a

Tribhuvan University

Faculty of Humanities and Social Sciences

NEPSE Stock Prediction System

A PROJECT REPORT

Submitted to

Department of Computer Application

Patan Multiple Campus, Patan Dhoka, Lalitpur

In partial fulfillment of the requirement for the Bachelors in Computer Application

Submitted by

Manish Bhusal: (Reg No: 6-2-22-740-2019)

August, 2022

Under the Supervision of

Mr. Dadhi Ghimire

Tribhuvan University

Faculty of Humanities and Social Sciences

Patan Multiple Campus

Patan Dhoka, Lalitpur

Bachelor in Computer Applications (BCA)

SUPERVISOR'S RECOMMENDATION

I hereby recommend that this project prepared under my supervision by **Manish Bhusal (Reg No 6-2-22-740-2019)** entitled "**NEPSE Stock Prediction (NESPEInsider)"** in the Partial Fulfillment of requirement for the degree of Bachelor in Computer Application is recommended for that final evaluation.

……..

Mr. Dadhi Ghimire.

Supervisor /Lecturer

Department of Bachelor in Computer Application

Patan Dhoka , Lalitpur, Nepal

Tribhuvan University

Faculty of Humanities and Social Science

Patan Multiple Campus, Patandhoka, Lalitpur

LETTER OF APPROVAL

This is to certify that this project prepared by **Manish Bhusal (Reg: 6-2-22-740-2019)** entitled **"NEPSE Stock Prediction (NESPEInsider)"** in partial fulfillment of the requirements for the degree of Bachelor in Computer Application has been evaluated. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

…….. **Mr. Dadhi Ghimire.** Supervisor /Lecturer Department of Computer Application Patan Multiple Campus, Patan Dhoka, Lalitpur	…….. **Mr. Bhoj Raj Joshi** Coordinator/Lecturer Department of Computer Application Patan Multiple Campus, Patan Dhoka , Lalitpur
…….. **Internal Examiner**	…….. **External Examiner**

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

We have taken effort in this project. However, it would not have been possible without the help and support of many individuals and organizations. We would like to extend our sincere thanks to all of them.

We are highly thankful to Patan Multiple Campus for providing us with this opportunity to showcase our learning through this project. We are also appreciative of the effort of our director for providing a learning environment contributing to the success of this project.

We would like to express our deepest sense of gratitude and sincere thanks to our highly respected supervisor **Mr. Dadhi Ghimire** for his valuable guidance, encouragement and help. His useful suggestions for this project and cooperative behavior are sincerely acknowledged.

ABSTRACT

"NepseInsider" is Nepal stock Exchange's stock prediction site for Nepal. It helps trader for predict stock for buy, hold or sell. It is an automated system that analyzes various data points such as market trends, company financials, and news to provide recommendations to investors. The system is designed to help investors make informed decisions by providing them with up-to-date and accurate information. This system has become increasingly popular as more people look to invest in the stock market. It can be used by both novice and experienced investors to help them make better investment decisions. In this day and age where stock markets are constantly fluctuating, having a reliable stock buy, hold, and sell suggestion system can be a valuable asset for any investor looking to maximize their returns. Site is simple and user friendly, user can easy signup and login to system and select the stock then system can predict the stock and show comparison result on screen. Site help to risk minimize by analyzing the graph.

ABBREVIATIONS

CSS : Cascading Style Sheet.

DFD : Data flow diagram.

ER : Entity relational diagram.

HTML : Hyper Text Markup Language.

HTTP : Hypertext transfer protocol.

IEEE : The Institute of Electrical and Electronics Engineers.

JS : Java script.

PHP : Hypertext preprocessor.

SQL : Structure Query Language.

URL : Universal resource locator.

NEPSEInsider : NEPSE stock Prediction system

Chapter1: Introduction

1.1Introduction

"NepseInsider" is Nepal stock Exchange's stock prediction site. It helps trader for predict stock for buy, hold or sell. It is an automated system that analyzes various data points such as market trends, company financials, and news to provide recommendations to investors. The system is designed to help investors make informed decisions by providing them with up-to-date and accurate information. This system has become increasingly popular as more people look to invest in the stock market. It can be used by both novice and experienced investors to help them make better investment decisions. In this day and age where stock markets are constantly fluctuating, having a reliable stock buy, hold, and sell suggestion system can be a valuable asset for any investor looking to maximize their returns.

1.2Problem Statement

Investing in the stock market can be a daunting task, especially for those who are not familiar with the intricacies of the financial world. Selecting the right stocks to buy and sell can be a challenging task, even for experienced investors. Some of the common problems that investors may face while selecting stocks to buy and sell include:

1.Lack of information: It can be difficult to gather all the necessary information about a company, its financials, and market trends. This lack of information can make it challenging to make informed decisions about investing in a particular stock.

2.Emotions and biases: Emotions and biases can influence investment decisions, leading investors to make choices based on factors such as personal preferences, news headlines, or social media trends rather than objective analysis.

3.Market volatility: The stock market is prone to fluctuations, and changes in market conditions can impact the performance of a particular stock. It can be challenging to predict these fluctuations, making it difficult to time investment decisions.

4.Lack of diversification: Investing in a single stock or a small number of stocks can increase risk, as the performance of the portfolio is tied to the performance of a few individual stocks.

1.3Objectives

The main objective of providing investors with reliable and accurate information to help them make informed decision about buying, holding or selling stocks.

- to providing investment advice, a stock buy, hold, and sell site may also offer educational resources to help investors better understand the stock market and make more informed decisions.

1.4Scope and Limitation

Scope

- Stock Prediction: The primary scope is to provide stock predictions to traders and investors. The system uses automated algorithms to analyze various data points, including market trends, company financials, and news, to generate buy, hold, or sell recommendations for specific stocks listed on the NEPSE.
- Informed Decision Making: The platform aims to assist investors in making informed decisions regarding their stock investments. By offering up-to-date and accurate information, it empowers users to navigate the stock market more effectively and potentially improve their investment outcomes
- User-Friendly: "NepseInsider" is designed to be user-friendly and accessible to both novice and experienced investors. The platform provides easy-to-understand recommendations, allowing users to utilize the system's insights regardless of their level of expertise in the stock market.

Limitation

- Market Volatility: While "NepseInsider" utilizes advanced algorithms to predict stock performance, it is essential to acknowledge that the stock market is inherently volatile and unpredictable. The system's accuracy may be affected during periods of extreme market fluctuations or unprecedented events.
- Past Performance vs. Future Outcomes: It relies on historical data and patterns to generate predictions. However, past performance does not guarantee future outcomes, and the stock market is influenced by various external factors that might not be fully captured in historical data analysis.
- Investment Risk: Even with the recommendations provided by "NepseInsider", investing in the stock market always involves an inherent level of risk. Users should exercise caution and perform their due diligence before making investment decisions.

- Data Reliability: The accuracy and reliability of "NepseInsider" predictions depend on the quality and authenticity of the data sources used. Inaccurate or outdated data could lead to less reliable recommendations.

  Not a Substitute for Professional Advice: While it aims to assist investors, it is not a substitute for personalized financial advice from qualified professionals. Users should consider consulting financial advisors or experts for comprehensive financial planning and investment strategies tailored to their individual needs and goals.

