

FUTURE GAINS - SHARE MARKET PREDICTOR

A PROJECT REPORT

*Submitted in partial fulfillment of the requirements
for the award of the degree of
Bachelor of Computer Applications (BCA)*

SUBMITTED BY

VIGNESH C

(2213141033095)

Under the guidance of

Mrs. D. THENMOZHI MCA.,M.Phil.,SET.,

ASSISTANT PROFESSOR

BACHELOR OF COMPUTER APPLICATIONS



GURU NANAK COLLEGE

(AUTONOMOUS)

Affiliated to University of Madras | Accredited at 'A++' Grade by NAAC

Approved by AICTE | An ISO 9001 2015 Certified Institution

Guru Nanak Salai, Velachery, Chennai – 600 042.

MARCH – 2025

GURU NANAK COLLEGE

(AUTONOMOUS)

Affiliated to University of Madras

Accredited at 'A++' Grade by NAAC | An ISO 9001 2015 Certified Institution

Guru Nanak Salai, Velachery, Chennai – 600 042.

School of Information Technology

Bachelor of Computer Applications (BCA)

BONAFIDE CERTIFICATE

This is to certify that, this is a bonafide record of work done by **VIGNESH C,**
2213141033095 for the Final Year Project during the Academic Year 2024-25.

PROJECT GUIDE

HEAD OF THE PROGRAMME

Submitted for the Project Viva Voce Examination held on _____

**at GURU NANAK COLLEGE (Autonomous), Guru Nanak Salai, Velachery,
Chennai - 600 042.**

Internal Examiner

External Examiner

Date:

Date:

DECLARATION

I **VIGNESH C (Register No.: 2213141033095)** studying III Year BCA at Guru Nanak College (Autonomous), Chennai hereby declare that this the Report of my Project entitled, **FUTURE GAINS - SHARE MARKET PREDICTOR** is the record of the original work carried out by me under the Guidance and Supervision of **Mrs. D. THENMOZHI** towards the partial fulfillment of the requirements of the award of the Degree of **BACHELOR OF COMPUTER APPLICATIONS**. I further declare that this has not been submitted anywhere for the award of Degree/Diploma or any other similar to this before.

PLACE: CHENNAI

VIGNESH C

DATE:

(2213141033095)

ACKNOWLEDGEMENT

I would like to thank the **Principal Dr. T. K. Avvai Kothai and Vice Principal Dr. P. V. Kumaraguru** for providing the necessary resources and facilities for the completion of this project.

My sincere thanks to **Dr. S. Nirmala Devi** for their support.

I extend my deepest thanks to **Ms. R. Caroline Kalaiselvi** whose guidance, support, and encouragement were invaluable throughout this endeavor. Her expertise and insights have been instrumental in shaping this project and enhancing its quality.

I owe my Guide **Mrs. D. Thenmozhi** a debt of gratitude for her invaluable guidance, patience, and encouragement. Her mentorship has been a beacon of light, steering me through the complexities of this project and helping me realize my potential.

I also like to extend my thanks to the **Faculty Members of BCA** for their valuable suggestion during the course of the study of my project.

Last but not least, I thank my **family and friends** for their unwavering encouragement and understanding during this journey.

ABSTRACT

The purpose of the project entitled as “Future Gains : Share Market Predictor” is an web application designed to analyze stock market trends using historical stocks data and predict future stock movements. The system integrates real-time stock data fetching, technical analysis, and predictive modeling to provide users with valuable insights into market fluctuations.

Responsive design for all screen sizes. These enhancements make the app more expressive by providing richer visual feedback, detailed analysis, and a more engaging user experience. The animations and interactive elements create a modern, dynamic feel while the additional technical information provides more depth for users interested in stock analysis.

FutureGains aims to empower investors, traders, and analysts by providing data-driven decision-making tools, enhancing their ability to navigate the stock market efficiently. This project showcases the practical application of data visualization in financial forecasting.

S.NO	TOPIC	PAGE NO.
1	INTRODUCTION 1.1 Introduction 1.2 Problem Identification 1.3 Objective and scope of the project	1 - 5
2	RQUIREMENTS SPECIFICATION 2.1 Hardware requirements 2.2 Software requirements	6 - 7
3	SYSTEM ANALYSIS 3.1 Existing system 3.2 Proposed system 3.3 Software specifications	8 – 11
4	SYSTEM DESIGN 4.1 System design 4.1.1 Introduction to UML 4.1.2 UML diagrams of the project	12 – 17
5	SYSTEM IMPLEMENTATION 5.1 Sample code	18 – 64
6	TESTING 6.1 Testing methods	65 – 67
7	SAMPLE SCREENSHOTS	68 – 72
8	CONCLUSION	73 – 74
9	BIBLIOGRAPHY	75 - 76

INTRODUCTION

1. INTRODUCTION

Forecasting of stock market is a way to predict future prices of stocks. It is a long time attractive topic for researchers and investors from its existence. The Stock prices are dynamic day by day, so it is hard to decide what is the best time to buy and sell stocks.

Machine Learning provides a wide range of algorithms, which has been reported to be quite effective in predicting the future stock prices. In this project, we explored different data mining algorithms to forecast stock market prices for NSE stock market.

Our goal is to compare various algorithms and evaluate models by comparing prediction accuracy. Based on the accuracy calculated using MACD indicator of all the models, we predicted prices of different industries. For forecasting, we used historical data of stock market and applied a few preprocessing methods to make prediction more accurate and relevant.

The goal of the application is to serve retail investors as a third party investment tool that uses machine learning to help them navigate in the fast changing stock market. No prediction is 100% accurate.

Therefore, the upper bound and lower bound of the stock prices will be displayed to illustrate the trading range the investors should be looking this application serves. Stock market prediction analysis are some of the most difficult jobs to complete.

There are numerous causes for this, including market volatility and variety of other dependent and independent variables that influence the value of a certain stock in the market.

1.2 PROBLEM IDENTIFICATION

The problem of the project is that it is set to predict the stock price for the next 1 day. “1DAY” is chosen as the time frame as short term price movements tend to depend more on trend momentum and price pattern, while long term price movements depend on the fundamentals. (e.g. company management capabilities, revenue model, market demand, macroeconomic factors, etc.).

The loss function of the training algorithm is the mean squared error of the predicted stock prices. The training algorithm or optimizer is set to minimize its value, and it serves as the basic performance metric for comparing different models.

Other scores are defined to provide more in-depth in sights on a model predictability performance and finance-domain-based comparisons between models for investors. prediction approaches are mainly tested, predicting the stock prices for the next day directly.

There are limited stocks are currently being updated in prediction that are top 50 of the US stock market .

1.3 OBJECTIVE

- 1) Fetch real-time stock market data using APIs.
- 2) Implement technical indicators for trend analysis.
- 3) Build an interactive web application with data visualization.
- 4) Ensure accuracy and efficiency using evaluation metrics like MAE, RMSE, and R^2 .

SCOPE OF THE PROJECT

1) Stock Prediction :

The primary function of the application is to predict the future stock price of a given company using historical data. It allows users to enter a stock symbol, submit a request, and receive predictions.

The app provides prediction results along with confidence levels, trend indicators, and model performance metrics (such as Mean Absolute Error, Root Mean Square Error, and R^2 Score).

The app analyzes the prediction confidence and strength, providing insights on the expected future trend.

2) Real-Time Data :

The app fetches real-time stock data, ensuring that users have the latest information available for analysis and prediction.

The app also displays the latest news related to the stock, including sentiment analysis (positive, negative, or neutral) to assist users in their decision-making process.

3) Data Export :

users can export the historical stock data to CSV format for further analysis, allowing them to leverage the data outside of the app.

4) User Interaction :

Users can search for stocks by symbol and name, view available stocks in a modal window, and select a stock to predict.

The app allows users to filter news articles by sentiment (positive, negative, or all), providing more focused insights.

5) Backend Integration :

The backend handles the requests for stock predictions (/predict), exporting historical data (/export), and retrieving stock symbols (/stocks), ensuring the frontend is populated dynamically.

REQUIREMENT SPECIFICATION

To be user friendly, all computer software needs certain hardware or other software resource to be present on the computer. These pre-requisites are known as system requirements and are often used as a guideline as opposed to an absolute rule. Most software sets two sets of system requirements : Minimum and Recommended. With increasing demand for higher processing power and resource in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer system than technological advancements.

2.1 HARDWARE REQUIREMENTS

PROCESSOR	:	Dual core i3, and above
RAM	:	8 GB RAM and above
HARD DISK	:	minimum 100 GB and above

2.2 SOFTWARE REQUIREMENTS

OPERATING SYSTEM	:	Windows 10 / 11
FRONTEND	:	HTML, CSS, java script .
WEB SERVER	:	Node.js

SYSTEM ANALYSIS

3. EXISTING SYSTEM

Existing systems do not usually display a confidence bar for stock price predictions. Predictions may be provided without any indication of their accuracy or certainty. Trend indicators like "upward" or "downward" trends are often absent, leaving users unsure about the strength or reliability of the predictions.

In the existing system, users may rely on external sources or manual efforts to find recent news related to stocks. Sentiment analysis is typically rudimentary and often not integrated with stock price predictions, leaving users to rely on their own judgment when interpreting news.

Features like the ability to search for stocks, view real-time data filter by sentiment may not be available, resulting in a less intuitive experience.

3.1 PROPOSED SYSTEM

The system offers advanced data visualizations for stock prices, confidence levels, prediction ranges, and historical performance in an interactive chart format. Users can see both historical trends and future stock predictions, making it easier to understand complex data.

The app pulls recent news articles related to the stock and performs sentiment analysis to gauge market sentiment (positive, negative, or neutral). Users can filter news based on sentiment, which helps in making informed decisions.

The system provides a confidence bar showing the predicted confidence level for each stock price prediction. It also includes trend indicators (upward or downward trends) to help users understand the strength of a stock's price movement.

The system allows users to export historical data (such as stock prices and predictions) in CSV format for offline analysis and record-keeping.

