report was generated based on the current news and affairs where user can have an insight on how the market can behave based on market sentiment. The project culminates in a user-friendly web application where users can view predictions and insights seamlessly. By continuously evaluating and refining the models, we aim to create a robust and efficient system for stock market forecasting.

Keywords: *stock, historical dataset, prediction, portfolio, analysis, web-application*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Network
BiGRU	Bidirectional Gated Recurrent Unit
CNN	Convolution Neural Network
CSS	Cascading Style Sheets
GRU	Gated Recurrent Unit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
SVM	Support Vector Machine

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND THEORY

The stock market is a system where people can buy and sell shares of publicly traded companies. Companies issue shares of stock to the public to raise funds for their operations and growth. The stock market operates on the principle of supply and demand. When demand for a company's stock is high, its share price rises, and vice versa. Investors purchase these shares because they believe the company will succeed and grow over time, increasing the value of their investment. If the company performs well, the stock price usually goes up, allowing investors to sell their shares at a higher price and make a profit.

The Nepal Stock Exchange (NEPSE) is the main stock exchange of Nepal. NEPSE opened its trading floor on 13 January 1994. It was established to provide a marketplace for the trading of securities in Nepal. The exchange facilitates transactions for both individual and institutional investors and plays a vital role in the economic development of Nepal by enabling companies to access public funds for expansion and growth.

Stock market prediction involves forecasting the future value of a company's stock, which is crucial for economic growth. By studying patterns in market data, we can predict when to buy stocks and potentially earn profits. This prediction relies on understanding supply and demand. By analyzing large amounts of market data, we can find patterns that give us a good chance of making accurate predictions. Stock market prediction consists of statistical analysis, machine learning, and sentiment analysis to forecast future stock prices using historical data and market sentiment. Integrating these methods and using machine learning helps to develop predictive models offering valuable insights of the market.

1.2 PROBLEM STATEMENT

The stock market is vast and unpredictable, with significant fluctuations making it challenging to anticipate whether prices will rise or fall the next day. Investing in stocks can result substantial profits or result in significant losses, making it a scary task for many. Manual methods of predicting future market trends are tedious and often inaccurate, especially when the financial markets are influenced by factors like politics, economics, and investor psychology.

1.3 OBJECTIVES

This project is created with the purpose to fulfill the following objectives:

- To analyze trends and patterns of historical trading data of companies (date, closing price, LTP) and sentimental data (news headlines, articles, posts, etc.) to capture market trends.
- To predict future price using the information from analyzed data implementing machine learning algorithms LSTM and CNN.

1.4 SCOPE AND APPLICATIONS

The applications of this project are as follows:

- To bridge communication between financial investors and industries as investors can find right companies for investments and companies can find investors for their growth
- To assist the investors and people to make sensible decisions for profitable investments through analysis of historical trading data.

CHAPTER 2: LITERATURE REVIEW

A study done by Polepally, Vijay & Jakka, Neha & Vishnukanth, Pendly & Raj, Rachakatla & Anish, Gudavelli presents a comprehensive study of machine learning algorithms for stock price prediction. In their 2023 study, they explored the effectiveness of the LSTM-RNN algorithm for predicting stock prices. Presented at the 7th International Conference on Intelligent Computing and Control Systems (ICICCS), their research highlights how LSTM-RNN can capture long-term dependencies in sequential data, making it highly suitable for stock market prediction. This algorithm is adept at handling time series data, which is critical for accurate forecasting in financial markets. However, their approach primarily focuses on historical data without incorporating sentiment analysis, potentially limiting its predictive capabilities in response to market sentiment shifts [1].

Lal and Timalsina's 2022 paper, presented at the 11th IOE Graduate Conference, investigated the use of a CNN-BIGRU method for stock price prediction. The combination of Convolutional Neural Networks (CNN) for feature extraction and Bidirectional Gated Recurrent Units (BIGRU) for handling sequential data shows promise in forecasting stock prices. Their method leverages CNN's strength in identifying important patterns in data and BIGRU's ability to process time-dependent sequences effectively. Despite its potential, this method can be complex and resource-intensive, and it does not explicitly include sentiment analysis, which could enhance prediction accuracy [2].

In the 2023 publication in the International Journal of Creative Research Thoughts (IJCRT), Kabir, Sobur, and Amin compared various machine learning models for stock price prediction. Their study evaluates the performance of different algorithms under diverse market conditions, providing insights into their strengths and weaknesses. While their comprehensive approach offers valuable comparisons, it does not integrate additional data sources such as sentiment analysis from news or social media, which can be crucial for improving the accuracy of stock predictions [3].

Choi’s 2018 research discusses a hybrid model combining ARIMA and LSTM for predicting stock price correlations. The hybrid ARIMA-LSTM model leverages the statistical strengths of ARIMA and the deep learning capabilities of LSTM, resulting in improved accuracy for correlation predictions between stocks. This approach demonstrates the benefits of combining traditional statistical methods with modern machine learning techniques. However, the model's complexity may pose challenges for practical implementation, and it focuses on correlation prediction rather than direct price forecasting [4].

Patel, J., Shah, S., Thakkar, P., & Kotecha, K. in their 2015 paper published in Expert Systems with Applications, Patel and colleagues examined various machine learning techniques, including ANN, SVM, random forest, and naive Bayes, for predicting stock prices and index movements. Their study emphasizes the importance of trend deterministic data preparation to enhance the accuracy of machine learning models. By comparing these techniques, the authors provide insights into their effectiveness for financial forecasting. Despite the comprehensive analysis, the study does not incorporate sentiment analysis, which could provide a more holistic view of market dynamics [5].

2.1 EXISTING SYSTEMS

2.1.1 QuantConnect

An algorithmic trading platform that allows users to create and back test trading strategies which requires a strong understanding of programming and algorithmic trading concepts, making it less accessible to beginners.

2.1.2 MetaTrader5

A popular trading platform offering tools for technical analysis and automated trading strategies that primarily focused on technical analysis and doesn't inherently support machine learning models or sentiment analysis.

2.1.3 AlphaSense

An AI-powered market intelligence platform that aggregates and analyzes financial data, news, and research reports but doesn't provide specific stock price prediction capabilities.

2.2 LIMITATIONS OF PREVIOUS SYSTEMS

- Many platforms like QuantConnect require advanced programming knowledge and have steep learning curves, making them less accessible to beginner users.
- Advanced platforms like AlphaSense have high subscription fees, which can be prohibitive for individual investors or small teams.
- Most tools focus primarily on technical analysis and do not natively support the integration of sentiment analysis from news and social media, limiting the comprehensiveness of predictions.

2.3 SOLUTIONS PROVIDED BY OUR SYSTEM

- Use of effective model: This project uses the LSTM algorithm, which excels in stock price forecasting due to its ability to handle time series data and avoid memory issues which is a major problem with RNN.

- Implementation of Sentiment Analysis: This system iimplements sentiment analysis from news and social media to provide the insights on how the market can behave according to market sentiment.
- Accessibility: Using widely known programming languages, frameworks, open-source libraries and tools keeps costs low, makes the solution more accessible to broader audience, individual users and small teams.

CHAPTER 3: RELATED THEORY

3.1 Machine Learning in Stock Prediction

3.1.1 Introduction to Machine Learning (ML)

Machine Learning (ML) enables computers to learn from data and make predictions. It is divided into:

- **Supervised Learning**: Models learn from labeled data (e.g., predicting stock prices based on historical data).
- **Unsupervised Learning**: Models identify patterns in unlabeled data (e.g., clustering similar stocks).
- **Reinforcement Learning**: Models make decisions based on rewards and penalties (e.g., algorithmic trading).

ML helps analyze large volumes of stock data efficiently, capturing complex dependencies that traditional models miss.

3.1.2 Time Series Forecasting

Time series forecasting involves predicting future values based on historical trends.

Common Models:

- Autoregressive (AR)**:** Predicts future values using a linear combination of past values.
- Moving Average (MA)**:** Models past errors to improve prediction accuracy.
- ARIMA (AutoRegressive Integrated Moving Average)**:** Combines AR and MA with differencing to handle non-stationary data.

Challenges in Time Series Prediction:

- Market volatility makes predictions uncertain.
- External factors (e.g., global events) impact stock movements unpredictably.
- Overfitting occurs if models learn noise instead of actual trends.

3.2 Machine Learning Models

3.2.1 Recurrent Neural Network (RNNs)

Recurrent Neural Networks (RNNs) are a type of neural network that are designed to process sequential data. They can analyze data with a temporal dimension, such as time series, speech, and text. RNNs can do this by using a hidden state passed from one timestep to the next. The hidden state is updated at each timestep based on the input and the previous hidden state. This structure allows RNNs to retain information about past inputs, effectively enabling them to exhibit a kind of memory over sequences.

However, standard RNNs face challenges when dealing with long sequences due to the problem of vanishing or exploding gradients during backpropagation. This makes it difficult for them to learn long-term dependencies. To address these issues, specialized architectures like Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) were developed. These architectures include mechanisms such as gates to better regulate the flow of information, allowing them to capture both short-term and long-term dependencies more effectively. RNNs are widely used in applications like language modeling, machine translation, video analysis, and stock price prediction. They are particularly well-suited for tasks where the order or context of data significantly impacts the output, as they inherently process data in sequence.

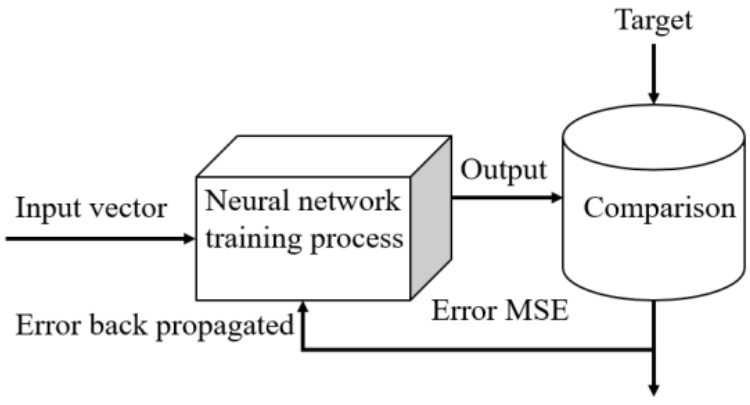


Figure 1: Recurrent Neural Network

3.2.2 Long Short-Term Memory (LSTM)

LSTMs Long Short-Term Memory is a type of RNNs that can detain long-term dependencies in sequential data. They use memory cell and gates to control the flow of information, allowing them to selectively retain or discard information as needed and thus avoid the vanishing gradient problem that plagues traditional RNNs. LSTMs are widely used in applications such as natural language processing, speech recognition, and time series forecasting.