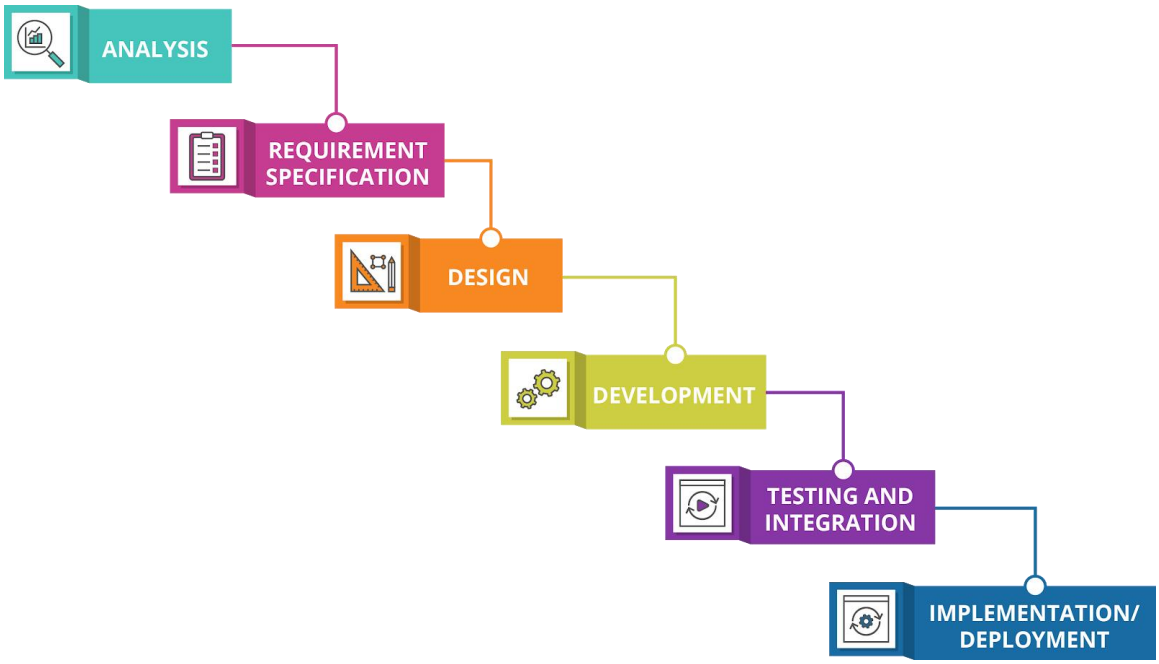
- Technical Issues: Like any automated system, "NepseInsider" may face technical glitches or downtime, which could temporarily disrupt access to its services.

- Limited Scope of Analysis: The system's predictions are based on the data points it analyzes, and there may be other relevant factors that could influence stock performance that are not accounted for in its analysis.

Overall, "NepseInsider" can be a valuable tool for investors seeking insights and recommendations for their stock investments. However, users should recognize its limitations and exercise their judgment in conjunction with other resources to make well-informed investment decisions.

1.5Development Methodology

"NepseInsider" is using waterfall methodology for developing system. it is a traditional and linear approach to software development, where each phase of the project must be completed before moving on to the next one.



. **Figure 1.5: NepseInsider developing methodology**

1.6Report Organization

The report is prepared following the guidelines provided by the Faculty of Humanities and Social Science, Tribhuvan University. The report is separated into different chapter Each chapter consists of various sub chapters with its content. The preliminary section of the report consists of Title Page, Acknowledgement, Abstract, Table of Contents, List of abbreviations, List of Figures and List of Tables. The main report is divided into 5 chapter, which include:

Chapter 1: Introduction

It includes the general overview of the system and the project as a whole. It includes the Introduction Problem Statement, Objective, Scope and Limitations of the project etc.

Chapter 2: Background Study and Literature Review

It includes the study of the current scenario the system will be deployed into. It includes the Review of the similar projects, theories done by other researchers.

Chapter 3: System Analysis and Design

It includes the requirement and feasibility analysis of the system that can be generated through the studies of previous two chapters. It also includes Data Modelling (ER-Diagram), Process Modelling Diagram (DFD) and Architectural Design, Database Schema Design, Interface Design. in design phase.

Chapter 4 Implementation and Testing

It includes the details of the different design and development tools used. The implementation details of the modules presented in the form of code snippets of functions, classes, it also includes the testing of the system with different test cases as per the requirement.

Chapter 5: Conclusion and Future Recommendations

It includes the summary of the system and the project as a whole It also includes the possibilities system can implement in the future.

The final part of the report consists of References and Appendices. The references are listed in accordance to the IEEE referencing standards and the Appendices includes the screenshots of the system.

\

Chapter 2: Background Study and Literature Review

2.1 Background Study

Investors faced challenges in interpreting the vast amount of financial data, tracking market trends, and accurately predicting stock movements. Many investors were also seeking a user-friendly platform that would cater to both experienced traders and those new to the stock market. The study highlighted the need for an automated system that could analyze diverse data sources and generate reliable stock predictions to empower investors in making well-informed decisions.

To address this gap, the concept of "NepseInsider" was conceived. A comprehensive background study was conducted to gain insights into the Nepalese stock market landscape, the challenges faced by investors, and the opportunities to provide valuable solutions. This study involved analyzing the performance of various stocks, historical market trends, trading volumes, and factors affecting stock price movements

2.2 Literature Review

"NepseInsider" involved a comprehensive examination of published academic research and professional literature related to stock market prediction, data analytics, and machine learning in finance. This review aimed to identify relevant methodologies, algorithms, and best practices used in similar stock prediction systems worldwide.

Numerous studies focused on the application of machine learning and data analytics techniques in financial markets, particularly in predicting stock prices and trends. These studies highlighted the significance of historical data analysis, sentiment analysis of financial news, and the incorporation of technical indicators as valuable predictors.

- Studies on “Predicting NEPSE index price using deep learning models”:
  This study addresses the challenges of predicting stock market performance, which is influenced by various interconnected factors such as global economic data, unemployment changes, monetary policy, immigration policy, natural disasters, and public health conditions. The authors propose a research framework that incorporates sentiment analysis of news from social media platforms to improve stock price predictions. They compare the performance of deep learning models, namely LSTM, GRU, and CNN, using a balanced set of input features, including fundamental data, macroeconomic data, technical indicators, and financial news sentiment scores. The study is conducted using data from the Nepal Stock Exchange and aims to build a reliable model for accurate stock price predictions. [1]
  The study compares the performance of three deep learning models, LSTM, GRU, and CNN, for time series prediction in the stock market. The results show that the LSTM model with 30 neurons outperforms the other models, exhibiting the smallest Root Mean Square Error (RMSE) of 10.4660, the lowest Mean Absolute Percentage Error (MAPE) of 0.6488, and the highest R score of 0.9874. The LSTM model also demonstrates lower variability in its

performance metrics. Actual vs. predicted plots indicate that the LSTM model provides a superior fit to the data compared to GRU and CNN models, capturing fluctuations more accurately. Overall, the study concludes that the LSTM model with 30 neurons is the best for stock price prediction in this scenario.