3.2 SOFTWARE SPECIFICATION

Frontend :

The frontend of the Stock Price Predictor application is responsible for user interaction, displaying predictions, news, charts, and historical data. It uses modern web technologies for a responsive, interactive user interface.

Technologies Used

HTML: Structure the web pages, including forms, buttons, and data display elements.

CSS : Styles the layout of the application to ensure a clean and visually appealing UI. Uses flexbox and grid for layout management.

JavaScript : Handles dynamic behavior, form submission, event handling, and data fetching.

Chart.js : Used for rendering interactive charts to visualize stock price trends and predictions.

Datalist : For suggesting stock symbols as users type in the input field.

BACKEND :

The backend is responsible for handling user requests, interacting with external APIs for stock data and predictions, and sending responses back to the frontend.

Technologies used

Node.js: The JavaScript runtime used for backend logic.

Express.js: A web framework for building RESTful APIs and handling HTTP requests (GET, POST).

API Integration: Fetch stock market data from external APIs like Alpha Vantage and Yahoo Finance.

CSV Export Library : Use a library like json2csv to convert stock data and predictions into CSV format for export.

SYSTEM DESIGN

4. SYSTEM DESIGN

4.1.1 INTRODUCTION TO UML DESIGN

UML , short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

4.1.2 UML APPROACH

UML Diagram :

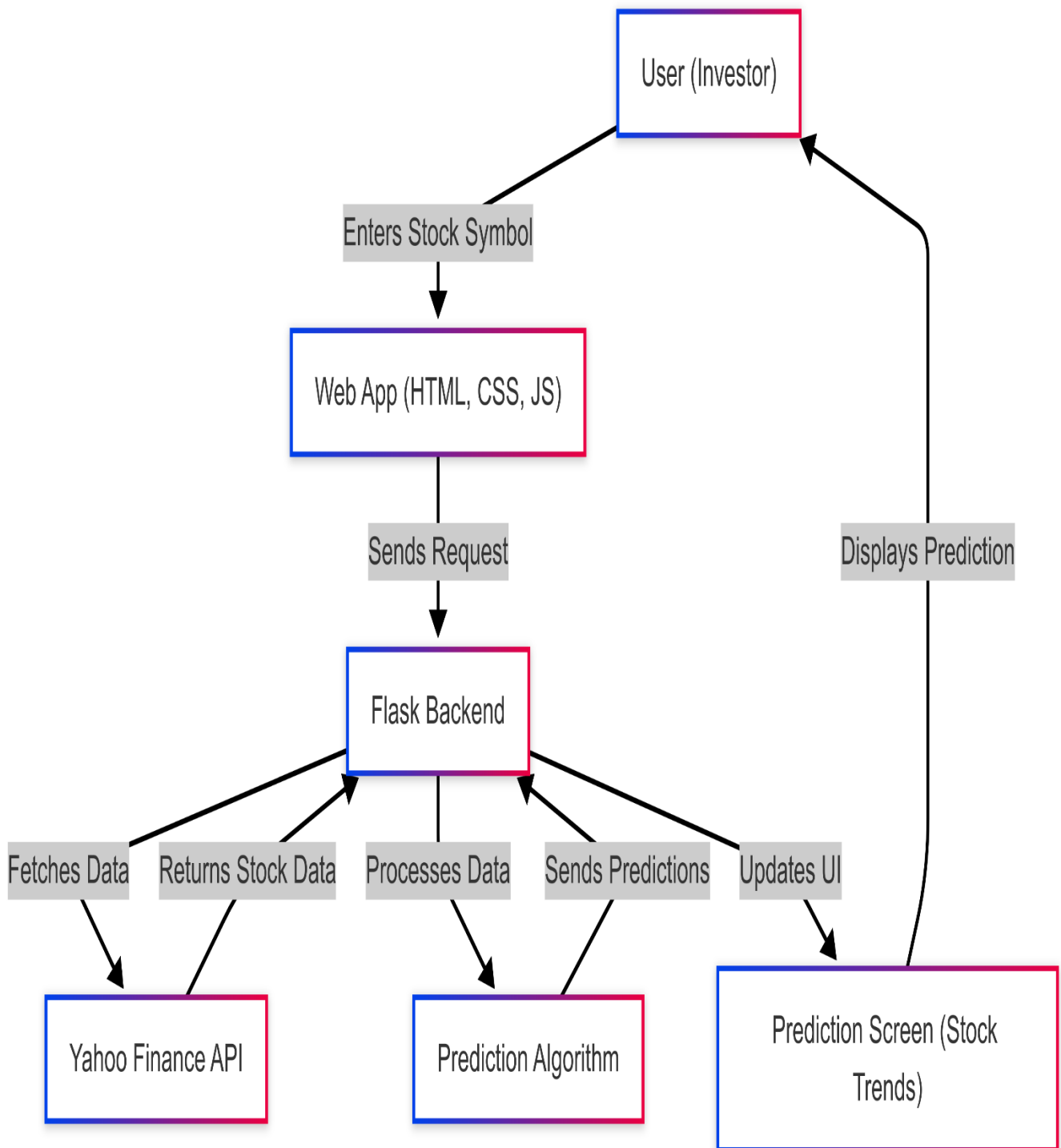
A UML diagram is a way to visualize systems and software using Unified Modeling Language (UML). Programmers create UML diagrams to understand the designs, code architecture, and proposed implementation of complex software systems. UML diagrams are also used to model workflows and business processes. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

The UML includes 9 such diagrams:

- ❖ Class Diagram
- ❖ Activity Diagram

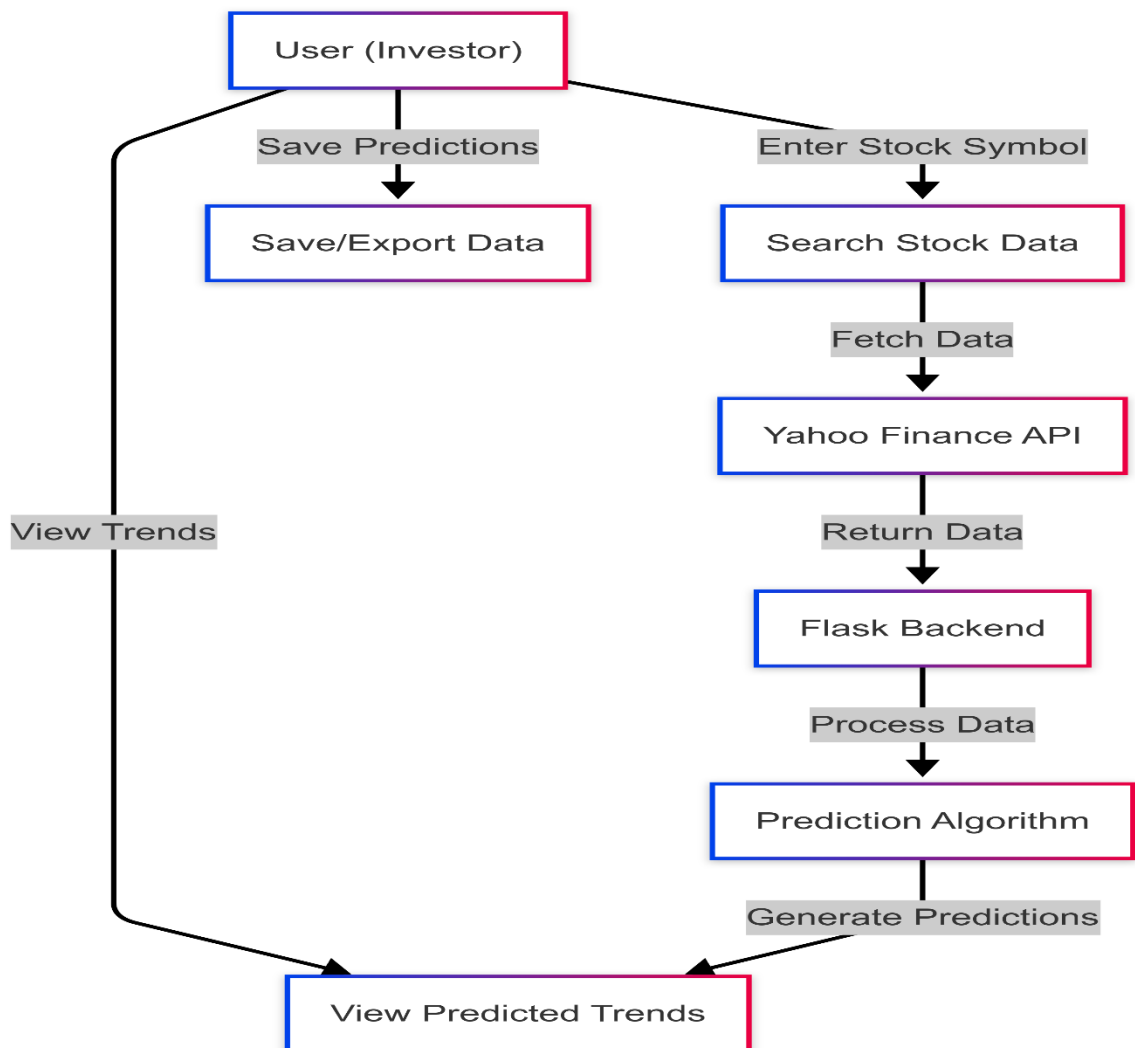
- ❖ Object Diagram
- ❖ Use case Diagram
- ❖ Sequence Diagram
- ❖ State Diagram
- ❖ Component Diagram
- ❖ Composite structure Diagram
- ❖ Interaction overview Diagram

DATA FLOW DIAGRAM :