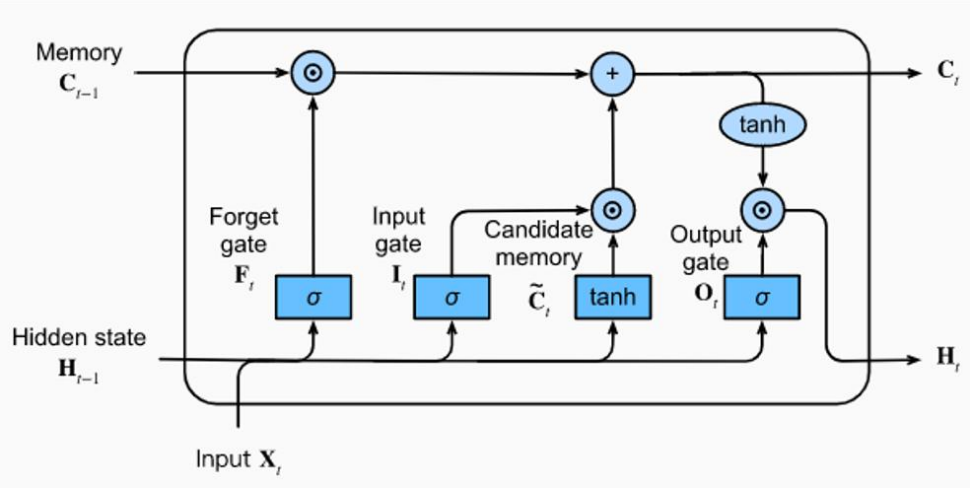


Figure 2: Long Short-Term Memory

There are three types of gates in an LSTM:

Input gate: It decides which information to store in the memory cell. It is trained to open when the input is important and close when it is not.

Forget gate: It decides which information to discard from the memory cell. It is trained to open when the information is no longer important and close when it is.

Output gate: It is responsible for deciding which information to use for the output of LSTM. It is trained to open when the information is important and close when it is not.

The gates in an LSTM are trained to open and close based on the input and the previous hidden state. This allows the LSTM to selectively retain or discard information, making it more effective at capturing long-term dependencies.

Stock prices exhibit temporal dependencies, where current values depend on past trends and patterns. LSTM is highly effective in capturing these dependencies because:

- It processes sequential data and remembers trends over long periods.
- It can model the fluctuations and volatility inherent in stock prices.

3.2.3 Other Deep Learning Models for Stock Prediction

- **GRU (Gated Recurrent Unit)**: A simpler version of LSTM with fewer parameters, making it computationally efficient.
- **Transformer Models**: Uses attention mechanisms for time series forecasting (e.g., Temporal Fusion Transformers for stock prediction).

3.3 Sentiment Analysis in Stock Market Prediction

3.3.1 Introduction to Sentiment Analysis

Sentiment analysis is the process of determining emotions and opinions in textual data, often used to measure investor sentiment.

Why Sentiment Matters in Stock Prediction?

- Positive news → Stock prices rise 📈
- Negative news → Stock prices fall 📉

3.3.2 Sources of Sentiment Data

- Financial news articles
- Social media posts

3.3.3 Natural Language Processing (NLP) in Sentiment Analysis

- **Text Preprocessing**: Tokenization, stopword removal, stemming, and lemmatization.
- **Feature Extraction**:
    - Bag of Words (BoW): Converts text into numerical vectors.

- TF-IDF (Term Frequency-Inverse Document Frequency): Weighs important words.
- Word Embeddings (Word2Vec, GloVe, BERT): Captures context and meaning.

- **Sentiment Classification Models**:
  - Traditional: Naïve Bayes, SVM
  - Deep Learning: LSTM, Transformer-based models (BERT, GPT)

3.3.4 Impact of Sentiment on Stock Prices

Studies have shown that public sentiment influences stock market trends. For example:

- Elon Musk's tweets affect Tesla stock prices.
- Negative news about a company causes mass sell-offs.

3.3.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are powerful deep learning models primarily used for image and text data analysis. They are particularly effective at processing data with a grid-like structure, such as images, by employing convolutional layers that apply filters to detect local patterns like edges, textures, and shapes. This capability allows CNNs to hierarchically extract features, from low-level details to high-level abstractions, making them ideal for tasks like object recognition, image classification, and natural language processing.

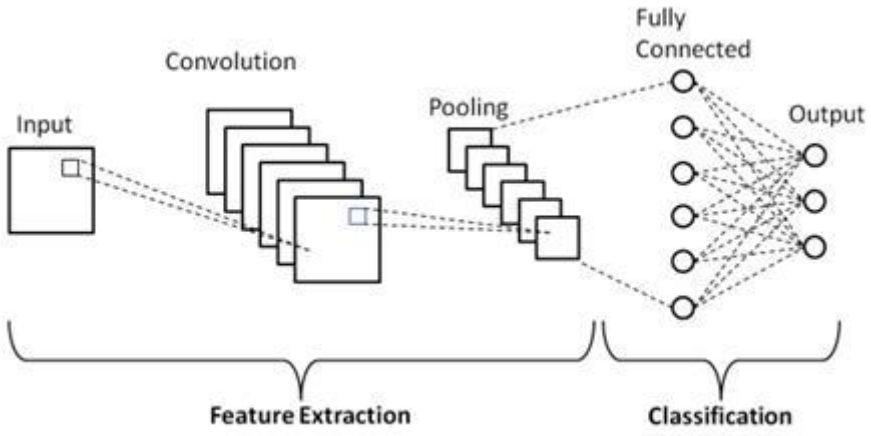


Figure 3: Convolution Neural Network

In the context of stock market prediction, CNNs excel at sentiment analysis by extracting meaningful patterns from textual data such as news articles, social media posts, and financial reports. Their ability to recognize contextual nuances in text enables them to classify sentiments as positive, negative or neutral associated with stock-related discussions. By identifying sentiments and trends, CNNs contribute significantly to improving predictive accuracy when integrated with time-series models like LSTM. This combination leverages CNNs for feature extraction and LSTMs for temporal modeling, providing a robust framework for understanding both static and dynamic aspects of market data.

Sentiment analysis involves processing textual data, such as news headlines and social media posts, to determine the sentiment (positive, negative, or neutral). CNN is well-suited for this task because:

- It can identify patterns and features in text, such as recurring sentiment indicators (e.g., "bullish," "growth").
- It is computationally efficient and can handle large volumes of text data.

Application in the Project:

In the context of this project, CNN is used to extract sentiment-related features from textual data:

- **Input Data:** News articles and social media posts are preprocessed into numerical representations (e.g., word embeddings).
- **Feature Extraction:** The CNN model analyzes the text to identify sentiment polarity and subjectivity, which are quantified into scores.
- **Output:** The sentiment scores used to determine the market behaviour.

3.4. Evaluation Metrics for Stock Prediction Models

3.4.1 Regression-Based Metrics

- **Mean Squared Error (MSE)**: Measures average squared prediction errors.
- **Root Mean Squared Error (RMSE)**: Square root of MSE, giving better interpretability.
- **Mean Absolute Error (MAE)**: Measures average absolute error, less sensitive to outliers.
- **R-squared (R^2) Score**: Measures goodness-of-fit of a model.

3.5 Challenges and Limitations in Stock Prediction

- **Data Quality Issues**: Missing or biased data impacts predictions.
- **Market Volatility**: Sudden crashes or booms disrupt models.
- **Computational Complexity**: Training deep learning models requires high processing power.

CHAPTER 4: METHODOLOGY

4.1 PROCESS MODEL

4.1.1 ITERATIVE MODEL

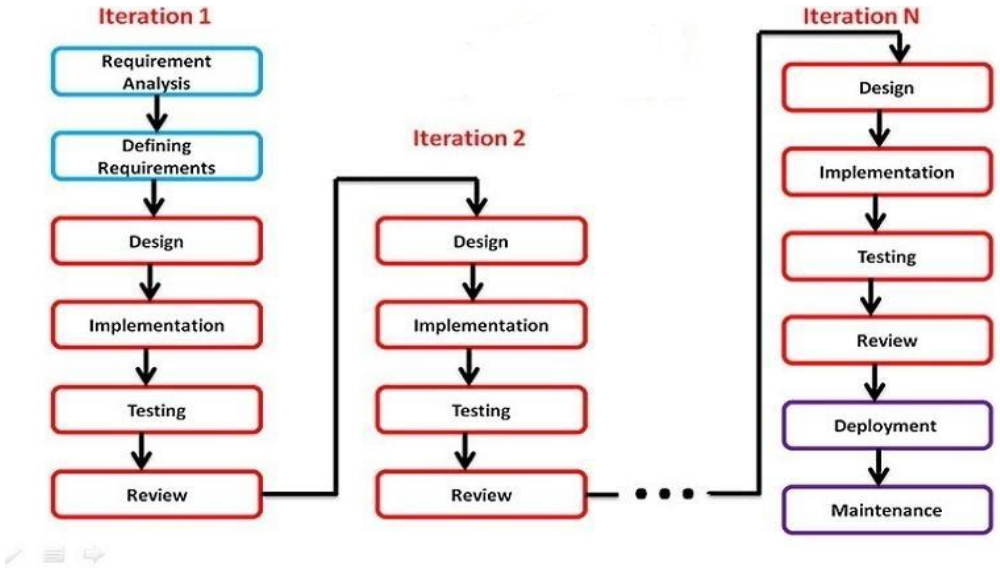


Figure 4: Iterative Process Model

In the iterative model, the process starts with a simple implementation of the software requirements and iteratively enhances the versions until the complete system is implemented and ready to be deployed.

The various phases of iterative model are as follows:

1. Requirement analysis: In the first phase, the identification of the key requirements for the stock prediction system i.e., historical data analysis and sentiment analysis is done. To develop the software under the incremental model, this phase performs a crucial role.

2. Design: In this phase, design of the features to be used in the model, such as moving averages from historical data and sentiment scores from text data are done along with simple LSTM model for time-series prediction using historical data and a basic sentiment analysis module to process textual data.

3. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that implements a basic LSTM model for predicting stock prices based on historical data and perform sentiment analysis based on news and articles.

4. Testing: The testing phase checks the performance of each existing function, tests the LSTM model's accuracy in predicting the stock prices and verifies that the sentiment scores are correctly influencing the predictions.

5. Review: After testing the model's performance, identifying areas for improvement, such as enhancing sentiment analysis or refining the LSTM model is done. Planning to integrate more advanced techniques or additional data sources in the next iteration is done if necessary.

6. Deployment: After completing all the phases, software is deployed to its work environment where users can interact with it and provide further feedback if any improvement or correction is required.

7. Maintenance: In the maintenance phase, after deployment of the software in the working environment there may be some bugs, some errors or new updates are required. Maintenance involves debugging, enhancement of models and expansion of the application.

Advantages of iterative process model:

- Repeated cycles of development allow for continuous improvement and refinement of the software.
- Early prototypes provide a possible product that user can review and provide feedback on.
- Frequent testing and evaluation throughout each iteration help identify and fix defects early.
- Flexibility in adjusting changes is increased, as each iteration provides an opportunity to adjust requirements based on user feedback and evolving needs.