- studies on "Analysis of look back period for stock price prediction with RNN variants: A case study on banking sector of NEPSE":
  This research compares three recurrent neural network models, VRNN, LSTM, and GRU, for stock price prediction. The stock price is influenced by supply and demand, making prediction challenging due to the difficulty in determining the exact factors driving increased demand and supply. Technical and fundamental analysis are the main approaches used by financial analysts. AI techniques have been developed to automate stock price prediction, utilizing deep learning models like LSTM and GRU for their ability to handle time series data without extensive feature engineering. The study analyzes the prediction performance and optimal look-back periods for NIB and NABIL banking stocks, aiming to improve automated forecasting in the stock market.

  the research demonstrates that GRU outperforms LSTM in predicting stock prices due to its moderate data requirements and fewer parameters. It also highlights the significance of selecting appropriate look-back periods for LSTM and GRU models. Furthermore, the study identifies several possibilities for extending the research, including incorporating additional indicators and principles to enhance prediction accuracy and explore long-term predictions. [2]

- User Experience and Feedback:
  I investigate user feedback and user experience studies related to existing popular NEPSE stock prediction system on "MeroLagani", "NepseAlpha", "ShareSanshar", "NepalStock" to understand user expectations and preferences. Most of user need simple user Interface with accurate information. Some user like to analyze data using table or some like data in graph. I found that most user like "NepseAlpha" interface which is easy to watch chart and stock prediction meter. Some user like "MeroLagani" because of its Nepse News and simple table format. All system is target for different age group user base on their requirements. [3]

By understanding the specific needs of Nepalese investors and drawing upon best practices and experiences from existing platforms, "NepseInsider" can be designed to be a relevant, reliable, and user-friendly tool for stock prediction in the Nepalese stock market

Chapter 3: System Analysis and Design

3.1 System Analysis

The process flow for this system includes analysis of the requirements, design, development, testing, and maintenance. All of the functional and non-functional needs are examined throughout the requirement analysis process, and the system is then designed according the requirements. In testing phase; if the testing is successful, the system is installed otherwise, some maintenance is required before the system can be used. As a software development model, the software employs the Structured Approach (Waterfall Model).

I. Functional requirements

A functional requirement defines a system or component's function, where a function is defined as a specification of behavior between inputs and outputs.

Admin:

- Admin can sign up for the account.
- Admin will manage NEPSE API.
- Admin will manage user data.

User:

- User can login and signup to the system.
- Only valid user is permitted to access the system.
- User can search required Stock.
- User gets stock Prediction.

Use Case Diagram:

This system allows two user privileges' i.e., user and admin. In the above point, we discussed the various activities that users and admin can perform in our system. The Use Case diagram below illustrates this in a graphical format.

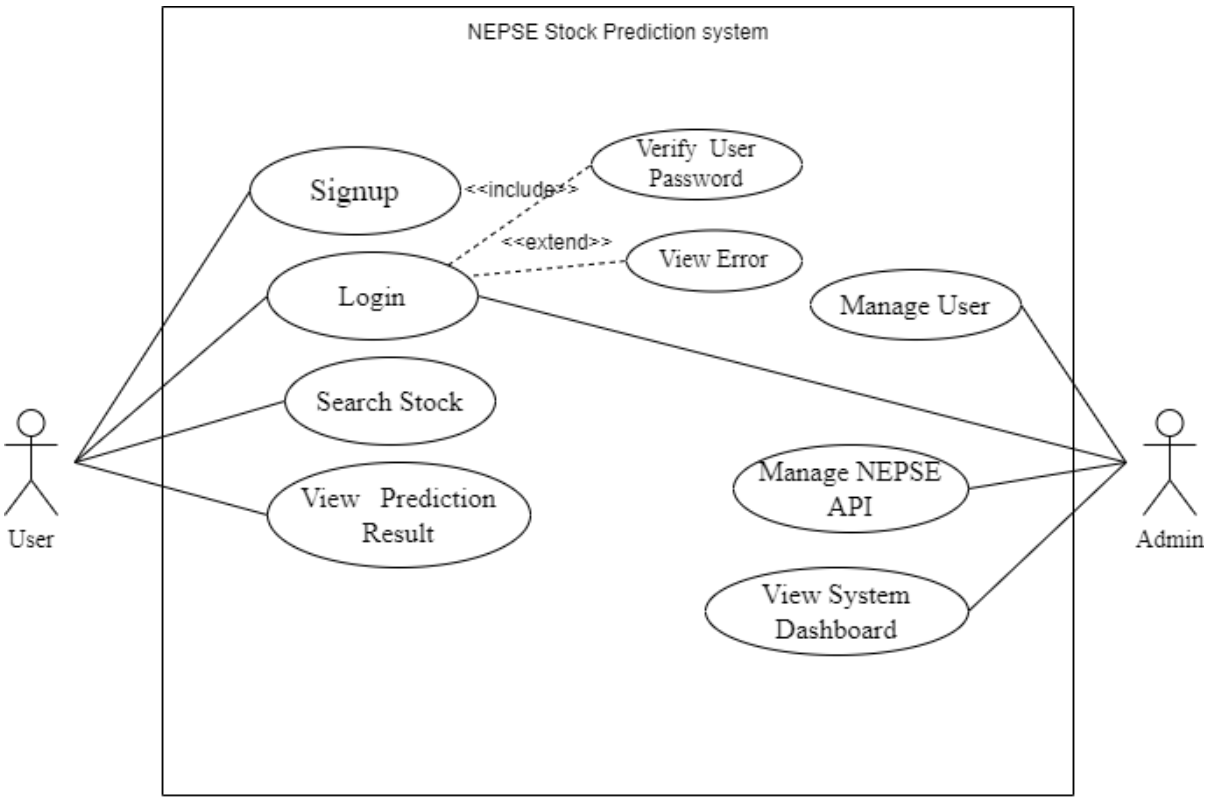


Figure 3.1.1: Use Case Diagram of NEPSEInsider

II. Non-functional requirements

Non-functional requirements are evaluated using following requirements:

Security Requirement:

Only admin is allowed to maintain the Nepse API and User. This system is extremely secure because it can only be operated by authorized admin.

Reliability Requirement:

This system will recommend the stock prediction base on the algorithm result.

Accuracy:

This system uses different machine learning algorithm which shows accurate data and can recommend accurately.

Availability Requirement:

As this system is online. So, it is available 24 hours a day and 7days a week. User can have used system any time they need.

Usability Requirement:

As an interface, the system makes use of web browser. Because all the user is familiar with the web browsers, no special training is required.

3.1.2. Feasibility Analysis

A feasibility study evaluates the system potential to success. Working to the project, following feasibility study is performed.

I. Technical Feasibility:

This system will be technically feasible as it meets the current technology standards. It will be compatible with all web browsers and devices and can run in all types of operating system.

II. Operational Feasibility:

This system will meet all the requirement of the recommendation system. No any skilled manpower is needed to use this system because of its simplicity and user-friendliness. It has simple and attractive UI so user will have great experience using it.

III. Economic Feasibility:

This system will be economic feasible because it uses open source and free resources. No any hardware or software should be deployed in its use.

3.1.3 Data modelling: ER Diagram

An entity relationship diagram depicts the relationships between entities in a database, such as people, things, or concepts. An ER will also depict the characteristics of these entities.[4]