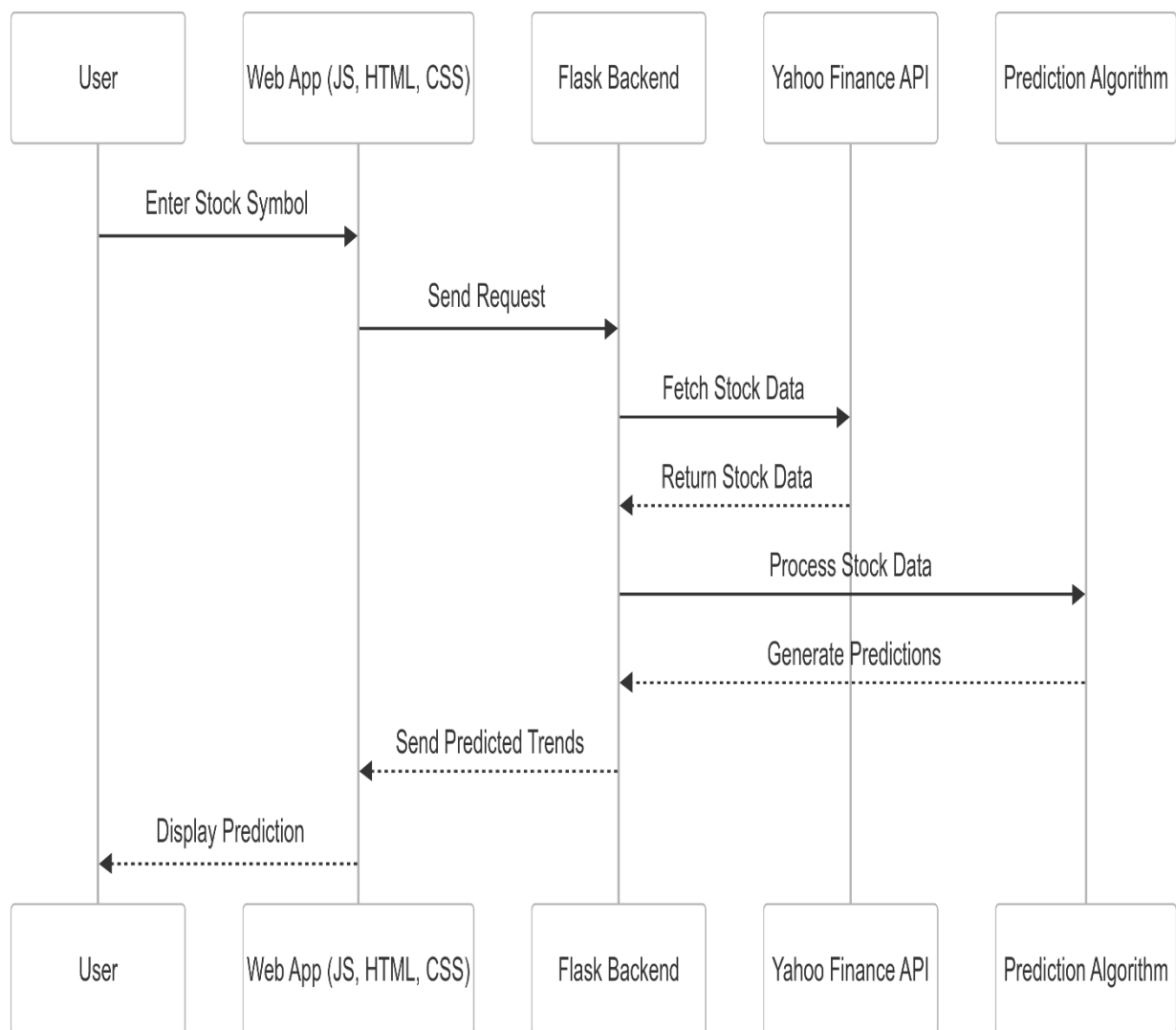
USE CASE DIAGRAM :

use case diagram is a graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well.



SEQUENCE DIAGRAM :

A Sequence Diagram is a key component of UML, used to visualize the interaction between objects in a sequential order. It focuses on how objects communicate with each other over time, making it an essential tool for modeling dynamic behavior in a system



SYSTEM IMPLEMENTATION

5. SYSTEM IMPLEMENTATION :

5.1 SAMPLE CODE :

CSS :

```
* {  
  
    margin: 0;  
  
    padding: 0;  
  
    box-sizing: border-box;  
  
}  
  
body {  
  
    font-family: Arial, sans-serif;  
    line-height: 1.6;  
    background-color: #f4f4f4;  
  
}  
  
.container {  
  
    max-width: 800px;  
    margin: 2rem auto;  
    padding: 2rem;  
  
    background-color: white;  
    border-radius: 10px;  
    box-shadow: 0 0 10px rgba(0,0,0,0.1);  
  
}
```

```
h1 {  
  text-align: center; color: #333;  
  margin-bottom: 2rem;  
}
```

```
.search-container {  
  text-align: center;  
  margin-bottom: 2rem;  
}
```

```
input[type="text"] {  
  padding: 0.8rem;  
  width: 300px;  
  border: 1px solid #ddd;  
  border-radius: 5px;  
  margin-right: 1rem;  
}
```

```
.search-container {  
  text-align: center;  
  margin-bottom: 2rem;  
}
```

```
input[type="text"] {
```

```
padding: 0.8rem;  
width: 300px;  
}  
  
button {  
  
padding: 0.8rem 1.5rem;  
background-color: #007bff;  
color: white;  
border: none;  
border-radius: 5px;  
cursor: pointer;  
transition: background-color 0.3s;  
}
```

```
button:hover {  
  
background-color: #0056b3;  
}
```

```
.price-container {  
display: flex;  
justify-content: space-around;  
margin-top: 2rem;  
}
```

```
.price-box {  
  
text-align: center; padding:
```

```
    1.5rem; background-color:
    #f8f9fa; border-radius: 5px;
    width: 45%;
}

.price-box h3 {
    color: #666;
    margin-bottom: 1rem;
}

.price-box p {
    font-size: 2rem;
    color: #28a745;
    font-weight: bold;
}

.hidden { display:
    none;
}

.loader {

    border: 4px solid #f3f3f3;
    border-top: 4px solid #3498db;
    border-radius: 50%;
    width: 40px;
    height: 40px;
    animation: spin 1s linear infinite;
    margin: 2rem auto;
}

.metric-label {
```

```
    font-size: 0.8rem;
    color: #666;
    font-weight: normal;
@keyframes spin {
    0% { transform: rotate(0deg); } 100%
    { transform: rotate(360deg); }
}
.error-message {
    color: #dc3545;
    text-align: center;
    margin-top: 1rem;
}

#stock-symbol {
    text-align: center;
    color: #444;
    margin-bottom: 1rem;
}

.metrics-container {
    margin: 2rem 0;
    padding: 1rem;
    background-color: #f8f9fa;
    border-radius: 5px;
}

.metrics-grid { display:
    grid;
    grid-template-columns: repeat(3, 1fr);

.metric-label {
```

```

    font-size: 0.8rem;
    color: #666;
    font-weight: normal;
    gap: 1rem; margin-
    top: 1rem;
}

.metric-box {

    text-align: center; padding:
    1rem; background-color:
    white; border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.metric-box h4 {
    color: #666;
    margin-bottom: 0.5rem;
    font-size: 0.9rem;
}

.metric-box p {
    color: #28a745;
    font-size: 1.2rem;
    font-weight: bold;
}

.chart-container {
    margin: 2rem 0;
    padding: 1rem;

```

```
metric-label {  
    background-color: white;  
    border-radius: 5px;  
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
}
```

```
.news-container {  
    margin: 2rem 0;  
}
```

```
.news-item {  
    padding: 1rem;  
    margin-bottom: 1rem;  
    background-color: white;  
    border-radius: 5px;  
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);  
}
```

```
.news-item h4 {  
    color: #333;  
    margin-bottom: 0.5rem;  
}
```

```
.news-item p {  
    color: #666;  
    font-size: 0.9rem
```

```
.news-item .news-date {
```

```
    color: #999;
    font-size: 0.8rem;
}
```

```
.prediction-range {
    font-size: 0.9rem;
    color: #666;
    margin-top: 0.5rem;
}
```

```
.export-container {
    text-align: center;
    margin-top: 2rem;
}
```

```
#export-btn {
    background-color: #28a745;
}
```

```
#export-btn:hover {
    background-color: #218838;
}
```

```
/* Add these styles to style.css */
```

```
.stock-select-container {
```



```
display: flex;

gap: 1rem;

margin-bottom: 1rem;

}
```

```
.modal { display:
  none;
  position: fixed;
  top: 0;
  left: 0;

  width: 100%;

  height: 100%;

  background-color: rgba(0,0,0,0.5); z-
  index: 1000;
}
```

```
.modal.show {
  display: block;
}
```

```
.modal-content {
  background-color: white;
  margin: 5% auto; padding:
  2rem;
  width: 80%;

  overflow-y: auto;

}
```

```
.close {  
  
    float: right;  
  
    font-size: 1.5rem;  
    cursor: pointer;  
    color: #666;  
}
```

```
.close:hover {  
    color: #000;  
}
```

```
#stock-search {  
    width: 100%;  
    margin: 1rem 0;  
    padding: 0.8rem; border:  
    1px solid #ddd; border-  
    radius: 5px;  
}
```

```
.stocks-grid { margin-  
    top: 1rem;  
}  
  
display: grid;  
  
grid-template-columns: repeat(4, 1fr);  
padding: 1rem;  
background-color: #f8f9fa;  
font-weight: bold;  
border-radius: 5px;
```

```
}
```

```
.stock-item {  
  display: grid;  
  grid-template-columns: repeat(4, 1fr);  
  padding: 1rem;  
  border-bottom: 1px solid #eee;  
  cursor: pointer;  
  transition: background-color 0.2s;  
  align-items: center;  
}
```

```
.stock-item:hover {  
  background-color: #f8f9fa;  
}
```

```
.stock-item span {  
  padding: 0 0.5rem;  
  overflow: hidden; text-  
  overflow: ellipsis;  
}
```

```
#show-stocks-btn { background-  
  color: #6c757d;  
}
```

```
#show-stocks-btn:hover {  
  background-color: #5a6268;  
}
```

```
.news-footer {  
  display: flex;  
  justify-content: space-between;  
  align-items: center;  
  margin-top: 0.5rem;  
}
```

```
.news-link {  
  color: #007bff;  
  text-decoration: none;  
  font-size: 0.9rem;  
}
```

```
.news-link:hover {  
.confidence-container {  
  margin: 2rem 0;  
  padding: 1.5rem;  
  background-color: #f8f9fa;  
  border-radius: 5px;  
  text-align: center;  
}
```

```
.confidence-meter {  
  width: 100%;  
  height: 20px;  
  background-color: #e9ecef;  
  border-radius: 10px; margin:  
  1rem 0;  
  position: relative;
```

```
}
```

```
.confidence-bar.low {  
  background-color: #dc3545;  
}
```

```
.momentum {  
  
  font-size: 0.8rem;  
  padding: 0.2rem 0.5rem;  
  border-radius: 3px;  
  margin-left: 0.5rem;  
}
```

```
.momentum.strong {  
  background-color: #28a745;  
  color: white;  
}
```

```
.momentum.moderate {  
  background-color: #ffc107;  
  color: black;
```

```
.momentum.neutral {  
  background-color: #6c757d;  
  color: white;  
}
```

```
.momentum.weak {  
  background-color: #dc3545;  
  color: white;  
}
```

```
.momentum.very-weak {  
  background-color: #dc3545;  
  color: white;  
  font-weight: bold;  
}
```

```
.metric-label {  
  
  font-size: 0.8rem;  
  color: #666;  
  font-weight: normal;  
}
```

