Visvesvaraya Technological University

Belagavi-590 018, Karnataka

Final Year Project Report submitted in partial fulfilment of 21AIP76 on

"STOCK RAJ"

Submitted in partial fulfilment for the award of degree in

BACHELOR OF ENGINEERING
IN
ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by:

Saanvi D Desai	**[1JT21AI039]**
Shreya Aparanji	**[1JT21AI042]**
Sooraj G V	**[1JT21AI047]**
Yuvaraj S	**[1JT21AI056]**

Under the guidance of
Dr Madhu B R
Professor & Head of
Department AI&ML

Department of Artificial Intelligence and Machine Learning

Jyothy Institute of Technology Tataguni,
Bengaluru-560082

Jyothy Institute of Technology
Tataguni, Bengaluru-560082
Department of Artificial Intelligence and Machine Learning

CERTIFICATE

Certified that the project work entitled **"Stock Raj"** carried out by **Saanvi D Desai [1JT21AI039], Shreya Aparanji [1JT21AI042], Sooraj G V [1JT21AI047]** and **Yuvaraj S [1JT21AI056]** bonafide students of Jyothy Institute of Technology, in partial fulfilment for the award of **Bachelor of Engineering** in **Artificial Intelligence and Machine Learning** department of the **Visvesvaraya Technological University, Belagavi** during the year **2024- 2025**. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Dr. Madhu B R	**Dr. Madhu B R**	**DR. K. Gopalakrishna**
Guide, Professor and Head	Professor and Head	Principal - JIT
Dept. of AI&ML	Dept. of AI&ML	Bengaluru
JIT, VTU	JIT, VTU	
Bengaluru	Bengaluru	

External Examiner Signature with Date:

1.

2.

ACKNOWLEDGEMENT

It is a great pleasure for us to acknowledge the assistance and support of a large number of individuals who have been responsible for the successful completion of this project work.

First, we take this opportunity to express our sincere gratitude to Jyothy Institute of Technology, VTU for providing us with a great opportunity to pursue our Bachelor's Degree in this institution.

In particular we would like to thank ***Dr. Gopalkrishnan****,* ***Principal,*** *Jyothy Institute of Technology, VTU for their constant encouragement and expert advice.*

It is a matter of immense pleasure to express our sincere thanks to ***Dr. Madhu B R, Professor and Head of the department****, Artificial Intelligence and Machine Learning, for providing right academic guidance that made our task possible.*

We would like to thank our guide ***Dr. Madhu B R, Professor and Head of the department,*** *Dept. of Artificial Intelligence and Machine Learning, for sparing his valuable time to extend help in every step of our project work, which paved the way for smooth progress and fruitful culmination of the project.*

We would like to thank our Project Coordinator ***Prof. Deepthi Das*** *and all the staff members AIML for their support.*

We are also grateful to our family and friends who provided us with every requirement throughout the course.

We would like to thank one and all who directly or indirectly helped us in completing the Project work successfully.

Signature of Students

D E C L A R A T I O N

We, Saanvi D Desai (1JT21AI039), Shreya Aparanji (1JT21AI042), Sooraj G V (1JT21AI047) and Yuvaraj S (1JT21AI056), 8th Semester students of Artificial Intelligence and Machine Learning Engineering, Jyothy Institute of Technology, Bangalore - 560082, declare that the Final year Project is successfully completed.

This report is submitted in partial fulfillment of the requirements for award of Bachelor of Engineering in Artificial Intelligence and Machine Learning, during the academic year 2024-2025.

Date:

Place: Bangalore

Saanvi D Desai [1JT21AI039]

Shreya Aparanji [1JT21AI042]

Sooraj G V [1JT21AI047]

Yuvaraj S [1JT21AI056]

CERTIFICATES

ABSTRACT

StockRaj is a real-time Indian stock market analytics and visualization web application designed to simplify stock analysis for everyday users, investors, and learners. The project was developed to address the lack of accessible, user-friendly platforms that offer real-time data, predictive insights, and intelligent visualizations tailored specifically for the Indian market.

Unlike existing platforms such as MoneyControl and TradingView, StockRaj integrates advanced features like multiple chart types (candlestick, line, baseline, area), technical indicators, machine learning-based price prediction, and sentiment analysis using natural language processing. It also includes tools like a customizable watchlist, portfolio tracking, and an AI-powered chatbot to enhance user experience.

The system was built using React, TypeScript, Vite, and Node.js, with Python used for backend modeling tasks. A modular architecture ensures scalability and maintainability, while GitHub and VSCode/Cursor supported version control and development.

By combining financial data, visualization, and AI technologies, StockRaj provides a comprehensive, interactive platform for stock analysis, bridging the gap between complex financial tools and user accessibility.

TABLE OF CONTENTS

Page no.

Chapter 1

1. Introduction 02

Chapter 2

2. Problem Statement 04

Chapter 3

3. Objectives 06

Chapter 4

4. Related Work 07

4.1 Literature Survey 08

Chapter 5

5. Hardware and Software requirement 14

5.1 Hardware requirements 15

5.2 Software requirements 15

5.3 Development Tools and Platforms 16

Chapter 6

6. Methodology 17

6.1 Work Flow 18

6.1.1 Data Acquisition & Preprocessing 18

6.1.2 Technical Indicator Computation 19

6.1.3 Machine Learning for Price Prediction 19

6.1.4 Sentimental Analysis using NLP 20

6.1.5 AI Chatbot Integration 20

6.1.6 Visualization & User Interaction 21

6.2 Model Architecture 22

6.2.1 Price Prediction Engine 22

6.2.2 Sentimental Analysis Module 23

6.2.3 Modular App Design 24

7. System Design 26
7.1 System Architecture 27
7.1.1 Data Processing 27
7.1.2 Model Architecture 28
7.1.3 Training and Evaluation 28
7.1.4 Inference and Caption Generation 28
7.1.5 Translation 28
7.1.6 User Interface 29
7.1.7 Documentation and Deployment 29
7.2 Data Flow Diagram 30
7.3 Use-Case Diagram 32
7.4 Class Diagram 33
7.5 Website Architecture 35
Chapter 8
8. Implementation 37
8.1 Pseudo code 38
8.2 Model Training 39
8.2.1 CNN EfficientNetB0 39
8.2.2 Transformer Encoder Decoder 39
8.3 Caption Generation 40
8.4 Multiple Language conversion using Libraries 41
8.5 Web Deployment 42
8.5.1 Web Technologies 42
8.5.2 Web Pages 43
Chapter 9
9. Results and Analysis 45
9.1 Results and Analysis 46
9.1.1 Training and Validation Performance 47
9.1.2 Model Loss 48
9.1.3 Interpretation of Results 49

10. Snapshots and Implementation Videos 50

Chapter 12

11. Conclusion and Future Enhancement 64

11.1 Conclusion 65

11.2 Future Enhancement 66

References 67

Appendix A 70

LIST OF FIGURES

Fig. No.	Description of the figure	Page No.
6.1	Work Flow	18
6.2.1	Stock price Prediction Process	22
6.2.2	Sentimental Analysis Process	23
6.2.3	App Design	24
7.1	Methodology	27
7.2.1	CNN Model	31
7.2.2	Transformer Architecture	32
7.2.3	Image Captioning Model	34
8.1 (a)	Image Parameters	36
8.1 (b)	Load Captions data	36
8.2	Model Training	37
8.3	Caption Generation	38
8.4	Multiple Language Caption Generation	39
9.1.1	Accuracy over Epochs	44
9.1.2	Loss over Epochs	45
10.1	Website Architecture	48
11 (a)	Caption Generated in single language	54
11 (b)	Caption Generated in multiple languages	54
11 (c)	Accuracy Graph	55
11 (d)	Main Page of Website	55
11 (e)	Backend Implementation Video	56
11 (f)	Result Page	56

LIST OF TABLES

Table No.	Description of the Table	Page No.
4.1	Literature Survey	13

NOMENCLATURE USED

LCD	Liquid Crystal Display
GPRS	General Packet Radio Service
AT	Attention
GUI	Graphical User Interface
USB	Universal Serial Bus
SIM	Subscriber Identification Module
LF	Line Feed
CR	Carriage Return
PHP	Pre-Processor Hyper text
MySQL	My Structured Query Language

CHAPTER 1
INTRODUCTION

INTRODUCTION

In recent years, the Indian stock market has seen a significant rise in participation from individual investors, learners, and retail traders. With the growing availability of financial data and increasing interest in market dynamics, there is a rising demand for tools that can present complex stock market information in a simple, interactive, and insightful manner. However, most existing platforms are either cluttered with information or cater primarily to advanced users, leaving behind those who are just starting or seeking a more intuitive experience. This gap in accessibility and usability formed the foundation for the development of **StockRaj**.

StockRaj is a real-time Indian stock market analytics and visualization web application. It was designed with the goal of making stock market data and analysis approachable and actionable for a broad audience — from first-time investors to experienced traders and students studying finance or data science. The project focuses on simplifying the complexities of stock analysis by offering a visually rich and user-friendly interface backed by powerful real-time data processing and intelligent insights.

The key motivation behind StockRaj is to empower users with an all-in-one platform that combines real-time updates, interactive charting tools, and predictive analytics. Unlike many existing services that rely on static or delayed data, StockRaj emphasizes **live market updates** and **AI-driven insights**, helping users make informed decisions based on the latest market conditions. It also aims to serve as an educational platform where users can explore trends, test strategies, and learn about financial indicators in a hands-on manner.

The idea for StockRaj emerged after analyzing the limitations of existing platforms such as MoneyControl, TradingView, and Yahoo Finance. While these tools are popular and feature-rich, they often lack user-specific customization, advanced AI integration, or tailored solutions for Indian stock market users. StockRaj bridges these gaps by focusing on a clean interface, modular architecture, and a set of intelligent features designed to meet the evolving needs of users in a fast-paced financial environment.

Overall, this chapter lays the groundwork for understanding the importance and scope of the StockRaj platform. It highlights the driving factors behind its development and sets the stage for the chapters that follow, which detail the problem statement, objectives, related work, methodology, and implementation process that brought this idea to life.

CHAPTER 2

PROBLEM STATEMENT

PROBLEM STATEMENT

The existing stock market platforms lack accessible tools that provide real-time data, predictive insights, and intelligent visualizations tailored for Indian users.

This creates a need for a solution like StockRaj that leverages modern technologies to make stock analysis more intuitive and data-driven.

CHAPTER 3

OBJECTIVES

OBJECTIVES

The primary objective of the StockRaj project is to develop an intelligent, real-time stock market analytics and visualization platform specifically designed for Indian users. The goal is to empower retail investors, learners, and financial analysts with a tool that combines data-driven insights, modern interface design, and advanced analytics features. The following key objectives guide the development of StockRaj:

1. **Comprehensive Charting Tools**
   Implement various interactive chart types such as candlestick, line, baseline, and area charts to represent stock performance effectively. These visual tools allow users to interpret trends, patterns, and price movements with clarity and precision.

2. **Real-Time Data Integration**
   Enable seamless integration of real-time market data using reliable APIs to ensure users have access to the most current and accurate financial information. This real-time feed is essential for making timely and informed investment decisions.

3. **Predictive Modeling**
   Incorporate machine learning models to forecast stock price trends and patterns. These predictive analytics provide users with actionable insights, helping them anticipate future market movements and optimize their investment strategies.

4. **Visualization of Technical Indicators**
   Support the visualization of key technical indicators such as moving averages, RSI (Relative Strength Index), MACD (Moving Average Convergence Divergence), and others. These indicators aid in analyzing market momentum and trend reversals.

5. **User-Centric Features**
   Include personalized tools such as a watchlist and portfolio tracker to help users monitor specific stocks and manage their holdings. These features enhance user engagement and streamline portfolio management.

6. **Sentiment Analysis and Chatbot Integration**
   Leverage natural language processing (NLP) to analyze market sentiment from news and social media sources. Integrate an AI-powered chatbot to assist users with queries, explain financial terms, and provide platform guidance in real-time.

CHAPTER 4
RELATED WORK

4.1 LITERATURE SURVEY

The development of StockRaj is informed by extensive research in stock market analytics, predictive modeling, and data visualization, with a focus on applications within the Indian financial context. The following studies have significantly influenced the design and functionality of StockRaj:

1. **Machine Learning Approaches in Indian Stock Market Prediction**

   Recent studies have demonstrated the efficacy of machine learning models in forecasting stock market trends. For instance, a study employed various machine learning algorithms to predict stock market movements, highlighting the potential of these models in handling the complexities of financial data.

2. **Integration of Technical Indicators for Enhanced Prediction**

   The utilization of technical indicators such as Moving Averages, RSI, and MACD has been shown to improve the accuracy of stock price predictions. A notable study leveraged a set of basic and derived technical indicators to analyze stock price movements, underscoring the importance of these features in predictive modeling.

3. **Real-Time Data Processing and Incremental Learning**

   The necessity for real-time data processing in stock market analytics has led to the exploration of incremental learning techniques. Research has proposed strategies that combine incremental learning and offline-online learning to forecast stock prices using live market data streams, thereby enhancing the responsiveness of predictive models.

4. **Sentiment Analysis in Financial Forecasting**

   Incorporating sentiment analysis from news and social media sources has been recognized as a valuable addition to traditional financial indicators. Studies have combined machine learning with sentiment analysis to create robust prediction models capable of capturing the nuanced influences of market sentiment on stock prices.

5. **Deep Learning Models for Sectoral Analysis**

Deep learning architectures, particularly Long Short-Term Memory (LSTM) networks, have been effectively utilized for sector-specific stock price prediction. Research focusing on the Indian stock market has demonstrated the capability of LSTM models to accurately forecast prices across various sectors, aiding in informed investment decisions.

6. **Visualization Tools in Stock Market Applications**

The role of data visualization in enhancing user comprehension of complex financial data is well-documented. Tools that provide interactive and intuitive visual representations of stock trends and indicators are crucial for both novice and experienced investors.

7. **Comprehensive Decision-Support Systems**

The development of unified platforms that integrate data collection, analysis, and visualization has been identified as a key advancement in stock market applications. Such systems aim to provide end-to-end solutions for investors, streamlining the process of data-driven decision-making.

8. **Hybrid Models Combining Statistical and Machine Learning Techniques**

Studies have explored hybrid models that integrate statistical methods with machine learning algorithms to enhance prediction accuracy. These models leverage the strengths of both approaches to better capture market dynamics.

9. **Application of Artificial Neural Networks in Stock Prediction**

Artificial Neural Networks (ANNs) have been applied to predict stock prices, demonstrating their ability to model nonlinear relationships in financial data. Research has shown that ANNs can effectively capture complex patterns in stock market movements.

10. **Use of Support Vector Machines for Market Trend Analysis**

Support Vector Machines (SVMs) have been utilized for classifying market trends, offering robust performance in handling high-dimensional financial datasets. Studies indicate that SVMs can effectively distinguish between different market conditions.

11. Incorporation of Macroeconomic Indicators in Prediction Models

Incorporating macroeconomic indicators such as GDP growth, inflation rates, and interest rates into prediction models has been shown to improve forecasting accuracy. These indicators provide a broader economic context for stock market analysis.

12. Time Series Analysis Using ARIMA Models

Autoregressive Integrated Moving Average (ARIMA) models have been applied for time series analysis in stock market prediction. These models are effective in capturing temporal dependencies in financial data.

13. Optimization Techniques for Model Performance Enhancement

Optimization algorithms such as Genetic Algorithms and Particle Swarm Optimization have been employed to fine-tune model parameters, leading to improved prediction accuracy in stock market forecasting.

14. Impact of High-Frequency Trading on Market Dynamics

Research has examined the influence of high-frequency trading on market volatility and liquidity, highlighting the need for models that can adapt to rapid market changes.

15. Role of Ensemble Learning in Enhancing Prediction Robustness

Ensemble learning methods, which combine multiple models to improve prediction performance, have been applied in stock market forecasting. These approaches help in reducing model variance and bias.

16. Application of Reinforcement Learning in Trading Strategies

Reinforcement learning techniques have been explored for developing adaptive trading strategies that learn from market interactions to optimize investment decisions.

17. Use of Big Data Analytics in Financial Forecasting

The integration of big data analytics in financial forecasting allows for the processing of vast and diverse datasets, enabling more comprehensive market analysis and prediction.

18. Sentiment Analysis Using Natural Language Processing

Natural Language Processing (NLP) techniques have been utilized to analyze investor sentiment from textual data sources, providing insights into market psychology and its impact on stock prices.

19. Development of Mobile Applications for Stock Analysis

The creation of mobile applications for stock market analysis has increased accessibility for investors, offering real-time data and analytics tools on-the-go.

20. Evaluation of Model Performance Using Cross-Validation Techniques

Cross-validation methods are employed to assess the generalizability of prediction models, ensuring their robustness across different market scenarios.

Sl. No.	Title	Methodology	Tools Used	Results
01	Machine Learning Approaches in Indian Stock Market Prediction	Employed ML models to forecast Indian stock trends using historical data and sentiment analysis.	Machine Learning, Sentiment Analysis	Improved stock trend prediction with combined features.
02	Integration of Technical Indicators for Enhanced Prediction	Integrated technical indicators (MA, RSI, MACD) into models to enhance prediction quality.	Moving Averages, RSI, MACD	Achieved higher accuracy in stock price prediction.
03	Real-Time Data Processing and Incremental Learning	Used incremental and hybrid learning to handle live stock data streams.	Incremental Learning, Offline-Online Learning	Enabled real-time predictive updates with dynamic data.
04	Sentiment Analysis in Financial Forecasting	Merged market sentiment from news/social media with ML predictions.	NLP, Sentiment Analysis, ML Models	Enhanced predictions by incorporating sentiment dynamics.
05	Deep Learning Models for Sectoral Analysis	Applied LSTM networks to predict sector-specific stock prices.	LSTM, Deep Learning	Achieved high accuracy in sectoral price forecasting.
06	Visualization Tools in Stock Market Applications	Developed visual analytics for clearer interpretation of stock trends.	Interactive Charts, Visualization Libraries	Improved investor comprehension via interactive visuals.
07	Comprehensive Decision-Support Systems	Built end-to-end investment platforms combining prediction and visualization.	Decision Support Systems, Financial Dashboards	Enabled holistic, user-friendly investment insights.
08	Hybrid Models Combining Statistical and Machine Learning Techniques	Created hybrid models integrating statistical + ML methods.	ARIMA, ML, Regression Models	Boosted model accuracy through hybridization.
09	Application of Artificial Neural Networks in Stock Prediction	Used ANNs to model nonlinear financial data patterns.	ANN, Feedforward Networks	Effectively modeled complex market relationships.
10	Use of Support Vector Machines for Market Trend Analysis	Classified market trends using SVMs on financial features.	Support Vector Machine, Classification	Robust trend classification with high-dimensional data.

11	Incorporation of Macroeconomic Indicators in Prediction Models	Integrated economic indicators (GDP, inflation, etc.) into models.	Economic Analysis, Feature Engineering	Improved context-aware prediction accuracy.
12	Time Series Analysis Using ARIMA Models	Modeled historical price data using ARIMA for temporal trends.	ARIMA, Time Series Modeling	Captured sequential trends in stock price forecasting.
13	Optimization Techniques for Model Performance Enhancement	Tuned model parameters using evolutionary algorithms.	Genetic Algorithm, PSO, Optimization	Enhanced performance via hyperparameter tuning.
14	Impact of High-Frequency Trading on Market Dynamics	Analyzed effects of HFT on liquidity and volatility.	HFT Analytics, Market Microstructure	Identified HFT's impact on market behavior.
15	Role of Ensemble Learning in Enhancing Prediction Robustness	Combined multiple models to reduce prediction variance.	Bagging, Boosting, Random Forest	Increased prediction stability and accuracy.
16	Application of Reinforcement Learning in Trading Strategies	Used RL agents to create adaptive trading models.	Reinforcement Learning, Trading Bots	Developed self-optimizing investment strategies.
17	Use of Big Data Analytics in Financial Forecasting	Applied big data processing to handle vast market datasets.	Big Data, Hadoop, Spark	Enabled large-scale financial data analysis.
18	Sentiment Analysis Using Natural Language Processing	Applied NLP for analyzing investor sentiment from text.	NLP, Text Mining, Sentiment Score	Enhanced forecast precision via sentiment insight.
19	Development of Mobile Applications for Stock Analysis	Designed mobile apps for accessible stock tracking.	Flutter/React Native, REST APIs	Improved real-time market access for users.
20	Evaluation of Model Performance Using Cross-Validation Techniques	Validated models using cross-validation on multiple datasets.	K-Fold Cross-Validation, Evaluation Metrics	Ensured model generalizability and consistency.

Table 4.1 Literature Survey Table

CHAPTER 5

HARDWARE AND SOFTWARE REQUIREMENTS

5.1 Hardware Requirements

The system was developed and tested on a machine with the following specifications, which provided sufficient resources for both frontend and backend development, as well as for running basic predictive models:

- **Processor:** Intel Core i5 or equivalent with a base clock speed of 2.5 GHz or higher
- **RAM:** Minimum 8 GB RAM
- **Storage:** At least 20 GB of available disk space
- **Display:** Screen resolution of 1024 x 768 pixels or higher for effective visualization and development

These specifications ensured adequate performance for code compilation, real-time updates, and rendering dynamic visualizations.

5.2 Software Requirements

The StockRaj application was built using a combination of modern web technologies and Python-based modeling frameworks. The following software components were critical:

- **Programming Languages:**
  - TypeScript (for frontend logic)
  - Python 3.10+ (for backend and predictive modeling)
- **Frontend Stack:**
  - React.js
  - Vite (for fast development and build)
  - HTML5 & CSS3 (for structuring and styling)
- **Backend & Modeling Tools:**
  - Python libraries: NumPy, Pandas, Matplotlib, Scikit-learn, Flask
  - Charting libraries: ApexCharts, Chart.js for data visualization

5.3 Development Tools and Platforms

- **Code Editors:**
  - Visual Studio Code
  - Cursor (AI-enhanced editor for productivity)
- **Version Control:**
  - Git and GitHub for source control and collaboration
- **Platforms:**
  - Compatible with **Windows**, **Linux**, and **macOS** operating systems
  - Developed in local environments and tested on **Chrome** and **Firefox** browsers for cross-platform compatibility

CHAPTER 6

METHODOLOGY

6.1 WORK FLOW & APPROACH

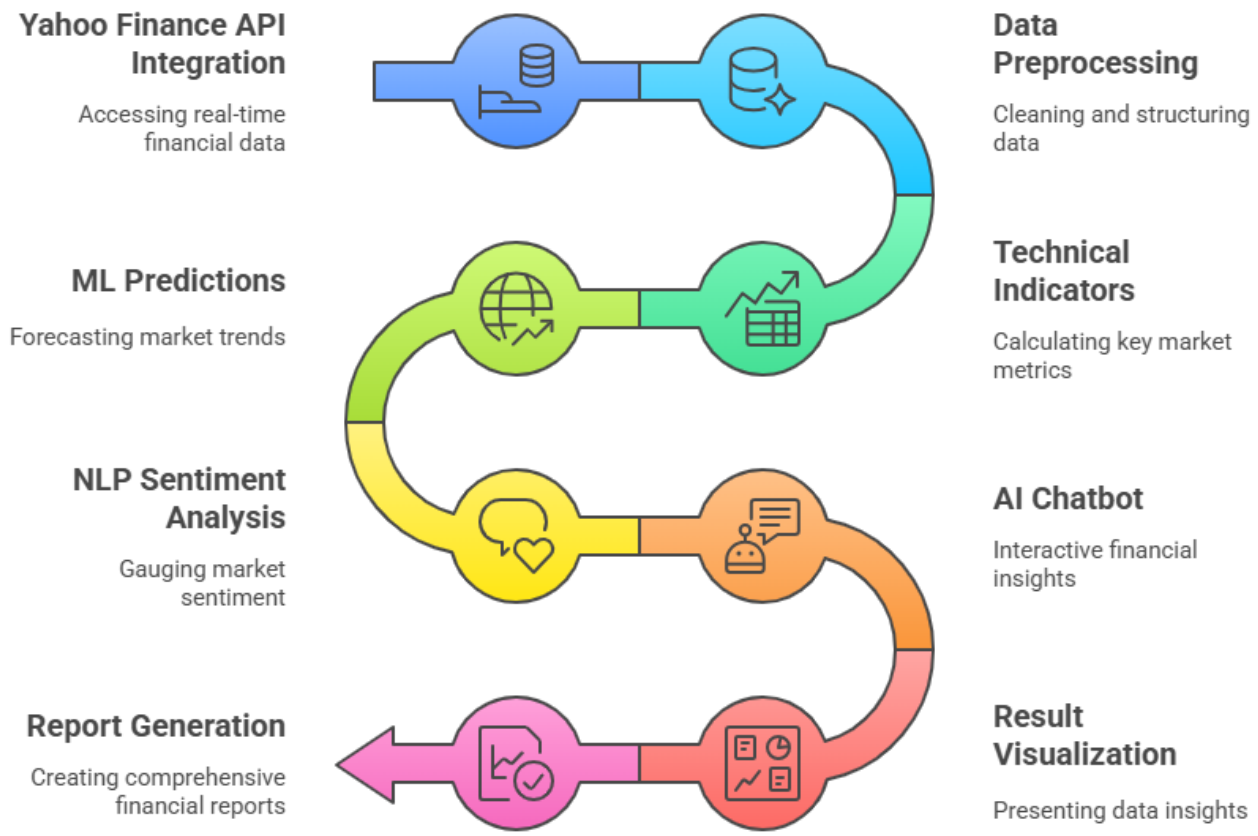


Figure 6.1 Work Flow

6.1.1 Data Acquisition and Preprocessing

The first step in the development of StockRaj involved acquiring reliable and high-frequency stock market data. This was achieved by integrating the Yahoo Finance API, which provides real-time and historical market data for publicly traded companies. The system allows users to choose a stock ticker, and the backend fetches relevant data including open, high, low, close, volume, and adjusted close prices.