4.2 BLOCK DIAGRAM

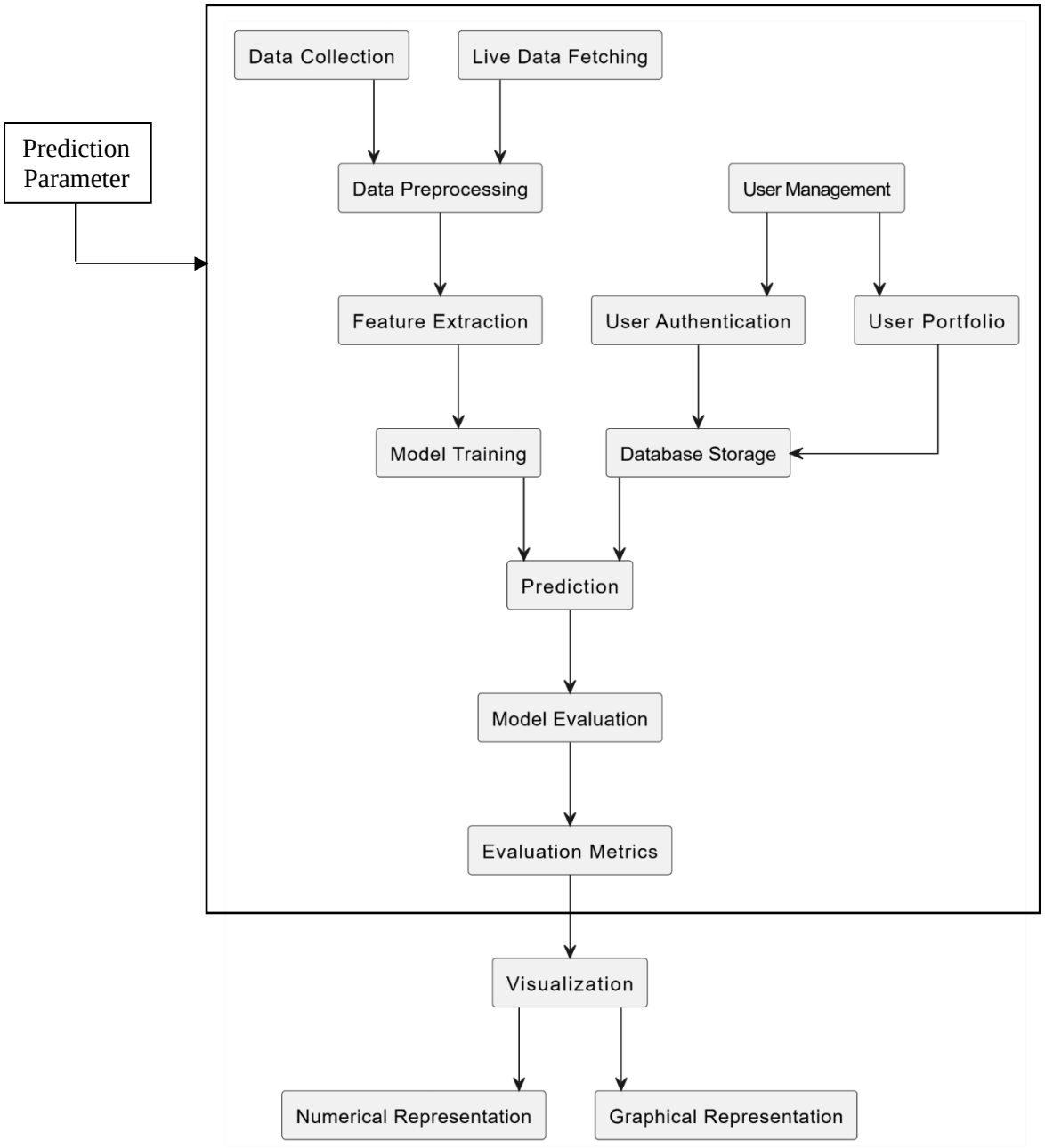


Figure 5: System Block Diagram

4.2.1 Data Collection

Data collection is the foundational step of our stock market prediction system, ensuring the availability of reliable and accurate data for both historical analysis and real-time updates. The data collection process is divided into two parts: historical stock price data collection and live data acquisition, which includes sentiment-related information.

1. Historical Data Collection:

To train our prediction model, we downloaded historical stock price data in CSV format from NepseAlpha, a reliable source for Nepal Stock Exchange (NEPSE) data. This dataset contains critical information such as opening price, closing price, high, low, and trading volume for individual stocks. This data was thoroughly verified to ensure its integrity and completeness before being used for model training. By focusing on high-quality data from NepseAlpha, we ensured that the model was built on a robust foundation of accurate historical trends.

- **Source:** NepseAlpha (a reliable financial data provider for NEPSE stocks)
- **Data Format:** CSV files containing stock prices with columns:
    - **Date:** The specific trading day to which the stock data corresponds.
    - **Open:** The stock's price at the beginning of the trading session on that date.
    - **High:** The highest price the stock reached during the trading session.
    - **Low:** The lowest price the stock traded at during the session.
    - **Close:** The stock's price at the end of the trading session.
    - **Percentage Change:** The relative change in the stock's price compared to the previous day, expressed as a percentage.
    - **Volume:** The total number of shares traded during the session, indicating the stock's market activity.
- **Storage:** The data is downloaded and stored in a local directory to be used for model training.

2. Live Stock Data Collection:

For real-time updates in the web application, live stock market data is acquired using a web scraping script. The script is designed to extract updated stock prices and other relevant information from ShareSansar, a trusted financial news and stock market analysis platform. To maintain accuracy and efficiency, the script is scheduled to run automatically at the end of each trading day, after the market closes. This new data is appended to the previously downloaded historical dataset, ensuring the model stays up-to-date with the latest market trends. The updated CSV file is then used by the web application to provide real-time predictions to users.

- **Source:** ShareSansar (a financial news and stock update platform)
- **Method:** Web scraping using **Selenium and BeautifulSoup**
- **Automation:**
    - The scraping script runs at the end of each trading day.
    - The new data is appended to the existing historical dataset.
    - The updated dataset is used for real-time predictions.

3. Sentiment Data Collection:

To implement sentiment analysis into the system, we utilize two main data sources:

1. Sentiment Word Dataset: A dataset containing a list of positive and negative words is fetched from GitHub repositories. This dataset serves as a reference for determining the sentiment polarity of textual data, helping the system classify whether a news article or post is positive, negative, or neutral.
2. Live News Articles and Social Media Data: Live textual data is collected using another web scraping script. This script is designed to fetch headlines and articles from various financial news websites and social media platforms. These data give insights into the public sentiment around specific companies and market conditions.

- **Sources:** Financial news websites & articles
- **Method:** Web scraping using Selenium & BeautifulSoup

- **Data Format:**
    - News headline
    - Summary
    - Full text
- **Usage:** The collected text data is used for sentiment analysis using the CNN model.

4. Automation and Scalability:

To ensure efficiency, the data collection process is largely automated:

- The web scraping scripts for both stock prices and news headlines are scheduled to run at specific intervals, ensuring the datasets are always up-to-date.
- The entire process is designed to be scalable, allowing easy integration of new stock symbols, additional data sources, or different markets in the future.

By combining high-quality historical data with regularly updated live data, this comprehensive data collection process forms the backbone of our prediction system. It ensures the availability of accurate, timely, and sentiment-enriched data, enabling the model to generate reliable predictions and insights.

4.2.2 Data Preprocessing

Data preprocessing is a critical phase in the stock market prediction system. It ensures that raw data is cleaned, standardized, and transformed into a format suitable for use in machine learning models like LSTM. The preprocessing steps applied to the historical stock price data are explained below, along with the planned steps for sentiment analysis preprocessing.

1. Preprocessing Historical Data for LSTM

a. Handling Missing Values: The historical stock price data collected from NepseAlpha was checked for missing values, such as blank entries for certain dates.

Missing data points were imputed using interpolation methods or ignored based on their relevance to ensure time-series continuity. This ensured that the model learned effectively from sequential data without any inconsistencies.

b. Sorting by Date: The data was sorted chronologically by the "Date" column to maintain the sequential order required for time-series analysis. Chronological order allowed the LSTM model to capture temporal dependencies in the data, which is critical for making future predictions.

c. Scaling the Data: The stock prices were normalized using the MinMaxScaler, scaling values between 0 and 1. This step ensured that all numerical features, particularly the "Close" price, were on a similar scale. Just the close price was use for the initial model training and testing which is to be further expanded for other relevant parameters as well. Normalization was necessary to ensure model works for every company's data as price range for every company varies from each other.

Normalized value of X= (X- Min value of X)/(MAX-MIN)

d. Splitting the Data into Training and Testing Sets: The dataset was divided into 80% for training and 20% for testing. The training set was used to teach the model patterns in the stock price data, while the testing set evaluated its performance on unseen data. This split ensured that the model's accuracy was tested on realistic scenarios.

e. Creating Sequences for LSTM: The data was structured into sequences of 60-time steps, where each sequence contained stock prices from the previous 60 days, and the target was the stock price on the 61st day. A custom function was implemented to create these sequences, and the resulting data was reshaped into a format suitable for LSTM input. This step allowed the model to learn relationships between past and future stock prices, capturing trends and temporal dependencies.

2. Future Prediction Data Preparation

To predict future prices, the last sequence of 60 days from the training data was extracted. This sequence served as the starting point for predicting prices iteratively,

day by day, for the desired number of future days. Each prediction was appended to the sequence to generate the next input, simulating how the model would perform in a real-world scenario.

3. Preprocessing for Sentiment Analysis

Preprocessing for sentiment analysis involves the following steps:

a. Text Cleaning: Special characters, numbers, and irrelevant text from news articles and social media posts are removed.

b. Tokenization: Text are splitted into individual words or phrases to enable easier analysis.

c. Stop-Word Removal and Lemmatization: Common words (e.g., "is," "the") are removed, and words are be converted to their base form (e.g., "running" to "run").

d. Vectorization: Processed text are converted into numerical data using techniques like TF-IDF or word embeddings.

These steps ensure that textual data is prepared for sentiment analysis, allowing sentiment features to be extracted effectively.

This structured preprocessing ensures that historical stock data is clean, normalized, and ready for training, while also laying the groundwork for sentiment analysis. By focusing on data quality, the system is optimized for accurate and reliable predictions.

4.2.3 Feature Extraction

Feature extraction is a crucial step in the stock prediction system that focuses on identifying and isolating meaningful patterns and indicators from the raw data. These features are then used as inputs for the machine learning models, enabling them to better understand trends and make accurate predictions. Below is a detailed breakdown of the feature extraction process in the project:

1. Feature Extraction from Historical Data

The historical stock price data contains several key parameters such as **Open**, **High**, **Low**, **Close**, and **Volume**. To enhance the predictive capability of the model, additional technical features were generated:

- **Moving Averages:** Calculated the average stock price over a specified time period (e.g., 7-day, 30-day moving averages). This smoothed out short-term fluctuations and highlighted overall trends in the data.
- **Percentage Change:** Measured the relative change in stock prices compared to the previous day. This feature provided insight into daily price volatility and market behavior.
- **Volume:** Included as an independent feature since it reflects market activity and liquidity, which often correlates with price movement.

These features were scaled and normalized along with the primary "Close" price to ensure consistency and compatibility with the LSTM model.

2. Feature Extraction from Sentiment Data

For sentiment analysis, textual data such as news articles and social media posts are processed to extract sentiment-based features. These include:

- **Sentiment Polarity:** This indicates whether the sentiment in the text is positive, neutral, or negative.
- **Sentiment Subjectivity:** This measures whether the sentiment is opinion-based (subjective) or fact-based (objective), helping in identifying text that is likely to influence market behavior.
- **Keyword-Based Features:** This identifies specific financial keywords and their context in the text (e.g., "profit," "loss," "growth"), helping to capture industry-specific sentiment relevant to stock predictions.

The process integrates technical indicators from historical stock data with sentiment-based insights, providing a holistic view of market trends. Through feature extraction, only the most relevant and meaningful data is selected, minimizing noise and enhancing

prediction accuracy. The extracted features from both historical and sentiment data are combined into a unified dataset, which serves as input for LSTM and CNN models. These models learn patterns and relationships that influence market movements, bridging the gap between raw data and actionable insights. This process lays a strong foundation for accurate and effective stock price predictions.

4.2.4 Model Training

Model training is a vital phase in the stock prediction system where the machine learning models are taught to recognize patterns and relationships in the data. This phase ensures that the models can generalize well to make accurate predictions on unseen data. The process of training models in the project, specifically the LSTM for historical data and the CNN for sentiment analysis, is explained in detail below.