JAVASCRIPT :

```
    let priceChart = null;  
  
document.getElementById('prediction-form').addEventListener('submit', async  
(e) => {  
  
  e.preventDefault();  
  
  const symbol = document.getElementById('symbol').value;  
  const loadingDiv = document.getElementById('loading');  
  const resultDiv = document.getElementById('result'); const  
  errorDiv = document.getElementById('error');  
  
  
  // Show loading, hide result and error  
  loadingDiv.classList.remove('hidden');  
  resultDiv.classList.add('hidden');  
  errorDiv.classList.add('hidden');
```

```

try {

    const formData = new FormData();
    formData.append('symbol', symbol);

    const response = await fetch('/predict', {
        method: 'POST',
        body: formData
    });

    const data = await response.json();

    if (data.error) {

        throw new Error(data.error);

    }

    updateResults(data);

    // Update chart
    data.performance.price_history);          updateChart(data.performance.dates,
    // Update news
    updateNews(data.news);

```

```

if (news.length === 0) {

    newsContainer.innerHTML = '<p>No recent news available</p>'; return;
}


news.forEach(item => {

    const sentiment = item.sentiment;

    const sentimentClass = sentiment > 0 ? 'positive' : sentiment < 0 ?
'negative' : '';

    const newsItem = document.createElement('div');
    newsItem.className = `news-item ${sentimentClass}`;

    const newsContent = `

        <h4>${item.title}</h4>

        <p>${item.summary}</p>

        <div class="news-meta">

            <span class="news-source">${item.source}</span>

            <span      class="news-date">${new
Date(item.time_published).toLocaleString()}</span>

        </div>

        <div class="news-footer">

            ${item.url ? `<a href="${item.url}" target="_blank" class="news-

```



```
link">Read More</a>` : "{
```

```
</div>
```

```
`;
```

```
newsItem.innerHTML = newsContent;
```

```
newsContainer.appendChild(newsItem);
```

```
});
```

```
}
```

```
// Add news filter functionality
```

```
document.querySelectorAll('.news-filter').forEach(button => {
```

```
  button.addEventListener('click', (e) => {
```

```
    // Update active button
```

```
    document.querySelectorAll('.news-filter').forEach(btn =>
```

```
    btn.classList.remove('active'));
```

```
    e.target.classList.add('active');
```

```
// Filter news items
```

```
const sentiment = e.target.dataset.sentiment;
```

```
const newsItems = document.querySelectorAll('.news-item');
```

```
newsItems.forEach(item => {
```

```
});ad available stocks when page loads document.addEventListener('DOMContentLoaded', loadStocks);
```

```
// Show stocks modal
```

```
document.getElementById('show-stocks-btn').addEventListener('click', ()  
=> {  
    document.getElementById('stocks-modal').classList.add('show');  
});
```

```
// Close modal
```

```
document.querySelector('.close').addEventListener('click', () => {  
    document.getElementById('stocks-modal').classList.remove('show');  
});
```

```
// Search stocks
```

```
document.getElementById('stock-search').addEventListener('input', (e) =>  
{  
    const searchTerm = e.target.value.toLowerCase();  
  
    const stockItems = document.querySelectorAll('.stock-item');  
  
    stockItems.forEach(item => {  
  
        const text = item.textContent.toLowerCase(); item.style.display =  
        text.includes(searchTerm) ? 'grid' : 'none';  
    });
```

```
});
```

```
// Load stocks function async
```

```
function loadStocks() {
```

```
  try {
```

```
    const response = await fetch('/stocks'); const
```

```
    data = await response.json();
```

```
    if (data.error) {
```

```
      throw new Error(data.error);
```

```
    }
```

```
    // Populate datalist
```

```
    const datalist = document.getElementById('stock-list');
```

```
    datalist.innerHTML = data.stocks.map(stock =>
```

```
      `<option    value="${stock.symbol}">${stock.name}  
    (${stock.symbol})</option>`
```

```
    ).join("");
```

```
    // Populate stocks list
```

```
    const stocksList = document.getElementById('stocks-list');
```

```

stocksList.innerHTML = data.stocks.map(stock => `

  <div class="stock-item" data-symbol="${stock.symbol}">

    <span>${stock.symbol}</span>

    <span>${stock.name}</span>

    <span>${stock.sector}</span>

    <span>${stock.current_price.toFixed(2)}</span>

  </div>

`).join("");

// Add click handlers for stock items
document.querySelectorAll('.stock-item').forEach(item => {
  item.addEventListener('click', () => {
    document.getElementById('symbol').value = item.dataset.symbol;
    document.getElementById('stocks-
modal').classList.remove('show');

    document.getElementById('prediction-form').dispatchEvent(new
Event('submit'));

  });

});

} catch (error) {

  console.error('Error loading stocks:', error);

}

```

```

document.getElementById('current-price').textContent =
`$$${data.current_price}`;

        document.getElementById('predicted-price').textContent =
`$$${data.prediction}`;


// Enhanced prediction range display

        const priceChange = ((data.prediction - data.current_price) /
data.current_price * 100).toFixed(2);

        const rangeSpread = ((data.upper_bound - data.lower_bound) /
data.prediction * 100).toFixed(2);


document.getElementById('prediction-range').innerHTML = `

        <div class="range-details">

                <span>Range:      $$${data.lower_bound}      -
$$${data.upper_bound}</span>

                <span class="spread">Spread: ${rangeSpread}%</span>

        </div>

        <div class="change-prediction ${priceChange >= 0 ? 'positive' :
'negative'}">

                Expected Change: ${priceChange}%

        </div>

`;

```

```

confidenceBar.style.width = `${data.confidence.level}%`;

confidenceBar.className = `confidence-bar ${data.confidence.level > 70
? 'high' : data.confidence.level > 40 ? 'medium' : 'low'}`;

confidenceLevel.textContent = `${data.confidence.level.toFixed(1)}%
Confidence`;

const trend = data.confidence.trend;

const trendIcon = trend.direction === 'up' ? '↑' : '↓';

const trendClass = trend.direction === 'up' ? 'trend-up' : 'trend-down';

trendDirection.className = trendClass;
trendDirection.innerHTML = `
    ${trendIcon} ${Math.abs(trend.percentage).toFixed(2)}%
    <span class="momentum
    ${trend.momentum}">${trend.momentum}</span>
`;

trendStrength.textContent = `${trend.strength > 5 ? 'Strong' : 'Moderate'}
${trend.direction.toUpperCase()}`;

// Update metrics with enhanced formatting

document.getElementById('mae').innerHTML =

```

```
`$$ {data.metrics.mae}<br><span class="metric-label">Mean Absolute Error</span>`;
```

```
document.getElementById('rmse').innerHTML =
`$$ {data.metrics.rmse}<br><span class="metric-label">Root Mean Square Error</span>`;
```

```
document.getElementById('r2').innerHTML =
`$ {data.metrics.r2}%<br><span class="metric-label">R2 Score</span>`
```

INDEX.HTML :

```
<!DOCTYPE html>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-
scale=1.0">

  <title>Stock Price Predictor</title>

  <link rel="stylesheet" href="{ { url_for('static',
filename='css/style.css') } }">

  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</head>

<body>

  <div class="container">

    <h1>Stock Price Predictor</h1>
```

```

<div class="search-container">

    <form id="prediction-form">

        <div class="stock-select-container">

            <input type="text" id="symbol" name="symbol" list="stock- list"
placeholder="Enter stock symbol (e.g., AAPL)" required>

            <datalist id="stock-list">

                <!-- Will be populated dynamically -->

            </datalist>

            <button type="button" id="show-stocks-btn">Show All
Stocks</button>

        </div>

        <button type="submit">Predict</button>

    </form>

</div>

<div id="stocks-modal" class="modal hidden">

    <div class="modal-content">

        <span class="close">&times;</span>

        <h2>Available Stocks</h2>

        <input type="text" id="stock-search" placeholder="Search
stocks...">

        <div class="stocks-grid">

```



```

<h3>Latest News</h3>

<div class="news-filters">
    <button class="news-filter active" data-sentiment="all">All
News</button>

                                <button    class="news-filter"    data-
sentiment="positive">Positive</button>

                                <button    class="news-filter"    data-
sentiment="negative">Negative</button>

    </div>

    <div id="news-list"></div>

</div>

<div class="export-container">

    <button id="export-btn">Export Historical Data (CSV)</button>

</div>

</div>

<div id="error" class="hidden">

    <p class="error-message"></p>

</div>

</div>

<script src="{{ url_for('static', filename='js/main.js') }}"></script>