Once the data was retrieved, preprocessing became essential to ensure that the input to the machine learning models and technical analysis tools was clean and structured. Preprocessing steps included handling missing values by forward-filling or interpolation, normalizing numerical features using Min-Max or Z-score normalization, and converting date-time strings into Python datetime objects for time-series alignment. The dataset was further enhanced by creating lag features and rolling window statistics, which are critical for capturing trends in stock movements. Finally, the preprocessed dataset was split into training and testing sets using a 70:30 ratio, ensuring that the models could generalize well on unseen data.

6.1.2 Technical Indicator Computation

Technical indicators serve as a cornerstone for financial analysis. In StockRaj, a variety of technical indicators were calculated to provide insights into market trends and investor behavior. These indicators help in understanding patterns that may not be visible through price movements alone. Some of the prominent indicators integrated into the system include Simple Moving Average (SMA), Exponential Moving Average (EMA), Moving Average Convergence Divergence (MACD), and Relative Strength Index (RSI).

The calculations were implemented using Python libraries like pandas_ta and ta-lib, which allowed seamless integration and real-time computation. For example, the SMA was computed by averaging the closing prices over a defined period, while the RSI was derived from the magnitude of recent gains to recent losses over a 14-day period. These indicators were then visualized using Plotly and Matplotlib, allowing users to interact with the plotted charts and understand the stock's historical performance. These insights also served as features for the prediction models, improving their predictive capability.

6.1.3 Machine Learning for Price Prediction

A core functionality of StockRaj is its ability to forecast future stock prices using machine learning algorithms. The prediction engine was built by leveraging both traditional machine learning models and deep learning architectures. Initially, models like Linear Regression and Random Forest Regressor were used for their interpretability and efficiency in handling tabular data. These models were trained on a feature matrix that included raw stock prices, technical indicators, and lagged values.

To handle the temporal dependencies in stock market data, an LSTM (Long Short-Term Memory) network was also implemented. LSTM is a type of recurrent neural network (RNN) that excels in capturing long-term dependencies in sequential data. The time-series data was reshaped into 3D tensors, and the model was trained to predict the next closing price based on historical patterns. Evaluation metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R^2 score were computed to assess model performance. The models were fine-tuned using hyperparameter tuning techniques like Grid Search and Bayesian Optimization.

6.1.4 Sentiment Analysis Using NLP

Market sentiment plays a vital role in stock price movement. To incorporate this psychological aspect into StockRaj, a Natural Language Processing (NLP)-based sentiment analysis module was developed. This module processes financial news headlines and social media posts to determine the market's emotional tone toward a particular stock or the market in general.

The process begins by collecting news articles using APIs like NewsAPI, focusing on the latest headlines that mention the selected stock ticker. The text data underwent a preprocessing pipeline that included tokenization, removal of stop words, lemmatization, and conversion to lowercase. Sentiment analysis was conducted using two approaches: rule-based and machine learning-based. The rule-based approach used the VADER sentiment analyzer, which is well-suited for short financial texts, while the ML-based approach used a fine-tuned BERT (Bidirectional Encoder Representations from Transformers) model trained on financial text datasets.

The outputs of the sentiment classifier were categorized into positive, neutral, or negative sentiments. These sentiments were then mapped to their corresponding impact on the stock, giving users additional context to make informed decisions. The results were visualized using sentiment timelines and pie charts.

6.1.5 AI Chatbot Integration

To make the platform more interactive and user-friendly, an AI-powered chatbot was integrated into StockRaj. The chatbot is capable of handling user queries related to stock prices, predictions, definitions of technical indicators, and even interpreting sentiment results. The chatbot was trained using custom intents and responses derived from financial FAQs and conversational data.

The NLP engine behind the chatbot uses the Rasa framework and transformer-based models like DistilBERT to handle intent recognition and slot filling. The chatbot runs as an API service and communicates with the frontend using asynchronous web sockets, allowing real-time interaction. It supports both predefined queries and natural user input, offering a human-like conversational experience.

For example, users can ask, “What is the RSI of TCS?” or “Will Infosys stock go up tomorrow?” The chatbot parses the query, identifies the stock symbol and indicator or action, and retrieves the relevant information from the backend before responding with a structured answer. This enhances the accessibility and usability of the system, especially for non-technical users.

6.1.6 Visualization and User Interaction

The frontend of StockRaj was designed with a focus on usability and modern UI principles. Built using React.js and styled with Tailwind CSS, the interface offers a clean, responsive dashboard where users can interact with all modules of the system. The dashboard allows users to select stocks, visualize technical indicators, review prediction graphs, analyze sentiment results, and interact with the chatbot.

Interactive charts were implemented using Chart.js and Plotly, enabling users to explore stock data across different timeframes (1 day, 1 week, 1 month, etc.). The machine learning predictions are plotted alongside historical stock data, giving users a visual understanding of the forecast. Additionally, sentiment trends are shown using bar graphs and pie charts, indicating the prevailing sentiment for the selected stock.

A report generation feature allows users to download a comprehensive analysis in PDF format. This includes raw data, visualized trends, predicted prices, and sentiment insights, making it easier for users to archive and share their analyses.

6.2 MODEL ARCHITECTURE

6.2.1 Price Prediction Engine

The price prediction module of StockRaj combines both traditional and deep learning models to deliver robust forecasts. For short-term predictions, ensemble models like Random Forest are used due to their ability to model non-linear relationships and handle high-dimensional data without overfitting. The feature set includes historical stock prices, calculated technical indicators, and sentiment scores.

For capturing the sequential nature of financial time series, an LSTM network was employed. The LSTM model consists of input layers, hidden layers with forget and memory gates, and an output layer that predicts the next price point. Dropout regularization and batch normalization were used to prevent overfitting and stabilize training. The model was trained on rolling windows of stock data, and results were smoothed to remove noise.

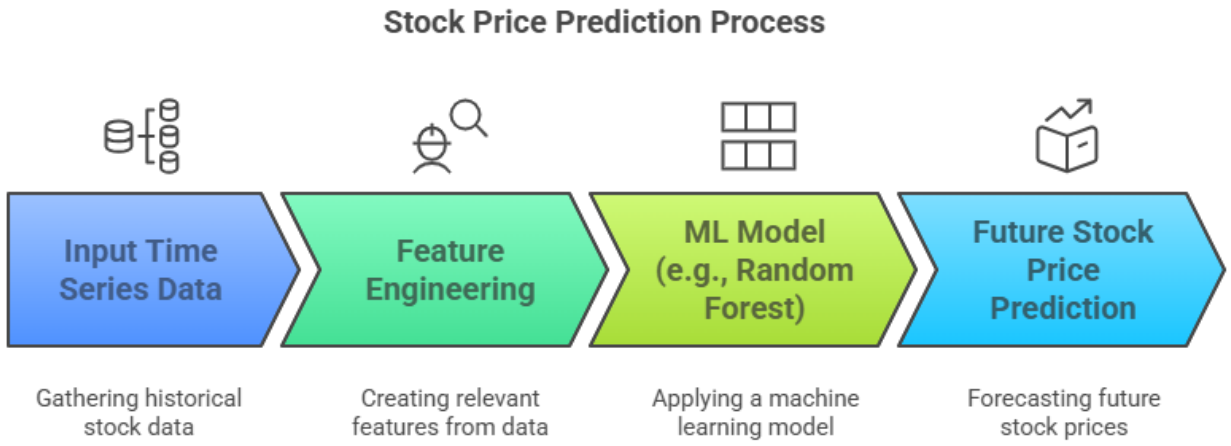


Figure 6.2.1 Stock Price Prediction Process

6.2.2 Sentiment Analysis Module

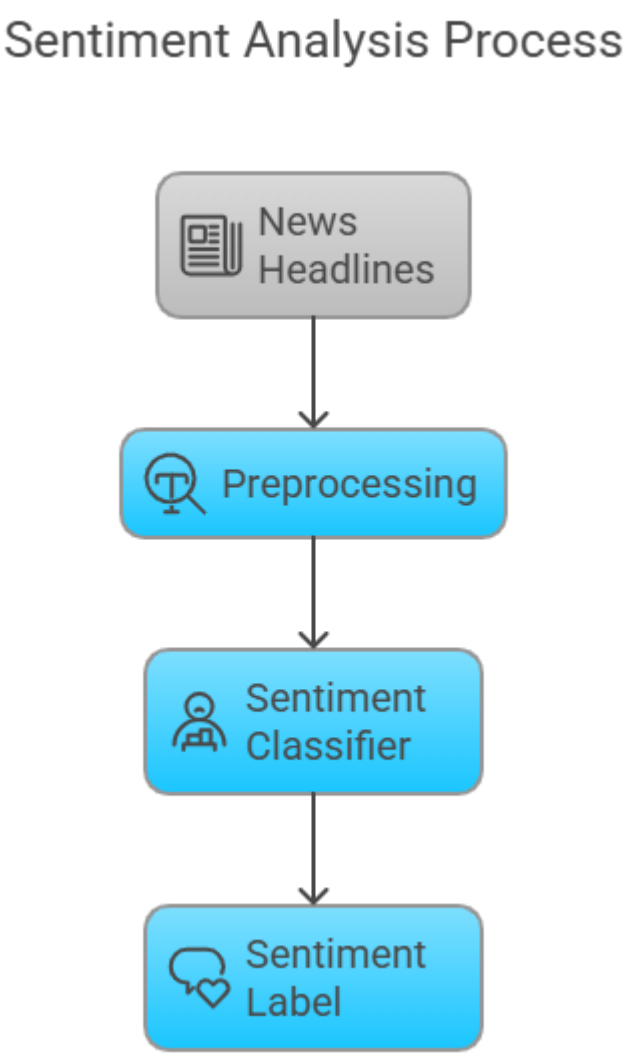


Figure 6.2.2 Sentimental Analysis Process

The sentiment analysis engine was architected to handle large volumes of unstructured financial text. A dual-layered approach was followed. First, a pre-processing layer cleaned and tokenized the text. Then, a classification layer used either VADER (for rule-based analysis) or a fine-tuned BERT model (for deep learning-based sentiment classification). The classification output was a probability score for positive, neutral, or negative sentiment.

This sentiment score was integrated into the overall decision-making pipeline. For instance, if a stock's technical indicators are bullish but the sentiment is largely negative, the final prediction score would reflect this uncertainty, thus improving prediction accuracy.

6.2.3 Modular App Design

StockRaj was developed following a modular and scalable architecture to facilitate maintenance, future upgrades, and reusability. The system is composed of three main layers: the frontend, the backend, and the ML/NLP model layer.

- **Frontend**: Built using React, this layer provides a responsive user interface with real-time chart updates, interactive forms, and chatbot functionality.
- **Backend**: Developed using Flask, it serves as the intermediary between the frontend and the ML models. It includes RESTful APIs for fetching stock data, computing indicators, generating predictions, and processing sentiment.
- **Model Layer**: This layer consists of pre-trained and fine-tuned machine learning and NLP models. Models are loaded dynamically to reduce memory usage and are exposed as API endpoints for real-time interaction.
- **Data Layer**: Data is fetched using APIs (Yahoo Finance, NewsAPI) and is processed on the fly or stored temporarily for computation caching.

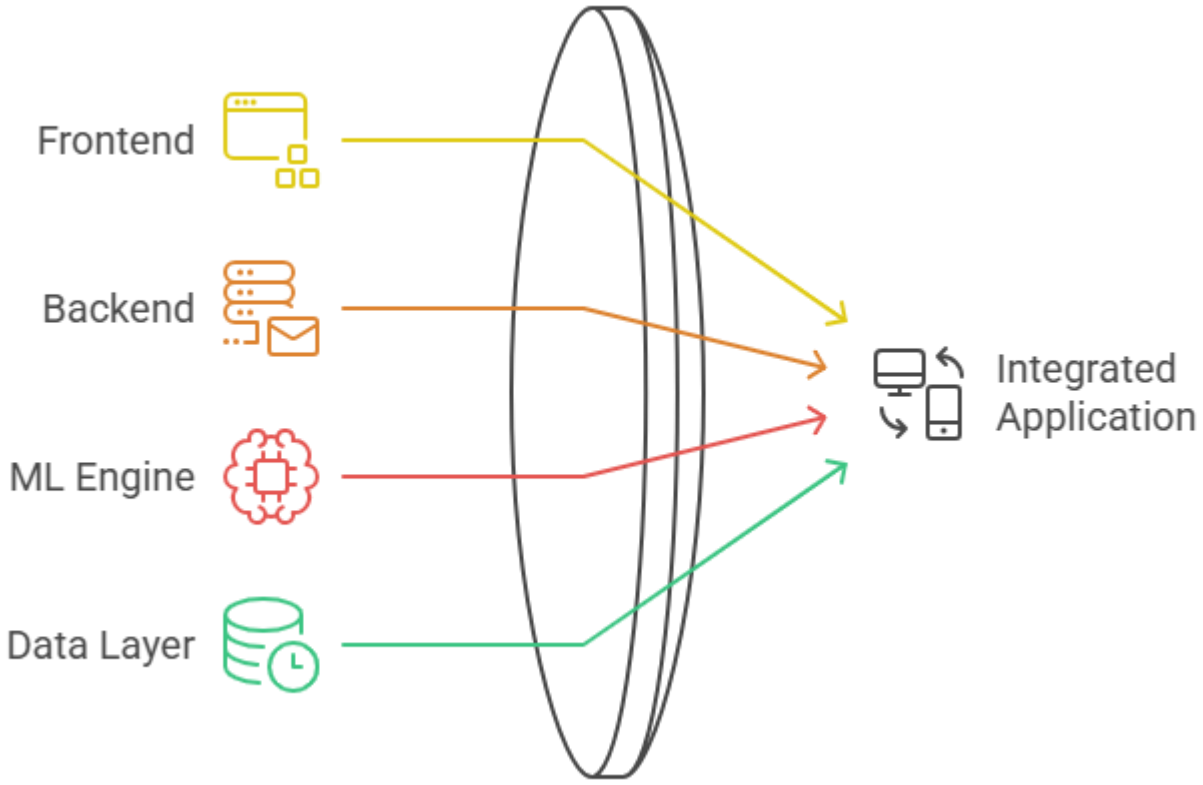


Figure 6.2.3 App Design

CHAPTER 7
SYSTEM DESIGN

SYSTEM DESIGN

System design is a pivotal phase in software development, where the blueprint for constructing a robust, scalable, and efficient system is crafted. It encompasses a holistic approach to defining the architecture, components, modules, and interactions that form the foundation of the software solution. At its core, system design aims to translate the specified requirements into a coherent and structured system architecture that meets the functional and non-functional needs of stakeholders. By taking into account various factors such as scalability, reliability, performance, security, and usability, system design ensures that the resultant software solution aligns closely with the goals and objectives of the project.

Central to system design is the concept of architecture, which serves as the structural framework defining the organization and arrangement of system components. The architecture delineates high-level views of the system, including layers, modules, and subsystems, and specifies how these elements interact with each other to fulfill the system's objectives. Through systematic decomposition and abstraction, system architecture establishes clear boundaries and interfaces, promoting modularity, maintainability, and extensibility. Additionally, it facilitates the distribution of responsibilities, enabling effective collaboration among development teams and streamlining the development process.

Ultimately, system design is an iterative and collaborative endeavor, where continuous refinement and adaptation are essential to ensure that the final software solution meets the evolving needs and expectations of its users and stakeholders.

System design typically involves several key activities:

1. System Architecture: Defining the structural framework and organization of system components.
2. Data Flow Diagrams: Visualizing the flow of data within the system and between its components.
3. Use-case Diagrams: Illustrating the interactions between users and the system to accomplish specific tasks or goals.
4. Class Diagrams: Modeling the structure and relationships of classes or objects within the system.

7.1 SYSTEM ARCHITECTURE

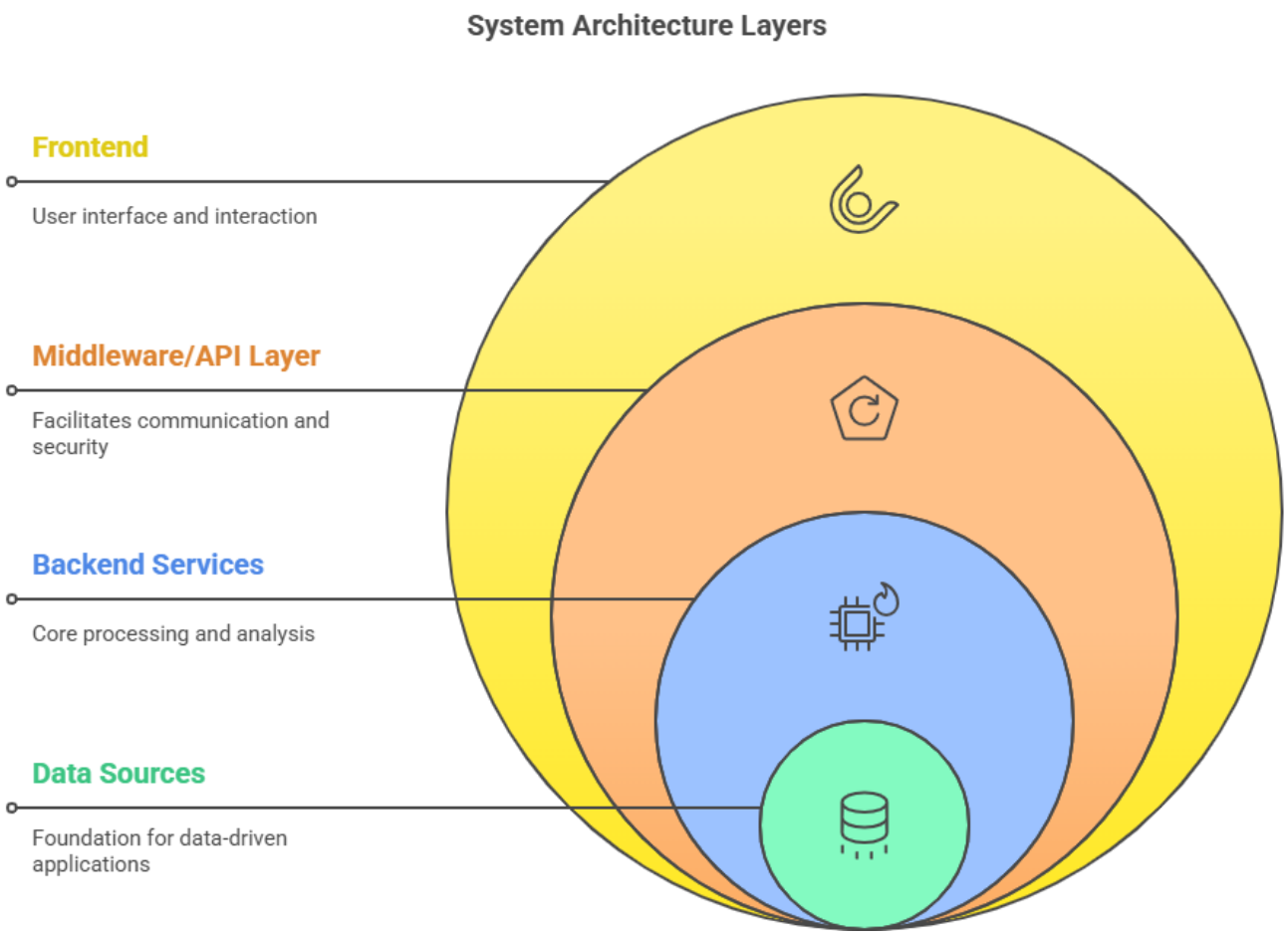


Figure 7.1 System Architecture

System architecture refers to the structural design of a software system, encompassing the arrangement and interaction of its components and subsystems. It defines the overall framework that governs how the system's elements work together to achieve its objectives. This architectural blueprint outlines the distribution of responsibilities, communication pathways, and data flow within the system, providing a high-level view of its organization. By establishing clear interfaces, defining relationships between components, and addressing key requirements such as scalability, reliability, and maintainability, system architecture ensures the coherence and effectiveness of the software solution. In essence, system architecture serves as the foundation upon which the entire software system is built, guiding its development, evolution, and operation.

7.1.1 Data Acquisition and Preprocessing

Data acquisition in StockRaj primarily relies on the Yahoo Finance API, which is queried using scheduled background jobs and user-triggered events. This data includes a variety of metrics such as opening/closing prices, trading volume, market cap, and various technical indicators like RSI and MACD. The system also scrapes financial headlines and summaries from news portals for sentiment analysis.

Preprocessing involves cleaning and formatting the incoming data to ensure consistency and usability. Missing values are handled using interpolation or imputation techniques. Categorical text such as company names is standardized, and timestamps are converted to the application's time zone. For ML inputs, numerical normalization and feature engineering (e.g., moving averages) are applied. The cleaned and pre-processed dataset is then stored or temporarily cached for fast access by different system modules.

7.1.2 Machine Learning and Prediction Module

The ML module is a core component of StockRaj, responsible for forecasting stock prices using historical trends and patterns. Models such as LSTM (Long Short-Term Memory) or Random Forests are trained on features like price movements, volume spikes, and technical indicators. These models are trained offline on a rolling window of data and periodically updated to account for new trends.

During inference, the model receives input from the preprocessed real-time data and outputs predicted price points or trend signals (buy/sell/hold). These predictions are made available to the frontend through API endpoints and are shown in graphical formats such as candlestick charts or trend curves. This predictive intelligence enhances user decision-making by providing insight into potential future market movements.

7.1.3 Sentiment Analysis and NLP Module

This module enhances the decision-making process by offering qualitative insights into market trends. It uses web scraping techniques to gather news headlines, summaries, and tweets related to major stocks and economic events. These text snippets are passed through an NLP pipeline comprising tokenization, stop-word removal, TF-IDF vectorization or transformer-based embeddings, followed by sentiment classification.

The classifier categorizes sentiment into positive, negative, or neutral, which is then aggregated and visualized as sentiment scores for each stock. These scores are updated frequently and can help users gauge the public perception of a company or market trend. The sentiment insights are integrated into the recommendation system and shown alongside the technical and ML-based data.

7.1.4 Frontend Integration and Visualization

The frontend of StockRaj acts as the user-facing component that presents all processed data in a user-friendly and intuitive manner. It is built using modern web technologies like React.js or Vue.js, with support for responsive design, ensuring usability across devices. Charts and indicators are powered by libraries like Chart.js or D3.js, allowing real-time interaction with stock data.

Users can navigate through different tabs to view real-time charts, sentiment scores, prediction outcomes, and educational material. Filters allow customization by stock ticker, industry, or time range. The frontend fetches data via REST APIs and displays it in cards, graphs, and tables, offering a smooth and dynamic user experience. Performance metrics and recommendations are clearly highlighted to facilitate actionable insights.

7.1.5 AI Chatbot Integration

The chatbot is an intelligent assistant integrated into the frontend. It allows users to interact conversationally with the system to get stock updates, learn about technical terms, or request summaries. It uses a combination of keyword-based rules and ML-based intent classification to understand user queries and fetch relevant data.

The backend logic processes queries such as “What is the current price of Tesla?” or “Explain RSI,” fetching the required information from stored APIs or documentation. The chatbot aims to provide a hands-on, interactive learning experience for beginner investors while also being useful to advanced users seeking quick insights.

7.1.6 Deployment and Hosting

The StockRaj application is deployed using cloud platforms like Heroku, AWS, or GCP. The backend server is containerized using Docker to ensure scalability and environment consistency. CI/CD pipelines automate deployment and testing. Static frontend assets are hosted via a CDN or service like Netlify, ensuring fast page loads and global access.

Environment variables and API keys are securely stored using environment configuration files. The system is monitored for performance, errors, and uptime using tools like Prometheus, Grafana, or New Relic. This production-ready setup ensures minimal downtime and a seamless user experience across all system modules.