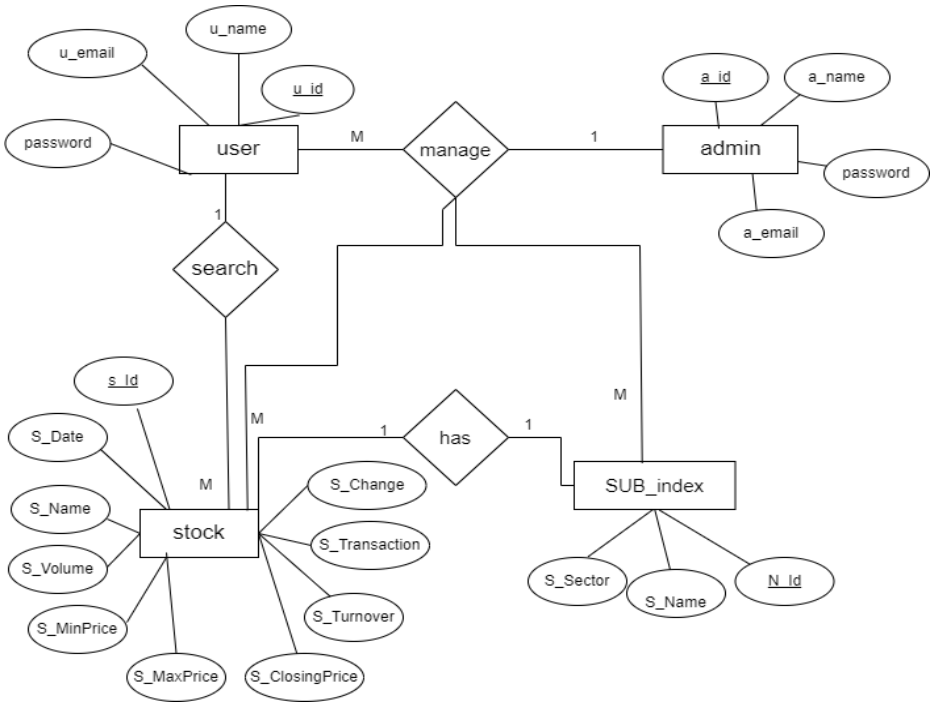


Figure 3.1.3 : ER diagram of NEPSEInsider

3.1.4 Process Modelling: DFD

A data-flow diagram depicts the flow of data through a process or system. The DFD also contains information about each entity's and the process's outputs and inputs. This system is demonstrated using 0 and 1 level DFD.

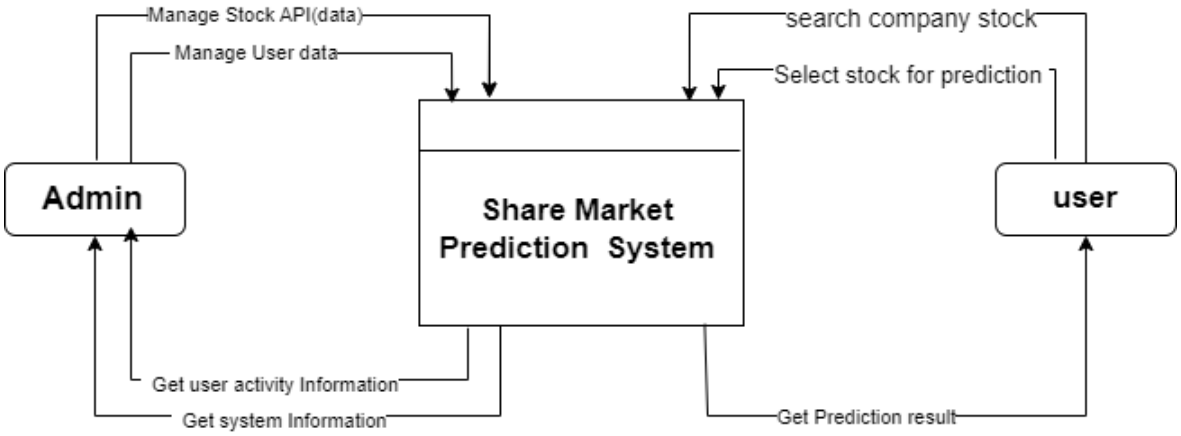


Figure 3.1.4.1: 0 level DFD of NEPSEInsider

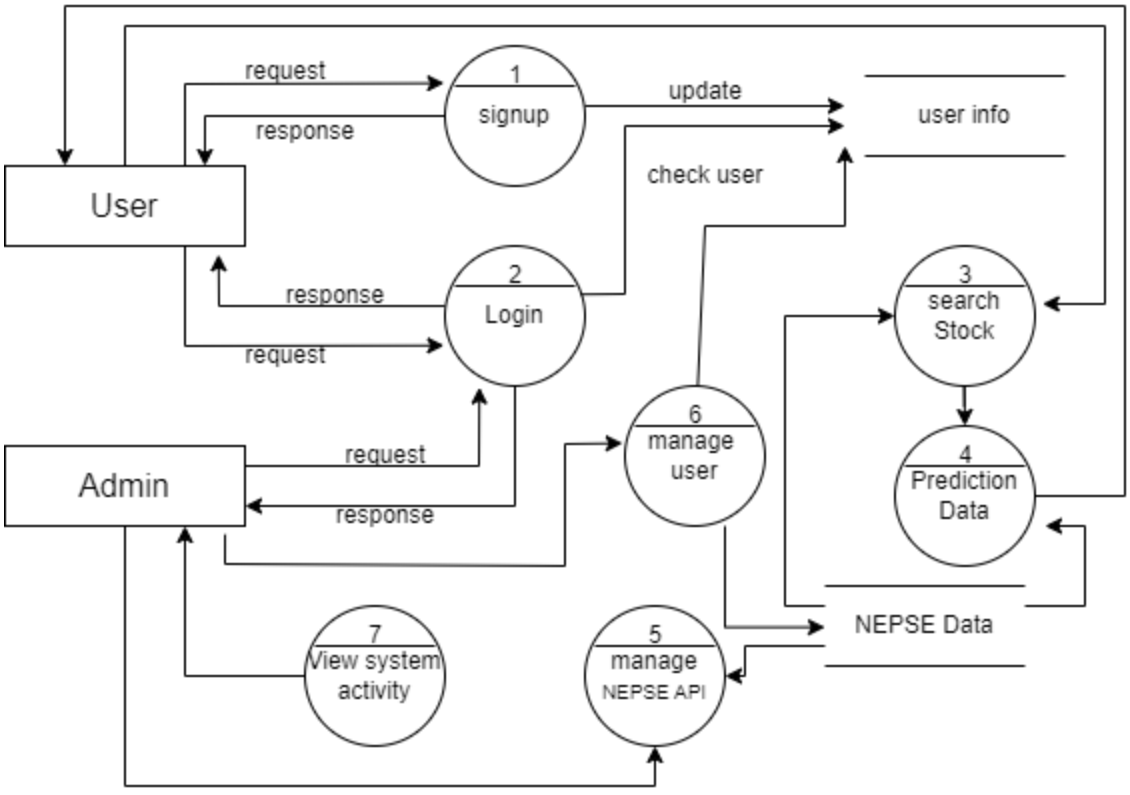


Figure 3.1.4.2: 1 level DFD of NEPSEInsider

3.2: System Design:

Different design diagram of NEPSEInsider have been created in order to graphically represent the functional requirement of the system.

3.2.1. Architectural Design:

Job recommendation system is designed under three tier architecture.