1. Training the LSTM Model for Historical Data

The Long Short-Term Memory (LSTM) model, a type of recurrent neural network (RNN), was used for analyzing sequential stock price data.

- **Input Data Preparation:** Historical stock data was preprocessed into sequences of 60-time steps, where each sequence represented stock prices over 60 days, and the target was the stock price on the 61st day. The data was divided into 80% training and 20% testing sets to ensure the model could generalize well to new data.
- **Model Architecture:** The LSTM model consisted of multiple layers: Two LSTM layers with 50 units each to capture temporal dependencies and a dense output layer to predict the closing price. The input shape was designed to match the sequence length (timesteps) and number of features.
- **Training Process:** The model was compiled with 'adam' optimizer to adjust weights efficiently during backpropagation and mean squared error (MSE) as the loss function to minimize prediction errors. It was trained over 100 epochs with a batch size of 64, balancing computation time and convergence.

- **Output:** The trained LSTM model was saved in .keras format to be integrated into the web application for real-time predictions.

2. CNN Model for Sentiment Analysis

The Convolutional Neural Network (CNN) is be employed to analyze textual data and extract sentiment-related features.

- **Input Data Preparation:** Textual data is be preprocessed (tokenized, vectorized, and converted into embeddings) to represent each word numerically. Sentiment labels (positive, neutral, or negative) are be assigned based on predefined datasets or manually labeled examples.
- **Model Architecture:** The CNN model include:
    - Convolutional layers to identify sentiment-related patterns in the text.
    - Pooling layers to reduce dimensionality and focus on important features.
    - Dense layers for classification of sentiment polarity and extraction of sentiment scores.
- **Training Process:** The model will be trained using labeled textual data from news articles and social media. Cross-entropy loss will be used as the loss function for classification tasks. Performance will be monitored through metrics like accuracy, precision, and recall.

3. Continuous Evaluation and Improvement

During training, evaluation metrics like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) were used to measure the LSTM model's accuracy. The training process also included hyperparameter tuning (e.g., adjusting the learning rate and number of epochs) to optimize the model's performance. After implementing sentiment analysis, the impact of sentiment on predictions will be analyzed to ensure its relevance and accuracy.

By designing and training models specifically for their tasks, the system leverages LSTM's strength in handling sequential data and CNN's ability to analyze textual data. This structured approach ensures the models are robust and capable of generating reliable stock market predictions.

4.2.5 Model Integration:

Model integration implements the outputs and features of the machine learning models into the web application, enabling users to see and analyze the output and interact with the system. This process implements the LSTM model, which analyzes historical stock price data and the CNN model, which processes sentiment data from news articles and social media. By implementing quantitative trends and qualitative sentiment insights, the users can get insights on the market behaviors.

Inputs from Each Model:

- **LSTM Model:** Processes time-series data from historical stock prices to predict future prices. It captures long-term and short-term trends, technical patterns, and market volatility.
- **CNN Model:** Processes textual data to extract sentiment-related features, such as polarity (positive, neutral, or negative) and subjectivity (fact-based or opinion-based). These features reflect market sentiment, which can influence short-term stock price changes.

4.2.6 Model Evaluation

Model evaluation ensures the accuracy and reliability of the stock prediction system by assessing the performance of the LSTM model for historical data and CNN model for sentiment analysis.

1. Evaluation Metrics for LSTM Model

- **Mean Squared Error (MSE):** Measures the average squared difference between predicted and actual prices, with lower values indicating better accuracy.
- **Root Mean Squared Error (RMSE):** Expresses error in the same units as stock prices, offering a clear understanding of prediction accuracy.
- **Mean Absolute Percentage Error (MAPE):** Shows the average percentage error, helping evaluate relative accuracy.

- **Graphical Analysis:** Plots actual vs. predicted prices for visual comparison, highlighting patterns and deviations.

2. Evaluation Metrics for Sentiment Analysis

- **Accuracy:** Percentage of correctly classified sentiments (positive, negative, neutral).
- **Precision, Recall, F1 Score:** Key metrics to evaluate sentiment classification performance.
- **Confusion Matrix:** Provides detailed insights into correct and incorrect classifications.

3. Tools and Techniques

- **Python Libraries:** scikit-learn for metrics and Matplotlib for visualization.
- **Cross-Validation:** Ensures robustness by splitting data for repeated training and validation.

By rigorously evaluating models, the system ensures reliable predictions, supports informed investment decisions, and identifies areas for refinement.

4.2.7 Prediction

Prediction is the final and most critical step of the stock market prediction system, where the trained models are used to forecast future stock prices. This step combines the outputs of historical data analysis and, eventually, sentiment analysis to generate accurate and insightful predictions that assist investors in making informed decisions.

1. Predictions Using LSTM Model

- **Historical Data Prediction:**
    - The trained LSTM model takes a sequence of past stock prices (e.g., data from the last 60 days) as input.

    - It predicts the price for the next day by identifying patterns and trends from the historical data.
    - This process is repeated iteratively to forecast prices for multiple future days (e.g., 30 or 60 days).
- **Trend Continuity:**
    - To ensure consistency, the predicted price for the first day in the future is used as the starting point for predicting the next day, maintaining a realistic trend.
    - An offset is applied to align future predictions with the last predicted price from historical data, ensuring smooth continuity.

2. Analysis with Sentiment Analysis

- **Sentiment Adjustment:**
    - Sentiment scores extracted from news articles or social media will be displayed as output for relative news articles.
    - Positive sentiment may boost predicted prices, while negative sentiment may lead to downward adjustments.

3. Outputs

- **Graphical Representation:**
    - The predicted prices are plotted alongside actual historical data for easy comparison.
    - A future price curve extends beyond the known data, providing a visual forecast of market trends.
- **Tabular Representation:**
    - Future predictions are presented in a table format, showing day-by-day forecasted prices for a specified period.

4. Tools and Implementation

- The trained LSTM model is loaded using TensorFlow/Keras, and predictions are generated programmatically.
- Chart.js and Matplotlib are used to visualize predictions in the web application, while tabular data is dynamically generated for clarity.

By leveraging prediction capabilities, the system provides actionable insights into future stock trends, helping users assess market opportunities and risks with confidence.

4.2.8 Visualization

Visualization plays a crucial role in the stock market prediction system, enabling users to interpret the model's predictions and insights in an intuitive and user-friendly manner. By presenting data through charts, tables, and graphs, the system effectively communicates trends, patterns, and predictions to assist users in making informed investment decisions.

1. Visualization of Historical and Predicted Data

- **Actual vs. Predicted Prices:** The model's predictions for historical stock prices (from the test dataset) are plotted alongside the actual stock prices on a line graph. This comparison allows users to visually assess the accuracy of the model. This graph is typically color-coded (e.g., blue for actual prices and red for predicted prices) to distinguish between the two data sets clearly.
- **Future Price Predictions:** Predicted prices for the future period (e.g., the next 7 days) are plotted as an extension of the historical graph. The future prediction curve starts from the last predicted value and highlights the projected stock price trend.
- **Tabular Representation:** Predicted prices for future days are also displayed in a table format, where each row corresponds to a day, and the columns show the day number and predicted stock price. This tabular data complements the graph by providing precise numerical values for each prediction.

2. Visualization of Sentiment Analysis Results

Sentiment Trends Over Time: Sentiment score will visualize sentiment polarity over time. Positive, neutral, and negative sentiment scores will be evaluated regularly to show how public sentiment fluctuates over time.

3. Web Application Integration

- The visualizations are integrated into the web application, where users can select a stock symbol and specify the prediction duration.
- Once the prediction is processed, the web application dynamically updates the visual elements, displaying:
    - A graph comparing actual vs. predicted prices and future predictions.
    - A table showing detailed numerical values for future predictions.

Dynamic Visualization of Predictions and Data

The web application dynamically takes input from the user and displays stock predictions and other relevant data in an interactive manner. The system ensures that users receive real-time, personalized insights into stock trends, both historical and predicted.

1. Input Selection by Users

- The frontend consists of a dropdown menu where users select the stock they want to analyze.
- A slider or input field allows users to choose the number of days for future predictions (e.g., 7, 15, or 30 days).
- When the user makes a selection, an AJAX request is sent to the Flask backend, triggering a function that loads the corresponding trained LSTM model and stock dataset.

2. Fetching Data Dynamically

- The Flask API receives the selected stock symbol and prediction duration.
- The backend:

- Loads the corresponding LSTM model from stored .keras files.
- Fetches the latest historical stock data from the database.
- Runs the prediction algorithm on the selected timeframe.

- The predicted stock prices are sent back to the frontend as JSON data.

3. Presenting Predictions in an Interactive Manner

- **Graphical Visualization (Chart.js):**
  - A line chart dynamically updates to show actual vs. predicted prices.
  - Color-coded lines differentiate between historical and future price trends.
  - The chart automatically resizes and scales based on user input.
- **Tabular Representation:**
  - The data is also displayed in a structured table format.
  - Each row corresponds to a predicted day with a date and expected price.
- **Historical Data Representation:**
  - Users can view historical stock trends using line graphs.

4. Real-Time Updates and User Interactivity

- The frontend updates dynamically without requiring a full page reload.
- When a user selects a different stock or time range, the new data is instantly fetched and displayed.
- The system ensures seamless navigation and data updates, enhancing user experience and usability.

4.3 User Login and Portfolio Management

The web application includes a user authentication system and personalized stock portfolio management. This allows users to securely log in and track their investments over time.

4.3.1 User Authentication System

1. Signup Process

- Users create an account by entering **username, email, and password**.
- The system:
    - Checks if the email already exists.
    - Hashes the password for security before storing it in the SQLite database.
    - Saves the user profile details.

2. Login Process

- Users enter their credentials to access their personalized dashboard.
- Flask-Login handles session management and ensures users stay logged in securely.
- Once logged in, users are redirected to their dashboard, where they can:
    - View their saved stock portfolio.
    - Update their profile information.
    - Access personalized predictions based on their investment preferences.

3. Logout Feature

- Users can log out, which clears their session data and redirects them to the homepage.

4.3.2 Portfolio Management

1. Adding Stocks to Portfolio

- Users can search and add stocks to their portfolio, specifying the quantity they own.
- The stock data is saved in the database, ensuring **persistent storage**.
- Duplicate stock entries are prevented using a unique constraint.

2. Viewing Portfolio Data

- The dashboard displays the list of stocks the user has invested in.
- Each stock entry includes:
    - Stock name
    - Quantity owned
    - Current stock price
    - Percentage change in value
- Users can see a summary of their total investment value.

3. Modifying and Removing Stocks

- Users can edit stock quantities if they buy or sell more shares.
- They can also remove stocks from their portfolio if they no longer track them.

4.3.3 Portfolio Prediction

The stock price prediction is extended to portfolio forecasting feature, allowing users to estimate their overall investment performance for the next seven days. Using the LSTM model, we predict individual stock prices, which are then aggregated based on the user's portfolio composition. The system calculates expected portfolio value changes by considering stock-wise predictions, weightage, and historical trends.

This feature helps investors anticipate short-term portfolio fluctuations, assess potential risks, and make informed decisions.

4.3.4 Secure and Personalized Experience

- Each user's portfolio is securely stored and accessible only to them.
- The application ensures data privacy using authentication mechanisms.
- Future enhancements may include email notifications and stock alerts based on portfolio performance.

4.4 ALGORITHMS

4.4.1 Recurrent Neural Network (RNNs)

Relevance to Our Project:

- Used for analyzing historical stock prices as stock data is sequential.
- Struggles with long-term dependencies, which leads to the use of LSTM.