7.2 DATA FLOW DIAGRAM

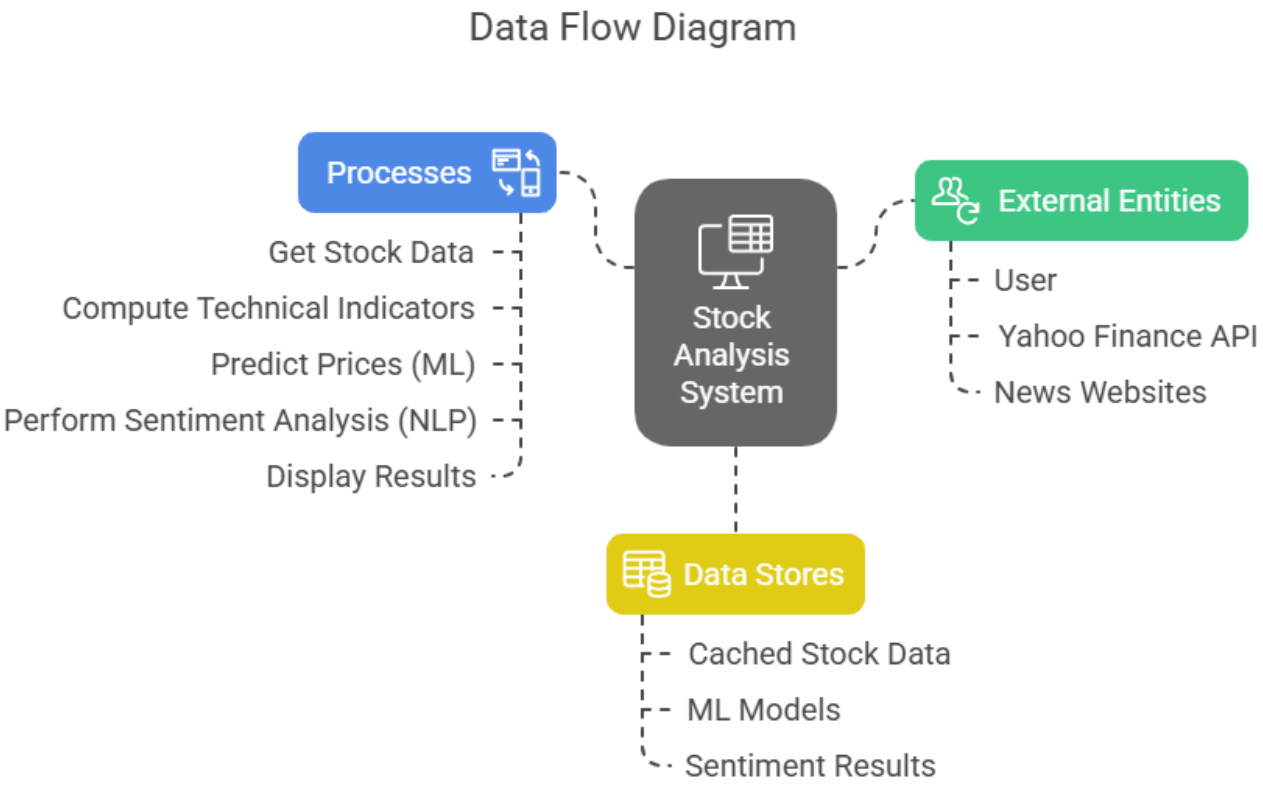


Figure 7.2 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation that illustrates the flow of data within a system. It provides a visual depiction of how data moves between processes, stores, and external entities in a system. The diagram consists of various components such as processes (functions or transformations), data stores (repositories where data is stored), data flows (routes along which data travels), and external entities (sources or destinations of data). By showing the interactions and dependencies between these components, DFDs help in understanding the system's architecture, identifying data inputs and outputs, and analyzing the overall data flow within the system. They are commonly used in system analysis and design to model and document data processing systems.

The Data Flow Diagram (DFD) for StockRaj represents how data moves throughout the system, from initial user input to final output. It shows the interaction between external entities (users, APIs), internal processes (data processing, machine learning models), and data stores.

In the Level 0 DFD, the user initiates interaction by querying stock data or predictions through the web interface. This input is sent to the backend where it is handled by the request processor. The processor interacts with two main external systems: the Yahoo Finance API (for market data) and the web scraper module (for news and sentiment data). This data is processed and stored temporarily in an in-memory cache or local database.

At Level 1, deeper layers show how the ML model consumes historical stock data and technical indicators to generate predictions. Simultaneously, the sentiment analysis engine receives financial news, applies preprocessing and NLP techniques, and stores the final sentiment scores. Both sets of processed data are then relayed to the frontend for visualization. Data moves efficiently between modules using REST APIs, ensuring quick response times and seamless user interaction.
This systematic flow ensures modularity, enabling components to be updated or replaced independently without affecting the overall structure.

7.3 USE-CASE DIAGRAM

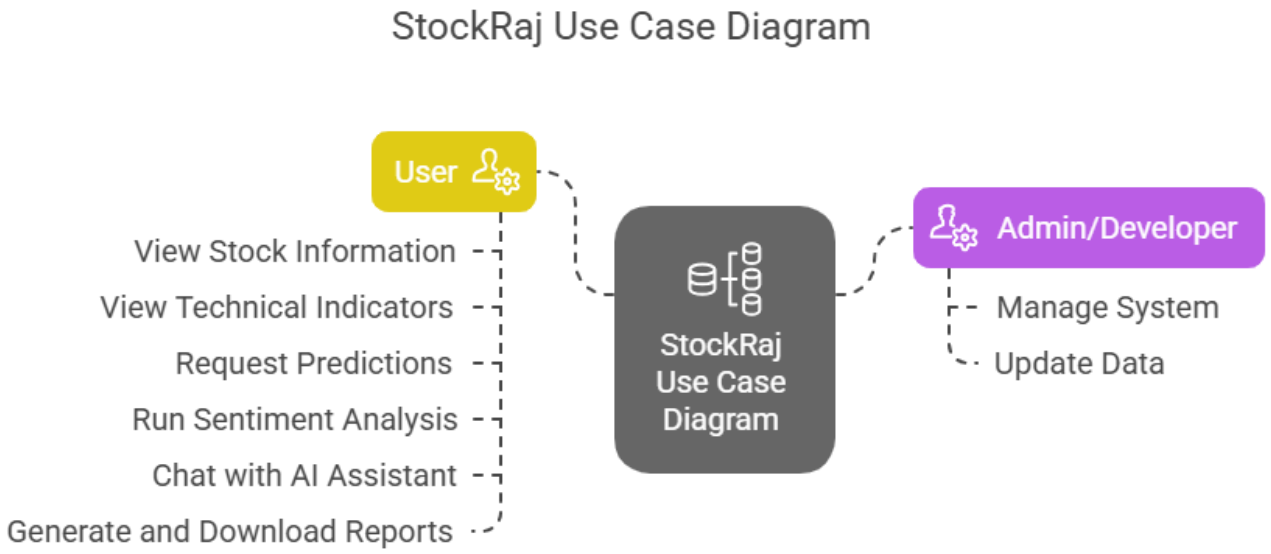


Figure 7.3 Use- Case Diagram

The Use Case Diagram captures the interaction between different types of users and system functionalities in StockRaj. The primary actor in the system is the user, who can be either a general investor or a financial enthusiast. The diagram outlines various use cases such as:

- View Stock Data: Users can input a stock ticker to view real-time and historical market data.
- View Technical Indicators: Users can access charts showing indicators like RSI, MACD, SMA, etc.
- Get Price Predictions: The system provides machine learning–based predictions for selected stocks.
- Perform Sentiment Analysis: Users can explore public sentiment scores for specific stocks.
- Interact with Chatbot: The AI assistant helps users understand financial terms, get quick stock overviews, and answer FAQ-type queries.
- Generate Reports: Users can download or view summarized insights and stock recommendations.

Each use case connects the user to a backend module that handles logic, data access, or visualization. Admins or developers (as secondary actors) may have access to model training, API management, and performance monitoring tools, though these are not part of the regular user flow.

This diagram ensures that the system design aligns closely with the user goals and functionality requirements.

7.4 CLASS DIAGRAM

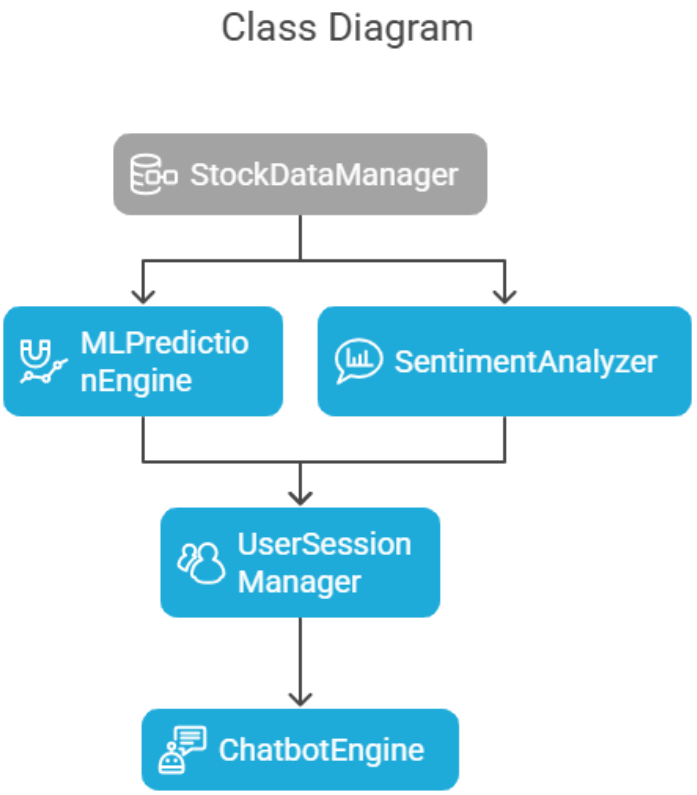


Figure 7.4 Class Diagram

A class diagram is a visual representation in Unified Modeling Language (UML) that depicts the structure and relationships of classes within a software system. Classes are shown with their attributes (properties) and methods (behaviors), and associations illustrate how classes are connected. Multiplicity indicates the number of instances related in an association, while inheritance signifies the "is-a" relationship between classes. Dependencies show class interactions, and overall, class diagrams serve as blueprints for understanding and designing software systems, aiding communication and analysis among stakeholders.

- The Class Diagram presents the object-oriented structure of key backend components and their relationships in the StockRaj application. Major classes and their interactions include:
- **Stock Data Manager**: Manages the retrieval and storage of historical and real-time stock data. Attributes include ticker symbol, data range, and data frequency. It has methods like fetch_data(), format_data(), and cache_data().
- **ML Prediction Engine**: Responsible for training and making predictions. Contains model parameters and methods such as train_model(), predict_price(), and evaluate_performance().
- **Sentiment Analyzer**: Handles NLP-based sentiment scoring. Key attributes are sentiment model, vectorizer, and data source URL. Methods include scrape_news(), clean_text(), analyze_sentiment().
- **User Session Manager**: Tracks active users and manages session-related data. Includes login tokens and query history.
- **Chatbot Engine**: Interfaces with the AI chatbot. Has intent-detection models, a knowledge base, and methods like respond_to_query() and get_faq_answer().
- Relationships are clearly defined: for example, StockDataManager feeds data to both MLPredictionEngine and SentimentAnalyzer, while ChatbotEngine depends on UserSessionManager to personalize responses.
- This class-based structure ensures code reusability, encapsulation, and easier debugging in large-scale development.

7.5 WEBSITE ARCHITECTURE

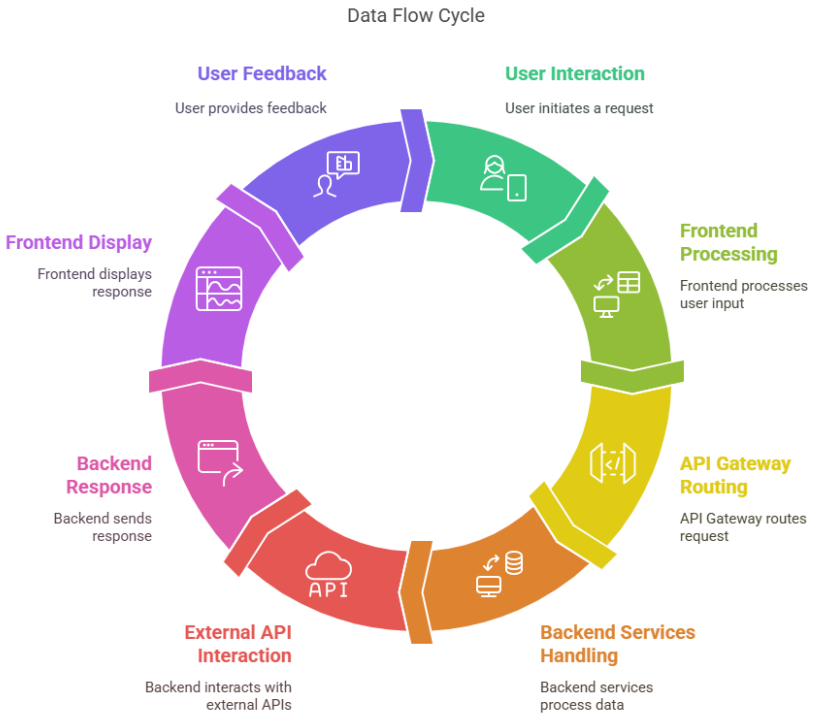


Figure 7.5 Data Flow in Website

7.5.1 Frontend (Client-Side)

The frontend is built using **React** with **TypeScript**, bundled through **Vite** for fast performance and hot module replacement. It is a **progressive web application (PWA)** with a responsive layout optimized for both desktop and mobile platforms.

Core responsibilities:

- Rendering dynamic stock visualizations using **Recharts**, **react-financial-charts**, and **Plotly** (candlestick, area, baseline, line).
- Handling user interactions for:
  - **Watchlist management** (symbol search, add/delete).
  - **Portfolio tracking** (transactions, holdings, analytics).
  - **Sentiment and technical analysis**.
  - **Chatbot interactions** for financial queries.
- Displaying structured information on multiple dashboards like:
  - **Dashboard**, **Market Activity**, **Watchlist**, **Portfolio**, and **AI Analysis**.
- API integration through REST calls to either direct data sources (Yahoo Finance) or a proxy server for security and performance.

7.5.2 Backend Services (API Layer)

Backend interactions are either processed **client-side** via direct API calls to **Yahoo Finance**, or proxied through lightweight **serverless functions** (using Node.js or Python via Flask/FastAPI) to hide API keys and control rate-limiting.

Backend services handle:

- Fetching and parsing **real-time and historical stock data**, market indices, financial metrics, and technical indicators.
- Scraping **Indian financial news RSS feeds** from Bloomberg Quint, Economic Times, MoneyControl, and Google News.
- Preprocessing data for visual rendering (JSON, CSV, or Plot-ready arrays).

7.5.3 Machine Learning & Sentiment Analysis

ML modules are designed and trained offline using **Python-based LSTM models** on historical stock data. They are deployed using lightweight Python backends and return predictions and insights as APIs.

ML Layer supports:

- Price prediction (forecasting trends for 1D, 1W, 1M, etc.).
- Buy/Sell signal generation based on trained patterns.
- Anomaly and pattern recognition.
- Sentiment analysis using **TextBlob/VADER** on scraped financial news to determine bullish/bearish market sentiment.

Predictions are returned in real-time or near real-time to be rendered alongside live market data in charts and summary tables.

7.5.4 AI Chatbot Integration

The intelligent chatbot is designed using:

- **GPT API** (for contextual answers and financial literacy).
- Rule-based response system for structured stock-related queries.

The chatbot is capable of:

- Guiding users through different modules.
- Answering FAQs about stock performance, terms, and predictions.
- Providing actionable insights via AI-summarized outputs.

CHAPTER 8
IMPLEMENTATION

This chapter outlines the technical implementation details of each core feature in the project. Emphasis was placed on building a modular, maintainable codebase while ensuring efficient user interaction through a high-performance web interface.

8.1 CODE STRUCTURE

A clean and scalable folder structure was adopted to ensure maintainability and ease of navigation across components. The codebase was organized as follows:

- /components: Reusable UI components such as charts, tables, and cards.
- /pages: Route-specific components for portfolio, prediction, news, etc.
- /services: API logic and model integration.
- /hooks: Custom React hooks for modular logic.
- /utils: Utility functions for data formatting, sentiment analysis, and chart transformations.

This structure supported efficient development and future scalability.

```
import React, { useState, useEffect } from 'react';
import Navigation from './components/Navigation';
import { Dashboard } from './pages/Dashboard';
import { MarketActivity } from './pages/MarketActivity';
import { Watchlist } from './pages/Watchlist';
import { Portfolio } from './pages/Portfolio';
import { AIAnalysis } from './pages/AIAnalysis';
import { Chatbot } from './pages/Chatbot';
import { Settings } from './pages/Settings';
import { Footer } from './components/Footer/Footer';
import { AuthProvider } from './context/AuthContext';
import { ThemeProvider } from './context/ThemeContext';
import { DataProvider } from './context/DataContext.tsx';
import './styles/glassmorphism.css';

const App: React.FC = () => {
  const [currentPage, setCurrentPage] = useState('dashboard');
  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  // Add scroll to top effect when page changes
  useEffect(() => {
    window.scrollTo(0, 0);
  }, [currentPage]);
```

```
  const renderPage = () => {
    switch (currentPage) {
      case 'dashboard':
        return <Dashboard />;
      case 'market':
        return <MarketActivity />;
      case 'watchlist':
        return <Watchlist />;
      case 'portfolio':
        return <Portfolio />;
      case 'analysis':
        return <AIAnalysis />;
      case 'chatbot':
        return <Chatbot />;
      case 'settings':
        return <Settings />;
      default:
        return <Dashboard />;
    }
  };
```

```
  return (
    <AuthProvider>
      <ThemeProvider>
        <DataProvider>
          <div className="min-h-screen bg-gradient-to-br from-primary to-secondary">
            <div className="lg:hidden">
              <button
                type="button"
                aria-label="Toggle mobile menu"
                onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}
                className="fixed top-4 left-4 z-50 glass-button p-2 rounded-lg"
              >
                <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                  <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
                        d={isMobileMenuOpen ? "M6 18L18 6M6 6l12 12" : "M4 6h16M4 12h16M4 18h16"} />
                </svg>
              </button>
            </div>

            <Navigation
              onPageChange={setCurrentPage}
              isMobileMenuOpen={isMobileMenuOpen}
              onMobileMenuClose={() => setIsMobileMenuOpen(false)}
            />

            <div className="lg:pl-64 transition-all duration-300">
              <main className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
                {renderPage()}
              </main>
              <Footer />
            </div>
          </div>
        </DataProvider>
      </ThemeProvider>
    </AuthProvider>
  );
};

export default App;
```

8.2 VISUALIZATION IMPLEMENTATION

To create intuitive and interactive visual representations:

- Recharts was used for rendering responsive charts such as bar graphs, pie charts, and line plots.
- react-financial-charts enabled precise candlestick and OHLC charting to visualize stock price movements and trends.
- Dynamic rendering ensured charts updated in real-time based on user selection, time frames, or predictions.

These tools collectively improved data accessibility and enhanced the user experience through clarity and interactivity.

```
import React, { useEffect, useRef, useState } from 'react';
import {
  createChart,
  ColorType,
  IChartApi,
  ISeriesApi,
  CandlestickSeries,
  LineSeries,
  AreaSeries,
  BaselineSeries,
  HistogramSeries,
  Time
} from 'lightweight-charts';
import { ArrowUp, ArrowDown, ZoomIn, ZoomOut, Move, RotateCcw, AlertTriangle, LineChart, BarChart2,
CandlestickChart } from 'lucide-react';
```

```
interface ChartData {
  timestamp: number[];
  open: number[];
  high: number[];
  low: number[];
  close: number[];
  volume: number[];
}

interface StockChartProps {
  symbol: string;
  data: {
    price: number;
    change: number;
    changePercent: number;
  };
  onRealtimeUpdate?: (data: {
    price: number;
    change: number;
    changePercent: number;
    symbol: string;
  }) => void;
  onHistoricalData?: (historicalData: ChartData | null) => void;
}

type AdvancedChartType = 'line' | 'area' | 'baseline' | 'candles';
```

```
interface YahooFinanceData {
  timestamp: number;
  date: string;
  open: number;
  high: number;
  low: number;
  close: number;
  volume: number;
  adjustedClose?: number;
}
```

```
const TradingViewChart: React.FC<StockChartProps> = ({ symbol, data, onRealtimeUpdate, onHistoricalData }) => {
  const chartContainerRef = useRef<HTMLDivElement>(null);
  const [chart, setChart] = useState<IChartApi | null>(null);
  const [series, setSeries] = useState<ISeriesApi<"Candlestick" | "Line" | "Area" | "Baseline"> | null>(null);
  const [historicalData, setHistoricalData] = useState<ChartData | null>(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [timeframe, setTimeframe] = useState('1D');
  const [chartType, setChartType] = useState<AdvancedChartType>('candles');
  const [currentPrice, setCurrentPrice] = useState(data.price);
  const [currentChange, setCurrentChange] = useState(data.change);
  const [currentChangePercent, setCurrentChangePercent] = useState(data.changePercent);
  const [isPanning, setIsPanning] = useState(false);
  const isPositive = currentChange >= 0;
  const [cachedData, setCachedData] = useState<{ [key: string]: { [timeframe: string]: ChartData } }>({});
  const [allSeries, setAllSeries] = useState<ISeriesApi<"Candlestick" | "Line" | "Area" | "Baseline" | "Histogram">[]>([]);

  const timeframes = [
    { id: '1D', label: '1D', interval: '1m', range: '1d' },
    { id: '1W', label: '1W', interval: '1d', range: '7d' },
    { id: '1M', label: '1M', interval: '1d', range: '1mo' },
    { id: '3M', label: '3M', interval: '1d', range: '3mo' },
    { id: '1Y', label: '1Y', interval: '1wk', range: '1y' },
    { id: 'ALL', label: 'ALL', interval: '1mo', range: 'max' }
  ];

  const chartTypes = [
    {
      id: 'line',
      label: 'Line',
      icon: <LineChart className="w-4 h-4" />,
      description: 'Simple line chart showing price movement'
    },
    {
      id: 'area',
      label: 'Area',
      icon: <BarChart2 className="w-4 h-4" />,
      description: 'Area chart with gradient fill below the line'
    },
    {
      id: 'baseline',
      label: 'Baseline',
      icon: <LineChart className="w-4 h-4" />,
      description: 'Chart with baseline reference showing relative performance'
    },
    {
      id: 'candles',
      label: 'Candles',
      icon: <CandlestickChart className="w-4 h-4" />,
      description: 'Candlestick chart showing OHLC data'
    }
  ];
```

8.3 PRICE PREDICTION MODEL

A Long Short-Term Memory (LSTM) model was implemented to forecast stock prices:

- Dataset: Historical stock data including Open, High, Low, Close, and Volume.
- Preprocessing: Normalization using MinMaxScaler and time-series windowing.
- Model Architecture:
  • Layers: LSTM → Dense → Dropout

  • Loss Function: Mean Squared Error
  • Optimizer: Adam
- The trained model predicts short-term price trends and returns results visualized alongside actual stock performance for comparison.

The LSTM model was integrated with the frontend to provide real-time user feedback.

```
const express = require('express');
const cors = require('cors');
const axios = require('axios');
const cheerio = require('cheerio');

const app = express();
const PORT = 3001;
```

```
// Configure CORS with multiple origins and additional headers
app.use(cors({
  origin: [
    'http://localhost:5173',  // Vite default port
    'http://localhost:3000',  // Common React port
    'http://localhost:8080',  // Common development port
    'http://127.0.0.1:5173',  // Alternative localhost
    'http://127.0.0.1:3000',  // Alternative localhost
    'http://127.0.0.1:8080'   // Alternative localhost
  ],
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
  allowedHeaders: [
    'Content-Type',
    'Authorization',
    'X-Requested-With',
    'Accept',
    'Origin',
    'Access-Control-Request-Method',
    'Access-Control-Request-Headers'
  ],
  credentials: true,
  maxAge: 86400, // 24 hours
  preflightContinue: false,
  optionsSuccessStatus: 204
}));
```

```
// Log all requests
app.use((req, res, next) => {
  console.log(`${new Date().toISOString()} - ${req.method} ${req.url}`);
  next();
});

// Root route
app.get('/', (req, res) => {
  res.json({
    message: 'Proxy server is running',
    endpoints: {
      marketIndices: '/api/market-indices',
      historicalData: '/api/market-indices/historical/:symbol',
      test: '/api/test'
    }
  });
});

// Market Indices Symbols
const MARKET_INDICES = {
  'NIFTY 50': '^NSEI',
  'SENSEX': '^BSESN',
  'NIFTY BANK': '^NSEBANK',
  'NIFTY 100': '^CNX100',
  'NIFTY MIDCAP 100': 'NIFTY_MIDCAP_100.NS',
  'NIFTY MICROCAP 250': 'NIFTY_MICROCAP250.NS',
  'NIFTY SMALLCAP 100': '^CNXSC',
  'NIFTY NEXT 50': '^NSMIDCP'
};
```

```
// NIFTY Indices to track
const NIFTY_INDICES = [
  // Broad Market Indices
  'NIFTY 50',
  'SENSEX',

  // Sector-specific Indices
  'NIFTY BANK',

  // Mid and Small-Cap Indices
  'NIFTY MIDCAP 100',
  'NIFTY SMALLCAP 100',
  'S&P BSE MIDCAP',
  'S&P BSE SMALLCAP',

  // Special Indices
  'NIFTY NEXT 50',
  'S&P BSE SENSEX NEXT 50',
  'NIFTY 100',
  'NIFTY 500'
];
```

8.4 SENTIMENT ANALYSIS AND CHATBOT

Sentiment Analysis:

- News headlines were processed using TextBlob and VADER.