```

```

</body>from datetime import datetime, timedelta import io
import csv
import logging

app = Flask(__name__)

# Configure logging
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

def validate_symbol(symbol):

    if not symbol or not isinstance(symbol, str):
        raise ValueError("Invalid stock symbol")
    # Remove any whitespace and convert to uppercase symbol =
    symbol.strip().upper()
    # Check if symbol contains only valid characters if
    not all(c.isalnum() or c in '.-' for c in symbol):
        raise ValueError("Invalid characters in stock symbol")
    return symbol

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST']) def
predict():

```

```

try:

    # Get and validate stock symbol

    symbol = validate_symbol(request.form.get('symbol', ''))

    # Get historical data stock
    = yf.Ticker(symbol)
    end_date = datetime.now()

    start_date = end_date - timedelta(days=365*2)

    df = stock.history(start=start_date, end=end_date)

    if df.empty:

        return jsonify({'error': f'No data found for symbol: {symbol}'})

    # Get prediction and metrics result
    = predict_stock_price(df)
    # Get historical performance

    performance = get_historical_performance(symbol)

    # Get news

    news = get_news(symbol)

    return jsonify({

        'prediction': round(result['prediction'], 2),

        'lower_bound': round(result['lower_bound'], 2),

        'upper_bound': round(result['upper_bound'], 2),

        'current_price': round(float(df['Close'].iloc[-1]), 2),

```

```

'symbol': symbol,
'metrics': {
    'mae': round(result['metrics']['mae'], 2),
    'rmse': round(result['metrics']['rmse'], 2),
    'r2': round(result['metrics']['r2'] * 100, 1)
},
'confidence': {
    'level': round(result['confidence']['level'], 1),
    'spread': round(result['confidence']['spread'], 1),
    'trend': result['confidence']['trend']
},
'performance': performance,
'news': news
})

```

```

except DataValidationError as e:

    logger.warning(f'Data validation error for symbol {request.form.get('symbol', "")}: {str(e)}") return jsonify({'error': str(e)})
except ValueError as e:
    logger.warning(f'Value error: {str(e)}")
    return jsonify({'error': str(e)})
except Exception as e:

    logger.error(f'Unexpected error: {str(e)}", exc_info=True)

    return jsonify({'error': 'An unexpected error occurred. Please try again

```

```

def export_data():
    try:
        symbol = validate_symbol(request.form.get('symbol', ''))
        stock = yf.Ticker(symbol)
        df = stock.history(period='1y')

        if df.empty:
            raise ValueError(f"No data available for symbol: {symbol}")

        # Create CSV in memory
        output = io.StringIO()
        writer = csv.writer(output)

        # Write headers
        writer.writerow(['Date', 'Open', 'High', 'Low', 'Close', 'Volume'])

        # Write data
        for index, row in df.iterrows():
            writer.writerow([
                index.strftime('%Y-%m-%d'),
                round(row['Open'], 2),
                round(row['High'], 2),
                round(row['Low'], 2),
                round(row['Close'], 2),
            ])
    
```

```

output.seek(0)
return send_file(
    io.BytesIO(output.getvalue().encode('utf-8')),
    mimetype='text/csv',
    as_attachment=True, download_name=f'{symbol}_historical_data.csv'
)

except Exception as e:

    logger.error(f"Error in export_data: {str(e)}", exc_info=True)
    return jsonify({'error': str(e)})

@app.route('/stocks')
def get_stocks():
    try:

        stocks = get_available_stocks()
        return jsonify({'stocks': stocks})
    except Exception as e:

        logger.error(f"Error fetching stocks list: {str(e)}", exc_info=True) return
        jsonify({'error': str(e)})

if __name__ == '__main__':
    app.run(debug=True)

```

MODEL.PY :

```
from sklearn.model_selection import train_test_split from
sklearn.preprocessing import MinMaxScaler from
sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

import pandas as pd
import yfinance as yf
import requests
from datetime import datetime, timedelta
import os
from dotenv import load_dotenv

# Load environment variables
load_dotenv()

# Get API keys from environment variables ALPHA_VANTAGE_API_KEY =
os.getenv('ALPHA_VANTAGE_API_KEY')
FINNHUB_API_KEY = os.getenv('FINNHUB_API_KEY')

class DataValidationError(Exception): pass

def create_features(df): try:
    # Reduce minimum data requirement from 30 to 20 days if
    len(df) < 20:
```

```
raise DataValidationError(f'Insufficient historical data (need at least 20
days, got {len(df)} days)')
```

```
df = df.copy()
```

```
# Handle missing values before calculations
```

```
df = df.fillna(method='ffill').fillna(method='bfill')
```

```
# Technical indicators with error handling and shorter windows try:
```

```
# Shorter SMA windows
```

```
df['SMA_7'] = df['Close'].rolling(window=7, min_periods=1).mean()
df['SMA_20'] = df['Close'].rolling(window=14,
min_periods=1).mean()
```

```
# RSI with shorter period
```

```
df['RSI'] = calculate_rsi(df['Close'], period=7)
```

```
# Shorter EMA windows
```

```
df['EMA_12'] = df['Close'].ewm(span=8, min_periods=1).mean()
```

```
df['EMA_26'] = df['Close'].ewm(span=17, min_periods=1).mean()
```

```
df['MACD'] = df['EMA_12'] - df['EMA_26']
```

```
# Volume and volatility with shorter windows
```



```
df['Volume_MA'] = df['Volume'].rolling(window=5,
min_periods=1).mean()
```

```
df['Daily_Return'] = df['Close'].pct_change().fillna(0)
```

```
df['Volatility'] = df['Daily_Return'].rolling(window=10,
min_periods=1).std()
```

```
# Create target variable (next day's closing price)
```

```
df['Target'] = df['Close'].shift(-1)
```

```
except Exception as e:
```

```
    raise DataValidationError(f"Error calculating technical indicators:
{str(e)}")
```

```
return df
```

```
except Exception as e:
```

```
    raise DataValidationError(f"Error in create_features: {str(e)}")
```

```
def calculate_rsi(prices, period=7): try:
```

```
    delta = prices.diff()
```

```
    delta = delta.fillna(0)
```

```
    gain = (delta.where(delta > 0, 0)).rolling(window=period,
min_periods=1).mean()
```

```
    loss = (-delta.where(delta < 0, 0)).rolling(window=period,
min_periods=1).mean()
```

```

# Avoid division by zero
loss = loss.replace(0, np.inf)
rs = gain / lossexcept
Exception as e:

raise DataValidationError(f"Error calculating RSI: {str(e)}")


def get_news(symbol):
    try:
        all_news = []

        # Try Yahoo Finance first
        try:
            stock = yf.Ticker(symbol)
            yahoo_news = stock.news
            if yahoo_news:
                for item in yahoo_news[:10]: # Get latest 10 news items #
                    Skip items without title or summary
                    if not item.get('title') or not item.get('summary'):
                        continue

                    cleaned_item = {

                        'title': item.get('title'),

                        'summary': item.get('summary', item.get('description',
                            ")).strip(),

                        'time_published':

```

```

datetime.fromtimestamp(item.get('providerPublishTime', 0)), 'url':
    item.get('link', ''),
    'source': item.get('publisher', 'Yahoo Finance'),
    'sentiment': analyze_sentiment(item.get('title', '') + ' ' +
    all_news.append(cleaned_item)
except Exception as e:
    print(f"Error fetching Yahoo news: {str(e)}")

# Try Finnhub as second source

if len(all_news) < 5 and FINNHUB_API_KEY:
    try:
        headers = {'X-Finnhub-Token': FINNHUB_API_KEY}
        end_date = datetime.now()
        start_date = end_date - timedelta(days=7)

        url = f"https://finnhub.io/api/v1/company-
news?symbol=
l={symbol}&from={start_date
. strftime('%Y-%m-%d')} &to={end_date
e.strftime('%Y-%m-%d')}"

        response = requests.get(url, headers=headers)
        finnhub_news = response.json()

        for item in finnhub_news[:10]:
            'source': item.get('source', 'Finnhub'),
            'sentiment': analyze_sentiment(item.get('headline', '') + ' ' +

```

```

item.get('summary', ""))

    }

    all_news.append(cleaned_item)
except Exception as e:
    print(f"Error fetching Finnhub news: {str(e)}")


# Try Alpha Vantage as last resort

if len(all_news) < 5 and ALPHA_VANTAGE_API_KEY:
    try:
        url =
f'https://www.alphavantage.co/query?function=NEWS_SENTIMENT&tickers={symbol}&apikey={ALPHA_VANTAGE_API_KEY}'

    response = requests.get(url, timeout=10)
    data = response.json()
    av_news = data.get('feed', [])

    for item in av_news[:10]:
        if not item.get('title') or not item.get('summary'):
            continue

        cleaned_item = {
            'title': item.get('title'),
            'summary': item.get('summary').strip(),
            'time_published': datetime.strptime(

```

```

datetime.now().strftime('%Y%m%dT%H%M%S')),
        '%Y%m%dT%H%M%S'
    ),
    'url': item.get('url', ''),
    'source': item.get('source', 'Alpha Vantage'),
    'sentiment': item.get('overall_sentiment_score', 0)
}

all_news.append(cleaned_item)
except Exception as e:
    print(f"Error fetching Alpha Vantage news: {str(e)}")

```

Filter out duplicates based on title

```
seen_titles = set()
```

```
filtered_news = []
```

```
for item in all_news:
```

```
    if item['title'] not in seen_titles:
```

```
        seen_titles.add(item['title'])
```

```
        filtered_news.append(item)
```

```
# Sort news by date and get most recent
filtered_news.sort(key=lambda x:
x['time_published'], reverse=True)
```