4.4.2 Long Short-Term Memory (LSTM)

Relevance to Our Project:

- Our model uses LSTM to predict future stock prices based on historical trends.
- LSTM captures long-term dependencies in stock price fluctuations more efficiently than standard RNNs.
- Helps in making more accurate stock price predictions based on past trends.

In this project, the LSTM model is used to predict future stock prices based on historical data.

- **Training Data:** The model is trained on sequences of past stock prices (e.g., the last 60 days) to predict the next day’s price.
- **Handling Time-Series Data:** The model effectively captures patterns such as moving averages, volatility, and trends over time.
- **Output:** After training, the LSTM model generates predictions for future stock prices, which are displayed as graphs and tables in the web application.

4.4.3 Convolutional Neural Networks (CNNs)

Sentiment analysis involves processing textual data, such as news headlines and social media posts, to determine the sentiment (positive, negative, or neutral). CNN is well-suited for this task because:

- It can identify patterns and features in text, such as recurring sentiment indicators (e.g., "bullish," "growth").
- It is computationally efficient and can handle large volumes of text data.

Application in the Project:

In the context of this project, CNN is used to extract sentiment-related features from textual data:

- **Input Data:** News articles and social media posts are preprocessed into numerical representations (e.g., word embeddings).
- **Feature Extraction:** The CNN model analyzes the text to identify sentiment polarity and subjectivity, which are quantified into scores.
- **Output:** The sentiment scores are used to enhance the accuracy of stock price forecasts.

4.5 System Architecture

The stock market prediction system consists of three core components:

1. **Backend (Flask):** Handles data processing, machine learning predictions, and database operations.
2. **Frontend (HTML/CSS/JavaScript):** Provides an interactive user interface for viewing predictions, managing portfolios, and tracking stock trends.
3. **Machine Learning (LSTM & CNN):** LSTM models trained for stock price predictions and CNN models in development for sentiment analysis.

4.5.1 Backend

1. Flask Initialization

The backend is built using Flask, a lightweight web framework for handling API requests, user authentication, and database interactions.

Key Features:

- **SQLAlchemy ORM:** Enables database interactions using Python classes instead of SQL queries.
- **Flask-Login:** Manages user authentication securely.
- **Session Management:** Encrypts session data for logged-in users.

2. Database Models

User Model

Column	Type	Description
Id	INTEGER	Primary key (Auto-increment)
Username	VARCHAR(50)	Unique username
Email	VARCHAR(100)	Unique email
password_hash	VARCHAR(128)	bcrypt hashed password

- **Security:** Passwords are stored as hashes using `pbkdf2:sha256` encryption.
- **Unique Constraints:** Email and username must be unique to prevent duplicate accounts.

Portfolio Model

Column	Type	Description
id	INTEGER	Primary key
user_id	INTEGER	Foreign key linking to Users
symbol	VARCHAR(10)	Stock symbol (e.g., NICL)
quantity	INTEGER	Number of shares held

- **User-Stock Relationship:** A one-to-many relationship exists between users and stocks.

- **Unique Constraint:** Prevents duplicate entries for the same stock under a single user.

3. Authentication Workflow

Signup Process

- User enters **username, email, and password** in the signup form.
- System checks for an existing email
- Password is **hashed** before storing in the database:

Login Process

- User submits **email and password**.
- System verifies credentials:
- **Session Management:** Flask-Login handles authentication and user session tracking.

4.5.2 Model Architecture

The model is built using LSTM (Long Short-Term Memory) networks, ideal for time-series forecasting.

```
x_train, y_train = create_sequences(train_data, seq_length)
x_test, y_test = create_sequences(test_data, seq_length)
x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 4))
x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 4))

model = Sequential([
       Input(shape=(seq_length, 4)),  # Input layer expects (60, 4)
       LSTM(100, return_sequences=True),
       Dropout(0.2),
       LSTM(100, return_sequences=False),
       Dropout(0.2),
       Dense(50, activation='relu'),
       Dense(4)  # 4 outputs (OHLC)
])
```