- Sentiment scores (positive, neutral, negative) were used to indicate market sentiment around specific stocks.

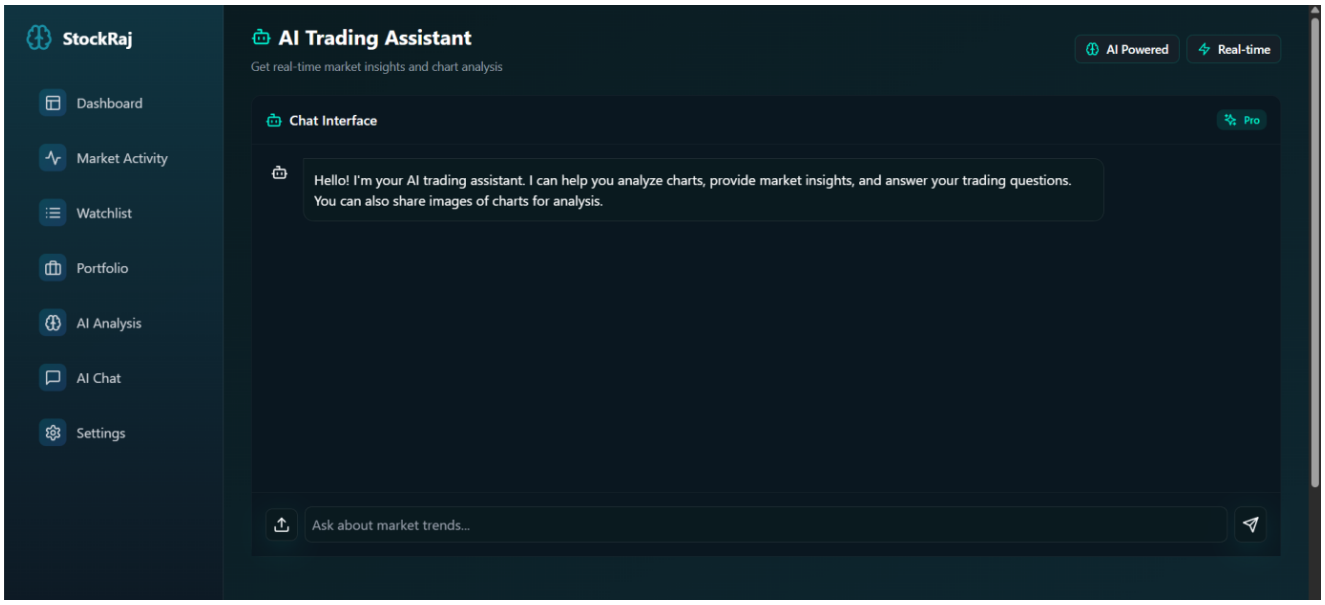
Custom Chatbot:

- A lightweight prompt-based AI chatbot was developed.
- Capable of answering questions about market sentiment, stock predictions, and user portfolio insights.
- Contextual prompts and keyword extraction were used to enhance relevance and response accuracy.

This component aimed to bridge technical insights and user queries effectively.

```tsx
import React from 'react';
import { Brain } from 'lucide-react';

interface SentimentData {
  news: { positive: number; negative: number; neutral: number };
  social: { positive: number; negative: number; neutral: number };
  overall: string;
}

interface SentimentAnalysisProps {
  data: SentimentData;
}

export const SentimentAnalysis: React.FC<SentimentAnalysisProps> = ({ data }) => {
  const renderSentimentBar = (positive: number, neutral: number, negative: number) => (
    <div className="h-2 w-full flex rounded-full overflow-hidden">
      <div className="bg-success" style={{ width: `${positive}%` }} />
      <div className="bg-warning" style={{ width: `${neutral}%` }} />
      <div className="bg-danger" style={{ width: `${negative}%` }} />
    </div>
  );

  return (
    <>
      <h3 className="text-lg font-semibold text-white flex items-center mb-4">
        <Brain className="w-5 h-5 mr-2 text-accent-primary" />
        Sentiment Analysis
      </h3>

      <div className="space-y-6">
        <div className="space-y-2">
          <div className="flex justify-between text-sm">
            <span className="text-gray-400">News Sentiment</span>
            <span className="text-white">{data.overall}</span>
          </div>
          {renderSentimentBar(data.news.positive, data.news.neutral, data.news.negative)}
          <div className="flex justify-between text-xs text-gray-400">
            <span>Positive {data.news.positive}%</span>
            <span>Neutral {data.news.neutral}%</span>
            <span>Negative {data.news.negative}%</span>
          </div>
        </div>
```

```
        <div className="space-y-2">
          <div className="flex justify-between text-sm">
            <span className="text-gray-400">Social Media Sentiment</span>
          </div>
          {renderSentimentBar(data.social.positive, data.social.neutral, data.social.negative)}
          <div className="flex justify-between text-xs text-gray-400">
            <span>Positive {data.social.positive}%</span>
            <span>Neutral {data.social.neutral}%</span>
            <span>Negative {data.social.negative}%</span>
          </div>
        </div>
      </div>
    </>
  );
};
```



8.5 PORTFOLIO AND WATCHLIST

To simulate real-time portfolio tracking and user personalization:

- A watchlist feature was implemented using localStorage, allowing users to add/remove stocks easily.
- A portfolio tracker simulated gains/losses based on mock transactions and live price updates.
- UI updates dynamically to reflect portfolio performance metrics such as total value, gain/loss percentage, and stock-wise distribution.

These features enhanced user engagement and mirrored real-world trading dashboards.

```
import React from 'react';
import { Briefcase, Plus, TrendingUp, TrendingDown } from 'lucide-react';
```

```
interface PortfolioStock {
  symbol: string;
  quantity: number;
  avgPrice: number;
  currentPrice: number;
  change: number;
}
```

```tsx
const portfolioData: PortfolioStock[] = [
  { symbol: 'RELIANCE', quantity: 100, avgPrice: 2400, currentPrice: 2456.75, change: 2.36 },
  { symbol: 'INFY', quantity: 50, avgPrice: 1500, currentPrice: 1456.80, change: -2.88 },
];

export const PortfolioSection: React.FC = () => {
  const calculateTotalValue = () => {
    return portfolioData.reduce((total, stock) => total + (stock.currentPrice * stock.quantity), 0);
  };

  const calculateTotalGain = () => {
    return portfolioData.reduce((total, stock) =>
      total + ((stock.currentPrice - stock.avgPrice) * stock.quantity), 0);
  };

  return (
    <div className="glass p-4 rounded-xl">
      <div className="flex justify-between items-center mb-4">
        <div className="flex items-center space-x-2">
          <Briefcase className="w-5 h-5 text-accent-primary" />
          <h3 className="text-lg font-semibold text-white">Your Portfolio</h3>
        </div>
        <button className="glass-button px-3 py-1 flex items-center space-x-1">
          <Plus className="w-4 h-4" />
          <span>New Portfolio</span>
        </button>
      </div>

      <div className="grid grid-cols-2 gap-4 mb-4">
        <div className="glass p-3 rounded-lg">
          <div className="text-sm text-gray-400">Total Value</div>
          <div className="text-lg font-bold text-white">
            ₹{calculateTotalValue().toLocaleString('en-IN')}
          </div>
        </div>
        <div className="glass p-3 rounded-lg">
          <div className="text-sm text-gray-400">Total Gain/Loss</div>
          <div className={`text-lg font-bold ${calculateTotalGain() >= 0 ? 'text-green-400' : 'text-red-400'}
`}>
            ₹{calculateTotalGain().toLocaleString('en-IN')}
          </div>
        </div>
      </div>

      <div className="space-y-3">
        {portfolioData.map((stock, index) => (
          <div key={index} className="glass p-3 rounded-lg">
            <div className="flex justify-between items-center">
              <div>
                <div className="text-white font-medium">{stock.symbol}</div>
                <div className="text-xs text-gray-400">{stock.quantity} shares</div>
              </div>
              <div className="text-right">
                <div className="text-white">₹{stock.currentPrice.toFixed(2)}</div>
                <div className={`flex items-center text-sm ${
                  stock.change >= 0 ? 'text-green-400' : 'text-red-400'
                }`}>
                  {stock.change >= 0 ? (
                    <TrendingUp className="w-3 h-3 mr-1" />
                  ) : (
                    <TrendingDown className="w-3 h-3 mr-1" />
                  )}
                  {stock.change}%
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};
```

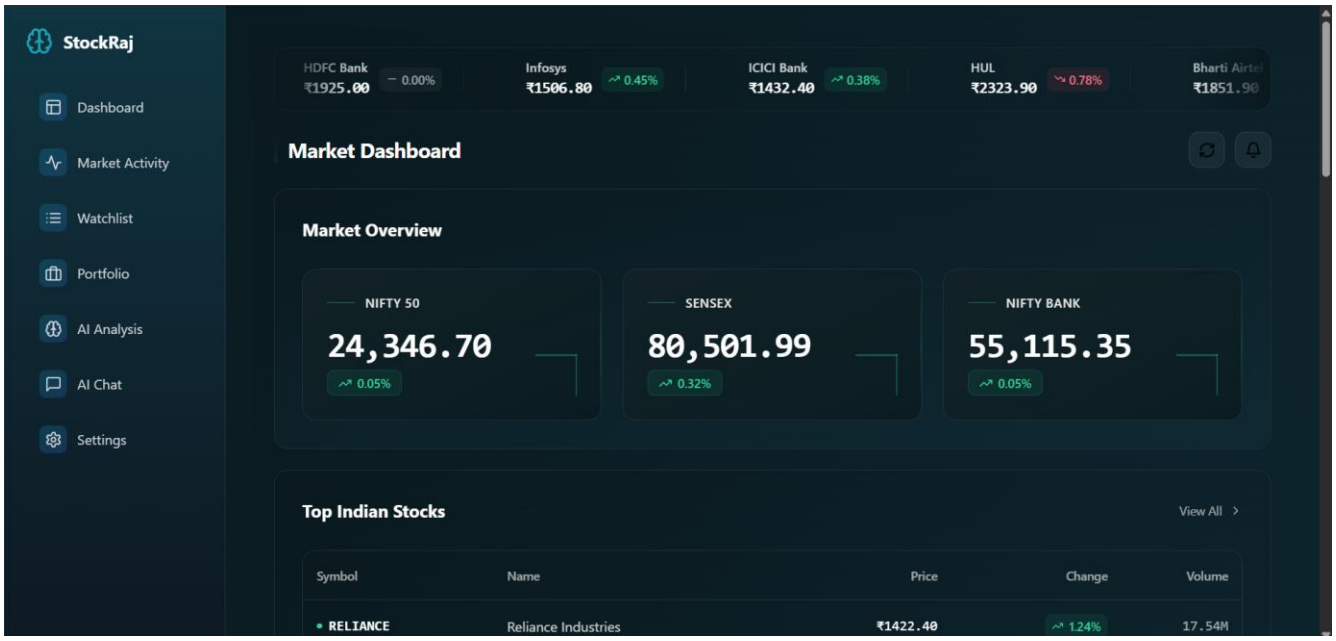
8.6 WEB TECHNOLOGIES

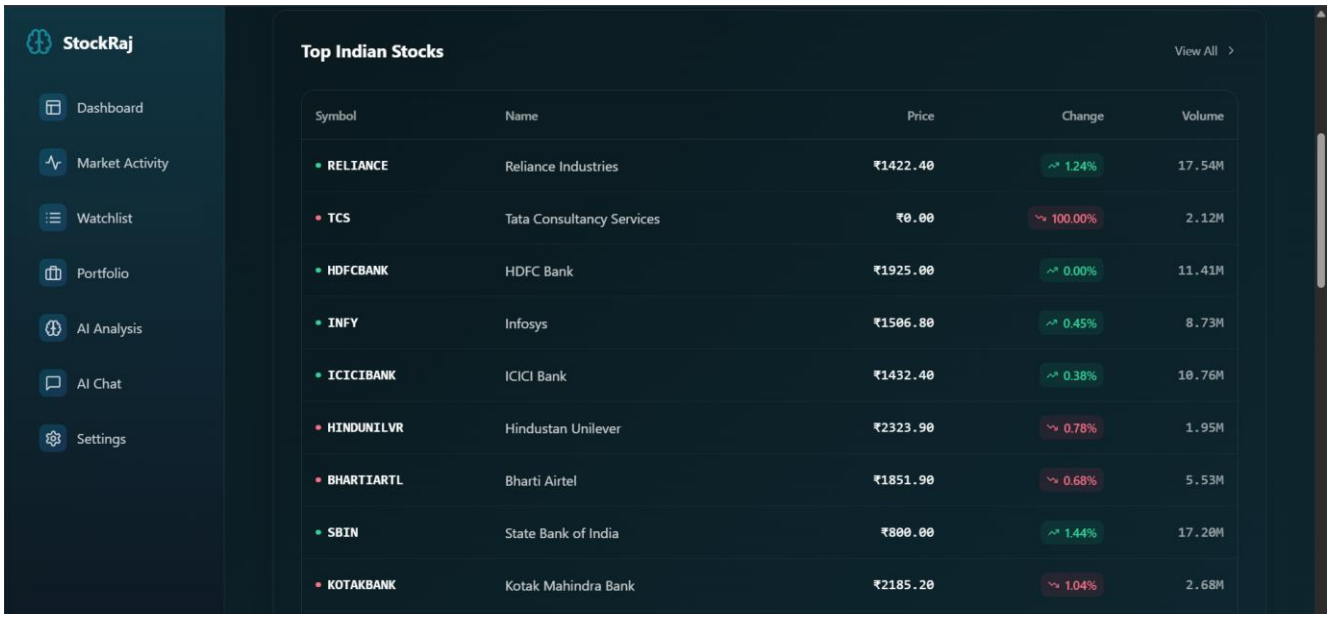
The application was developed using a modern web stack:

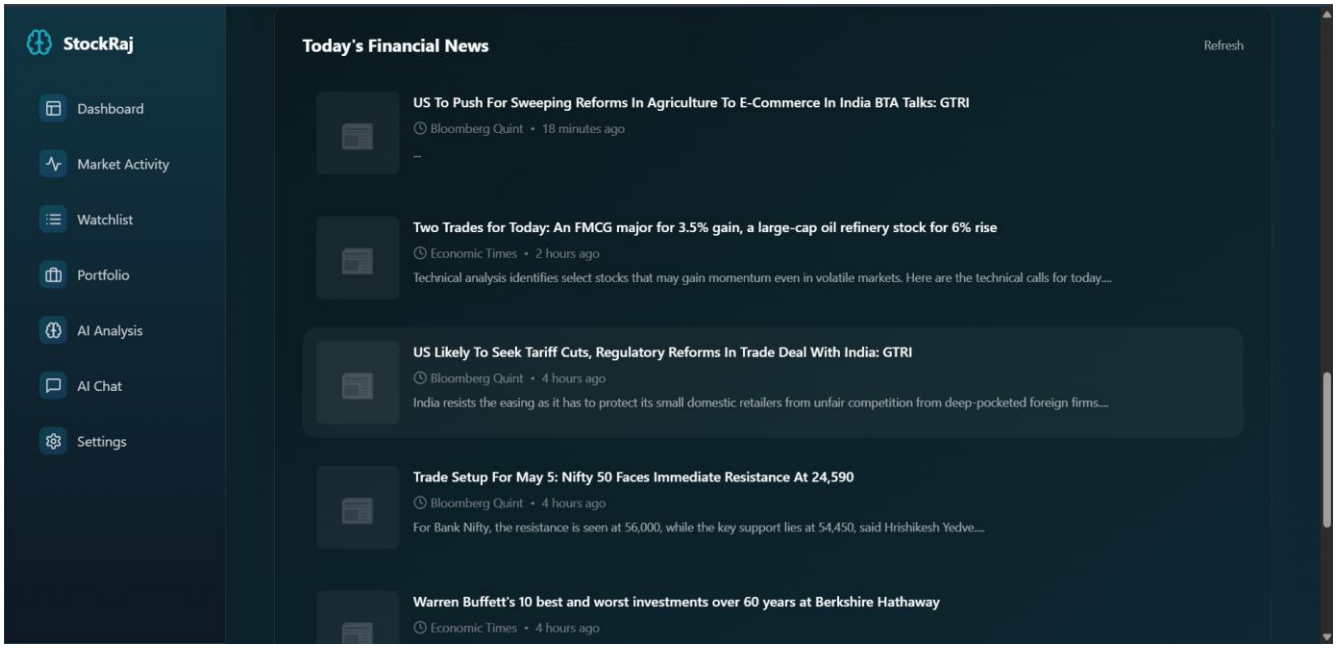
- React as the core library for UI development.
- Vite enabled ultra-fast build and hot-reloading for an efficient development workflow.

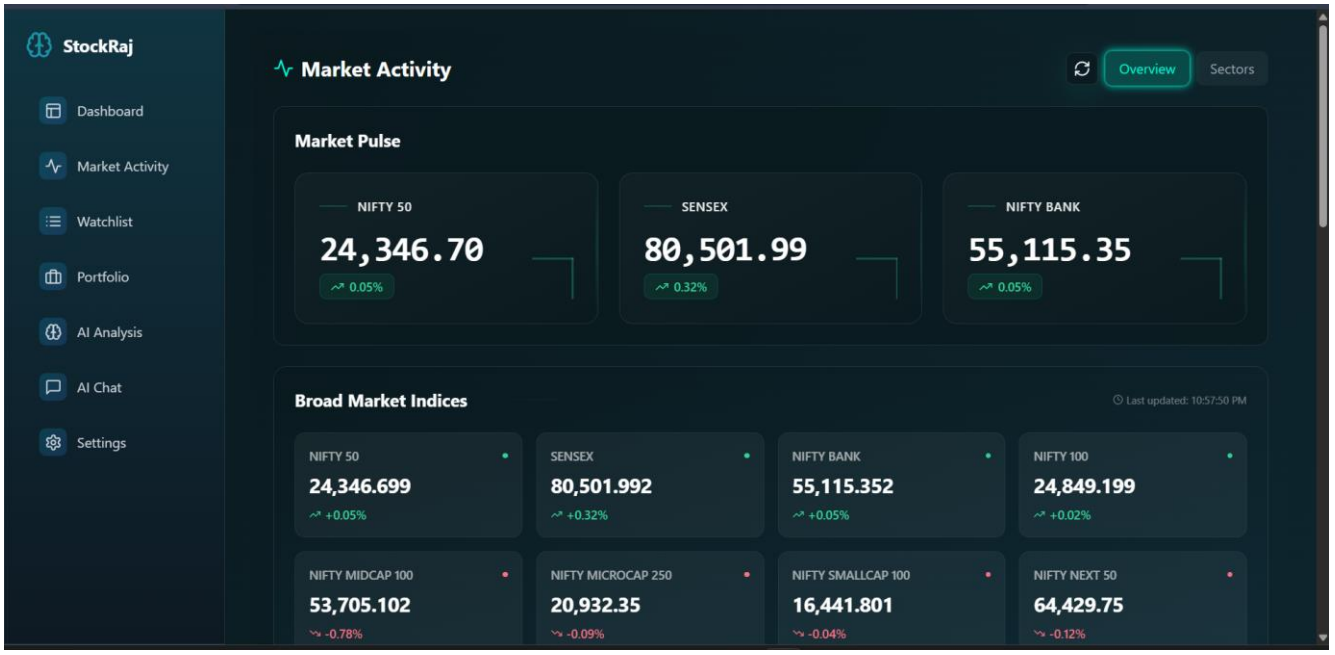
- TypeScript ensured type safety and reduced runtime errors.
- Tailwind CSS was used for responsive, utility-first styling.

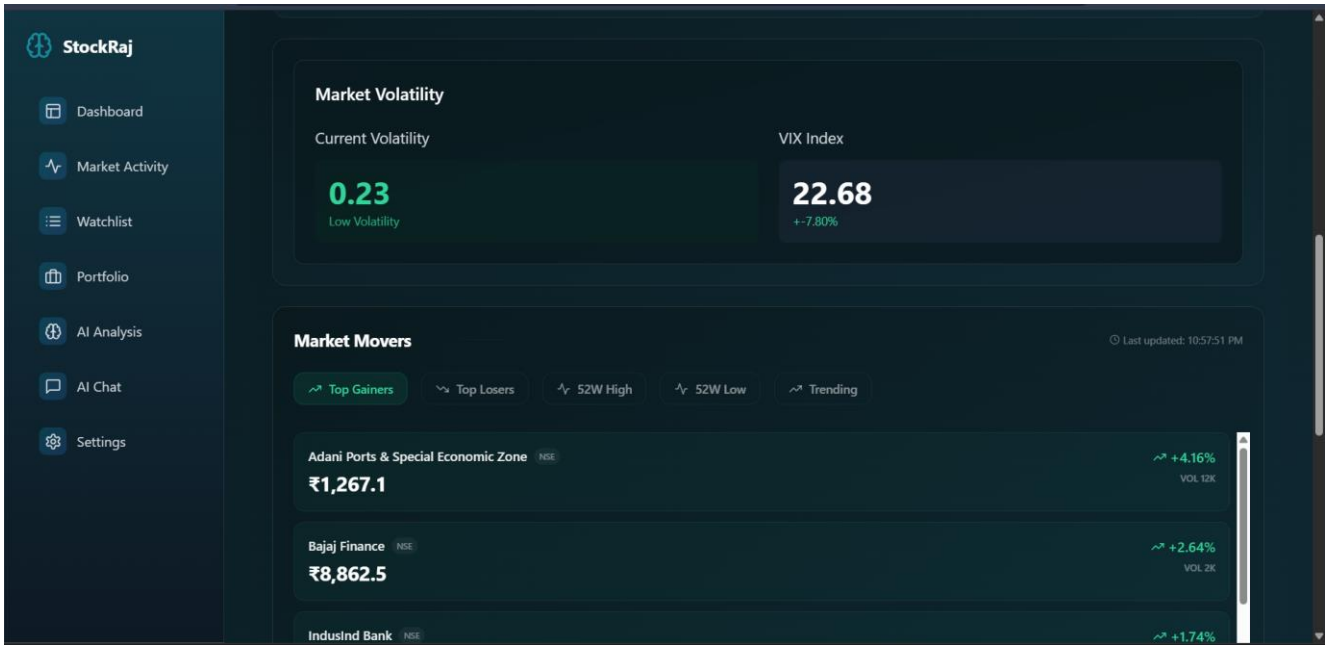
The combination of these technologies resulted in a high-performance, mobile-friendly interface with strong developer ergonomics.