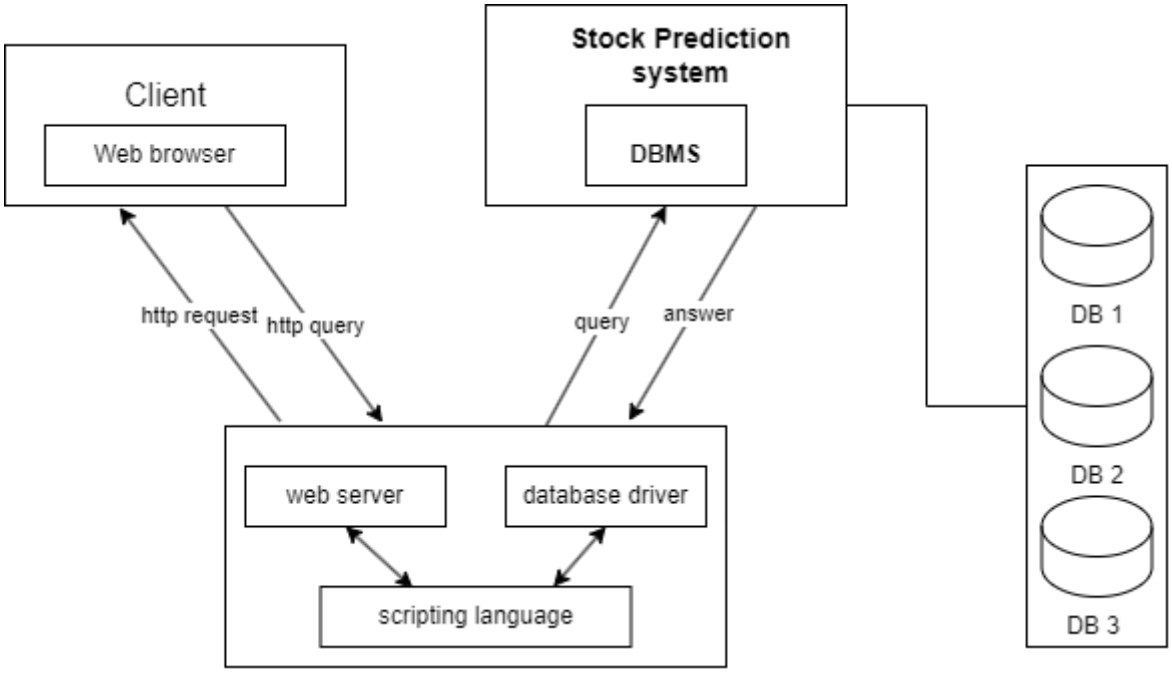


Figure 3.2.2: 3 –tire Architectural Design of NEPSEInsider

3.2.2. Database Schema Design

The database schema of Job NEPSEInsider is the skeleton structure that represents the logical view of the complete database. It specifies how data is arranged and how relationships between them are defined. It defines all of the constraints that will be applied to the data.

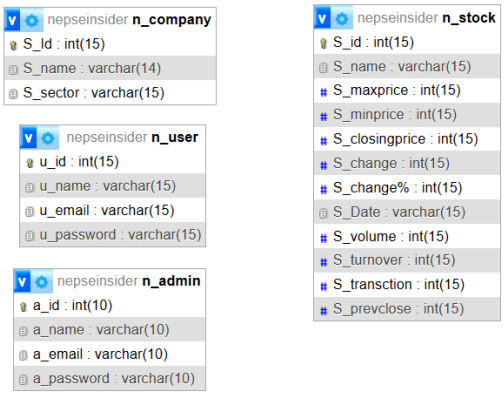
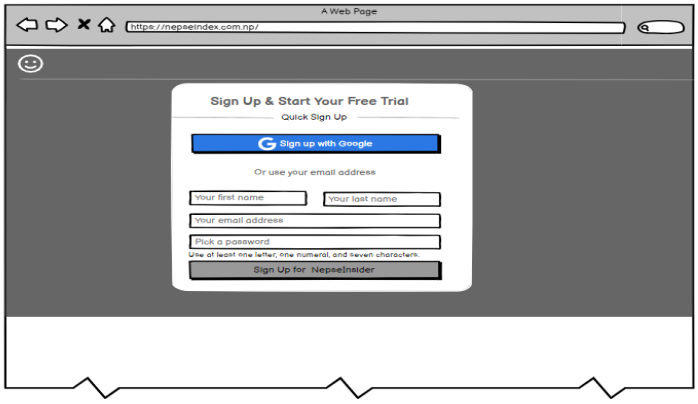
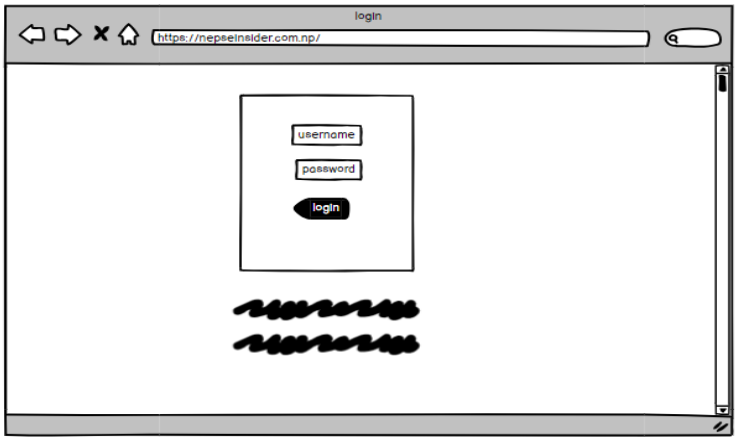


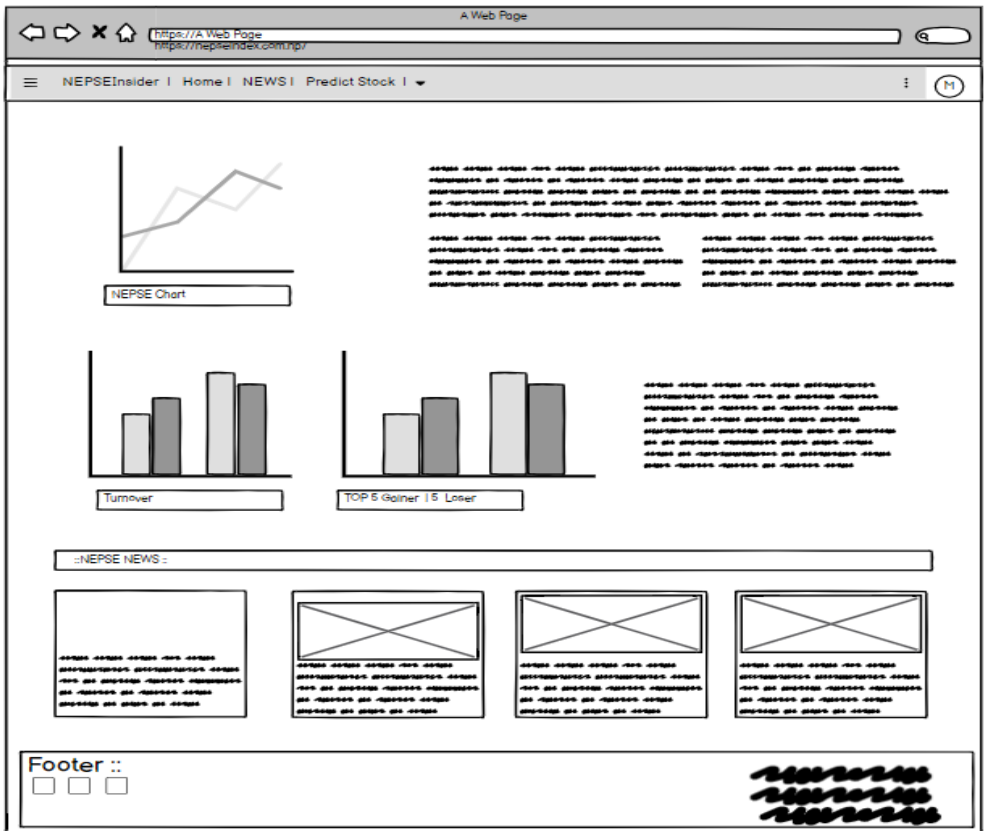
Figure 3.2.2: Database Schema of NEPSEInsider

3.2.3. Interface Design

Different tool is used for "NepseInsider" In this balsamiq is used to design the system wireframe/layout