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=32, epochs=50, verbose=1,
validation_data=(x_test, y_test))
model.save(os.path.join(model_dir, f"{symbol}.keras"))
```

- **LSTM Layers:** Capture sequential dependencies in stock prices.
- **Dense Layer:** Outputs a single predicted price.

Training Process:

1. **Data Preparation:**
    - Historical stock data is split (80% training, 20% testing).
    - Sequences of 60 days are used to predict the 61st day.
2. **Hyperparameters:**
    - **Epochs:** 100 iterations
    - **Batch Size:** 64 samples per update
3. **Model Saving:**
    - Each model is saved as `[stock_symbol].keras` for efficient retrieval.

4.6 Step-by-Step Project Flow

4.6.1 Data Collection & Storage

The system gathers data from multiple sources to build an accurate and up-to-date stock prediction model.

1. Historical Data Collection

- **Source:** NepseAlpha (a reliable financial data provider for NEPSE stocks)
- **Data Format:** CSV files containing stock prices with columns:
    - Date
    - Open price
    - High price
    - Low price
    - Close price
    - Trading volume
- **Storage:** The data is downloaded and stored in a local directory to be used for model training.

2. Live Data Collection

- **Source:** ShareSansar (a financial news and stock update platform)
- **Method:** Web scraping using Selenium and BeautifulSoup
- **Automation:**
    - The scraping script runs at the end of each trading day.
    - The new data is appended to the existing historical dataset.
    - The updated dataset is used for real-time predictions.

3. Sentiment Data Collection

- **Sources:** Financial news websites & articles
- **Method:** Web scraping using Selenium & BeautifulSoup
- **Data Format:**
    - News headline
    - Summary

    - Full text
- **Usage:** The collected text data is used for sentiment analysis using the CNN model.

4.6.2 Data Preprocessing & Feature Engineering

Before feeding the data into the models, it undergoes various preprocessing steps.

1. Stock Data Preprocessing

- **Handling Missing Data:**
    - If any values are missing, they are either interpolated or removed.
- **Sorting by Date:**
    - Ensures stock prices are in chronological order for time-series prediction.
- **Feature Scaling:**
    - Uses MinMaxScaler to normalize stock prices between 0 and 1.
- **Creating Sequences for LSTM:**
    - Converts stock data into 60-day sequences, where X (input) = past 60 days' prices and Y (output) = next day's price.

2. Sentiment Data Preprocessing

- **Text Cleaning:**
    - Removes special characters, numbers, and redundant words.
- **Tokenization:**
    - Splits text into words for analysis.
- **Vectorization:**
    - Converts text into numerical form using word embeddings.
- **Labeling Sentiments:**
    - Classifies news articles as positive, neutral, or negative.

4.6.3 Model Training & Evaluation

1. LSTM Model for Stock Prediction

- **Architecture:**
  - Two LSTM layers with 50 neurons each.
  - Dense layer to output predicted stock prices.
- **Training Process:**
  - 80% of data used for training, 20% for testing.
  - Trained for 100 epochs with an Adam optimizer.
- **Evaluation Metrics:**
  - Mean Absolute Error (MAE)
  - Root Mean Squared Error (RMSE)
  - Mean Absolute Percentage Error (MAPE)
- **Model Storage:**
  - Each trained model is saved as a .keras file for later use.

2. CNN Model for Sentiment Analysis

- **Architecture:**
  - Convolutional layers extract features from text.
  - Fully connected layers classify sentiment.
- **Training:**
  - Sentiment-labeled text data is fed into the model.
- **Integration:**
  - The sentiment score will later be implemented in web application to refine stock forecasts.

4.6.4 Web Application Development

1. Backend (Flask & Database Management)

- **Flask Framework:** Handles API requests and serves web pages.
- **Flask-Login & Flask-SQLAlchemy:**
  - Manages user authentication (login, signup, session handling).
  - Stores user profile & stock portfolio in an SQLite database.
- **Prediction API:**
  - Receives stock symbol & days from frontend.
  - Loads the corresponding LSTM model.
  - Returns predicted stock prices to the frontend.

2. Frontend (HTML, CSS, JavaScript, Chart.js)

- **Dropdown Menu:** Allows users to select a stock symbol.
- **Prediction Table:** Displays forecasted stock prices in a table format.
- **Graphical Visualization (Chart.js):**
  - Line charts show actual vs. predicted stock prices.
  - Displays historical trends & future projections.
- **Accuracy Metrics:** Shows MAE, RMSE, and MAPE to assess prediction reliability.

3. User Features

- **Profile Page:**
  - Users can update username, bio, and portfolio.
- **Portfolio Management:**
  - Users can add/remove stocks they are tracking.
- **Live Stock Updates:**
  - Displays latest scraped stock prices.
- **News Sentiment Analysis:**
  - Will display sentiment scores alongside stock trends.

4.6.5 Deployment & Scalability

- **Model Deployment:**
  - LSTM models are loaded on-demand for real-time predictions.
- **Web Scraping Automation:**
  - Runs daily to update datasets.
- **Scalability:**
  - Supports new stock symbols by training new models.
- **Cloud Hosting (Future Work):**
  - The system will be deployed on cloud platforms like AWS/GCP for scalability.

4.7 FLOWCHART

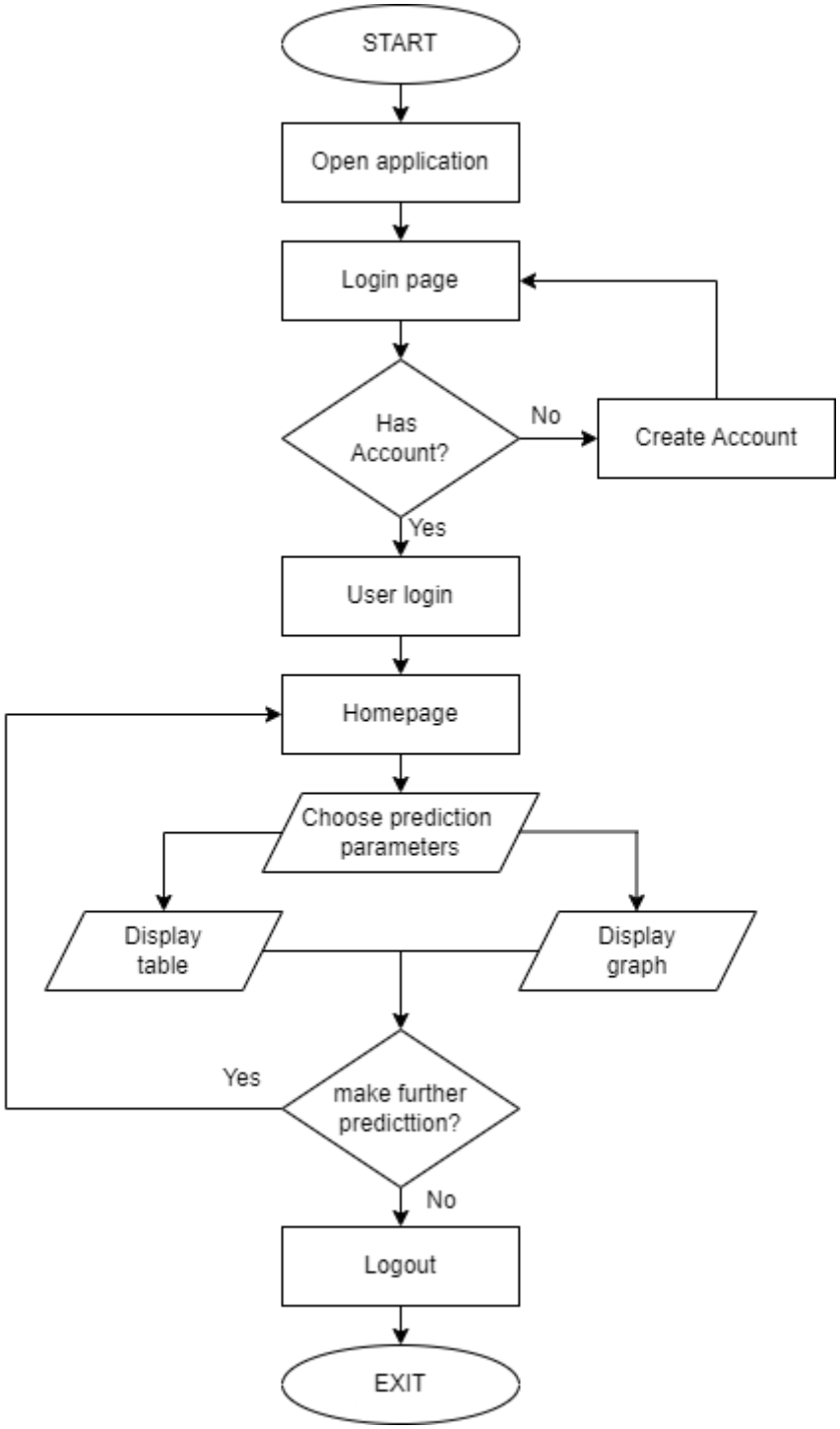


Figure 6: Flowchart