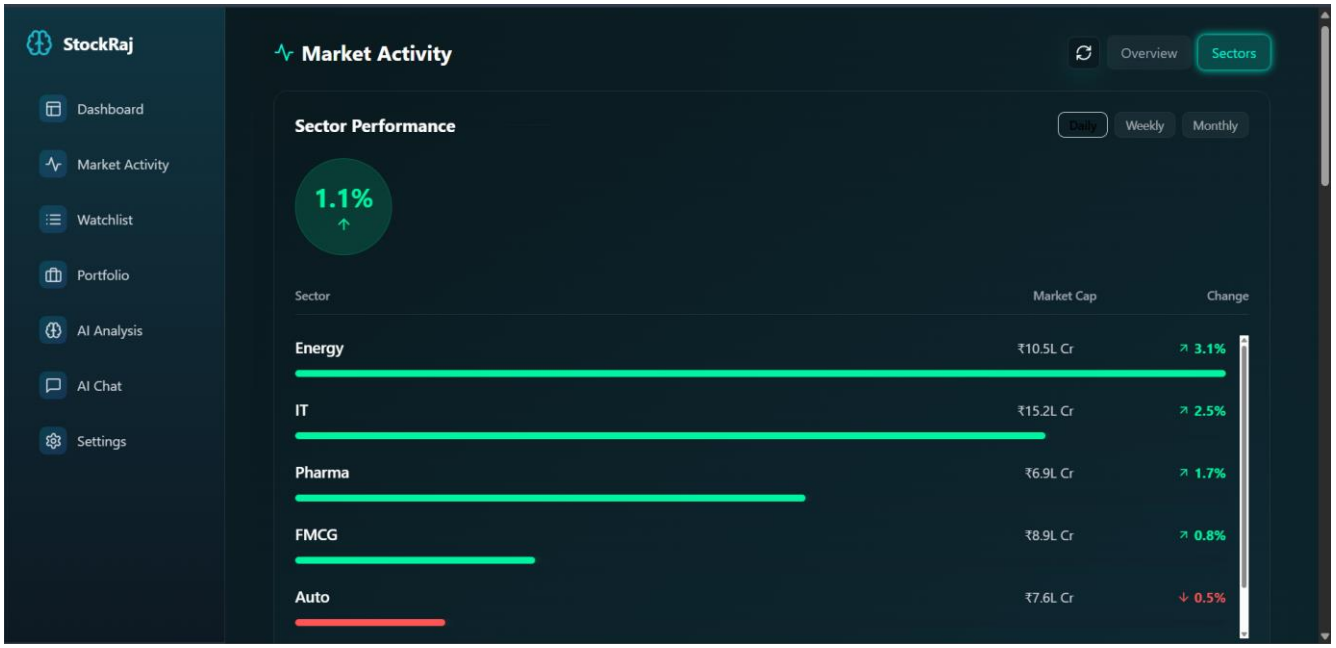


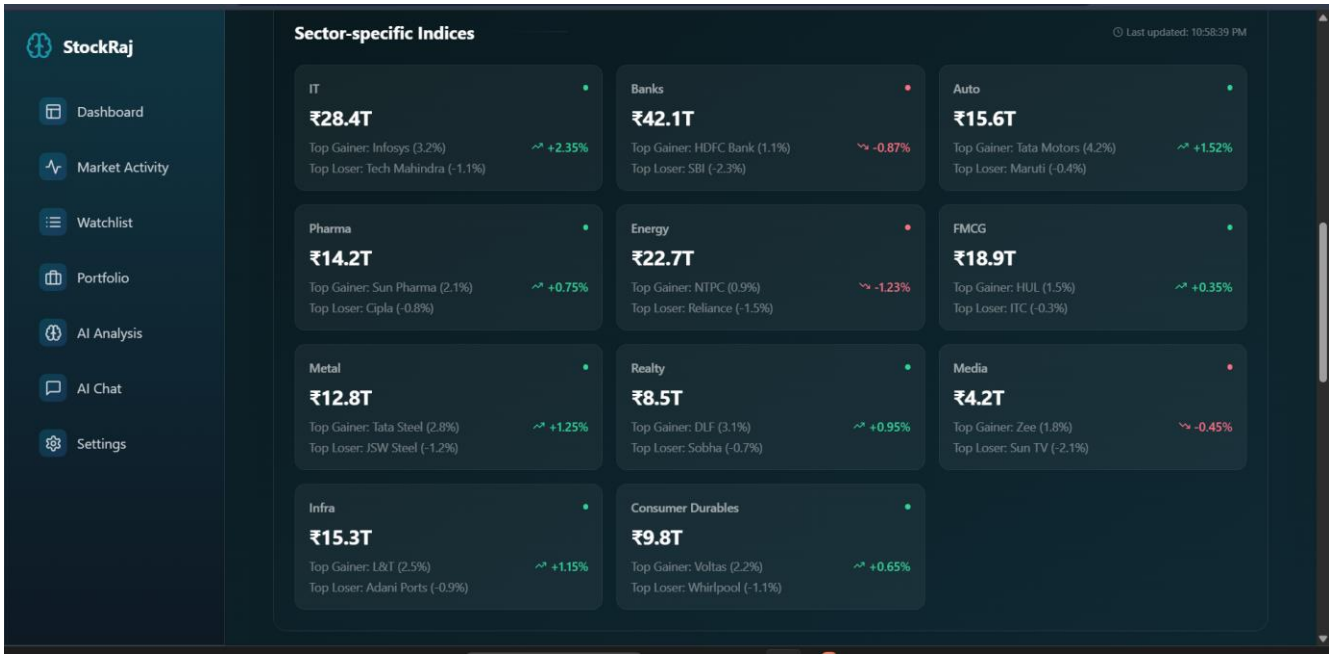


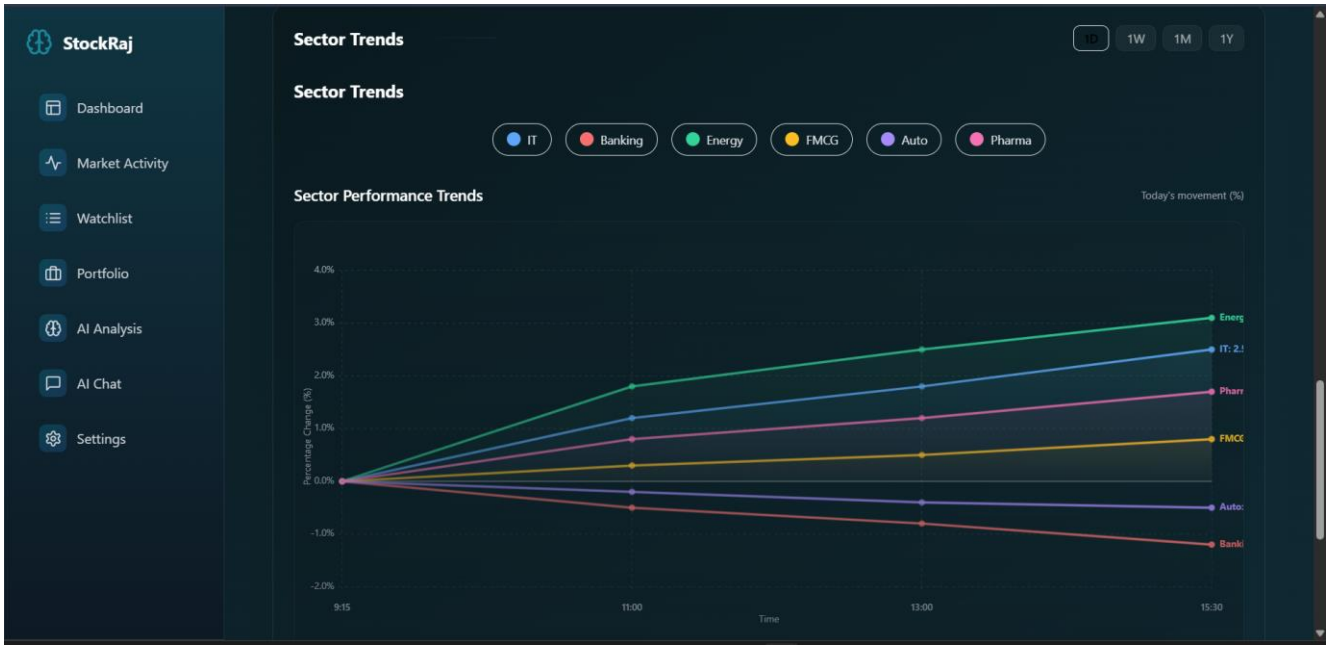

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Today's Financial News
Refresh
US To Push For Sweeping Reforms In Agriculture To E-Commerce In India BTA Talks: GTRI
Bloomberg Quint • 18 minutes ago
Two Trades for Today: An FMCG major for 3.5% gain, a large-cap oil refinery stock for 6% rise
Economic Times • 2 hours ago
Technical analysis identifies select stocks that may gain momentum even in volatile markets. Here are the technical calls for today....
US Likely To Seek Tariff Cuts, Regulatory Reforms In Trade Deal With India: GTRI
Bloomberg Quint • 4 hours ago
India resists the easing as it has to protect its small domestic retailers from unfair competition from deep-pocketed foreign firms....
Trade Setup For May 5: Nifty 50 Faces Immediate Resistance At 24,590
Bloomberg Quint • 4 hours ago
For Bank Nifty, the resistance is seen at 56,000, while the key support lies at 54,450, said Hrishikesh Yedve....
Warren Buffett's 10 best and worst investments over 60 years at Berkshire Hathaway
Economic Times • 4 hours ago

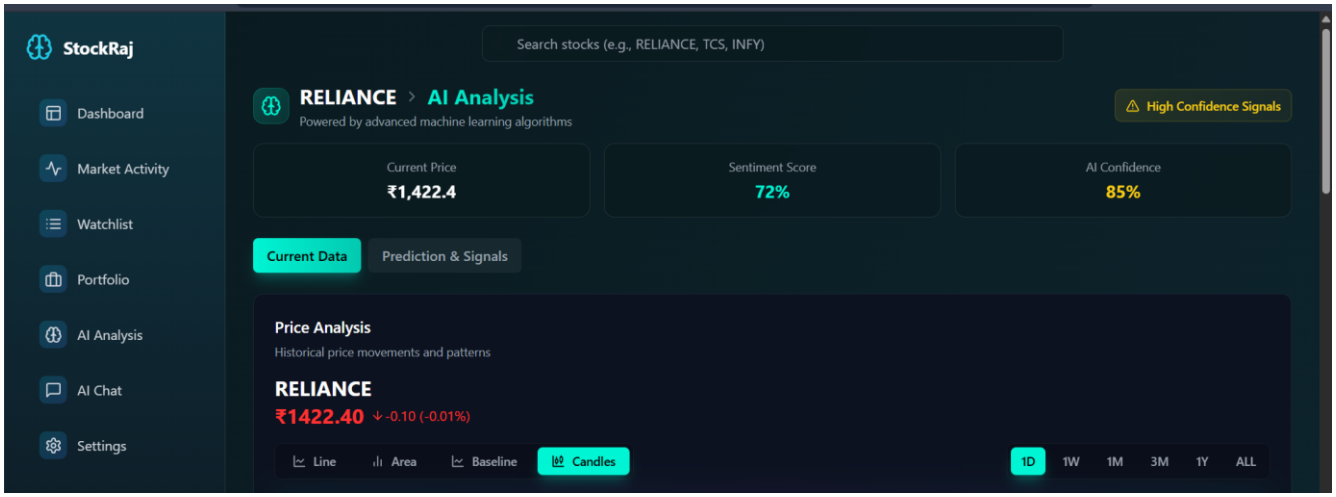

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Market Activity
Overview
Sectors
Market Pulse
NIFTY 50
24,346.70
0.05%
SENSEX
80,501.99
0.32%
NIFTY BANK
55,115.35
0.05%
Broad Market Indices
Last updated: 10:57:50 PM
NIFTY 50
24,346.699
+0.05%
SENSEX
80,501.992
+0.32%
NIFTY BANK
55,115.352
+0.05%
NIFTY 100
24,849.199
+0.02%
NIFTY MIDCAP 100
53,705.102
-0.78%
NIFTY MICROCAP 250
20,932.35
-0.09%
NIFTY SMALLCAP 100
16,441.801
-0.04%
NIFTY NEXT 50
64,429.75
-0.12%

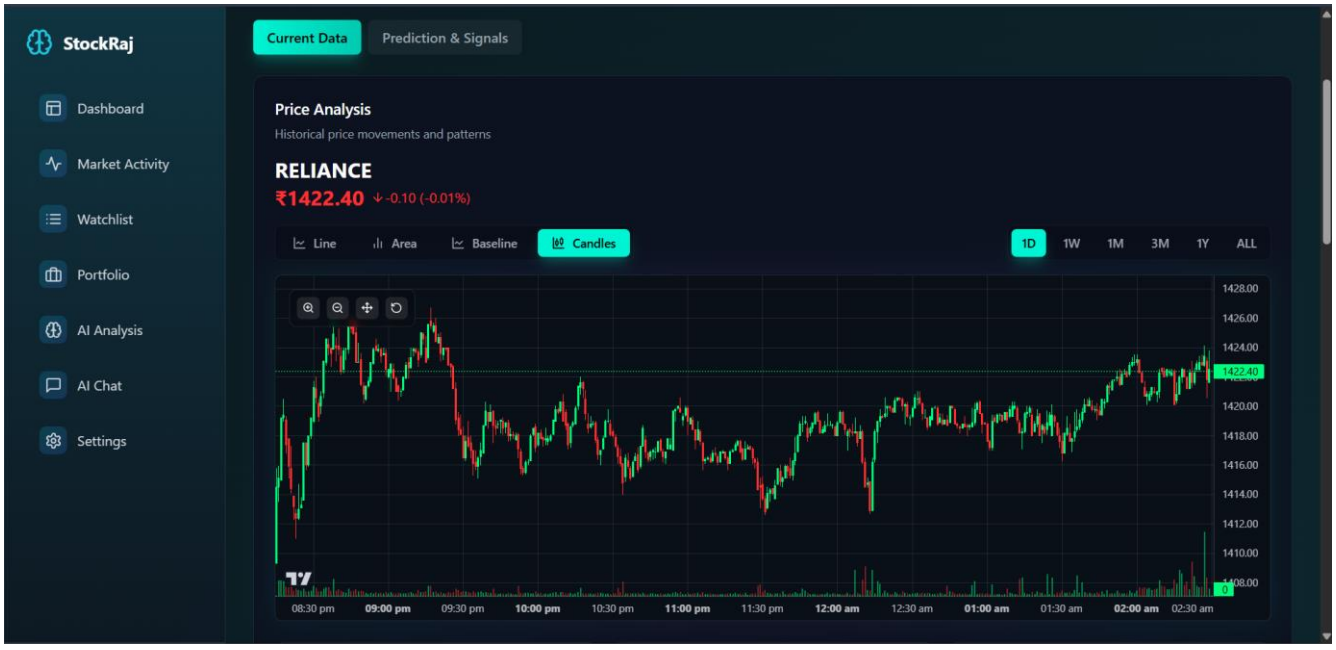

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Market Volatility
Current Volatility
0.23
Low Volatility
VIX Index
22.68
+-7.80%
Market Movers
Last updated: 10:57:51 PM
Top Gainers
Top Losers
52W High
52W Low
Trending
Adani Ports & Special Economic Zone NSE
₹1,267.1
+4.16%
VOL 12K
Bajaj Finance NSE
₹8,862.5
+2.64%
VOL 2K
IndusInd Bank NSE
+1.74%

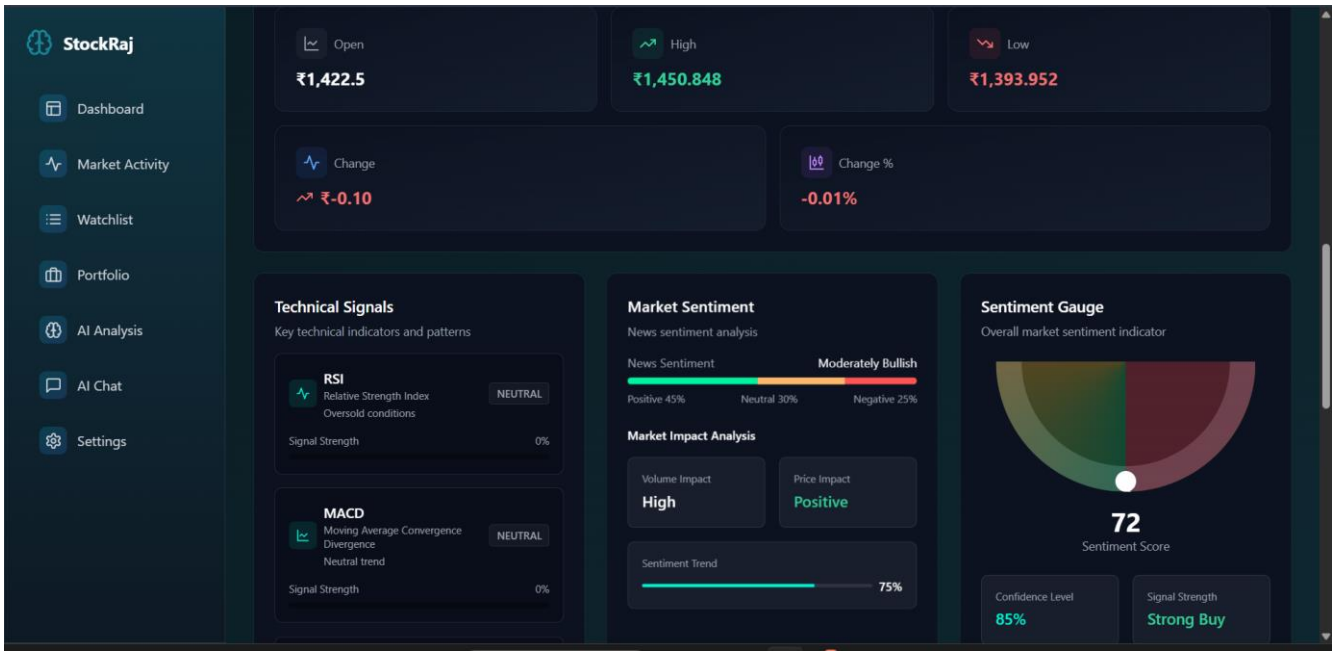

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Market Activity
Overview
Sectors
Sector Performance
Daily
Weekly
Monthly
1.1%
Sector
Market Cap
Change
Energy
₹10.5L Cr
3.1%
IT
₹15.2L Cr
2.5%
Pharma
₹6.9L Cr
1.7%
FMCG
₹8.9L Cr
0.8%
Auto
₹7.6L Cr
0.5%

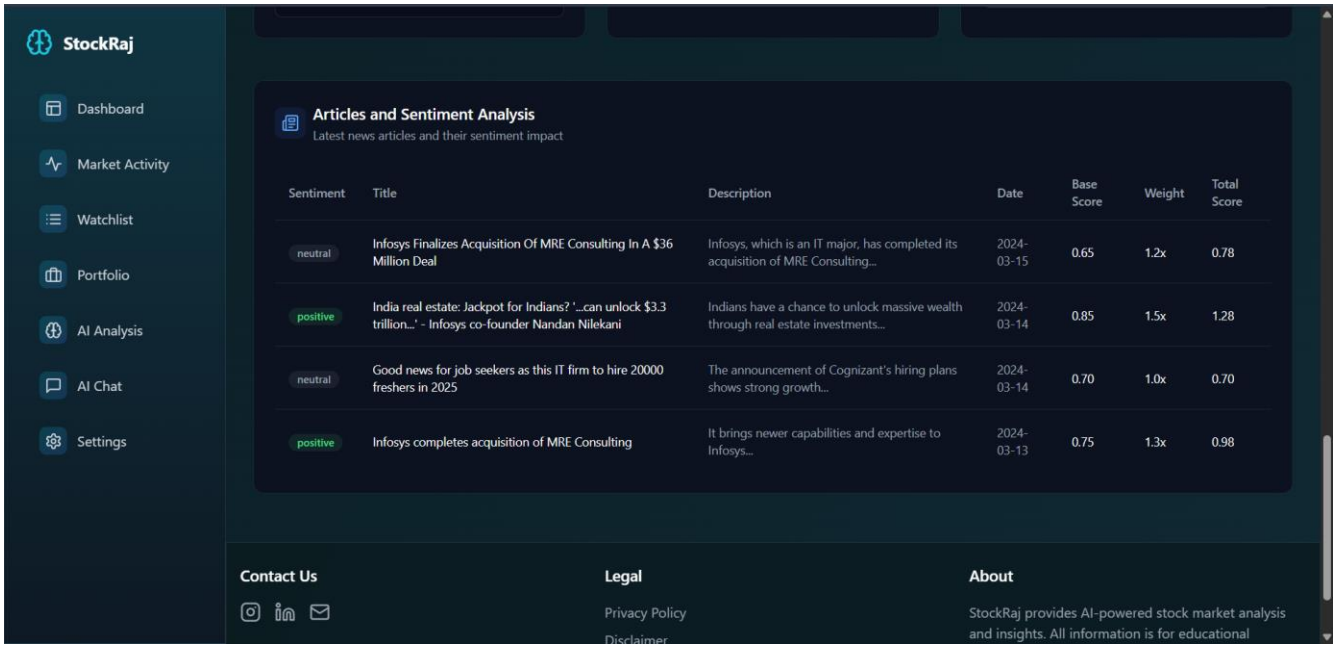

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Sector-specific Indices
Last updated: 10:58:39 PM
IT
₹28.4T
Top Gainer: Infosys (3.2%) +2.35%
Top Loser: Tech Mahindra (-1.1%)
Banks
₹42.1T
Top Gainer: HDFC Bank (1.1%) -0.87%
Top Loser: SBI (-2.3%)
Auto
₹15.6T
Top Gainer: Tata Motors (4.2%) +1.52%
Top Loser: Maruti (-0.4%)
Pharma
₹14.2T
Top Gainer: Sun Pharma (2.1%) +0.75%
Top Loser: Cipla (-0.8%)
Energy
₹22.7T
Top Gainer: NTPC (0.9%) -1.23%
Top Loser: Reliance (-1.5%)
FMCG
₹18.9T
Top Gainer: HUL (1.5%) +0.35%
Top Loser: ITC (-0.3%)
Metal
₹12.8T
Top Gainer: Tata Steel (2.8%) +1.25%
Top Loser: JSW Steel (-1.2%)
Realty
₹8.5T
Top Gainer: DLF (3.1%) +0.95%
Top Loser: Sobha (-0.7%)
Media
₹4.2T
Top Gainer: Zee (1.8%) -0.45%
Top Loser: Sun TV (-2.1%)
Infra
₹15.3T
Top Gainer: L&T (2.5%) +1.15%
Top Loser: Adani Ports (-0.9%)
Consumer Durables
₹9.8T
Top Gainer: Voltas (2.2%) +0.65%
Top Loser: Whirlpool (-1.1%)

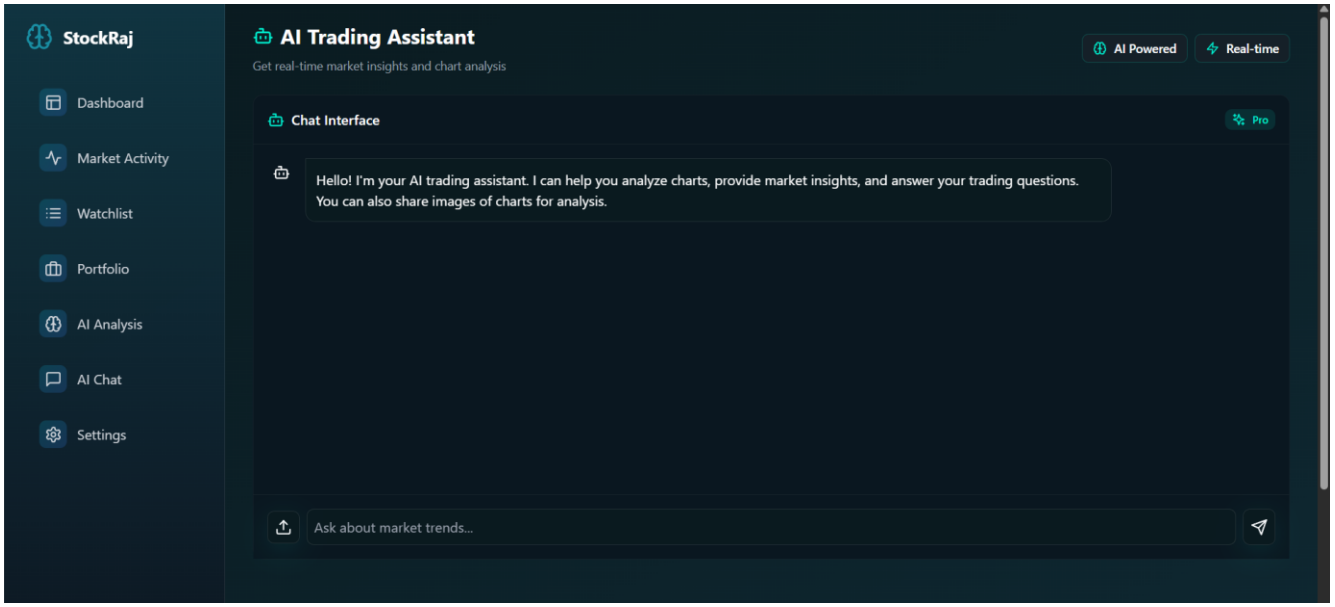

StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Sector Trends
1D
1W
1M
1Y
Sector Trends
IT
Banking
Energy
FMCG
Auto
Pharma
Sector Performance Trends
Today's movement (%)
4.0%
3.0%
2.0%
1.0%
0.0%
-1.0%
-2.0%
Percentage Change (%)
9:15
11:00
13:00
15:30
Time


StockRaj
Search stocks (e.g., RELIANCE, TCS, INFY)
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
RELIANCE › AI Analysis
Powered by advanced machine learning algorithms
High Confidence Signals
Current Price
₹1,422.4
Sentiment Score
72%
AI Confidence
85%
Current Data
Prediction & Signals
Price Analysis
Historical price movements and patterns
RELIANCE
₹1422.40 -0.10 (-0.01%)
Line
Area
Baseline
Candles
1D
1W
1M
3M
1Y
ALL


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Current Data
Prediction & Signals
Price Analysis
Historical price movements and patterns
RELIANCE
₹1422.40 -0.10 (-0.01%)
Line
Area
Baseline
Candles
1D
1W
1M
3M
1Y
ALL
1428.00
1426.00
1424.00
1422.40
1420.00
1418.00
1416.00
1414.00
1412.00
1410.00
08:30 pm
09:00 pm
09:30 pm
10:00 pm
10:30 pm
11:00 pm
11:30 pm
12:00 am
12:30 am
01:00 am
01:30 am
02:00 am
02:30 am


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Open
₹1,422.5
High
₹1,450.848
Low
₹1,393.952
Change
₹-0.10
Change %
-0.01%
Technical Signals
Key technical indicators and patterns
RSI
Relative Strength Index
Oversold conditions
NEUTRAL
Signal Strength
0%
MACD
Moving Average Convergence Divergence
Neutral trend
NEUTRAL
Signal Strength
0%
Market Sentiment
News sentiment analysis
News Sentiment
Moderately Bullish
Positive 45%
Neutral 30%
Negative 25%
Market Impact Analysis
Volume Impact
High
Price Impact
Positive
Sentiment Trend
75%
Sentiment Gauge
Overall market sentiment indicator
72
Sentiment Score
Confidence Level
85%
Signal Strength
Strong Buy


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Articles and Sentiment Analysis
Latest news articles and their sentiment impact
Sentiment
Title
Description
Date
Base Score
Weight
Total Score
neutral
Infosys Finalizes Acquisition Of MRE Consulting In A $36 Million Deal
Infosys, which is an IT major, has completed its acquisition of MRE Consulting...
2024-03-15
0.65
1.2x
0.78
positive
India real estate: Jackpot for Indians? '...can unlock $3.3 trillion...' - Infosys co-founder Nandan Nilekani
Indians have a chance to unlock massive wealth through real estate investments...
2024-03-14
0.85
1.5x
1.28
neutral
Good news for job seekers as this IT firm to hire 20000 freshers in 2025
The announcement of Cognizant's hiring plans shows strong growth...
2024-03-14
0.70
1.0x
0.70
positive
Infosys completes acquisition of MRE Consulting
It brings newer capabilities and expertise to Infosys...
2024-03-13
0.75
1.3x
0.98
Contact Us
Legal
Privacy Policy
Disclaimer
About
StockRaj provides AI-powered stock market analysis and insights. All information is for educational


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
AI Trading Assistant
Get real-time market insights and chart analysis
AI Powered
Real-time
Chat Interface
Pro
Hello! I'm your AI trading assistant. I can help you analyze charts, provide market insights, and answer your trading questions. You can also share images of charts for analysis.
Ask about market trends...