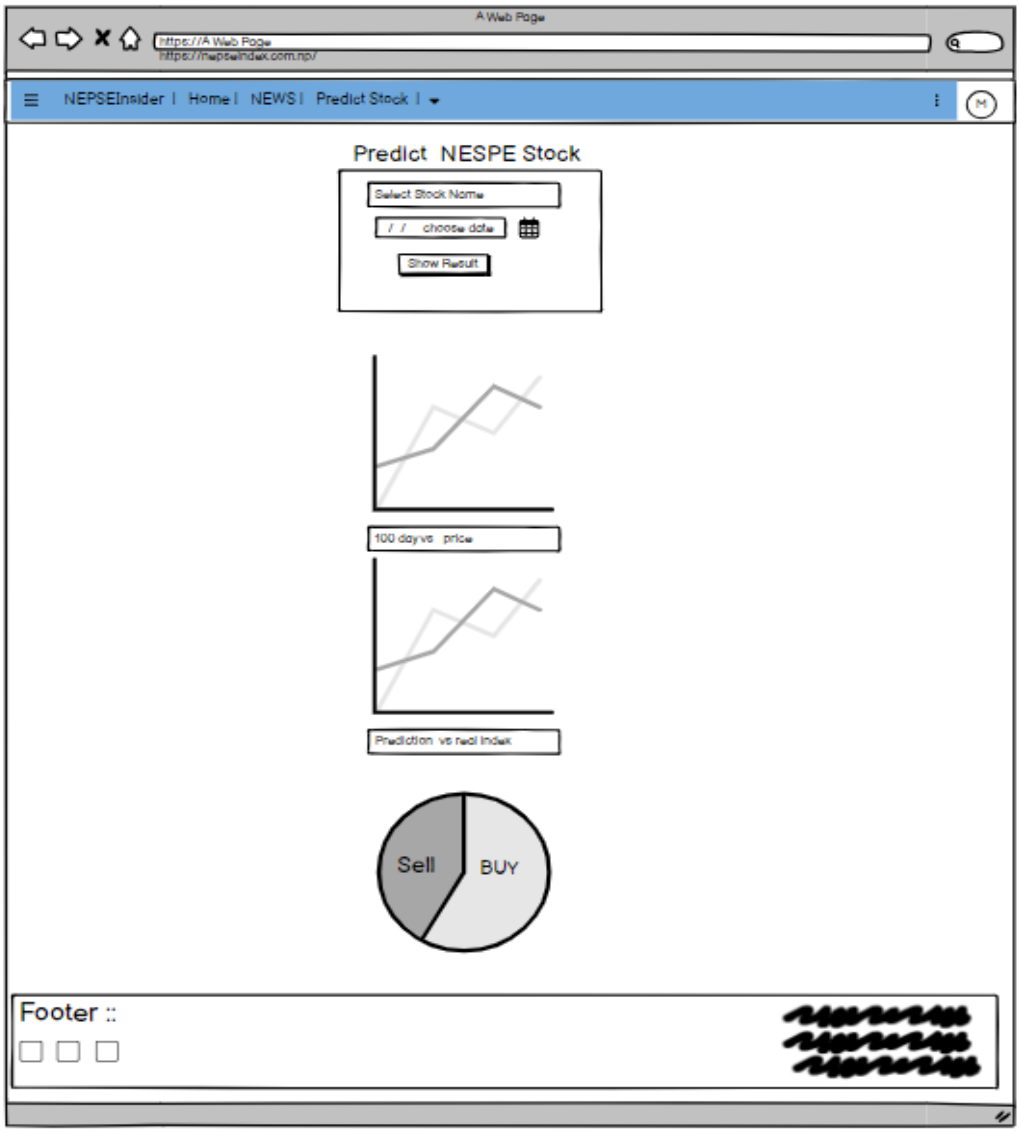


Figure 3.3.3: WireFrame Diagram of NEPSEInsider

3.3 . Algorithm

In NEPSEInsider, LSTM Algorithm is used for predict the stock data.

Long Short-Term Memory (LSTM) is a type of recurrent neural network (RNN) architecture designed to capture long-term dependencies in data. Unlike traditional feedforward neural networks, LSTM networks have feedback connections. It can process not only single data points but also entire sequences of data, making them well-suited for tasks involving sequences and time series prediction.[5]

Here are the key components of an LSTM network:

1. Cell State:
LSTMs have a cell state, which serves as the memory of the network. It runs straight down the entire chain of the LSTM, with only minor linear interactions. Information can be added or removed to the cell state via structures called gates.

2. Gates:
LSTMs have three types of gates to control the flow of information:

Forget Gate: It decides what information in the cell state should be thrown away or kept.
Input Gate: It updates the cell state with new information.
Output Gate: Based on the cell state and the input, it decides what the next hidden state should be.
These gates are neural networks themselves, allowing them to learn which information is relevant to keep or discard.

3. Hidden State:
The hidden state, also known as the output state, is a filtered version of the cell state. It's the LSTM's way of outputting only the relevant information.

4. Training:
During training, LSTMs use backpropagation through time (BPTT), an extension of backpropagation, to update their weights. BPTT works by unrolling the network through time and applying regular backpropagation.

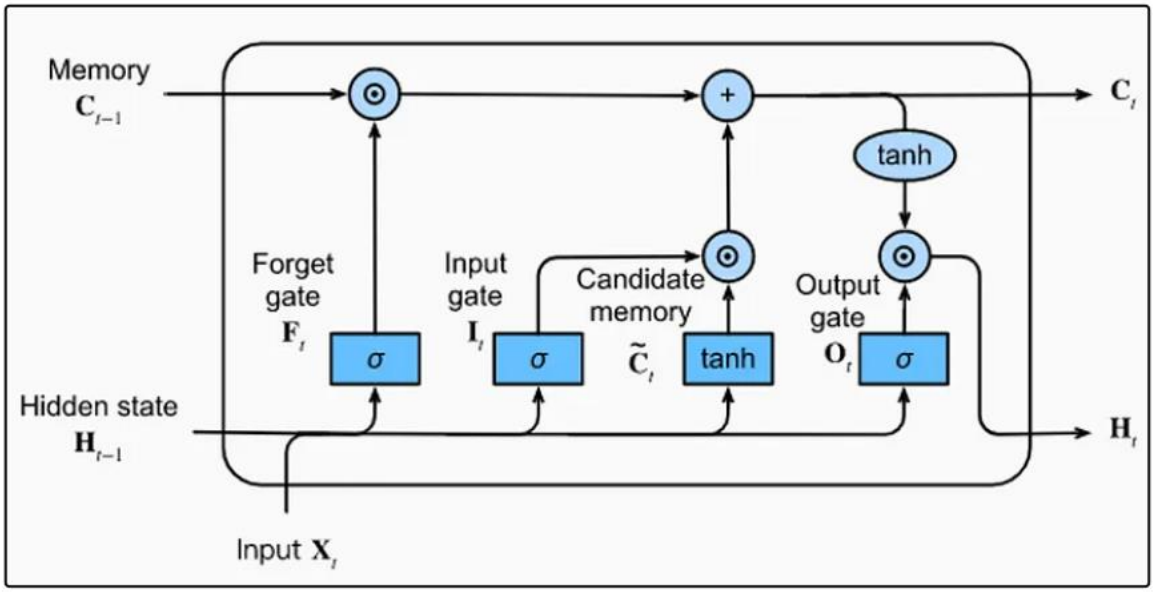


Figure 3.3: LSTM Algorithm of NEPSEInsider

CHAPTER 4: IMPLEMENTATION AND TESTING

4.1: Implementation

Implementation is the phase in which the system is actually built. We studied and analyzed all of the information we gathered before putting a system in place for users. Coding, testing, installation, and documentation are all part of the implementation process. Several tools are used in the development of this system, which is discussed further below.

4.1.1: Tools used:

Different front-end and backend tools are used in development of the system.