Return at least one placeholder if no news found if

```
not filtered_news:
```

```
    return [{
```

```
        'title': 'No recent news available',
```

```
        'url': "",
        'source': 'System',
        'sentiment': 0
    ]}]
```

```
return filtered_news[:5] # Return 5 most recent news items
```

```
except Exception as e:
```

```
    print(f"Error fetching news: {str(e)}")
    return [{
        'title': 'Error fetching news',
        'summary': 'Temporarily unable to fetch news. Please try again later.',
        'time_published': datetime.now(),
        'url': "",
        'source': 'System',
        'sentiment': 0
    }]
```

```
def analyze_sentiment(text):
```

```
    """Simple sentiment analysis based on keywords"""

    positive_words = set(['up', 'rise', 'gain', 'positive', 'growth', 'profit', 'success',
                           'higher'])

    negative_words = set(['down', 'fall', 'loss', 'negative', 'decline', 'drop', 'lower',
                           'risk'])

    words = text.lower().split()
```

```

positive_count = sum(1 for word in words if word in positive_words)
negative_count = sum(1 for word in words if word in negative_words)
total = positive_count + negative_count if
total == 0:
    return 0

return (positive_count - negative_count) / total

def predict_stock_price(df):
    try:
        if len(df) < 30:
            raise DataValidationError(f"Insufficient data for reliable prediction (need at
least 30 days, got {len(df)} days)")

        # Create features with error handling try:
        df = create_features(df) df
        = df.dropna()
    except Exception as e:
        raise DataValidationError(f"Error preparing data: {str(e)}")

    if len(df) == 0:
        raise DataValidationError("No valid data after preprocessing")

    features = ['Close', 'SMA_7', 'SMA_20', 'RSI', 'EMA_12', 'EMA_26',
'MACD', 'Volume_MA', 'Volatility']

```

```

# Verify all features exist

missing_features = [f for f in features if f not in df.columns] if
missing_features:
    raise ValueError(f'Missing features: {',
'.join(missing_features)}')

X = df[features]
y = df['Target']

# Split and scale data

X_train, X_test, y_train, y_test = train_test_split(X[:-1], y[:-1],
test_size=0.2, random_state=42)

scaler = MinMaxScaler()

X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train model with more estimators and better parameters model
= RandomForestRegressor(
    n_estimators=200, # Increased from 100
    max_depth=15,    # Added parameter
    min_samples_split=5,
    min_samples_leaf=2,
    random_state=42
)

model.fit(X_train_scaled, y_train)

```



```

# Calculate base metrics

y_pred = model.predict(X_test_scaled)

mae = mean_absolute_error(y_test, y_pred)

rmse = np.sqrt(mean_squared_error(y_test, y_pred)) r2
= r2_score(y_test, y_pred)

# Make prediction for next day last_data =
scaler.transform(X.iloc[-1:])

# Get prediction and confidence interval
predictions = []
for estimator in model.estimators_: pred
    = estimator.predict(last_data)
    predictions.append(pred[0])

predictions = np.array(predictions)
prediction = float(np.mean(predictions))
conf_interval = np.percentile(predictions, [5, 95])

# Calculate confidence metrics
pred_std = np.std(predictions)
    pred_spread = (conf_interval[1] - conf_interval[0]) / prediction if
prediction != 0 else 1

# Calculate weighted confidence level

volatility_factor = df['Volatility'].iloc[-1] / df['Volatility'].mean() volume_factor

```

```

= df['Volume'].iloc[-1] / df['Volume'].mean()

confidence_level = max(min(

    (1 - pred_spread) * 0.3 +          # Prediction spread weight
    (r2 * 0.3) +                      # Model accuracy weight
    (1 - volatility_factor) * 0.2 +    # Volatility weight
    1.0
), 0.0) * 100

# Calculate trend and momentum
current_price = float(df['Close'].iloc[-1])
price_change = ((prediction - current_price) / current_price * 100) if
current_price != 0 else 0

momentum = calculate_momentum(df)

return {

    'prediction': float(prediction),
    'lower_bound': float(conf_interval[0]),
    'upper_bound': float(conf_interval[1]),
    'metrics': {
        'mae': float(mae),
        'rmse': float(rmse),
        'r2': float(r2)
    },
    'confidence': {

```

```

        'level': float(confidence_level),
        'spread': float(pred_spread * 100),
        'trend': {
            'direction': 'up' if prediction > current_price else 'down', 'strength':
            float(abs(price_change)),
            'percentage': float(price_change),
        }
    except DataValidationError as e: raise
        DataValidationError(str(e))
    except Exception as e:

        raise DataValidationError(f"Error in prediction process: {str(e)}")

```

```
def calculate_momentum(df):
```

```

    """Calculate price momentum indicator""" try:
        # Calculate short and long term momentum

        short_term = df['Close'].pct_change(5).iloc[-1] # 5-day momentum
        medium_term = df['Close'].pct_change(10).iloc[-1] # 10-day momentum
        long_term = df['Close'].pct_change(20).iloc[-1]    # 20-day momentum

        # Weight the momentum periods
        weighted_momentum = (
            short_term * 0.5 +    # 50% weight to short term
            medium_term * 0.3 + # 30% weight to medium term
            long_term * 0.2      # 20% weight to long term
        )

        # Classify momentum strength

        if abs(weighted_momentum) < 0.02: return

```

```

        'neutral'
    elif weighted_momentum > 0:
    else:

        return 'weak' if weighted_momentum > -0.05 else 'very weak'

except Exception:
    return 'neutral'

def get_historical_performance(symbol, period='1y'): try:
    stock = yf.Ticker(symbol)

    hist = stock.history(period=period)

    if len(hist) == 0:

        raise DataValidationError("No historical data available")

    performance = {

        'start_price': float(hist['Close'].iloc[0]),

        'end_price': float(hist['Close'].iloc[-1]),

        'return': float((((hist['Close'].iloc[-1] - hist['Close'].iloc[0]) /
hist['Close'].iloc[0]) * 100),

        'highest_price': float(hist['High'].max()),

        'lowest_price': float(hist['Low'].min()),

        'average_volume': float(hist['Volume'].mean()),

        'price_history': [float(x) for x in hist['Close'].tolist()],

```

```

        'dates': [x.strftime('%Y-%m-%d') for x in hist.index.tolist()]

    }

    return performance
except Exception as e:
    raise DataValidationError(f"Error fetching historical performance:
{str(e)}")

def get_available_stocks():
    try:
        # List of popular stocks (expanded list)
        available_stocks = [
            # Technology
            'AAPL', 'MSFT', 'GOOGL', 'AMZN', 'META', 'NVDA', 'TSLA',
            'AVGO', 'ORCL', 'CSCO', 'ADBE', 'CRM', 'AMD', 'INTC',

            # Finance
            'JPM', 'BAC', 'WFC', 'GS', 'MS', 'BLK', 'C', 'AXP', 'V', 'MA',

            # Healthcare
            'JNJ', 'UNH', 'PFE', 'MRK', 'ABBV', 'LLY', 'TMO', 'ABT', 'DHR', 'BMY',

            # Consumer
            'PG', 'KO', 'PEP', 'WMT', 'MCD', 'DIS', 'NFLX', 'NKE', 'SBUX',
            'HD',

            # Industrial
            'XOM', 'CVX', 'GE', 'BA', 'CAT', 'MMM', 'HON', 'UPS', 'RTX',

```

```
'LMT',  
  
    # Others  
  
    'VZ', 'T', 'IBM', 'QCOM', 'TXN', 'PYPL', 'BABA', 'TSM', 'ASML', 'TMUS'  
  
]
```

```
# Get basic info for each stock
```

```
stocks_info = []
```

```
for symbol in available_stocks:
```

```
    try:
```

```
        stock = yf.Ticker(symbol)
```

```
        info = stock.info
```

```
    # Get current price
```

```
    current_price = info.get('currentPrice', 0) if
```

```
    not current_price:
```

```
        current_price = info.get('regularMarketPrice', 0)
```

```
    stocks_info.append({
```

```
        'symbol': symbol,
```

```
        'name': info.get('shortName', info.get('longName', symbol)),
```

```
        'sector': info.get('sector', info.get('industry', 'N/A')), 'market_cap':
```

```
        info.get('marketCap', 0),
```

```
        'current_price': current_price, 'currency':
```

```
    }
```

SYSTEM TESTING

6 . SOFTWARE TESTING

Software testing ensures that all components of FutureGain function correctly and reliably. This section outlines the testing strategies implemented to validate system functionality, performance, and security.

Types of Testing :

Stock prediction applications require various types of testing to ensure accuracy, reliability, and performance.

Unit Testing

- Tests individual functions.
- Ensures expected outcomes are given for user inputs.
- Example: Verifying if stock price normalization functions work correctly.

Integration Testing

- Ensures that different components (APIs, datalink) work together correctly.
- Example: Testing whether the backend correctly fetches stock data and passes it to the client side for prediction.

System Testing (End-to-End Testing)

- Ensures the entire stock prediction system functions as a whole.
- Validates data retrieval, processing, and visualization.
- Example: A complete test where a stock symbol is entered, and the correct prediction is displayed.

Performance Testing

- Measures system response time, data processing speed, and model execution time.

- Example: Evaluating how many stock predictions can be processed per second under load.

Security Testing

- Checks for vulnerabilities such as API abuse, SQL injection, and XSS attacks.
- Example: Ensuring the system does not accept malicious inputs that could crash the application.