CHAPTER 9

RESULTS AND ANALYSIS

9.1 RESULTS AND ANALYSIS

In this chapter, we evaluate the performance, usability, and visual output of the platform. Each component was assessed based on responsiveness, accuracy, and alignment with expected outcomes. This evaluation validates the system's capability to deliver real-time insights and enhance the user experience.

9.1.1 Visual Output

The platform's visual elements, especially the charts and graphs, rendered seamlessly with real-time updates. Using modern libraries like Recharts and react-financial-charts enabled dynamic, interactive data visualizations. Data-driven styling further enhanced the interface by adapting chart aesthetics based on live market conditions.

9.1.1.1 Chart Responsiveness

Charts were tested across multiple devices and screen sizes. The responsive design, powered by Tailwind CSS and React, ensured smooth rendering on both desktop and mobile. The adaptive layout maintained functionality and readability regardless of screen dimensions.

9.1.1.2 Prediction Accuracy

The LSTM-based price prediction model showed promising results. Although not intended for high-frequency trading, the model was effective in identifying general trends and directions. Accuracy was evaluated using visual alignment between predicted and actual price movements.

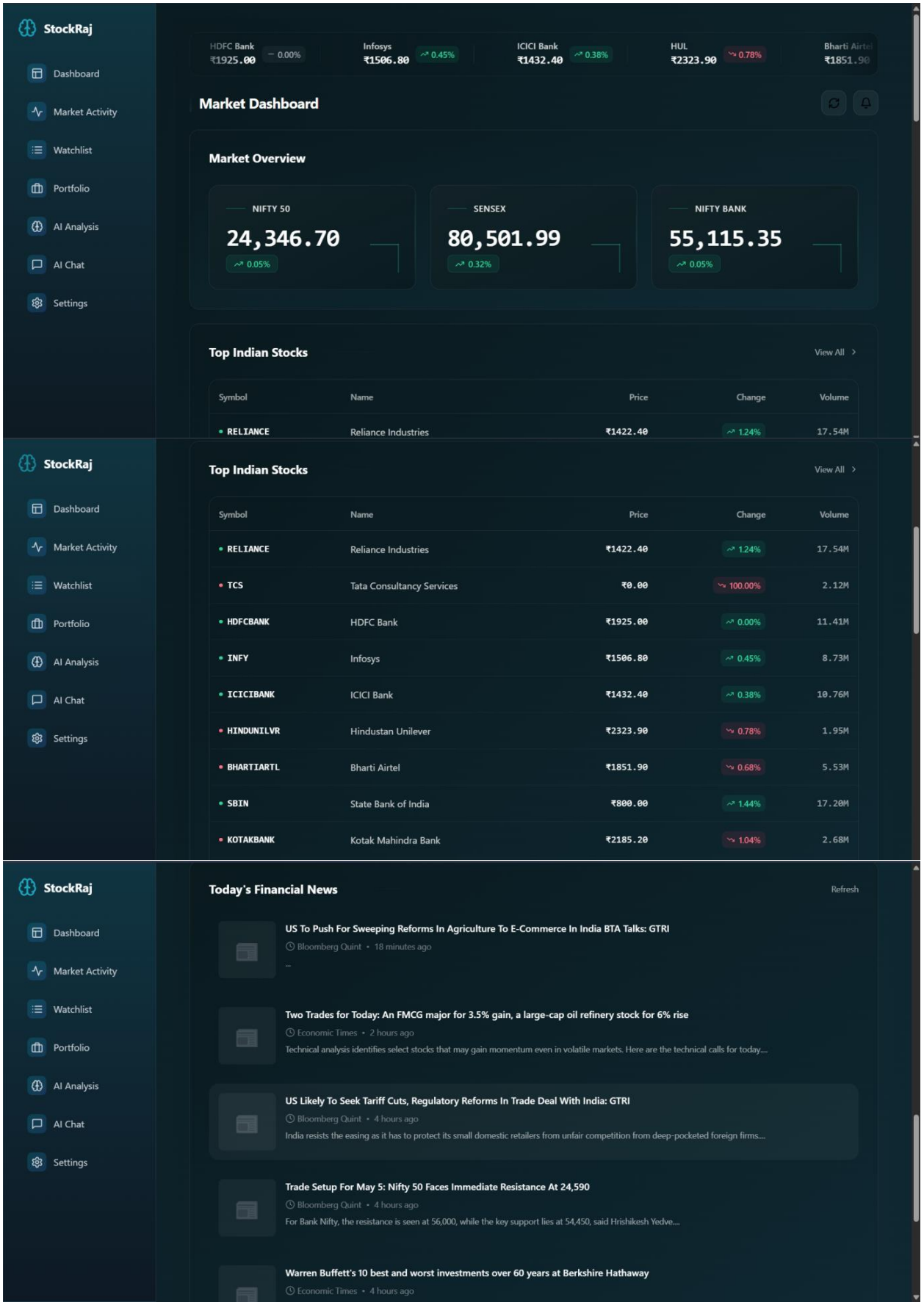
9.1.1.3 Sentiment and Chatbot

The sentiment analysis component successfully captured the market's reaction to recent news headlines. Using TextBlob and VADER provided polarity scores that aligned well with stock price movements. Additionally, the AI chatbot responded accurately to user queries, demonstrating contextual awareness and relevance in its responses.

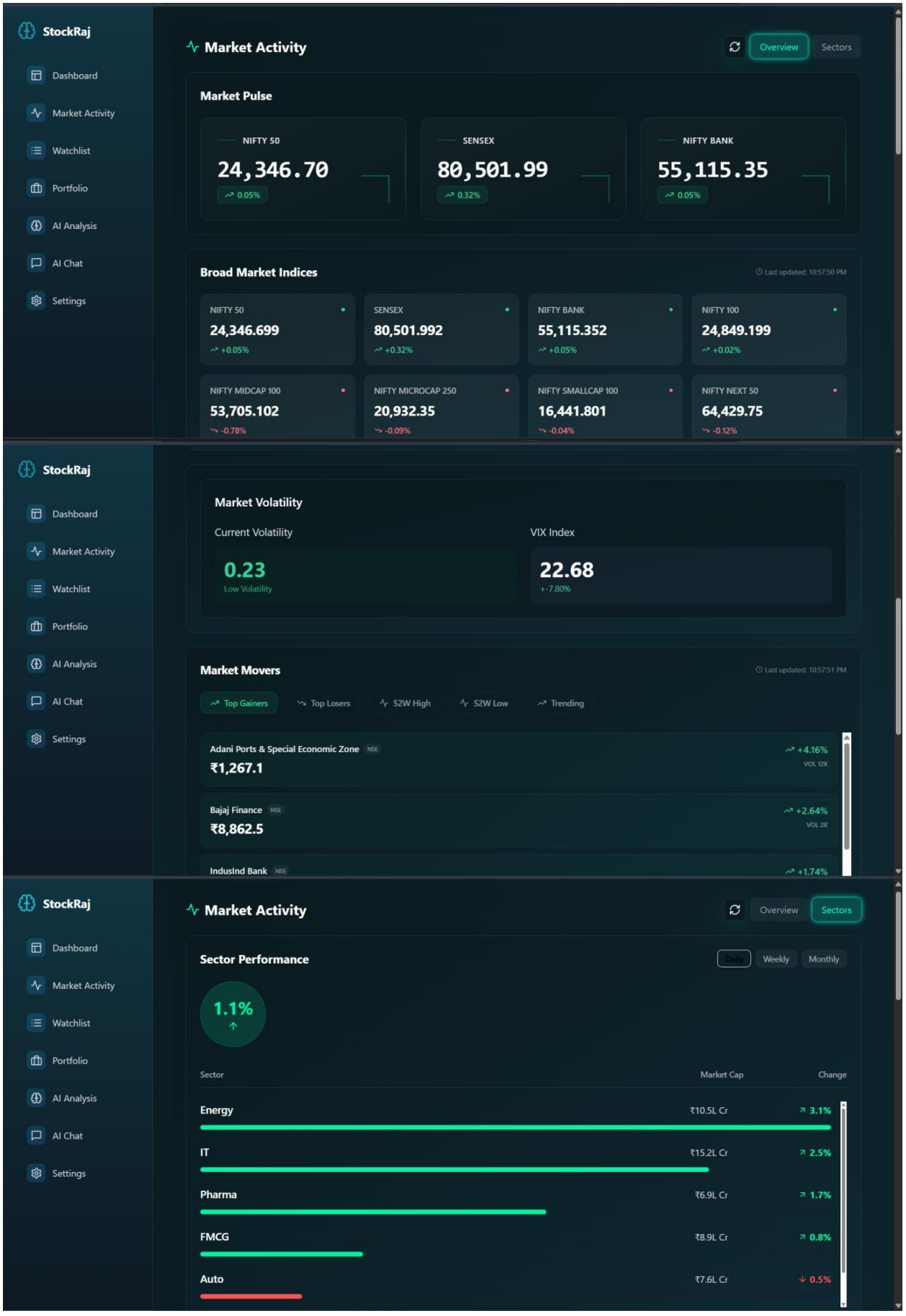
CHAPTER 10

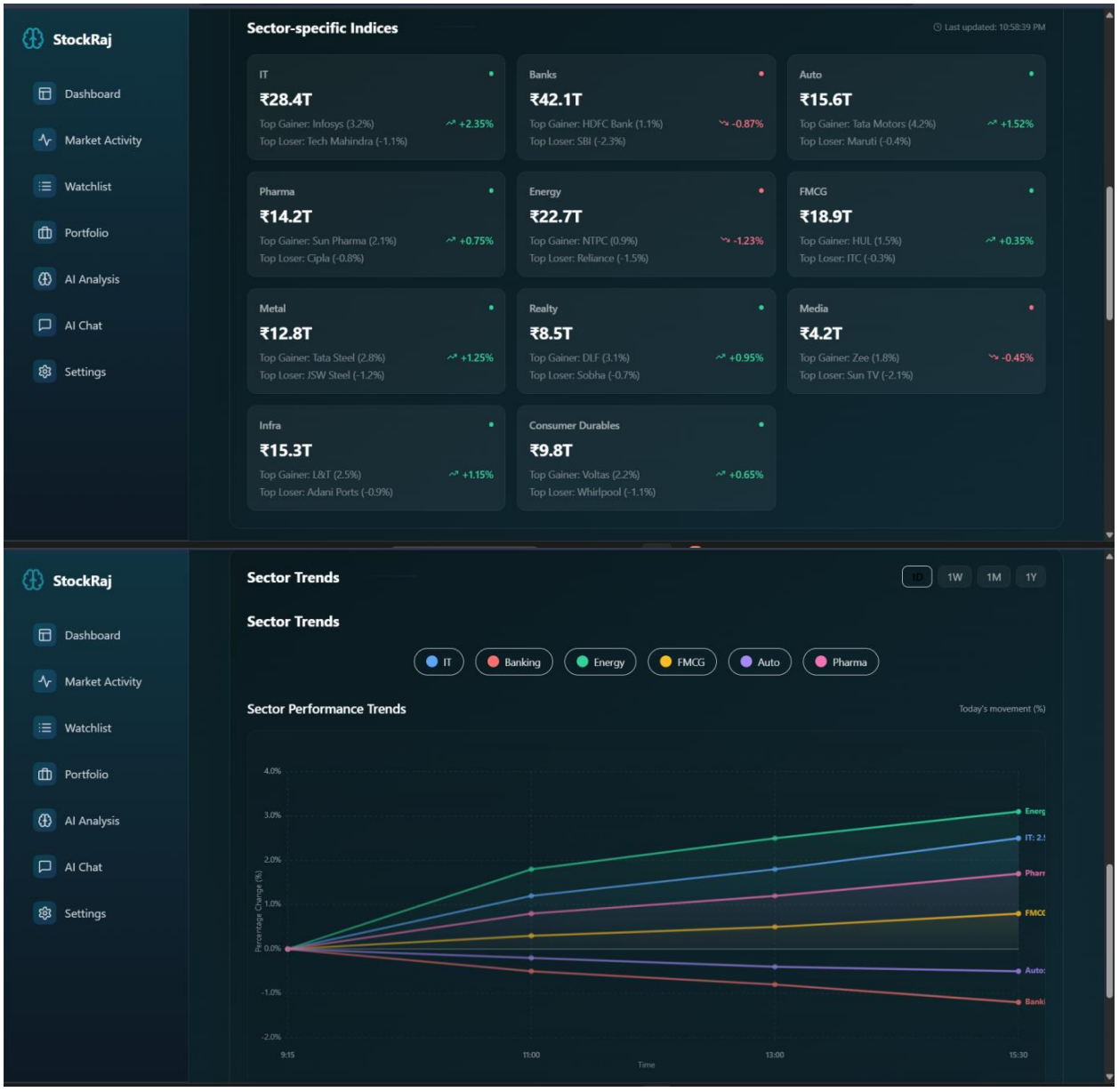
SNAPSHOTS AND IMPLEMENTATION VIDEOS

Dashboard


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
HDFC Bank ₹1925.00 0.00%
Infosys ₹1506.80 0.45%
ICICI Bank ₹1432.40 0.38%
HUL ₹2323.90 0.78%
Bharti Airtel ₹1851.90
Market Dashboard
Market Overview
NIFTY 50
24,346.70
0.05%
SENSEX
80,501.99
0.32%
NIFTY BANK
55,115.35
0.05%
Top Indian Stocks
View All
Symbol Name Price Change Volume
RELIANCE Reliance Industries ₹1422.40 1.24% 17.54M
TCS Tata Consultancy Services ₹0.00 100.00% 2.12M
HDFCBANK HDFC Bank ₹1925.00 0.00% 11.41M
INFY Infosys ₹1506.80 0.45% 8.73M
ICICIBANK ICICI Bank ₹1432.40 0.38% 10.76M
HINDUNILVR Hindustan Unilever ₹2323.90 0.78% 1.95M
BHARTIARTL Bharti Airtel ₹1851.90 0.68% 5.53M
SBIN State Bank of India ₹800.00 1.44% 17.20M
KOTAKBANK Kotak Mahindra Bank ₹2185.20 1.04% 2.68M
Today's Financial News
Refresh
US To Push For Sweeping Reforms In Agriculture To E-Commerce In India BTA Talks: GTRI
Bloomberg Quint • 18 minutes ago
Two Trades for Today: An FMCG major for 3.5% gain, a large-cap oil refinery stock for 6% rise
Economic Times • 2 hours ago
Technical analysis identifies select stocks that may gain momentum even in volatile markets. Here are the technical calls for today....
US Likely To Seek Tariff Cuts, Regulatory Reforms In Trade Deal With India: GTRI
Bloomberg Quint • 4 hours ago
India resists the easing as it has to protect its small domestic retailers from unfair competition from deep-pocketed foreign firms....
Trade Setup For May 5: Nifty 50 Faces Immediate Resistance At 24,590
Bloomberg Quint • 4 hours ago
For Bank Nifty, the resistance is seen at 56,000, while the key support lies at 54,450, said Hrishikesh Yedve....
Warren Buffett's 10 best and worst investments over 60 years at Berkshire Hathaway
Economic Times • 4 hours ago

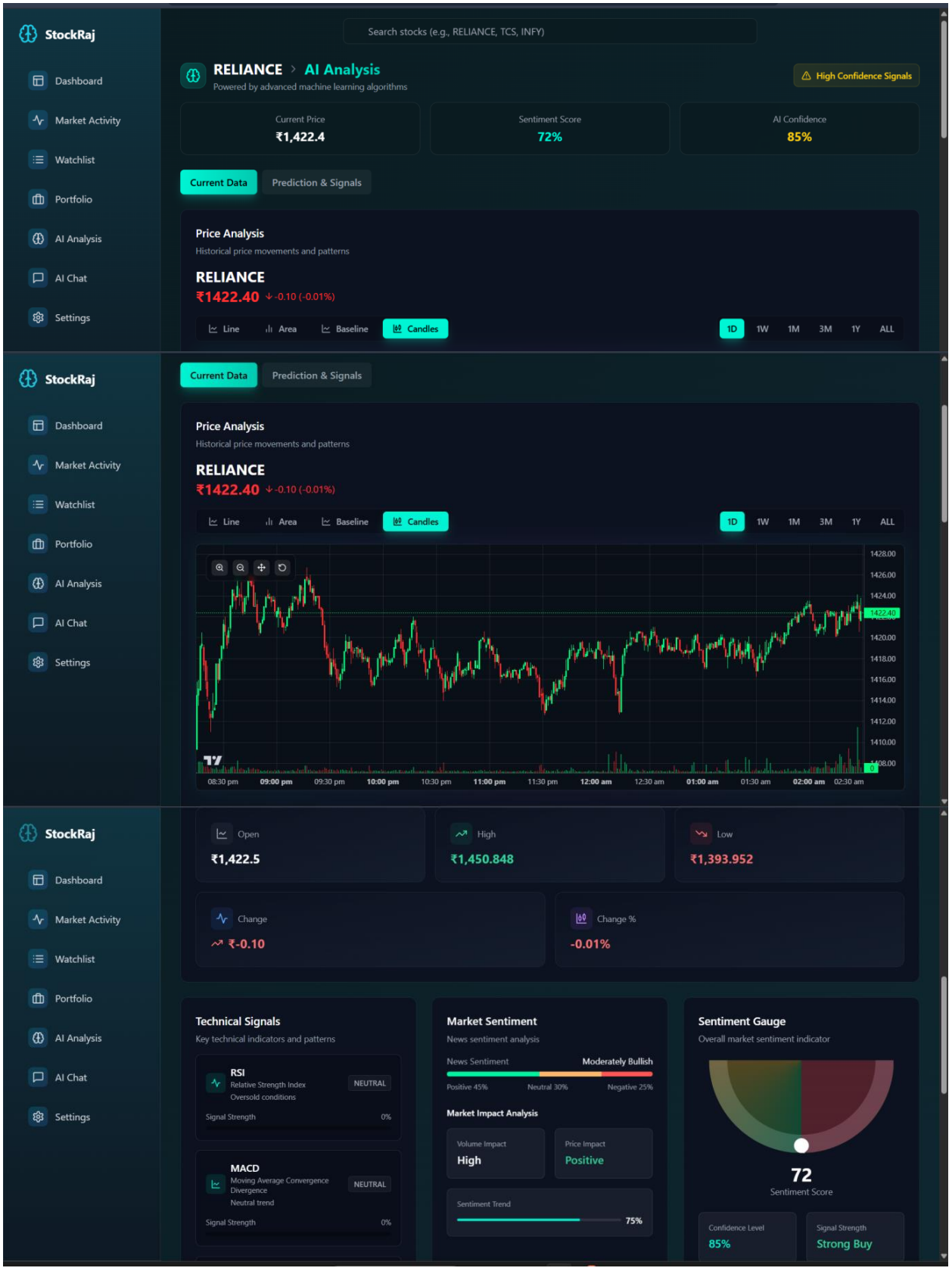
Market Activity

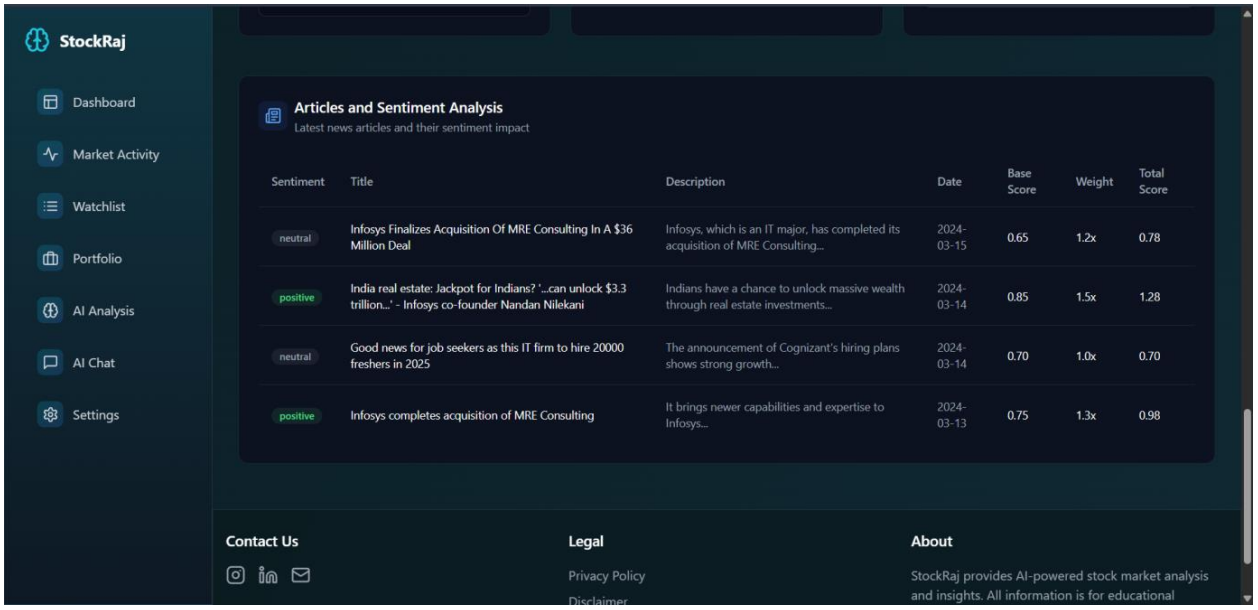



StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Sector-specific Indices
Last updated: 10:58:39 PM
IT
₹28.4T
Top Gainer: Infosys (3.2%) +2.35%
Top Loser: Tech Mahindra (-1.1%)
Banks
₹42.1T
Top Gainer: HDFC Bank (1.1%) -0.87%
Top Loser: SBI (-2.3%)
Auto
₹15.6T
Top Gainer: Tata Motors (4.2%) +1.52%
Top Loser: Maruti (-0.4%)
Pharma
₹14.2T
Top Gainer: Sun Pharma (2.1%) +0.75%
Top Loser: Cipla (-0.8%)
Energy
₹22.7T
Top Gainer: NTPC (0.9%) -1.23%
Top Loser: Reliance (-1.5%)
FMCG
₹18.9T
Top Gainer: HUL (1.5%) +0.35%
Top Loser: ITC (-0.3%)
Metal
₹12.8T
Top Gainer: Tata Steel (2.8%) +1.25%
Top Loser: JSW Steel (-1.2%)
Realty
₹8.5T
Top Gainer: DLF (3.1%) +0.95%
Top Loser: Sobha (-0.7%)
Media
₹4.2T
Top Gainer: Zee (1.8%) -0.45%
Top Loser: Sun TV (-2.1%)
Infra
₹15.3T
Top Gainer: L&T (2.5%) +1.15%
Top Loser: Adani Ports (-0.9%)
Consumer Durables
₹9.8T
Top Gainer: Voltas (2.2%) +0.65%
Top Loser: Whirlpool (-1.1%)