Visual Studio Code:

In the project, Visual Studio Code was used as the main code editor to write the codes for the platform. We chose Visual Studio Code as the primary code editor for the project as it is an open-source software with a user-friendly interface and support for multiple programing languages and added functionality like debugging, syntax, highlighting, intelligent code completion, snippets, code refactoring embedded GIT etc.

Balsamiq:

Balsamiq is the open source designing tools. It is used to create wireframe design of museum ticket booking system.

Front End:

The frontend section is built with HTML, CSS, Bootstrap and JavaScript.

HTML:

HTML is a markup language used to format text documents on the web. It is used to design the layout of a web page.

CSS:

CSS3 is a style sheet language that is used to style the basic layout created by HTML5. It aids in the creation of more appealing and beautiful layouts.

Bootstrap:

Bootstrap is used to style and make responsive websites.

JavaScript:

JavaScript is a client-side programming language that enables us to add dynamic behavior to our webpages. It enables client-side scripts to interact with users and control browsers. It is also used to validate various forms.

Back End:

The backend is implemented using Python and flask.

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically

typed and garbage-collected. It supports multiple programming paradigms, including structured, object-oriented and functional programming.

Flask

Flask is used for developing web applications using python, implemented on Werkzeug and Jinja2. Advantages of using Flask framework are: There is a built-in development server and a fast debugger provided.

Database:

My SQL is used as database tool.

My SQL:

MySQL is a widely used relational database management system (RDBMS). MySQL is free and open-source. MySQL is ideal for both small and large applications.

4.1.2. Implementation Details of Modules

A module is a high-level description of a functional area that consists of a group of processes that describe the module's functionality. The proposed system consists of several modules, including user modules, admin modules, login modules, and booking modules.

User Module:

After entering to the website, user have to login or signup to the system. To create a user account, the user must fill out a form that includes fields such as Username and Password. If the details entered while filling out the form are identical to those already in the database, the form will be submitted and if the details entered by user doesn’t match in database then it throws error. When the form is successfully submitted, all of the information is saved in the database. After that, the user can access the system whenever they want.

Signup Module:

User can signup/register to the system through signup module. The signup module has 5 fields i.e. Name, Email, Phone, Password, Confirm Password. User should fill this all required fields to register to the system. After entering all these data users can now login to the system and use the system.

```
@app.route('/', methods=['GET', 'POST'])
def signup():
    if 'logged_in' in session:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor()
```

```
        cursor.execute("INSERT INTO users (username, password) VALUES (%s,
%s)", (username, password))
        mysql.connection.commit()
        cursor.close()
        flash('You are now registered and can log in', 'success')
        return redirect(url_for('login'))
    return render_template('signup.html')
```

Login Module:

After successfully creating an account, the user can log in to the system via the login module. The login module has two fields: username and password. When the username and password entered by the user match the database username and password, the user is logged in. To access the system, the user must enter the correct username and password.

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if 'logged_in' in session:
        return redirect(url_for('dashboard'))

    if request.method == 'POST':
        username = request.form['username']
        password_candidate = request.form['password']
        cursor = mysql.connection.cursor()
        result = cursor.execute("SELECT * FROM users WHERE username = %s",
[username])
        if result > 0:
            data = cursor.fetchone()
            password = data['password']
            user_status = data['status']

            if password_candidate == password:
                session['logged_in'] = True
                session['username'] = username
                session['status'] = user_status
                flash('You are now logged in', 'success')
                return redirect(url_for('dashboard'))
            else:
                error = 'Invalid login'
                return render_template('login.html', error=error)
            cursor.close()
        else:
            error = 'Username not found'
            return render_template('login.html', error=error)
    return render_template('login.html')
```

Prediction Module:

After login to website, landing page in open. User can select stock from list. Prediction method start prediction on the given stock. It will show prediction Chart and prediction price and actual price chart on screen.

```
@app.route('/dashboard')
def dashboard():
    if 'logged_in' not in session:
        return redirect(url_for('login'))

    # Check if the user is an admin (status '1')
    if session.get('status') == 1:
        # Display admin dashboard
        return render_template('adminDashboard.html')
    else:
        # Display regular user dashboard
        db = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="NepseStock"
        )

        # Load the pre-trained model
        model = load_model("my_Stock.keras")
        scaler = MinMaxScaler(feature_range=(0, 1))

        cursor = db.cursor()
        cursor.execute("SELECT Date, Close FROM Stock")
        data = cursor.fetchall()
        cursor.close()

        # Convert data to a pandas DataFrame
        df = pd.DataFrame(data, columns=["Date", "Close"])
        df["Close"] = df["Close"].astype(float)

        # Perform predictions
        scaled_data = scaler.fit_transform(df["Close"].values.reshape(-1, 1))
        x_test = []
        for i in range(100, len(scaled_data)):
            x_test.append(scaled_data[i-100:i, 0])
        x_test = np.array(x_test)
        x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], 1))
        predictions = model.predict(x_test)
```

```
        predictions = scaler.inverse_transform(predictions)

        # Prepare data for HTML rendering
        dates = df.iloc[100:, 0].values
        actual_prices = df.iloc[100:, 1].values
        predicted_prices = predictions.flatten()

        return render_template('index.html', dates=dates,
actual_prices=actual_prices, predicted_prices=predicted_prices)
```

Admin Module:

Authentication in the admin module is done using the admin's email and password. If the admin enters the correct email and password, he will be able to access his dashboard. The system is managed by the administrator, who has access to a dashboard from which he or she can manage users, view booking details, edit notices, add and delete museums, and so on.

AdminUser Module:

admin can manage user account and admin can convert and user to admin if required. Admin has full control on user account. Admin delete and block user from access the site.

```
@app.route('/adminUser', methods=['GET'])
def get_all_users_route():
    users = get_all_users()
    return render_template('adminUser.html', users=users)
# Route to retrieve a specific user by ID
@app.route('/adminUser/<int:user_id>', methods=['GET'])
def get_user_by_id_route(user_id):
    user = get_user_by_id(user_id)
    return render_template('user_detail.html', user=user
# Route to update user information
@app.route('/adminUser/update/<int:user_id>', methods=['POST'])
def update_user_route(user_id):
    new_username = request.form['new_username']
    new_password = request.form['new_password']
    update_user(user_id, new_username, new_password)
    return "User updated successfully!
# Route to delete a user by ID
@app.route('/adminUser/delete/<int:user_id>', methods=['POST'])
def delete_user_route(user_id):
    delete_user(user_id)
    return "User deleted successfully!"
```

AdminStock Model:

Admin can upload and update Stock data into the database. Using this model admin can manage all stock data.