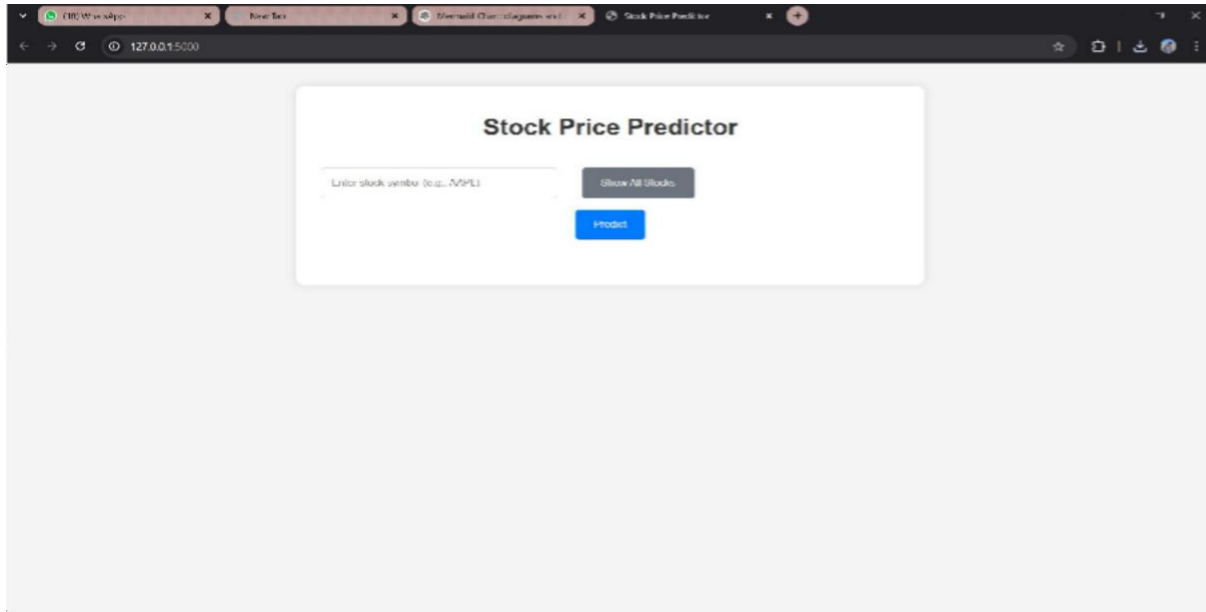
User Acceptance Testing (UAT)

- Validates the application with real-world users.
- Example: Testing if traders and analysts find the stock predictions useful and reliable.

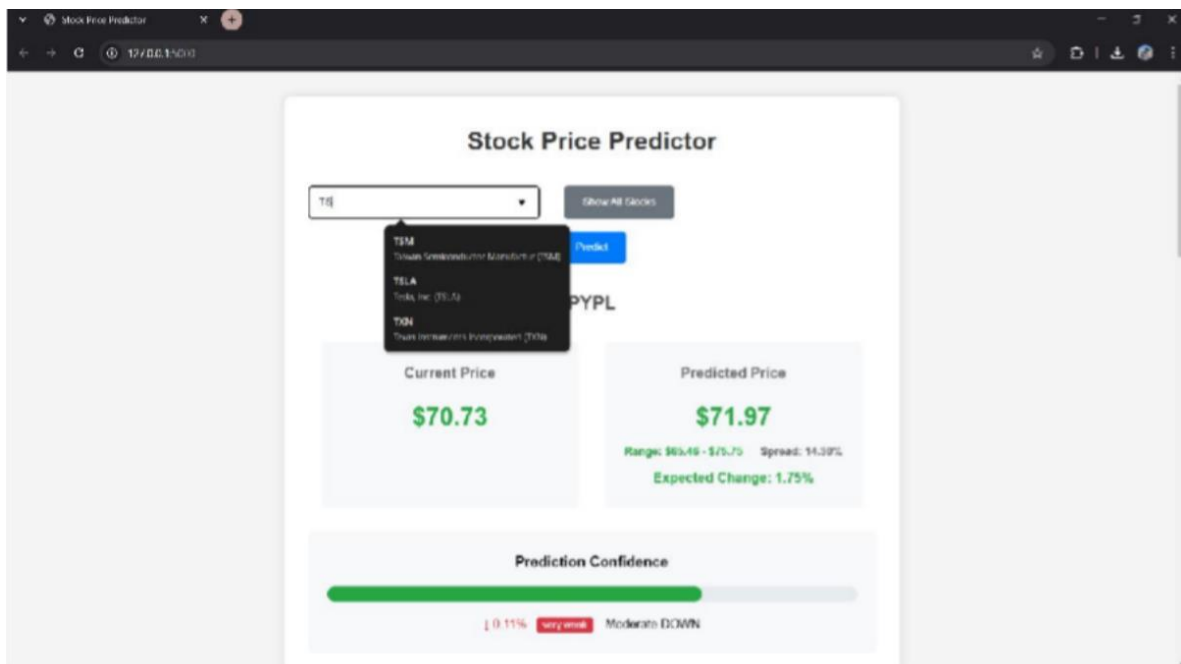
SCREENSHOTS

7. SCREENSHOTS

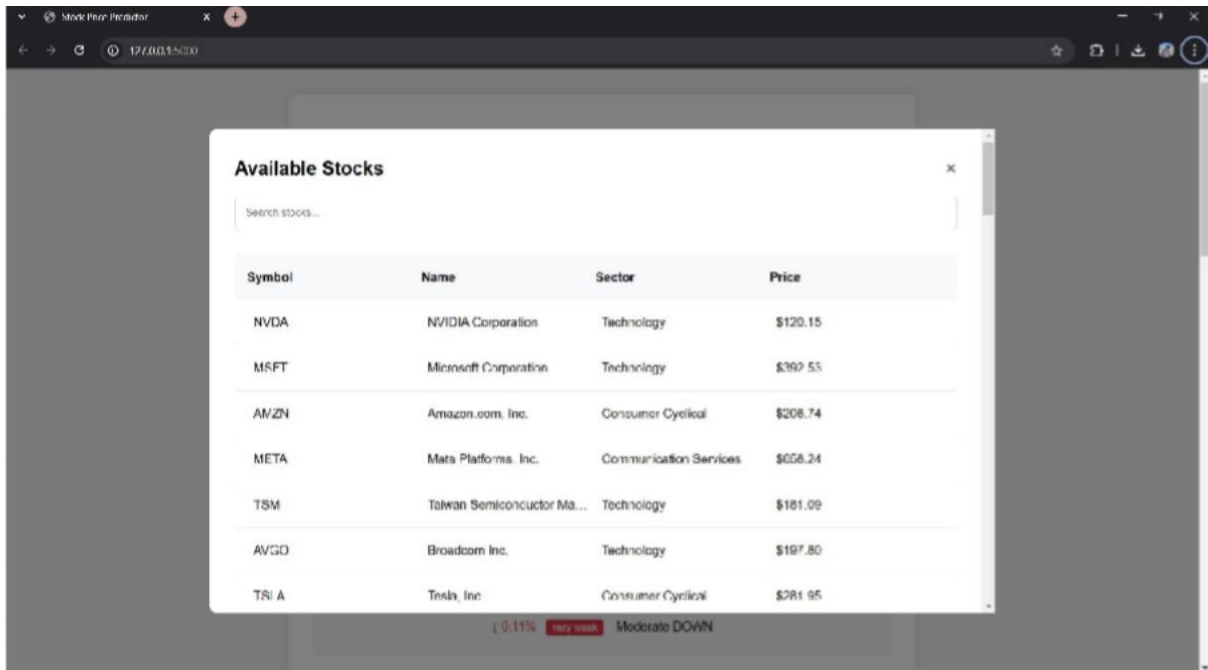
STOCKS SEARCH PAGE :



STOCKS SUGGESTION ON SEARCH:



AVAILABLE STOCKS :