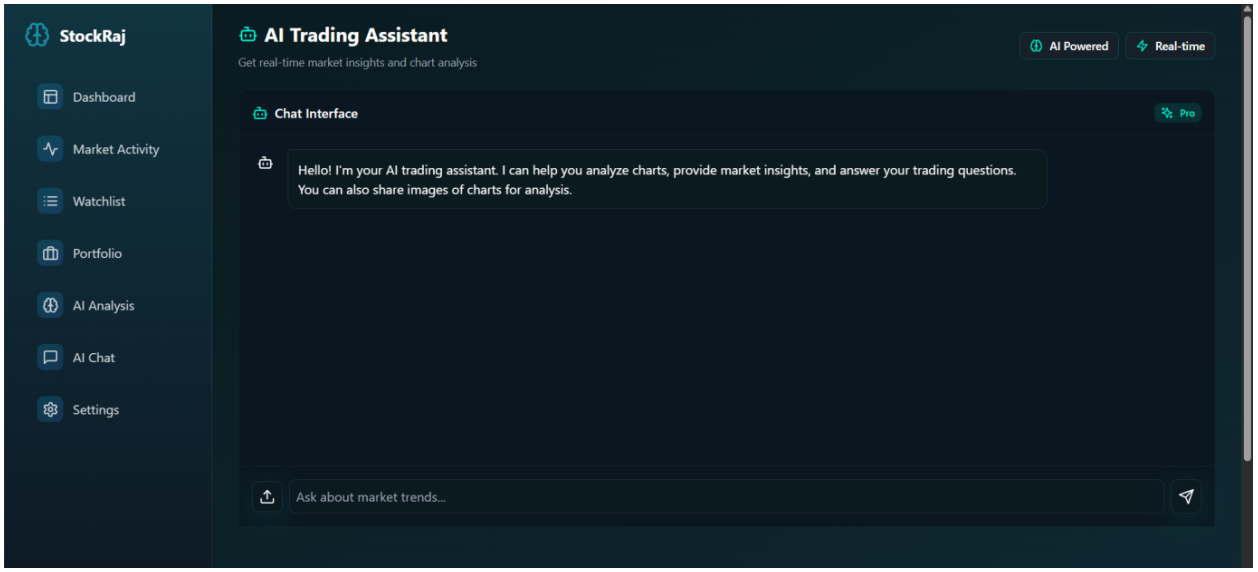
StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Sector Trends
1D
1W
1M
1Y
Sector Trends
IT
Banking
Energy
FMCG
Auto
Pharma
Sector Performance Trends
Today's movement (%)
4.0%
3.0%
2.0%
1.0%
0.0%
-1.0%
-2.0%
Percentage Change (%)
9:15
11:00
13:00
15:30
Time

AI Analysis


StockRaj
Search stocks (e.g., RELIANCE, TCS, INFY)
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
RELIANCE › AI Analysis
Powered by advanced machine learning algorithms
High Confidence Signals
Current Price
₹1,422.4
Sentiment Score
72%
AI Confidence
85%
Current Data
Prediction & Signals
Price Analysis
Historical price movements and patterns
RELIANCE
₹1422.40 -0.10 (-0.01%)
Line
Area
Baseline
Candles
1D
1W
1M
3M
1Y
ALL
Open
₹1,422.5
High
₹1,450.848
Low
₹1,393.952
Change
₹-0.10
Change %
-0.01%
Technical Signals
Key technical indicators and patterns
RSI
Relative Strength Index
Oversold conditions
NEUTRAL
Signal Strength
0%
MACD
Moving Average Convergence Divergence
Neutral trend
NEUTRAL
Signal Strength
0%
Market Sentiment
News sentiment analysis
News Sentiment
Moderately Bullish
Positive 45%
Neutral 30%
Negative 25%
Market Impact Analysis
Volume Impact
High
Price Impact
Positive
Sentiment Trend
75%
Sentiment Gauge
Overall market sentiment indicator
72
Sentiment Score
Confidence Level
85%
Signal Strength
Strong Buy


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
Articles and Sentiment Analysis
Latest news articles and their sentiment impact
Sentiment
Title
Description
Date
Base Score
Weight
Total Score
neutral
Infosys Finalizes Acquisition Of MRE Consulting In A $36 Million Deal
Infosys, which is an IT major, has completed its acquisition of MRE Consulting...
2024-03-15
0.65
1.2x
0.78
positive
India real estate: Jackpot for Indians? '...can unlock $3.3 trillion...' - Infosys co-founder Nandan Nilekani
Indians have a chance to unlock massive wealth through real estate investments...
2024-03-14
0.85
1.5x
1.28
neutral
Good news for job seekers as this IT firm to hire 20000 freshers in 2025
The announcement of Cognizant's hiring plans shows strong growth...
2024-03-14
0.70
1.0x
0.70
positive
Infosys completes acquisition of MRE Consulting
It brings newer capabilities and expertise to Infosys...
2024-03-13
0.75
1.3x
0.98
Contact Us
Legal
Privacy Policy
Disclaimer
About
StockRaj provides AI-powered stock market analysis and insights. All information is for educational

AI Chat


StockRaj
Dashboard
Market Activity
Watchlist
Portfolio
AI Analysis
AI Chat
Settings
AI Trading Assistant
Get real-time market insights and chart analysis
AI Powered
Real-time
Chat Interface
Pro
Hello! I'm your AI trading assistant. I can help you analyze charts, provide market insights, and answer your trading questions.
You can also share images of charts for analysis.
Ask about market trends...

Code Snippets

App.tsx

```tsx
import React, { useState, useEffect } from 'react';
import Navigation from './components/Navigation';
import { Dashboard } from './pages/Dashboard';
import { MarketActivity } from './pages/MarketActivity';
import { Watchlist } from './pages/Watchlist';
import { Portfolio } from './pages/Portfolio';
import { AIAnalysis } from './pages/AIAnalysis';
import { Chatbot } from './pages/Chatbot';
import { Settings } from './pages/Settings';
import { Footer } from './components/Footer/Footer';
import { AuthProvider } from './context/AuthContext';
import { ThemeProvider } from './context/ThemeContext';
import { DataProvider } from './context/DataContext.tsx';
import './styles/glassmorphism.css';

const App: React.FC = () => {
  const [currentPage, setCurrentPage] = useState('dashboard');
  const [isMobileMenuOpen, setIsMobileMenuOpen] = useState(false);

  // Add scroll to top effect when page changes
  useEffect(() => {
    window.scrollTo(0, 0);
  }, [currentPage]);

  const renderPage = () => {
    switch (currentPage) {
      case 'dashboard':
        return <Dashboard />;
      case 'market':
        return <MarketActivity />;
      case 'watchlist':
        return <Watchlist />;
      case 'portfolio':
        return <Portfolio />;
      case 'analysis':
        return <AIAnalysis />;
      case 'chatbot':
        return <Chatbot />;
      case 'settings':
        return <Settings />;
      default:
        return <Dashboard />;
    }
  };

  return (
    <AuthProvider>
      <ThemeProvider>
        <DataProvider>
          <div className="min-h-screen bg-gradient-to-br from-primary to-secondary">
            <div className="lg:hidden">
              <button
                type="button"
                aria-label="Toggle mobile menu"
                onClick={() => setIsMobileMenuOpen(!isMobileMenuOpen)}
                className="fixed top-4 left-4 z-50 glass-button p-2 rounded-lg"
              >
                <svg className="w-6 h-6" fill="none" stroke="currentColor" viewBox="0 0 24 24">
                  <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2}
                        d={isMobileMenuOpen ? "M6 18L18 6M6 6l12 12" : "M4 6h16M4 12h16M4 18h16"} />
                </svg>
              </button>
            </div>

            <Navigation
              onPageChange={setCurrentPage}
              isMobileMenuOpen={isMobileMenuOpen}
              onMobileMenuClose={() => setIsMobileMenuOpen(false)}
            />

            <div className="lg:pl-64 transition-all duration-300">
              <main className="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-6">
                {renderPage()}
              </main>
              <Footer />
            </div>
          </div>
        </DataProvider>
      </ThemeProvider>
    </AuthProvider>
  );
};

export default App;
```

Sentiment

```tsx
import React from 'react';
import { Brain } from 'lucide-react';

interface SentimentData {
  news: { positive: number; negative: number; neutral: number };
  social: { positive: number; negative: number; neutral: number };
  overall: string;
}

interface SentimentAnalysisProps {
  data: SentimentData;
}

export const SentimentAnalysis: React.FC<SentimentAnalysisProps> = ({ data }) => {
  const renderSentimentBar = (positive: number, neutral: number, negative: number) => (
    <div className="h-2 w-full flex rounded-full overflow-hidden">
      <div className="bg-success" style={{ width: `${positive}%` }} />
      <div className="bg-warning" style={{ width: `${neutral}%` }} />
      <div className="bg-danger" style={{ width: `${negative}%` }} />
    </div>
  );

  return (
    <>
      <h3 className="text-lg font-semibold text-white flex items-center mb-4">
        <Brain className="w-5 h-5 mr-2 text-accent-primary" />
        Sentiment Analysis
      </h3>

      <div className="space-y-6">
        <div className="space-y-2">
          <div className="flex justify-between text-sm">
            <span className="text-gray-400">News Sentiment</span>
            <span className="text-white">{data.overall}</span>
          </div>
          {renderSentimentBar(data.news.positive, data.news.neutral, data.news.negative)}
          <div className="flex justify-between text-xs text-gray-400">
            <span>Positive {data.news.positive}%</span>
            <span>Neutral {data.news.neutral}%</span>
            <span>Negative {data.news.negative}%</span>
          </div>
        </div>

        <div className="space-y-2">
          <div className="flex justify-between text-sm">
            <span className="text-gray-400">Social Media Sentiment</span>
          </div>
          {renderSentimentBar(data.social.positive, data.social.neutral, data.social.negative)}
          <div className="flex justify-between text-xs text-gray-400">
            <span>Positive {data.social.positive}%</span>
            <span>Neutral {data.social.neutral}%</span>
            <span>Negative {data.social.negative}%</span>
          </div>
        </div>
      </div>
    </>
  );
};
```

Technical_indicators

```tsx
import React, { useEffect, useState, useRef } from 'react';
import { Activity, Waves, LineChart } from 'lucide-react';
import styles from './TechnicalIndicators.module.css';

export interface HistoricalData {
  timestamp: number[];
  open: number[];
  high: number[];
  low: number[];
  close: number[];
  volume: number[];
}

interface Indicator {
  name: string;
  value: number;
  signal: 'buy' | 'sell' | 'neutral';
  strength: number;
  description: string;
  icon: React.ReactNode;
}

interface TechnicalIndicatorsProps {
  historicalData: HistoricalData | null;
  technicalIndicators?: {
    rsi: number;
    macd: { macd: number; signal: number; histogram: number };
    bollingerBands: { upper: number; middle: number; lower: number };
  } | null;
}

export const TechnicalIndicators: React.FC<TechnicalIndicatorsProps> = ({ historicalData, technicalIndicators
 }) => {
  const [indicators, setIndicators] = useState<Indicator[]>([
    {
      name: 'RSI',
      value: 0,
      signal: 'neutral',
      strength: 0,
      description: 'Relative Strength Index',
      icon: <Activity className="w-4 h-4" />
    },
    {
      name: 'MACD',
      value: 0,
      signal: 'neutral',
      strength: 0,
      description: 'Moving Average Convergence Divergence',
      icon: <LineChart className="w-4 h-4" />
    },
    {
      name: 'Bollinger',
      value: 0,
      signal: 'neutral',
      strength: 0,
      description: 'Bollinger Bands Position',
      icon: <Waves className="w-4 h-4" />
    }
  ]);
```

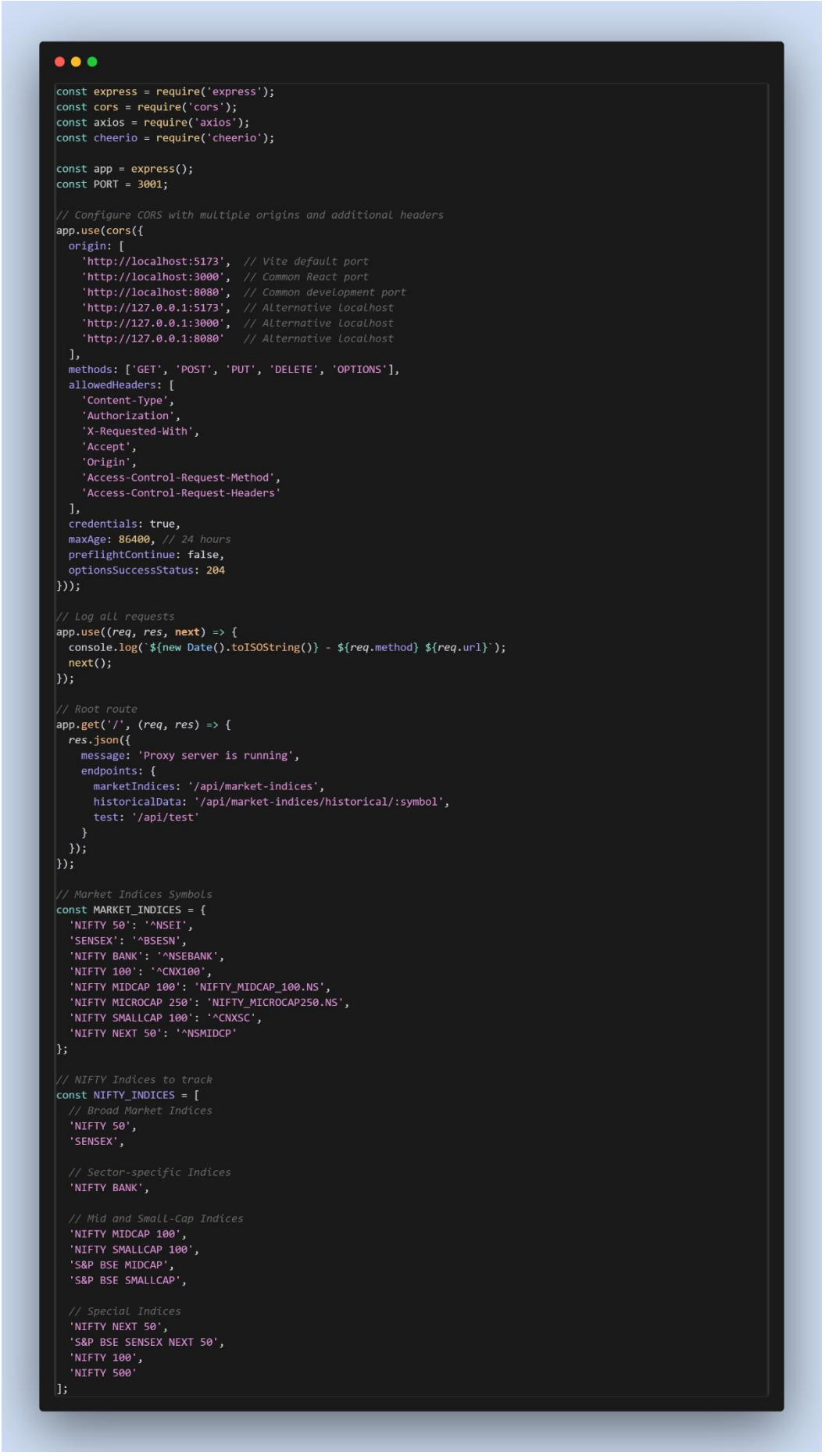
TradingView Chart

```tsx
import React, { useEffect, useRef, useState } from 'react';
import {
  createChart,
  ColorType,
  IChartApi,
  ISeriesApi,
  CandlestickSeries,
  LineSeries,
  AreaSeries,
  BaselineSeries,
  HistogramSeries,
  Time
} from 'lightweight-charts';
import { ArrowUp, ArrowDown, ZoomIn, ZoomOut, Move, RotateCcw, AlertTriangle, LineChart, BarChart2, CandlestickChart } from 'lucide-react';

interface ChartData {
  timestamp: number[];
  open: number[];
  high: number[];
  low: number[];
  close: number[];
  volume: number[];
}

interface StockChartProps {
  symbol: string;
  data: {
    price: number;
    change: number;
    changePercent: number;
  };
  onRealtimeUpdate?: (data: {
    price: number;
    change: number;
    changePercent: number;
    symbol: string;
  }) => void;
  onHistoricalData?: (historicalData: ChartData | null) => void;
}

type AdvancedChartType = 'line' | 'area' | 'baseline' | 'candles';

interface YahooFinanceData {
  timestamp: number;
  date: string;
  open: number;
  high: number;
  low: number;
  close: number;
  volume: number;
  adjustedClose?: number;
}

const TradingViewChart: React.FC<StockChartProps> = ({ symbol, data, onRealtimeUpdate, onHistoricalData }) => {
  const chartContainerRef = useRef<HTMLDivElement>(null);
  const [chart, setChart] = useState<IChartApi | null>(null);
  const [series, setSeries] = useState<ISeriesApi<"Candlestick" | "Line" | "Area" | "Baseline"> | null>(null);
  const [historicalData, setHistoricalData] = useState<ChartData | null>(null);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);
  const [timeframe, setTimeframe] = useState('1D');
  const [chartType, setChartType] = useState<AdvancedChartType>('candles');
  const [currentPrice, setCurrentPrice] = useState(data.price);
  const [currentChange, setCurrentChange] = useState(data.change);
  const [currentChangePercent, setCurrentChangePercent] = useState(data.changePercent);
  const [isPanning, setIsPanning] = useState(false);
  const isPositive = currentChange >= 0;
  const [cachedData, setCachedData] = useState<{ [key: string]: { [timeframe: string]: ChartData } }>({});
  const [allSeries, setAllSeries] = useState<ISeriesApi<"Candlestick" | "Line" | "Area" | "Baseline" | "Histogram">[]>([]);

  const timeframes = [
    { id: '1D', label: '1D', interval: '1m', range: '1d' },
    { id: '1W', label: '1W', interval: '1d', range: '7d' },
    { id: '1M', label: '1M', interval: '1d', range: '1mo' },
    { id: '3M', label: '3M', interval: '1d', range: '3mo' },
    { id: '1Y', label: '1Y', interval: '1wk', range: '1y' },
    { id: 'ALL', label: 'ALL', interval: '1mo', range: 'max' }
  ];

  const chartTypes = [
    {
      id: 'line',
      label: 'Line',
      icon: <LineChart className="w-4 h-4" />,
      description: 'Simple line chart showing price movement'
    },
    {
      id: 'area',
      label: 'Area',
      icon: <BarChart2 className="w-4 h-4" />,
      description: 'Area chart with gradient fill below the line'
    },
    {
      id: 'baseline',
      label: 'Baseline',
      icon: <LineChart className="w-4 h-4" />,
      description: 'Chart with baseline reference showing relative performance'
    },
    {
      id: 'candles',
      label: 'Candles',
      icon: <CandlestickChart className="w-4 h-4" />,
      description: 'Candlestick chart showing OHLC data'
    }
  ];
```

Portfolio

```tsx
import React from 'react';
import { Briefcase, Plus, TrendingUp, TrendingDown } from 'lucide-react';

interface PortfolioStock {
  symbol: string;
  quantity: number;
  avgPrice: number;
  currentPrice: number;
  change: number;
}

const portfolioData: PortfolioStock[] = [
  { symbol: 'RELIANCE', quantity: 100, avgPrice: 2400, currentPrice: 2456.75, change: 2.36 },
  { symbol: 'INFY', quantity: 50, avgPrice: 1500, currentPrice: 1456.80, change: -2.88 },
];

export const PortfolioSection: React.FC = () => {
  const calculateTotalValue = () => {
    return portfolioData.reduce((total, stock) => total + (stock.currentPrice * stock.quantity), 0);
  };

  const calculateTotalGain = () => {
    return portfolioData.reduce((total, stock) =>
      total + ((stock.currentPrice - stock.avgPrice) * stock.quantity), 0);
  };

  return (
    <div className="glass p-4 rounded-xl">
      <div className="flex justify-between items-center mb-4">
        <div className="flex items-center space-x-2">
          <Briefcase className="w-5 h-5 text-accent-primary" />
          <h3 className="text-lg font-semibold text-white">Your Portfolio</h3>
        </div>
        <button className="glass-button px-3 py-1 flex items-center space-x-1">
          <Plus className="w-4 h-4" />
          <span>New Portfolio</span>
        </button>
      </div>

      <div className="grid grid-cols-2 gap-4 mb-4">
        <div className="glass p-3 rounded-lg">
          <div className="text-sm text-gray-400">Total Value</div>
          <div className="text-lg font-bold text-white">
            ₹{calculateTotalValue().toLocaleString('en-IN')}
          </div>
        </div>
        <div className="glass p-3 rounded-lg">
          <div className="text-sm text-gray-400">Total Gain/Loss</div>
          <div className={`text-lg font-bold ${calculateTotalGain() >= 0 ? 'text-green-400' : 'text-red-400'}
`}>
            ₹{calculateTotalGain().toLocaleString('en-IN')}
          </div>
        </div>
      </div>

      <div className="space-y-3">
        {portfolioData.map((stock, index) => (
          <div key={index} className="glass p-3 rounded-lg">
            <div className="flex justify-between items-center">
              <div>
                <div className="text-white font-medium">{stock.symbol}</div>
                <div className="text-xs text-gray-400">{stock.quantity} shares</div>
              </div>
              <div className="text-right">
                <div className="text-white">₹{stock.currentPrice.toFixed(2)}</div>
                <div className={`flex items-center text-sm ${
                  stock.change >= 0 ? 'text-green-400' : 'text-red-400'
                }`}>
                  {stock.change >= 0 ? (
                    <TrendingUp className="w-3 h-3 mr-1" />
                  ) : (
                    <TrendingDown className="w-3 h-3 mr-1" />
                  )}
                  {stock.change}%
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};
```

Proxy Server/Express/Endpoints/Backend



```
const express = require('express');
const cors = require('cors');
const axios = require('axios');
const cheerio = require('cheerio');

const app = express();
const PORT = 3001;

// Configure CORS with multiple origins and additional headers
app.use(cors({
  origin: [
    'http://localhost:5173',  // Vite default port
    'http://localhost:3000',  // Common React port
    'http://localhost:8080',  // Common development port
    'http://127.0.0.1:5173',  // Alternative localhost
    'http://127.0.0.1:3000',  // Alternative localhost
    'http://127.0.0.1:8080'   // Alternative localhost
  ],
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'OPTIONS'],
  allowedHeaders: [
    'Content-Type',
    'Authorization',
    'X-Requested-With',
    'Accept',
    'Origin',
    'Access-Control-Request-Method',
    'Access-Control-Request-Headers'
  ],
  credentials: true,
  maxAge: 86400, // 24 hours
  preflightContinue: false,
  optionsSuccessStatus: 204
}));

// Log all requests
app.use((req, res, next) => {
  console.log(`${new Date().toISOString()} - ${req.method} ${req.url}`);
  next();
});

// Root route
app.get('/', (req, res) => {
  res.json({
    message: 'Proxy server is running',
    endpoints: {
      marketIndices: '/api/market-indices',
      historicalData: '/api/market-indices/historical/:symbol',
      test: '/api/test'
    }
  });
});

// Market Indices Symbols
const MARKET_INDICES = {
  'NIFTY 50': '^NSEI',
  'SENSEX': '^BSESN',
  'NIFTY BANK': '^NSEBANK',
  'NIFTY 100': '^CNX100',
  'NIFTY MIDCAP 100': 'NIFTY_MIDCAP_100.NS',
  'NIFTY MICROCAP 250': 'NIFTY_MICROCAP250.NS',
  'NIFTY SMALLCAP 100': '^CNXSC',
  'NIFTY NEXT 50': '^NSMIDCP'
};

// NIFTY Indices to track
const NIFTY_INDICES = [
  // Broad Market Indices
  'NIFTY 50',
  'SENSEX',

  // Sector-specific Indices
  'NIFTY BANK',

  // Mid and Small-Cap Indices
  'NIFTY MIDCAP 100',
  'NIFTY SMALLCAP 100',
  'S&P BSE MIDCAP',
  'S&P BSE SMALLCAP',

  // Special Indices
  'NIFTY NEXT 50',
  'S&P BSE SENSEX NEXT 50',
  'NIFTY 100',
  'NIFTY 500'
];
```