```
def add_company_name(name):
    cursor = mysql.connection.cursor()
    cursor.execute("INSERT INTO S_company (Name) VALUES (%s)", [name])
    mysql.connection.commit()
    cursor.close()

# Function to upload stock data from CSV file
def upload_stock_data(file):
    df = pd.read_csv(file)

    cursor = mysql.connection.cursor()
    for _, row in df.iterrows():
        cursor.execute(
            "INSERT INTO N_stock (sn, Name, Date, Txn, MaxPrice, MinPrice,
Close, Volume, Turnover, PreClose, Change, ChangePercent) VALUES (%s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)",
            (row['sn'], row['Name'], row['Date'], row['Txn'],
row['MaxPrice'], row['MinPrice'], row['Close'], row['Volume'],
row['Turnover'], row['PreClose'], row['Change'], row['Change%']))
    mysql.connection.commit()
    cursor.close()

# Function to retrieve all stock data
def get_all_stock_data():
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM StockData")
    stock_data = cursor.fetchall()
    cursor.close()
    return stock_data

# Function to update stock data by ID
def update_stock_data(stock_id, new_data):
    cursor = mysql.connection.cursor()
    cursor.execute("UPDATE N_stock SET sn=%s, Name=%s, Date=%s, Txn=%s,
MaxPrice=%s, MinPrice=%s, Close=%s, Volume=%s, Turnover=%s, PreClose=%s,
Change=%s, ChangePercent=%s WHERE id=%s",
                   (new_data['sn'], new_data['Name'], new_data['Date'],
new_data['Txn'], new_data['MaxPrice'], new_data['MinPrice'],
new_data['Close'], new_data['Volume'], new_data['Turnover'],
new_data['PreClose'], new_data['Change'], new_data['ChangePercent'],
stock_id))
```

```
    mysql.connection.commit()
    cursor.close()

# Function to delete stock data by ID
def delete_stock_data(stock_id):
    cursor = mysql.connection.cursor()
    cursor.execute("DELETE FROM N_stock WHERE id=%s", [stock_id])
    mysql.connection.commit()
    cursor.close()
```

```
@app.route('/adminStock/add_company', methods=['POST'])
def add_company_route():
    name = request.form['name']
    add_company_name(name)
    return "Company name added successfully!"

# Route to upload stock data from CSV file
@app.route('/adminStock/upload_data', methods=['POST'])
def upload_stock_data_route():
    if 'file' not in request.files:
        return "No file part"

    file = request.files['file']

    if file.filename == '':
        return "No selected file"

    upload_stock_data(file)
    return "Stock data uploaded successfully!"

# Route to retrieve all stock data
@app.route('/adminStock', methods=['GET'])
def get_all_stock_data_route():
    stock_data = get_all_stock_data()
    # Render a template or return JSON response with stock data
    # ...
    return render_template('adminStock.html', stock_data=stock_data)

# Route to update stock data by ID
@app.route('/adminStock/update/<int:stock_id>', methods=['POST'])
def update_stock_data_route(stock_id):
    new_data = {
        'sn': request.form['sn'],
```

```
        'Name': request.form['Name'],
        'Date': request.form['Date'],
        'Txn': request.form['Txn'],
        'MaxPrice': request.form['MaxPrice'],
        'MinPrice': request.form['MinPrice'],
        'Close': request.form['Close'],
        'Volume': request.form['Volume'],
        'Turnover': request.form['Turnover'],
        'PreClose': request.form['PreClose'],
        'Change': request.form['Change'],
        'ChangePercent': request.form['ChangePercent']
    }
    update_stock_data(stock_id, new_data)
    return "Stock data updated successfully!"

# Route to delete stock data by ID
@app.route('/adminStock/delete/<int:stock_id>', methods=['POST'])
def delete_stock_data_route(stock_id):
    delete_stock_data(stock_id)
    return "Stock data deleted successfully!"
```

4.2: Testing

Testing is done to check the behavior of a complete and fully integrated software product based on the software requirement specification document. There are many types of tests to be carried out on a web application from performance, functionality, database loading time, response time, user's action and many others. We will not carry out all types of tests for the application considering the time scale to present this project. We will focus the test cases on functionality, security and performance. Some of the types of testing we did are:

4.2.1. Test Cases for Unit Testing

Unit testing is a software development process in which the small part of an application, called units are individually and independently examined for proper operation.

Table 4.2: Test Table for admin access.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail

1	Check admin access	1)Enter URL of admin panel 2)Enter username & password	1)http://localhost:5000/ admin/ admin.html 2)username: admin123 Password:1!232#AA	To Redirect to admin dashboard	Redirected to admin dashboard	Pass

Table 4.3: Test Table for Register of user.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
2	Check Register	1)Enter URL of Register 2)Enter Name email Phone Password Confirm password	1) http://localhost:5000/ 2) 2)name: Ram Email: ram@gmail.com password: 1#24324@AF	To Register user.	Redirected to index page	Pass

Table 4.3: Test Table for Login of user.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail
3	Check login	1)Enter URL of login 2)Enter username & password	1) http://localhost:5000/ Login.html 2)username: ram@gmail.com password: 1#24324@AF	To Redirect to user Dashboard	Redirected to user panel	Pass

Table 4.4: Test Table for Stock Prediction.

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Result	Actual Result	Pass/Fail

4	Check Predict stock	1)Enter URL of Prediction page 2)Enter required field	1) http://localhost:5000/ NepseIndex-predction.html 2)select Sock which want to predict.	Predict stock	Predict stock	Pass

4.2.2 Test Cases for System Testing

System testing process involved testing the application after development.

Table 4.5: For adding Company.

Test Case 1	Add Stock
Test Data	Company : NTC Submit
Expected Result	A message should be displayed saying "Success!! company is update".
Test Result	Success ! new Company is Add.

Table 4.5: For adding Stock Data.

Test Case 1	Add Stock Data
Test Data	Select Company : NTC Upload Data : NTC.CVS
Expected Result	A message should be displayed saying "Success!! Stock data is update".
Test Result	Success!! Stock data is update.

Table 4.6: For Editing user details.

Test Case 2	Edit user details.
Test Data	Museum name: Ram Email: ram@gmail.com Status : 1 (admin)
Expected Result	Should be update user status to admin. He has full access of system.
Test Result	Success!! Ram update to admin

Table 4.7: For user delete.

Test Case	Expected Result	Test Result
On click of Delete	Delete the user	Successful

Chapter 5: Conclusion and Future Recommendations

5.1 Lesson learnt/ Outcome

With the completion of this project, it was possible to achieve project goal. After registering to the system user ca predict stock and get good analysis of stock. User can calculate the risk the of stock and maximize profit.

5.2 Conclusion

NepseInsider has been designed to address the challenges faced by beginner investors in the complex world of stock prediction. Traditionally, novice investors had to rely on multiple Nepse sites and online news sources for stock information, leading to time-consuming and often inaccurate self-analysis. This process not only consumed valuable time but also resulted in doubts and uncertainties about their predictions, increasing the risk of financial loss.

To tackle these issues, NepseInsider was developed using the structured waterfall model. The development process began with thorough analysis, followed by meticulous design, development, and rigorous testing before deployment. The goal was to create a user-friendly platform that simplifies the stock prediction process, ensuring accurate results for investors.

The website employs a variety of frontend tools such as HTML, CSS, JavaScript, Bootstrap, and JQuery for seamless user interaction. On the backend, Python is utilized along with MYSQL for efficient database connectivity. Additionally, Balsamiq was employed for designing the wireframes and layout of the system.

NepseInsider operates as a digital platform, storing all records in a centralized and secure manner, minimizing the risk of data loss. The user-friendly interface eliminates the need for specialized knowledge, making it accessible to a wide range of users. By achieving its intended objectives, NepseInsider stands out as an exceptional stock prediction website, offering reliable and convenient solutions to investors.

5.2 Future Recommendations

There are many things that can be added in future to improve this website. There is more to be done, thus this application can be seen more useful. The system can be updated based on the user's requirements/recommendation for this system. Here are some changes that can be made in near future:

1. Adding review and feedback option so that users can give their genuine review.

2. Adding stock Newsletter where user can get more update about share market.

REFERENCES

[1] https://www.sciencedirect.com/science/article/pii/S2666827022000706

[2] https://www.sciencedirect.com/science/article/pii/S1877050920308851

[3] https://journalofbigdata.springeropen.com/articles/10.1186/s40537-020-00333-6

[4] https://drawio-app.com/blog/entity-relationship-diagrams-with-draw-io

[5] https://blog.mlreview.com/understanding-lstm-and-its-diagrams-37e2f46f1714