4.8 Necessary Diagrams

4.8.1 Data Flow Diagram

The Data Flow Diagram of the Stock Market Prediction is shown below:

Data Flow Diagram - 0

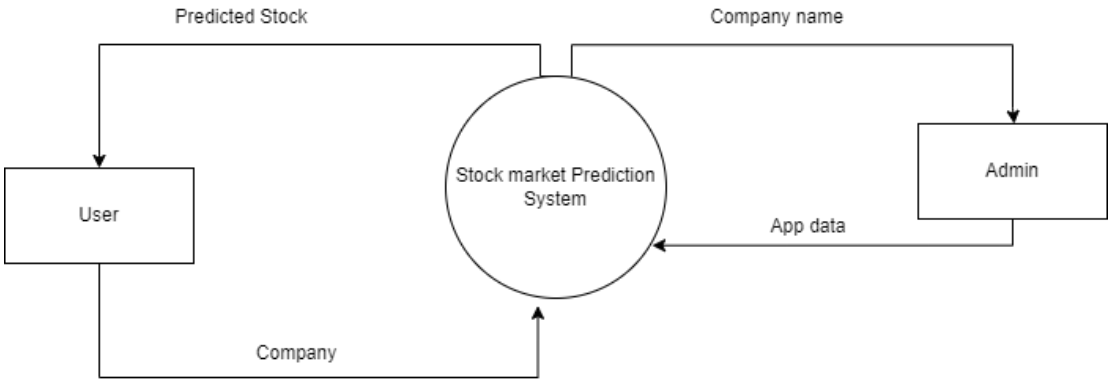


Figure 7: Data Flow Diagram 0

Data Flow Diagram – 1

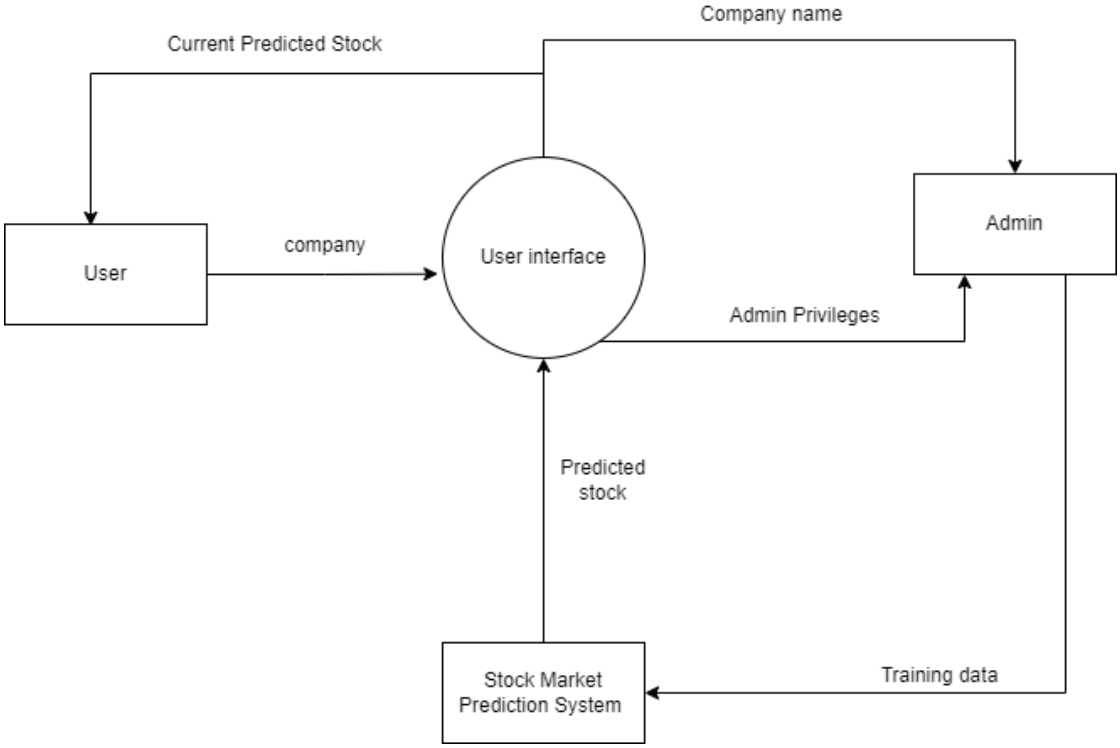


Figure 8: Data Flow Diagram Level - 1

Data Flow Diagram Level – 2

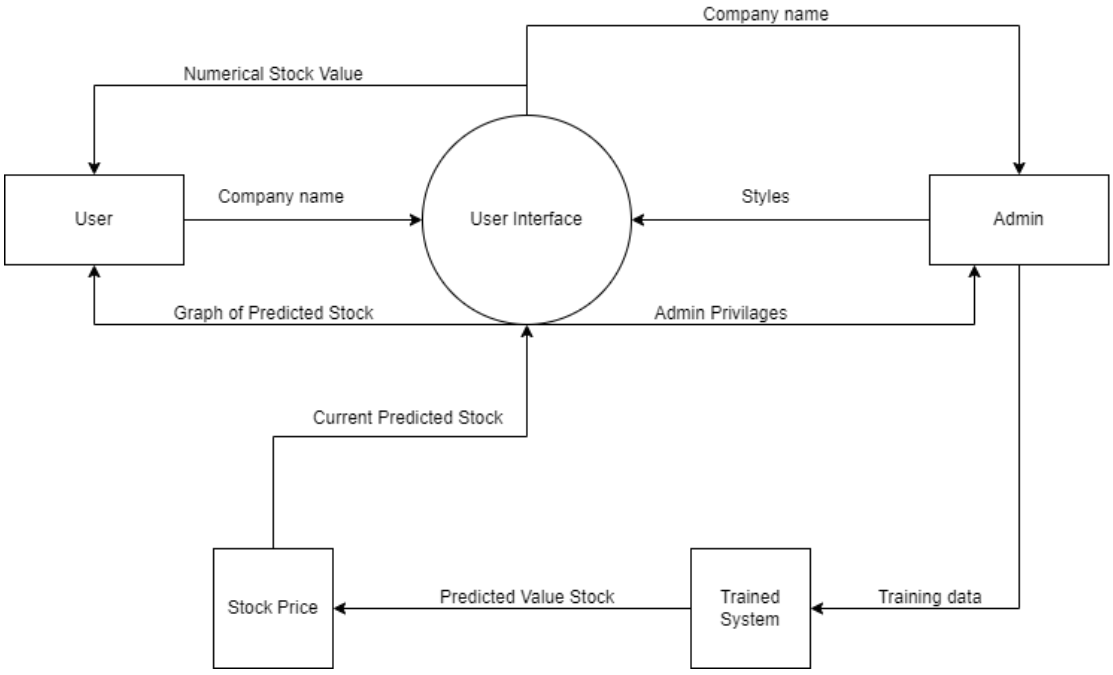


Figure 9: Data Flow Diagram Level-2

4.8.2 Use Case Diagram

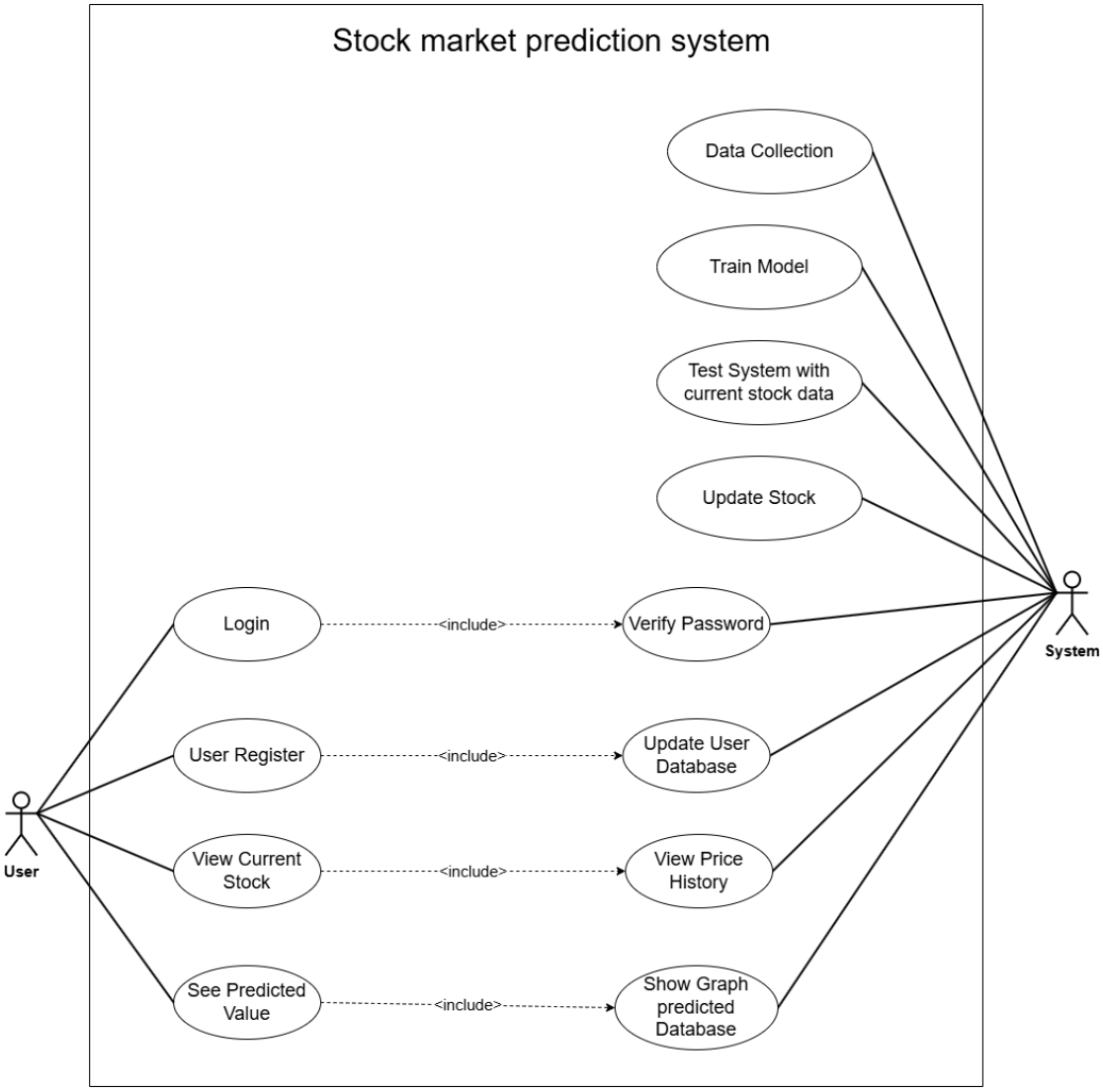


Figure 10: Use Case Diagram

4.8.3 Sequence Diagram

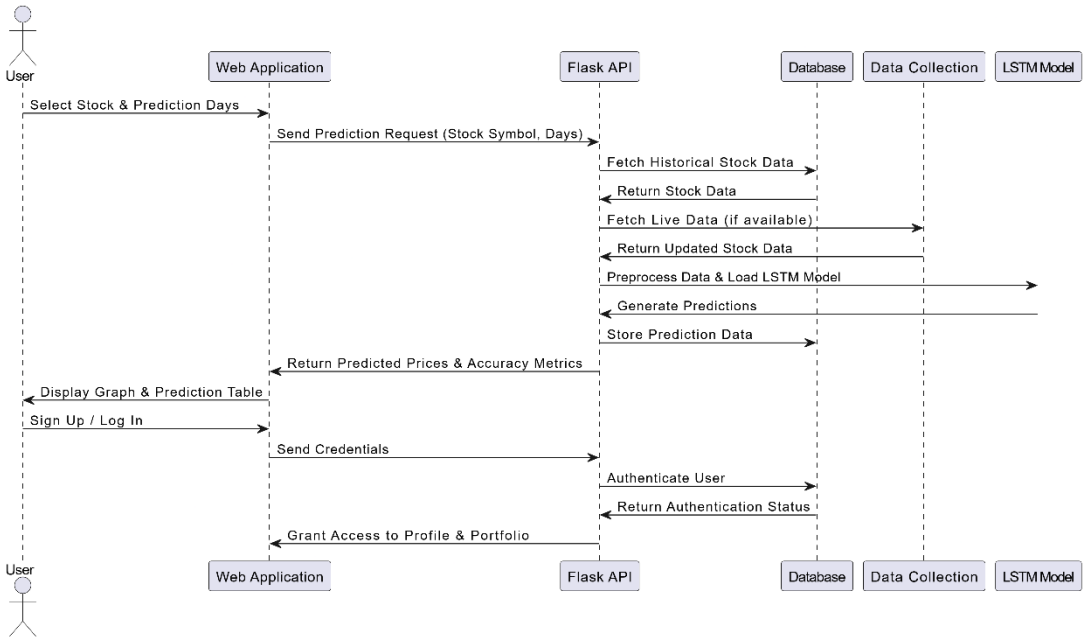


Figure 11: Sequence Diagram

4.9 TOOLS USED

4.9.1 Python

Python has been used for most of the works in this project which has been explained below:

- Building and visualizing the LSTM model using libraries pandas, numpy, sklearn, tensorflow, matplotlib.
- Training individual LSTM models for multiple companies.
- Implementing and training a Convolutional Neural Network (CNN) for sentiment analysis using TensorFlow and Keras.
- Integrating the model into the web application using Flask.
- Fetching live and updated datasets via a web scraping script using BeautifulSoup and Selenium.
- Managing the SQLite database for storing user profiles and investment records using Flask-SQLAlchemy.

4.9.2 HTML/CSS/JavaScript

HTML, CSS, and JavaScript have been used to create the interactive and easy-to-use frontend UI of the web application.

- **HTML**: Used to structure the content of web pages.
- **CSS**: Applied for styling and formatting the appearance of the webpages.
- **JavaScript**: Utilized to fetch and display numerical and graphical data from the backend.
- **Chart.js**: Used to visualize stock price predictions and historical trends.
- **AJAX & Fetch API**: Implemented to dynamically update data on the frontend without reloading the page.

4.9.3 Flask & Flask-Login

- **Flask**: Used as the web framework for handling requests, routing, and rendering templates.
- **Flask-Login**: Integrated to manage user authentication, including login, signup, and session handling.
- **Flask-SQLAlchemy**: Used to manage the SQLite database for storing user data and portfolio records.

4.9.4 GitHub

- **Dataset Source**: GitHub has been a source for certain datasets required for model training.
- **Version Control**: Used as an online repository for the project, ensuring proper collaborative workflow among the team members.
- **Code Management**: Facilitates version control, collaboration, and tracking changes in scripts.

4.9.5 Selenium & BeautifulSoup

- **Selenium**: Used for automating the process of fetching live stock price data from financial websites.
- **BeautifulSoup**: Utilized for parsing and extracting textual data from news articles for sentiment analysis.

4.9.6 SQLite

- **Database Management**: SQLite has been used to store user profiles, stock portfolios, and investment records.
- **Data Persistence**: Ensures that user data and stock-related records remain stored across multiple sessions.

4.9.7 TensorFlow & Keras

- **LSTM Model Training**: Used for building and training deep learning models for stock price prediction.

- **CNN for Sentiment Analysis**: Employed to process financial news data and classify sentiment as positive, negative, or neutral.
- **Model Deployment**: TensorFlow facilitates saving and loading models in the web application for real-time predictions.

4.9.8 Chart.js

- **Data Visualization**: Integrated into the frontend to display stock trends and prediction results in graphical formats.
- **Historical Data Representation**: Used to plot actual vs. predicted prices for better user insights.

By integrating these tools and technologies, the project ensures an efficient, scalable, and user-friendly stock prediction system that incorporates historical data analysis, sentiment-based insights, and interactive web functionalities.

CHAPTER 5: EPILOGUE

5.1 RESULT, ANALYSIS AND CONCLUSION

The project involved several key components, starting with **data collection**, where historical stock market data was gathered for analysis and modeling. Additionally, datasets for natural language processing were collected from news articles to aid in sentiment analysis.

For **web scraping**, a script was developed to fetch live stock market data, ensuring real-time updates. Another script was created to scrape news headlines and tweets, providing relevant data for sentiment analysis.

In **model development**, an LSTM model was trained using historical stock market datasets. This trained model was then integrated into the web application to enable real-time stock price predictions.

The **user interface** was designed and implemented with an interactive and visually appealing layout, enhancing the overall user experience.

The **database setup** involved establishing a database to manage user information within the web application and integrating the prediction model into it. This allows users to create profiles, maintain their investment portfolios efficiently and view estimated investment performance for future.

For **sentiment analysis** model, we used CNN based model to process Nepali stock market news headline and predict its sentiment as positive, negative or neutral.

NepseDarshan

Sign Up Login Profile About

Stock Prediction

Please choose stock name:

ADBL

Number of days to predict:

10 Days

Predict

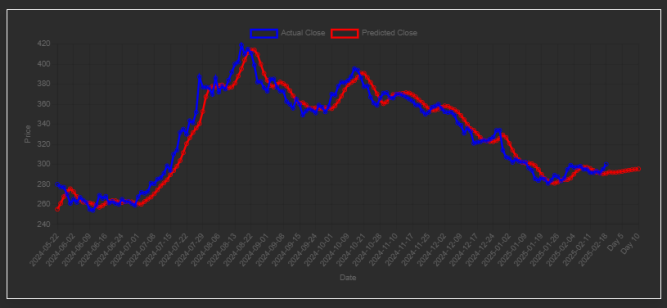