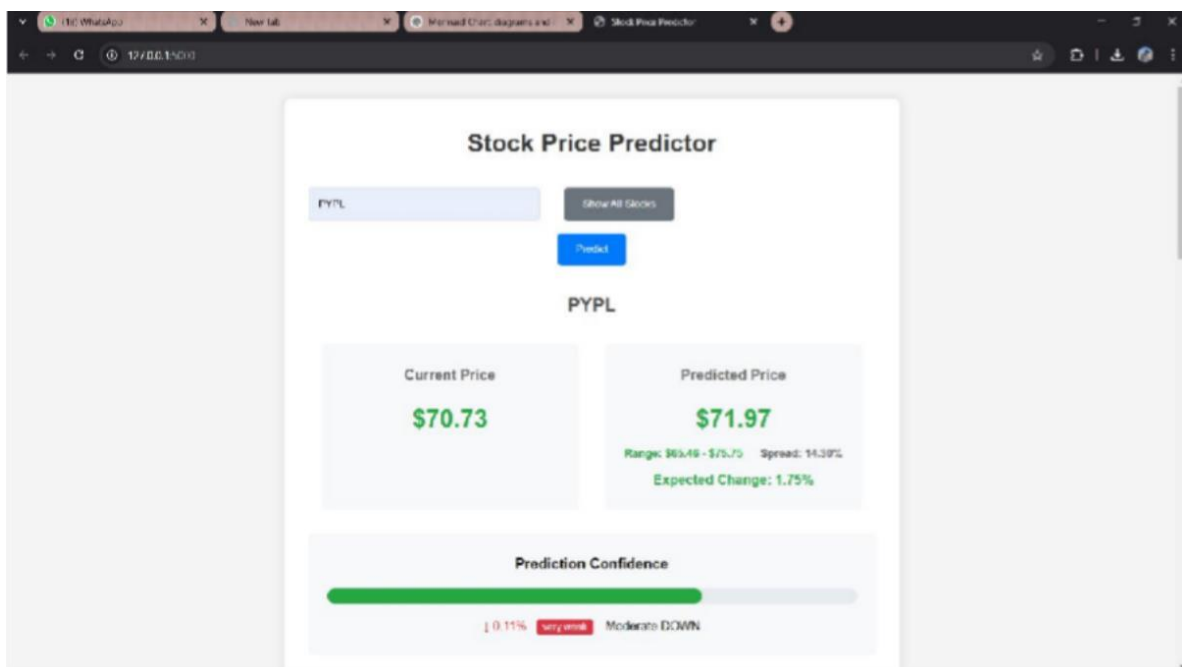
Available Stocks

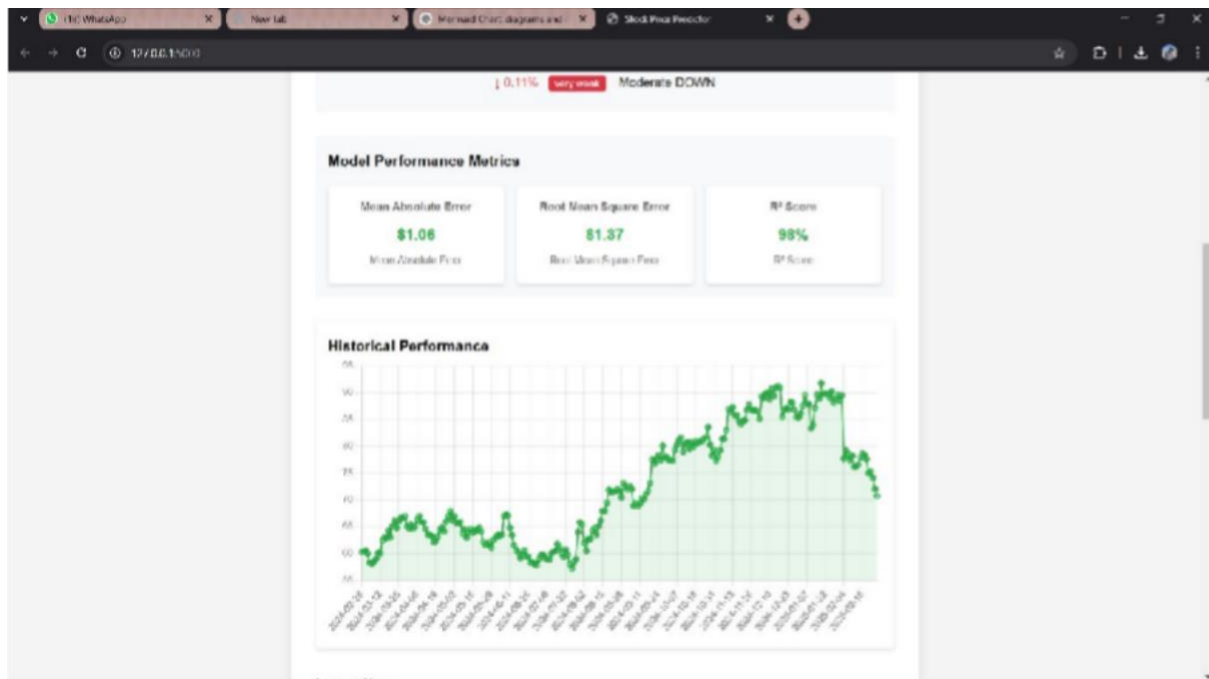
Search stocks...

Symbol	Name	Sector	Price
NVDA	NVIDIA Corporation	Technology	\$120.15
MSFT	Microsoft Corporation	Technology	\$397.53
AMZN	Amazon.com, Inc.	Consumer Cyclical	\$208.74
META	Meta Platforms, Inc.	Communication Services	\$258.24
TSM	Taiwan Semiconductor Ma...	Technology	\$181.09
AVGO	Broadcom Inc.	Technology	\$197.80
TSIA	Tesla, Inc.	Consumer Cyclical	\$281.95

↓ 0.11% Very weak Moderate DOWN

PREDICTION RESULT :

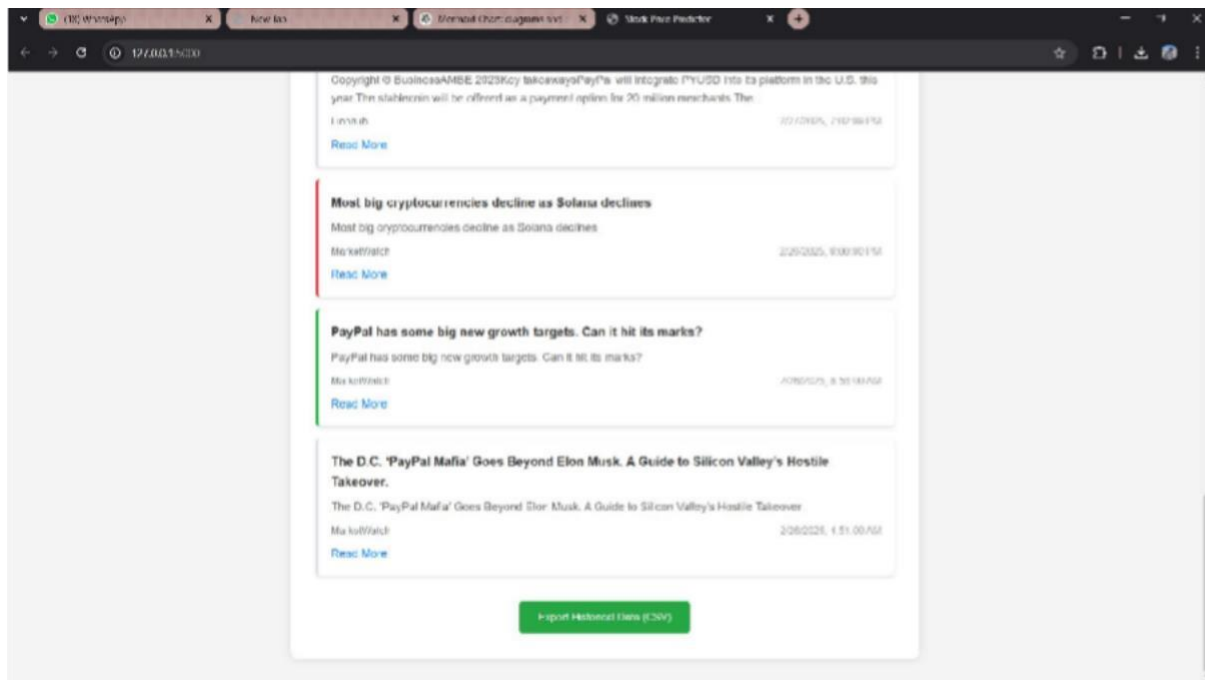




STOCK NEWS PAGE :

The screenshot shows a "Stock News Page" with a "Latest News" section. It features a list of four news items, each with a title, a brief description, a byline, and a timestamp. The first item is "Building A \$1,000,000 Dividend Portfolio: A Strategy For Growth, Income And Risk Management" by SpektigAlphe, dated 2/26/2023, 5:03:08 AM. The second is "PayPal integrates PYUSD in U.S. to strengthen presence in stablecoin market" by Copyright © BusinessAMSE 2023Key takeaways, dated 2/27/2023, 7:02:48 PM. The third is "Most big cryptocurrencies decline as Solana declines" by Ma.kul@iand, dated 2/28/2023, 6:00:40 PM. The fourth is "PayPal has some big new growth targets. Can it hit its marks?" by Merwan@stet, dated 2/28/2023, 8:16:08 AM. Each item has a "Read More" link.

CSV FILE DOWNLOAD PAGE:



CSV FILE DATA :

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Date	Open	High	Low	Close	Volume									
2	28-02-2024	59.7	60.9	59.44	60.25	9851600									
3	29-02-2024	60.6	61.87	60.01	60.34	1.7E+07									
4	01-03-2024	60.53	61.64	60.21	60.54	1.7E+07									
5	04-03-2024	60.41	60.6	58.86	59.98	1.7E+07									
6	05-03-2024	59.51	60.83	57.87	58.27	1.8E+07									
7	06-03-2024	58.75	58.92	57.91	58.12	1.7E+07									
8	07-03-2024	58.46	59.06	57.95	58.5	1.9E+07									
9	08-03-2024	58.72	59.56	58.24	59.01	1.8E+07									
10	11-03-2024	59	60.22	58.94	60.08	1.1E+07									
11	12-03-2024	59.8	60.47	59.3	60.03	1.3E+07									
12	13-03-2024	59.98	63.3	59.81	62.45	2E+07									
13	14-03-2024	62.51	64.2	62.71	63	1.9E+07									
14	15-03-2024	62.6	63.83	62.43	62.85	2E+07									
15	18-03-2024	63.12	64.76	63.1	64.23	1.4E+07									
16	19-03-2024	63.77	64.05	62.8	63.01	9089300									
17	20-03-2024	63.58	65.36	63.12	65.05	1.3E+07									
18	21-03-2024	65.3	66.99	65.05	66.14	1.4E+07									
19	22-03-2024	66.44	67.35	64.75	64.77	1E+07									

CONCLUSION

8. CONCLUSION

The Future GainS - Share Market Predictor successfully integrates real- time stock market data offering users insightful trend analysis and forecasts. the application provides a user-friendly platform for investors and traders to make informed decisions based on data-driven insights.

While the model offers valuable predictions, stock market movements are inherently uncertain due to external factors like economic events, news, and investor sentiment. Future improvements could involve incorporating deep learning models, sentiment analysis from financial news, and advanced technical indicators to enhance prediction accuracy.

Overall, FutureGain serves as a powerful tool for stock market enthusiasts, combining technology and financial analytics to aid decision-making in an ever-changing market landscape.

BIBLIOGRAPHY

9. BIBLIOGRAPHY

Flanagan, D. (2020). JavaScript: The Definitive Guide (7th ed.). O'Reilly Media.

McFarland, D. (2018). HTML and CSS: Design and Build Websites. John Wiley & Sons.

Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python (2nd ed.). O'Reilly Media.

Freeman, E., & Robson, E. (2020). Head First HTML and CSS (2nd ed.). O'Reilly Media.

Mozilla Developer Network (MDN). (n.d.). JavaScript, HTML, and CSS Documentation. Retrieved from <https://developer.mozilla.org>

Flask Documentation. (n.d.). Flask Web Framework. Retrieved from <https://flask.palletsprojects.com>

Python Software Foundation. (n.d.). Python Official Documentation. Retrieved from <https://docs.python.org/3/>

Bootstrap Documentation. (n.d.). Bootstrap: Frontend Framework for Responsive Design. Retrieved from <https://getbootstrap.com>