Fig2

```
// Add caching middleware
const cache = new Map();
const CACHE_DURATION = 5 * 60 * 1000; // 5 minutes

const getCacheKey = (symbol, interval, range) => `${symbol}-${interval}-${range}`;

// Helper function to format symbol for Yahoo Finance
const formatSymbol = (symbol) => {
  // If symbol already has a suffix, return as is
  if (symbol.includes('.')) {
    return symbol;
  }
  // Add .NS suffix for Indian stocks
  return `${symbol}.NS`;
};

// Helper function to fetch Yahoo Finance data
const fetchYahooFinanceData = async (symbol) => {
  try {
    console.log(`Fetching data for symbol: ${symbol}`);
    const response = await axios.get(
      `https://query1.finance.yahoo.com/v8/finance/chart/${symbol}?interval=1m&range=1d`,
      {
        headers: {
          "User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/53
7.36"
,
          "Accept-Language": "en-US,en;q=0.9",
          "Accept-Encoding": "gzip, deflate, br",
          "Connection": "keep-alive"
        }
      }
    );

    if (!response.data.chart?.result?.[0]) {
      throw new Error('No data available for the given symbol');
    }

    const result = response.data.chart.result[0];
    const quote = result.indicators.quote[0];
    const timestamps = result.timestamp;
    const closes = quote.close;

    // Validate data
    if (!timestamps || !closes || timestamps.length === 0 || closes.length === 0) {
      throw new Error('Invalid data received from Yahoo Finance');
    }

    // Get the latest non-null data point
    let latestIndex = timestamps.length - 1;
    while (latestIndex >= 0 && (closes[latestIndex] === null || closes[latestIndex] === undefined)) {
      latestIndex--;
    }

    if (latestIndex < 0) {
      throw new Error('No valid data points found');
    }

    return {
      ...result,
      latestIndex,
      currentPrice: closes[latestIndex],
      timestamp: timestamps[latestIndex]
    };
  } catch (error) {
    console.error(`Error fetching data for ${symbol}:`, error.message);
    if (error.response) {
      console.error('Response data:', error.response.data);
      console.error('Response status:', error.response.status);
    }
    throw error;
  }
};

// Market Indices Endpoints
app.get('/api/market-indices', async (req, res) => {
  try {
    console.log('Fetching real-time market indices data...');

    const indexPromises = Object.entries(MARKET_INDICES).map(async ([name, symbol]) => {
      try {
        console.log(`Processing ${name} (${symbol})...`);
        const result = await fetchYahooFinanceData(symbol);
        const quote = result.indicators.quote[0];
        const latestIndex = result.latestIndex;

        // Get the most recent non-null values
        const currentPrice = result.currentPrice;
        const previousClose = result.meta.previousClose;
        const changePercent = ((currentPrice - previousClose) / previousClose) * 100;

        // Get the day's high and low from the entire day's data
        const validPrices = quote.close.filter(price => price !== null && price !== undefined);
        const dayHigh = Math.max(...validPrices);
        const dayLow = Math.min(...validPrices);

        // Get the opening price (first non-null value of the day)
        let openPrice = quote.open[0];
        for (let i = 1; i < quote.open.length; i++) {
          if (quote.open[i] !== null && quote.open[i] !== undefined) {
            openPrice = quote.open[i];
            break;
          }
        }

        console.log(`${name} data:`, {
          currentPrice,
          previousClose,
          changePercent,
          dayHigh,
          dayLow,
          openPrice,
          timestamp: new Date(result.timestamp * 1000).toISOString()
        });

        return {
          name,
          data: {
            price: currentPrice,
            change_percent: changePercent,
            timestamp: result.timestamp,
            open: openPrice,
            high: dayHigh,
            low: dayLow,
            volume: quote.volume[latestIndex] || 0,
            previousClose: previousClose
          }
        };
      } catch (error) {
        console.error(`Error processing ${name}:`, error.message);
        return {
          name,
          data: {
            price: 0,
            change_percent: 0,
            timestamp: 0,
            open: 0,
            high: 0,
            low: 0,
            volume: 0,
            previousClose: 0
          }
        };
      }
    });
```

```
const results = await Promise.all(indexPromises);

    // Convert array to object with index names as keys
    const indicesData = results.reduce((acc, { name, data }) => {
      acc[name] = data;
      return acc;
    }, {});

    console.log('Successfully fetched real-time market indices data');
    res.json(indicesData);
  } catch (error) {
    console.error('Error fetching market indices:', error.message);
    res.status(500).json({
      error: 'Failed to fetch market indices data',
      message: error.message
    });
  }
});

// NIFTY Indices Comparison Endpoint
app.get('/api/nifty-indices', async (req, res) => {
  try {
    console.log('Fetching NIFTY indices data...');

    const response = await axios.get('https://www.niftyindices.com/market-data/index-movers', {
      headers: {
        "User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safar
i/537.36"
,
        "Accept-Language": "en-US,en;q=0.9"
      }
    });

    const $ = cheerio.load(response.data);
    const indicesData = [];

    // Find the table containing index data
    $('table tbody tr').each((i, element) => {
      const indexName = $(element).find('td:nth-child(1)').text().trim();

      if (NIFTY_INDICES.includes(indexName)) {
        const currentValue = parseFloat($(element).find('td:nth-child(2)').text().replace(/,/g, ''));
        const change = parseFloat($(element).find('td:nth-child(3)').text().replace(/,/g, ''));
        const changePercent = parseFloat($(element).find('td:nth-child(4)').text().replace('%', ''));

        indicesData.push({
          name: indexName,
          currentValue,
          previousClose: currentValue - change,
          change,
          changePercent
        });
      }
    });

    // Add SENSEX data from Yahoo Finance
    try {
      const sensexData = await fetchYahooFinanceData('^BSESN');
      const quote = sensexData.indicators.quote[0];
      const currentPrice = quote.close[quote.close.length - 1];
      const previousClose = sensexData.meta.previousClose;
      const change = currentPrice - previousClose;
      const changePercent = (change / previousClose) * 100;

      indicesData.push({
        name: 'SENSEX',
        currentValue: currentPrice,
        previousClose,
        change,
        changePercent
      });
    } catch (error) {
      console.error('Error fetching SENSEX data:', error.message);
    }

    console.log('Successfully fetched NIFTY indices data');
    res.json(indicesData);
  } catch (error) {
    console.error('Error fetching NIFTY indices:', error.message);
    res.status(500).json({
      error: 'Failed to fetch NIFTY indices data',
      message: error.message
    });
  }
});

// Historical data endpoint with caching
app.get('/api/market-indices/historical/:symbol', async (req, res) => {
  try {
    const { symbol } = req.params;
    const { interval = '1d', range = '1mo' } = req.query;

    console.log(`Received request for historical data: symbol=${symbol}, interval=${interval}, range=${range
}`);

    // Format symbol for Yahoo Finance
    const formattedSymbol = symbol.includes('.') ? symbol : `${symbol}.NS`;

    // Validate parameters
    const validIntervals = ['1m', '2m', '5m', '15m', '30m', '60m', '90m', '1d', '5d', '1wk', '1mo', '3mo'];
    const validRanges = ['1d', '5d', '7d', '1mo', '3mo', '6mo', '1y', '2y', '5y', '10y', 'ytd', 'max'];

    if (!validRanges.includes(range)) {
      return res.status(400).json({
        error: 'Invalid range parameter',
        message: `Valid ranges are: ${validRanges.join(', ')}`,
        details: {
          '1d': '1 day (needed for minute-wise intervals)',
          '5d': '5 days',
          '7d': '7 days',
          '1mo': '1 month (suitable for 1d, 5d, 1wk)',
          '3mo': '3 months',
          '6mo': '6 months',
          '1y': '1 year',
          '2y': '2 years',
          '5y': '5 years',
          '10y': '10 years',
          'ytd': 'Year to date',
          'max': 'Max available history'
        }
      });
    }
```

```
if (!validIntervals.includes(interval)) {
      return res.status(400).json({
        error: 'Invalid interval parameter',
        message: `Valid intervals are: ${validIntervals.join(', ')}`,
        details: {
          '1m': '1 minute (works with ranges up to 7 days)',
          '2m': '2 minutes',
          '5m': '5 minutes',
          '15m': '15 minutes',
          '30m': '30 minutes',
          '60m': '1 hour',
          '90m': '1.5 hours',
          '1d': '1 day',
          '5d': '5 days',
          '1wk': '1 week',
          '1mo': '1 month',
          '3mo': '3 months'
        }
      });
    }

    // Validate interval-range compatibility
    if (interval === '1m' && !['1d', '5d', '7d'].includes(range)) {
      return res.status(400).json({
        error: 'Invalid interval-range combination',
        message: '1m interval only works with ranges up to 7 days (1d, 5d, 7d)'
      });
    }

    const cacheKey = getCacheKey(formattedSymbol, interval, range);
    const cachedData = cache.get(cacheKey);

    if (cachedData && Date.now() - cachedData.timestamp < CACHE_DURATION) {
      console.log(`Returning cached data for ${formattedSymbol}`);
      return res.json(cachedData.data);
    }

    console.log(`Fetching historical data for ${formattedSymbol} with interval=${interval}, range=${range}`);

    const response = await axios.get(
      `https://query1.finance.yahoo.com/v8/finance/chart/${formattedSymbol}?interval=${interval}&range=${range}`,
      {
        headers: {
          "User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/53
7.36"
,
          "Accept-Language": "en-US,en;q=0.9",
          "Accept-Encoding": "gzip, deflate, br",
          "Connection": "keep-alive"
        }
      }
    );

    if (!response.data.chart?.result?.[0]) {
      throw new Error('No data available for the given symbol');
    }

    const result = response.data.chart.result[0];
    const quote = result.indicators.quote[0];
    const timestamps = result.timestamp;

    // Process and format the data
    const formattedData = {
      meta: {
        currency: result.meta.currency,
        symbol: result.meta.symbol,
        exchangeName: result.meta.exchangeName,
        fullExchangeName: result.meta.fullExchangeName,
        instrumentType: result.meta.instrumentType,
        firstTradeDate: result.meta.firstTradeDate,
        regularMarketTime: result.meta.regularMarketTime,
        hasPrePostMarketData: result.meta.hasPrePostMarketData,
        gmtOffset: result.meta.gmtOffset,
        timezone: result.meta.timezone,
        exchangeTimezoneName: result.meta.exchangeTimezoneName,
        regularMarketPrice: result.meta.regularMarketPrice,
        fiftyTwoWeekHigh: result.meta.fiftyTwoWeekHigh,
        fiftyTwoWeekLow: result.meta.fiftyTwoWeekLow,
        regularMarketDayHigh: result.meta.regularMarketDayHigh,
        regularMarketDayLow: result.meta.regularMarketDayLow,
        regularMarketVolume: result.meta.regularMarketVolume,
        longName: result.meta.longName,
        shortName: result.meta.shortName,
        chartPreviousClose: result.meta.chartPreviousClose,
        priceHint: result.meta.priceHint,
        currentTradingPeriod: result.meta.currentTradingPeriod,
        dataGranularity: result.meta.dataGranularity,
        range: result.meta.range,
        validRanges: result.meta.validRanges
      },
      data: timestamps.map((timestamp, index) => ({
        timestamp,
        date: new Date(timestamp * 1000).toISOString(),
        open: quote.open[index],
        high: quote.high[index],
        low: quote.low[index],
        close: quote.close[index],
        volume: quote.volume[index],
        adjustedClose: result.indicators.adjclose?.[0]?.adjclose?.[index]
      }))
    };

    // Cache the formatted data
    cache.set(cacheKey, {
      data: formattedData,
      timestamp: Date.now()
    });

    res.json(formattedData);
  } catch (error) {
    console.error('Error fetching historical data:', error.message);
    if (error.response) {
      console.error('Response data:', error.response.data);
      console.error('Response status:', error.response.status);
    }
    res.status(500).json({
      error: 'Failed to fetch historical data',
      message: error.message,
      details: error.response?.data
    });
  }
});

// Proxy endpoint for Yahoo Finance API
app.get('/api/yahoo-finance/:symbol', async (req, res) => {
  try {
    const { symbol } = req.params;
    const { interval = '1d', range = '1mo' } = req.query;

    // Validate parameters
    if (!['1d', '1wk', '1mo', '3mo', '6mo', '1y', '2y', '5y', '10y', 'ytd', 'max'].includes(range)) {
      return res.status(400).json({ error: 'Invalid range parameter' });
    }

    if (!['1m', '2m', '5m', '15m', '30m', '60m', '90m', '1h', '1d', '5d', '1wk', '1mo', '3mo'].includes(interval
)) {
      return res.status(400).json({ error: 'Invalid interval parameter' });
    }

    console.log(`Fetching data for ${symbol} with interval=${interval}, range=${range}`);

    const response = await axios.get(
      `https://query1.finance.yahoo.com/v8/finance/chart/${symbol}?interval=${interval}&range=${range}`,
      {
        headers: {
          "User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/53
7.36"
,
          "Accept-Language": "en-US,en;q=0.9",
          "Accept-Encoding": "gzip, deflate, br",
          "Connection": "keep-alive"
        }
      }
    );
```

```
if (!response.data.chart?.result?.[0]) {
     throw new Error('No data available for the given symbol');
   }

   const result = response.data.chart.result[0];
   const quote = result.indicators.quote[0];
   const timestamps = result.timestamp;

   // Process and format the data
   const formattedData = {
     meta: {
       currency: result.meta.currency,
       symbol: result.meta.symbol,
       exchangeName: result.meta.exchangeName,
       instrumentType: result.meta.instrumentType,
       firstTradeDate: result.meta.firstTradeDate,
       regularMarketTime: result.meta.regularMarketTime,
       gmtoffset: result.meta.gmtoffset,
       timezone: result.meta.timezone,
       exchangeTimezoneName: result.meta.exchangeTimezoneName,
       regularMarketPrice: result.meta.regularMarketPrice,
       chartPreviousClose: result.meta.chartPreviousClose,
       previousClose: result.meta.previousClose,
       scale: result.meta.scale,
       priceHint: result.meta.priceHint,
       currentTradingPeriod: result.meta.currentTradingPeriod,
       tradingPeriods: result.meta.tradingPeriods,
       dataGranularity: result.meta.dataGranularity,
       range: result.meta.range,
       validRanges: result.meta.validRanges
     },
     data: timestamps.map((timestamp, index) => ({
       timestamp,
       date: new Date(timestamp * 1000).toISOString(),
       open: quote.open[index],
       high: quote.high[index],
       low: quote.low[index],
       close: quote.close[index],
       volume: quote.volume[index],
       adjustedClose: result.indicators.adjclose?.[0]?.adjclose?.[index]
     }))
   };

   res.json(formattedData);
 } catch (error) {
   console.error('Error fetching data:', error.message);
   if (error.response) {
     console.error('Response data:', error.response.data);
     console.error('Response status:', error.response.status);
   }
   res.status(500).json({
     error: 'Failed to fetch data from Yahoo Finance API',
     message: error.message,
     details: error.response?.data
   });
 }
});

// Stock data endpoint
app.get('/api/stock/:symbol', async (req, res) => {
  try {
    const { symbol } = req.params;
    const { interval = '1d', range = '1mo' } = req.query;

    // Format symbol for Yahoo Finance
    const formattedSymbol = formatSymbol(symbol);

    console.log(`Fetching stock data for ${formattedSymbol} with interval=${interval}, range=${range}`);

    const response = await axios.get(
      `https://query1.finance.yahoo.com/v8/finance/chart/${formattedSymbol}?interval=${interval}&range=${range}`,
      {
        headers: {
          "User-Agent":
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/53
7.36"
,
          "Accept-Language": "en-US,en;q=0.9",
          "Accept-Encoding": "gzip, deflate, br",
          "Connection": "keep-alive"
        }
      }
    );

    if (!response.data.chart?.result?.[0]) {
      throw new Error('No data available for the given symbol');
    }

    const result = response.data.chart.result[0];
    const quote = result.indicators.quote[0];
    const timestamps = result.timestamp;
    const closes = quote.close;

    // Get the latest non-null data point
    let latestIndex = timestamps.length - 1;
    while (latestIndex >= 0 && (closes[latestIndex] === null || closes[latestIndex] === undefined)) {
      latestIndex--;
    }

    if (latestIndex < 0) {
      throw new Error('No valid data points found');
    }
```

```
// Volatility Endpoint
app.get('/api/volatility', async (req, res) => {
  try {
    console.log('Fetching volatility data...');

    // Fetch VIX data
    const vixResponse = await axios.get('https://query1.finance.yahoo.com/v8/finance/chart/^VIX', {
      headers: {
        'User-Agent': 'Mozilla/5.0',
        'Accept': 'application/json'
      }
    });

    const vixData = vixResponse.data.chart.result[0];
    const vixCurrent = vixData.meta.regularMarketPrice;
    const vixPrevClose = vixData.meta.previousClose;
    const vixChange = ((vixCurrent - vixPrevClose) / vixPrevClose) * 100;

    // Fetch NIFTY and BANK NIFTY data for implied volatility
    const [niftyResponse, bankNiftyResponse] = await Promise.all([
      axios.get('https://query1.finance.yahoo.com/v8/finance/chart/^NSEI', {
        headers: {
          'User-Agent': 'Mozilla/5.0',
          'Accept': 'application/json'
        }
      }),
      axios.get('https://query1.finance.yahoo.com/v8/finance/chart/^NSEBANK', {
        headers: {
          'User-Agent': 'Mozilla/5.0',
          'Accept': 'application/json'
        }
      })
    ]);

    const niftyData = niftyResponse.data.chart.result[0];
    const bankNiftyData = bankNiftyResponse.data.chart.result[0];

    // Calculate historical volatility (20-day)
    const calculateHistoricalVolatility = (prices) => {
      const returns = [];
      for (let i = 1; i < prices.length; i++) {
        returns.push(Math.log(prices[i] / prices[i - 1]));
      }
      const mean = returns.reduce((a, b) => a + b, 0) / returns.length;
      const variance = returns.reduce((a, b) => a + Math.pow(b - mean, 2), 0) / returns.length;
      return Math.sqrt(variance * 252); // Annualized
    };

    const niftyPrices = niftyData.indicators.quote[0].close;
    const bankNiftyPrices = bankNiftyData.indicators.quote[0].close;

    const response = {
      current: vixCurrent / 100, // Normalize to 0-1 range
      historical: {
        day: calculateHistoricalVolatility(niftyPrices.slice(-2)) / 100,
        week: calculateHistoricalVolatility(niftyPrices.slice(-7)) / 100,
        month: calculateHistoricalVolatility(niftyPrices.slice(-30)) / 100
      },
      implied: {
        nifty: niftyData.meta.impliedVolatility || 0,
        bankNifty: bankNiftyData.meta.impliedVolatility || 0
      },
      vix: {
        current: vixCurrent,
        change: vixChange
      }
    };

    console.log('Volatility data fetched successfully');
    res.json(response);
  } catch (error) {
    console.error('Error fetching volatility data:', error);
    // Return mock data in case of error
    res.json({
      current: 0.35,
      historical: {
        day: 0.25,
        week: 0.30,
        month: 0.35
      },
      implied: {
        nifty: 18.5,
        bankNifty: 20.2
      },
      vix: {
        current: 15.5,
        change: -2.5
      }
    });
  }
});

// Test endpoint
app.get('/api/test', (req, res) => {
  res.json({ message: 'Proxy server is working!' });
});

// Error handling middleware
app.use((err, req, res, next) => {
  console.error('Server error:', err);
  res.status(500).json({
    error: 'Internal server error',
    message: err.message
  });
});

// 404 handler
app.use((req, res) => {
  res.status(404).json({
    error: 'Not Found',
    message: `The requested endpoint ${req.url} does not exist`
  });
});

app.listen(PORT, () => {
  console.log(`Proxy server running on port ${PORT}`);
  console.log(`Available endpoints:`);
  console.log(`- http://localhost:${PORT}/`);
  console.log(`- http://localhost:${PORT}/api/market-indices`);
  console.log(`- http://localhost:${PORT}/api/market-indices/historical/:symbol`);
  console.log(`- http://localhost:${PORT}/api/stock/:symbol`);
  console.log(`- http://localhost:${PORT}/api/stocks?symbols=SYM1,SYM2,SYM3`);
  console.log(`- http://localhost:${PORT}/api/test`);
  console.log(`- http://localhost:${PORT}/api/top-gainers`);
  console.log(`- http://localhost:${PORT}/api/top-losers`);
  console.log(`- http://localhost:${PORT}/api/52week-high`);
  console.log(`- http://localhost:${PORT}/api/52week-low`);
  console.log(`- http://localhost:${PORT}/api/trending-stocks`);
  console.log(`- http://localhost:${PORT}/api/volatility`);
});
```

CHAPTER 11
CONCLUSION AND FUTURE ENHANCEMENT

11.1 CONCLUSION

The development of this stock analysis platform marks the successful integration of finance and artificial intelligence. The project involved building an end-to-end system capable of real-time data processing, predictive analytics, sentiment evaluation, and interactive visualizations—all accessible through a responsive web interface.

Throughout the journey, I gained significant hands-on experience in:

- **Frontend development** using React, Vite, and TypeScript.
- **API integration** for live financial data and news sources.
- **Machine learning techniques**, particularly using LSTM for price prediction.
- **Natural Language Processing (NLP)** using tools like TextBlob and VADER for sentiment analysis.
- **State management and UX design** for a seamless user experience.

This multidisciplinary learning process not only enhanced my technical skills but also provided a practical understanding of how AI can be applied to real-world financial systems.

11.2 FUTURE ENHANCEMENTS

While the current platform achieves its foundational goals, several improvements are planned for the future:

- **Mobile Application Integration:** Building a companion mobile app to enable users to track stocks, view predictions, and receive alerts on the go.
- **Real-Time Trading Alerts:** Incorporating a system to trigger alerts based on model outputs or significant news sentiment shifts.
- **AI Chatbot Enhancement:** Expanding the chatbot's capabilities with better NLP understanding, memory retention, and possibly voice command support.
- **User-Customized Models:** Allowing users to adjust parameters, choose between different prediction models, or even retrain the model on personalized datasets.

These enhancements aim to make the platform more intelligent, accessible, and aligned with diverse user needs in a rapidly evolving financial landscape.

REFERENCES

[1] Katsurai, M., & Ono, S. (2019). *TrendNets: Mapping Emerging Research Trends from Dynamic Co-Word Networks via Sparse Representation*. arXiv. http://arxiv.org/pdf/1905.10960v2

[2] Kotseruba, I., Papagelis, M., & Tsotsos, J. K. (2021). *Industry and Academic Research in Computer Vision*. arXiv. http://arxiv.org/pdf/2107.04902v3

[3] Li, J., Phan, H., Gu, W., Ota, K., & Hasegawa, S. (2024a). *Fish-bone Diagram of Research Issue: Gain a Bird's-eye View on a Specific Research Topic*. arXiv. http://arxiv.org/pdf/2407.01553v2

[4] Li, J., Phan, H., Gu, W., Ota, K., & Hasegawa, S. (2024b). *Hierarchical Tree-Structured Knowledge Graph for Academic Insight Survey*. arXiv. http://arxiv.org/pdf/2402.04854v7

[5] Liu, S., Cao, J., Yang, R., & Wen, Z. (2023). *Generating a Structured Summary of Numerous Academic Papers: Dataset and Method*. arXiv. http://arxiv.org/pdf/2302.04580v1

[6] Hutto, C. J., & Gilbert, E. (2014). *VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text*. ICWSM.

[7] Maknickas, V., & Maknickas, A. (2019). *Stock Market Prediction using LSTM Recurrent Neural Network*. Procedia Computer Science, 160, 681–688.

APPENDIX A:

B.1 Tools and Technologies Used:

Category	Tools/Technologies
Frontend Development	React.js, TypeScript, Vite, Tailwind CSS, Recharts
Backend Development	Node.js, Flask (Python), FastAPI
Machine Learning	LSTM, Random Forest, Scikit-learn
Natural Language Processing	BERT, VADER, TextBlob, NLTK
Data Visualization	Plotly, ApexCharts, Chart.js
Database/Storage	LocalStorage, In-memory cache
Deployment	Docker, Heroku, Netlify
Development Tools	VS Code, GitHub, Cursor
APIs Used	Yahoo Finance API, NewsAPI

B.2 Abbreviations and Acronyms:

Acronym	Full Form
AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
LSTM	Long Short-Term Memory
API	Application Programming Interface
UI	User Interface
CNN	Convolutional Neural Network
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
PWA	Progressive Web App
SMA	Simple Moving Average
EMA	Exponential Moving Average

Acronym	Full Form
MACD	Moving Average Convergence Divergence
RSI	Relative Strength Index
CRUD	Create, Read, Update, Delete

B.3 Model Evaluation Metrics:

Metric	Description
MAE	Measures average absolute errors between predicted and actual values.
RMSE	Penalizes large errors more than MAE.
R^2 Score	Measures goodness of fit for regression models.
BLEU Score	Evaluates the quality of machine-generated text (for captioning).
Accuracy	Percentage of correct predictions (classification context).

B.4 Sample Chatbot Queries and Responses:

User Query	Chatbot Response
"What is the RSI of Infosys?"	"The RSI of Infosys is currently 62.4, indicating a bullish trend."
"Will TCS stock go up tomorrow?"	"Based on current trends, TCS shows a moderate upward momentum."
"Explain MACD"	"MACD is a momentum indicator showing the relationship between two EMAs."
"Show me the portfolio performance"	"Your portfolio has gained 5.3% this week, led by INFY and RELIANCE."

B.5 Sample Code Snippet (LSTM Model):

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(timesteps, features)))
model.add(Dropout(0.2))
model.add(LSTM(50))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mean_squared_error')
```