Historical Data Statistics

Statistic	Open	High	Low	Close
Average	360.58	365.15	354.65	359.35
Min	225	225	223	223.4
Max	604.8	620	590	597
Standard Deviation	94.85	96.27	92.91	94.56
Variance	8996.05	9268.74	8632.54	8940.75

Prediction Results

Day	Open	High	Low	Close
1	292.65	295.74	289.46	292.24
2	292.23	295.36	289.18	291.8
3	292.33	295.43	289.26	291.93
4	292.71	295.76	289.57	292.36
5	293.2	296.19	289.95	292.92
6	293.71	296.64	290.35	293.5
7	294.21	297.08	290.73	294.05
8	294.67	297.49	291.07	294.56
9	295.06	297.85	291.38	294.98
10	295.4	298.17	291.64	295.33

Prediction Graph (LTP)



Accuracy Metrics

Metric	Open	High	Low	Close
MAE (Rs)	6.36	7.49	5.91	7.8
RMSE (Rs)	8.46	10.41	7.78	10.27
MAPE (%)	1.91%	2.24%	1.8%	2.34%

MAE: Mean Absolute Error

RMSE: Root Mean Square Error

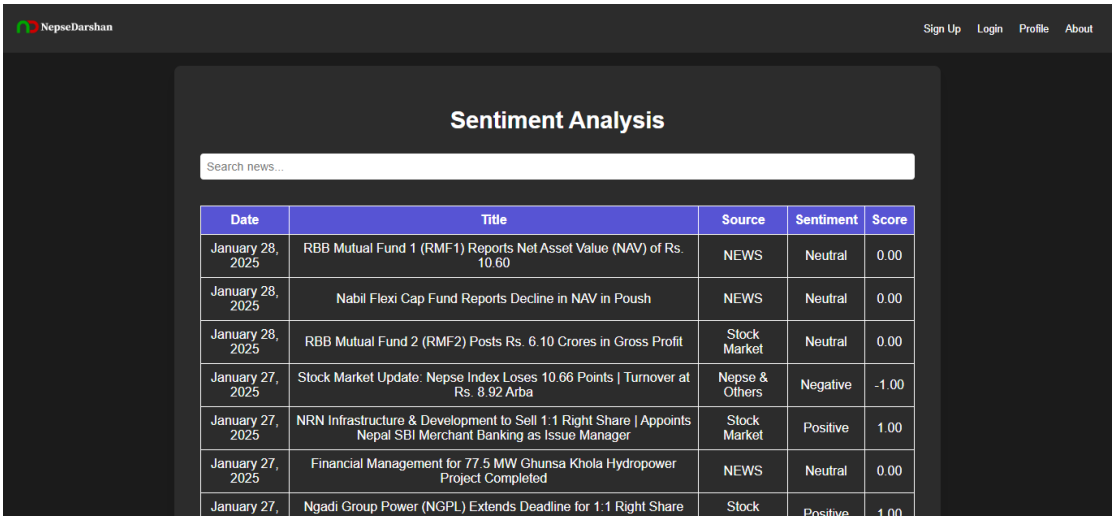
MAPE: Mean Absolute Percentage Error

Historical Data Statistics Table: This table summarizes key statistical parameters of past stock prices for the selected company (e.g., ADBL). It includes average, minimum, maximum, standard deviation, variance. These statistics help users understand past market behavior and price volatility.

Prediction Results Table: This table presents the model's stock price forecasts for the next selected number of days (e.g., 10 days). It includes day, open, high, low, close stock prices (predicted) for each trading day. These predictions help investors anticipate price trends and make informed decisions.

Prediction Graph (LTP - Last Traded Price): This graph visually compares actual vs. predicted closing prices over time. The blue line represents actual past closing prices and the red line represents the LSTM model's predicted closing prices. A close overlap of the two lines indicates the model's accuracy in tracking historical trends and forecasting future prices.

Accuracy Metrics Table: This section evaluates the model's performance using three key metrics: MAE (Mean Absolute Error), RMSE (Root Mean Square Error) and MAPE (Mean Absolute Percentage Error). These metrics help assess the reliability of the model's predictions.



The sentiment analysis table presents news articles related to the stock market, categorized based on their sentiment. The key components include:

1. Date: The publication date of the news.
2. Title: The headline of the news article.
3. Source: The origin of the news (e.g., NEWS, Stock Market).
4. Sentiment: Classified as Positive, Neutral, or Negative based on the article's content.
5. Score: A numerical representation of sentiment
    - Positive (1.00) – Indicates favorable market conditions.
    - Neutral (0.00) – No significant impact on market sentiment.
    - Negative (-1.00) – Suggests a possible downturn or risk in the market.

Users can search for specific news to analyze market trends and make informed investment decisions.

5.2 FUTURE ENHANCEMENT

Integration of sentiment analysis with historical prediction:

Future improvements can include sentiment analysis for individual stocks, providing more precise market insights. Sentiment scores from news and social media can be integrated with the existing prediction model to enhance accuracy. This will create a more comprehensive tool for informed investment decisions. It can be further improved by expanding data sources and refining sentiment analysis techniques.

Adding admin privileges:

- **User Management** – Admins can manage user accounts and control access.
- **Stock Data Management** – Add, update, or remove stock data for accuracy.
- **Monitoring & Moderation** – Track user activity and prevent misuse.
- **Model Updates** – Evaluate and retrain prediction models as needed.
- **Security & Access Control** – Implement MFA and activity logging for admins.

This ensures better data accuracy, security, and system reliability.

REFERENCES

[1] Polepally, Vijay & Jakka, Neha & Vishnukanth, Pendly & Raj, Rachakatla & Anish, Gudavelli. (2023). An Efficient Way to Predict Stock Price using LSTM-RNN Algorithm. *7th International Conference on Intelligent Computing and Control Systems (ICICCS), 1240-1244*, 2023.

[2] Janak Kumar Lal, Arun Kumar Timalsina. (2022). A CNN-BIGRU Method for Stock Price Prediction. Proceedings of 11th IOE Graduate Conference, *2350-8914 (Online), 2350-8906 (Print),* 2022.

[3] Kabir, Md Humayun & Sobur, Abdus & Amin, Md Ruhul. (2023). Stock Price Prediction Using the Machine Learning. *International Journal of Creative Research Thoughts (IJCRT). f946-f950,* 2023

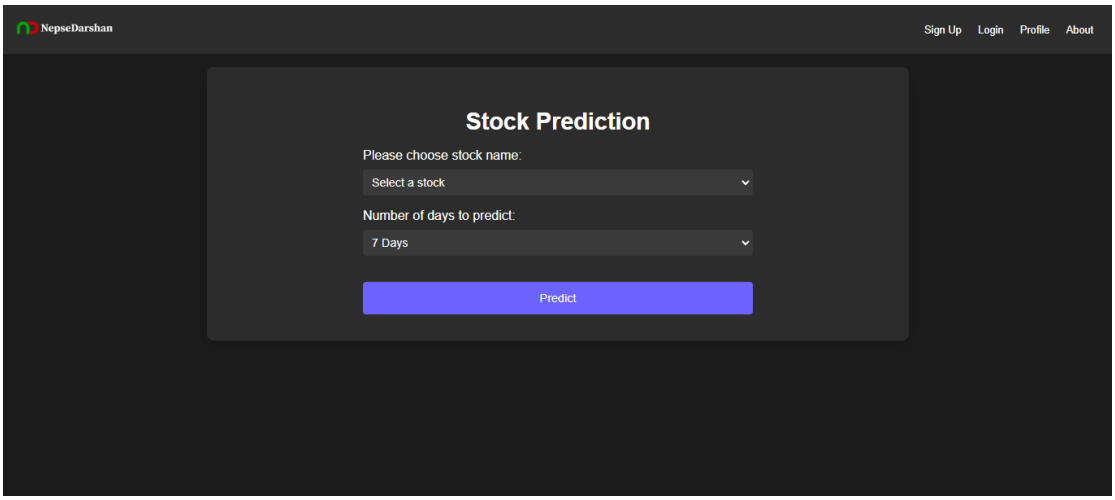
[4] Choi, Hyeong. (2018). Stock Price Correlation Coefficient Prediction with ARIMA-LSTM Hybrid Model, 2018.

[5] Patel, J., Shah, S., Thakkar, P., & Kotecha, K. (2015). Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Systems with Applications, 42(1), 259-268.

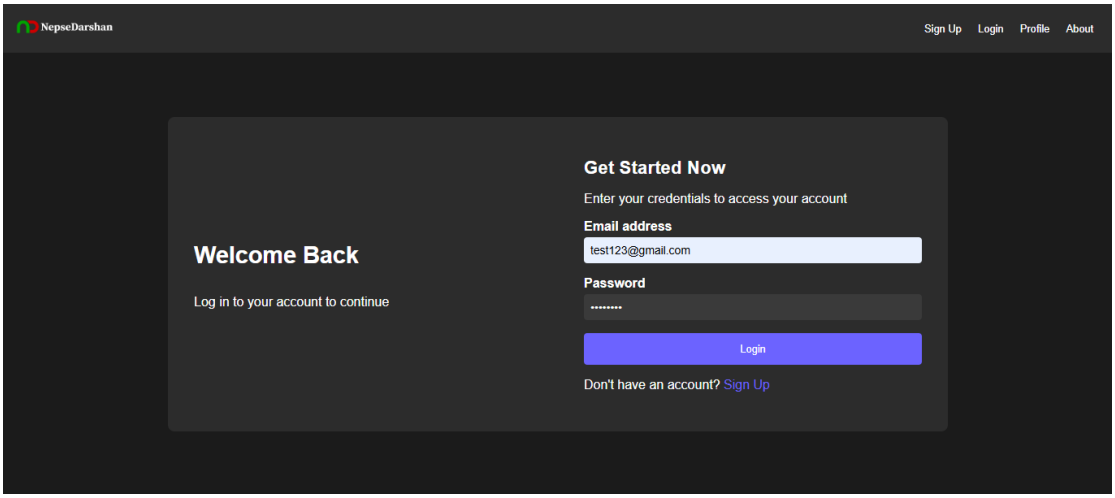
BIBLIOGRAPHY

- https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/lstm
- https://www ijcrt.org/papers/IJCRT2307700.pdf
- https://ieeexplore.ieee.org/abstract/document/10142902
- https://www.youtube.com/watch?v=b61DPVFX03I
- http://conference.ioe.edu.np/publications/ioegc11/ioegc-11-004-11005.pdf
- https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning

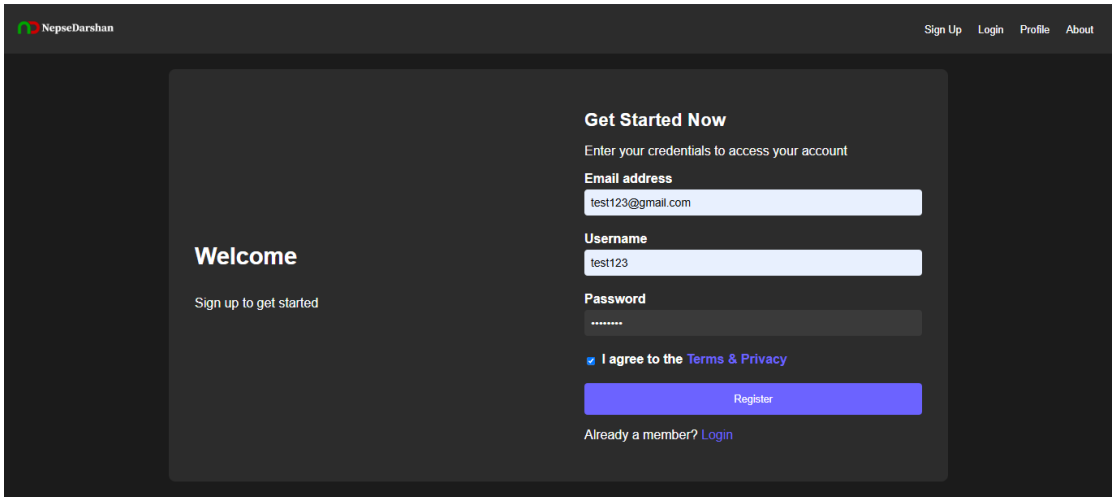
SCREENSHOTS



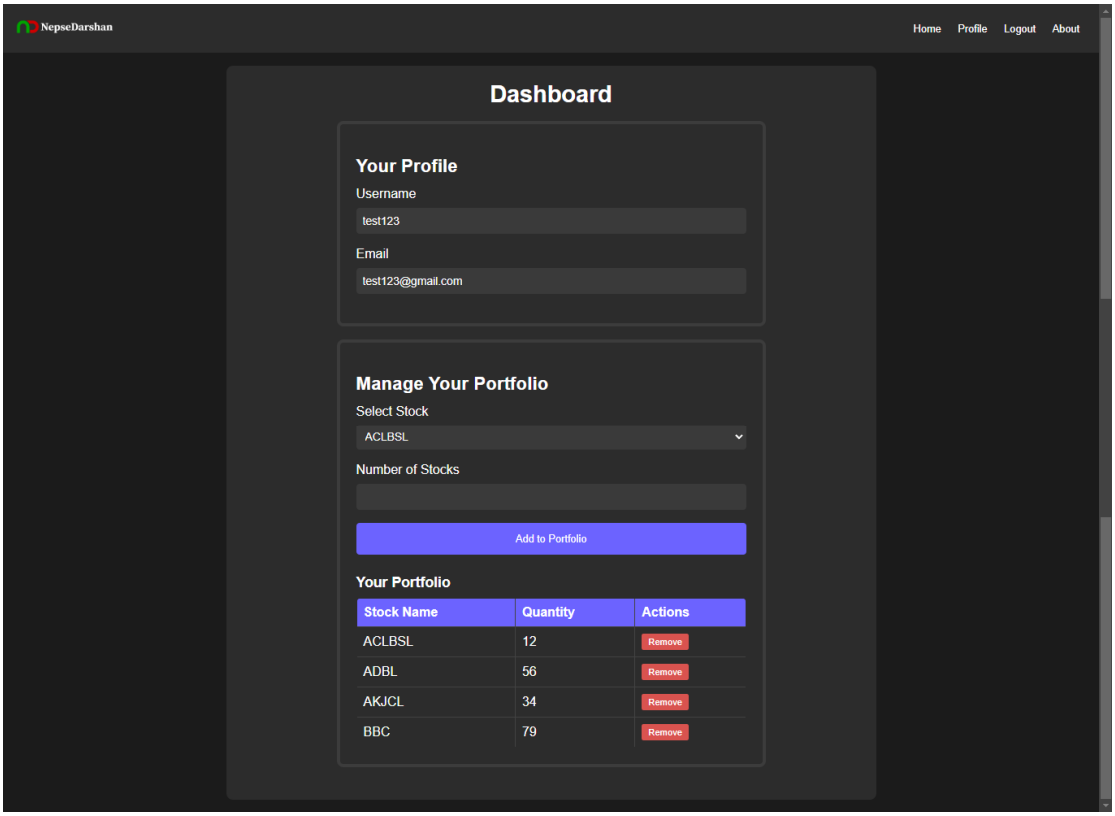
Homepage of the Web application



Login Page



Sign up page



Portfolio page of the user