“Stock Market Analysis”

A

Project Report

submitted

in partial fulfillment

for the award of the Degree of

Bachelor of Technology

in Department of Computer Science and Engineering (AI)

Project Mentor:
Ms. Garima Garg
Assistant Professor

Submitted By :
Aaditya Sharma (21ESKCA002)
Ansh Tyagi (21ESKCA017)
Bhavya Gupta (21ESKCA032)

Department of Computer Science and Engineering(AI)
Swami Keshvanand Institute of Technology, M & G, Jaipur
Rajasthan Technical University, Kota
Session 2024-2025

Swami Keshvanand Institute of Technology,
Management & Gramothan, Jaipur
Department of Computer Science and Engineering(AI)

CERTIFICATE

This is to certify that Mr. Aaditya Sharma, a student of B.Tech (Computer Science & Engineering(AI)) 8th semester has submitted his Project Report entitled **"Stock Market Analysis"** under my guidance.

Mentor:	**Coordinator:**
Ms. Garima Garg	Mr. Sumit Kumar
Assistant Professor, CSE	Assistant Professor, CSE
Signature. . [illegible]	Signature. [illegible]

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur
Department of Computer Science and Engineering(AI)

CERTIFICATE

This is to certify that Mr. Ansh Tyagi, a student of B.Tech (Computer Science & Engineering(AI)) 8th semester has submitted his Project Report entitled **"Stock Market Analysis"** under my guidance.

Mentor:
Ms. Garima Garg
Assistant Professor, CSE
Signature.. [illegible]

Coordinator:
Mr. Sumit Kumar
Assistant Professor, CSE
Signature. [illegible]

Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur

Department of Computer Science and Engineering(AI)

CERTIFICATE

This is to certify that Mr. Bhavya Gupta, a student of B.Tech (Computer Science & Engineering(AI)) 8th semester has submitted his Project Report entitled **"Stock Market Analysis"** under my guidance.

Mentor:

Ms. Garima Garg

Assistant Professor, CSE

Signature....... Garima

Coordinator:

Mr. Sumit Kumar

Assistant Professor, CSE

Signature........ [illegible]

DECLARATION

We hereby declare that the report of the project entitled ”Stock Market Analysis” is a record of an original work done by us at Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur under the mentorship of ”Ms. Garima Garg” (Dept. of Computer Science and Engineering) and coordination of ”Mr. Sumit Kumar” (Dept. of Computer Science and Engineering). This project report has been submitted as the proof of original work for the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology (B.Tech) in the Department of Computer Science and Technology. It has not been submitted anywhere else, under any other program to the best of our knowledge and belief.

Team Members	**Signature**
Aaditya Sharma (21ESKCA002)	[illegible]
Ansh Tyagi (21ESKCA017)	[illegible]
Bhavya Gupta (21ESKCA032)	Bhavya

Acknowledgement

A project of such a vast coverage cannot be realized without help from numerous sources and people in the organization. We take this opportunity to express our gratitude to all those who have been helping us in making this project successful.

We are highly indebted to our faculty mentor Ms. Garima Garg. She has been a guide, motivator, and source of inspiration for us to carry out the necessary proceedings for the project to be completed successfully. We also thank our project coordinator Mr. Sumit Kumar for his co-operation, encouragement, valuable suggestions, and critical remarks that galvanized our efforts in the right direction.

We would also like to convey our sincere thanks to Dr. Mehul Mahrishi, HOD, Department of Computer Science and Engineering, for facilitating, motivating, and supporting us during each phase of the development of the project. Also, we pay our sincere gratitude to all the Faculty Members of Swami Keshvanand Institute of Technology, Management and Gramothan, Jaipur, and all our colleagues for their co-operation and support.

Last but not least, we would like to thank all those who have directly or indirectly helped and cooperated in accomplishing this project.

Team Members
Aaditya Sharma (21ESKCA002)
Ansh Tyagi (21ESKCA017)
Bhavya Gupta (21ESKCA032)

Abstract

This project aims to develop a stock market analysis system that predicts future stock prices using historical data. We compared three machine learning models Support Vector Machine (SVM), Random Forest (RF), and Long Short-Term Memory (LSTM) networks chosen for their ability to handle time series data and nonlinear relationships. To improve predictions, we incorporated technical indicators like RSI, MACD, and Bollinger Bands, which provide valuable into market momentum and volatility.

We built a backend using Flask to generate stock price predictions, exposing APIs for easy integration with a frontend. The mobile application, developed using Kotlin, displays live stock data for five selected tickers, showing key metrics such as opening and closing prices, daily highs/lows, trading volume, and trend graphs. This allows users to visualize and interpret complex market data in a straightforward manner.

The app also includes user-friendly features like secure login, a customizable watchlist, and a news section that gathers market headlines. Designed to be simple and responsive, the app provides an easy-to-navigate interface for tracking stocks and accessing real-time data. By combining machine learning, real-time data, and mobile development, this project offers a practical tool for both casual investors and professional traders.

Keywords: Stock Market, LSTM, SVM, Random Forest, Price Prediction, Android App, Real-Time Data, Machine Learning, Kotlin

Table of Content

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Problem Statement and Objective

Predicting stock prices is difficult because the market changes constantly. Prices are affected by many factors and can go up or down quickly. New investors often find it hard to understand charts and indicators, and even experienced traders struggle to find consistent trends in the data.

The project aims to create a Machine Learning model that uses past stock data to predict future prices. This model is connected to a mobile app, built using Kotlin for Android and a Flask-based backend, giving users an easy way to get stock price insights.

The mobile app lets users search for stock symbols, view future price trends, create personalized watchlists, and stay updated with the latest financial news. By providing real-time insights directly on users' mobile devices, the app helps investors make smarter decisions without needing advanced knowledge.

In summary, the objective of this project is to develop an LSTM-based model that leverages historical stock prices and technical indicators to generate accurate insights. These insights are then made easily accessible through a user-friendly mobile application, empowering users to make informed investment decisions.

- To built and trained an LSTM model using 10 years of historical stock data, focusing only on price changes, without using sentiment analysis or timestamp-based features.
- To improve accuracy, we have added well-known technical indicators like RSI, MACD, Bollinger Bands, and both SMA and EMA into the model.

- The mobile app is designed for a smooth experience, showing live prices, historical trends, and future predictions with clear, interactive visuals.
- The backend, built with Flask (Python), manages everything from data analysis to user authentication and communication with the app.
- To support both beginners and experienced traders, our app provides reliable insights, intuitive visualizations, and a user-friendly interface to help you make informed decisions.

1.2 Literature Survey / Market Survey / Investigation and Analysis

To create a useful and competitive solution, existing platforms were explored, and academic research was reviewed. This will helped us in understanding the current market, identifying limitations in existing systems, and creating a solution that offers real value, not just another interface for displaying stock data.

➤ **Market Research**

- Platforms like Zerodha, Upstox, Sharekhan, and Moneycontrol are popular in the Indian stock market. They offer real-time updates, advanced charts, and easy trade execution.
- These platforms are feature-rich, but they mainly focus on trading. They do not simplify complex data or help users understand market trends using smart prediction models.
- While powerful, the interfaces of these platforms can be overwhelming, especially for beginners who are just starting to explore the stock market without much guidance.
- Most of these platforms don't offer personalized insights or recommendations based on patterns in historical stock data, which is exactly where we step in.

➤ **Academic Research**

- In academic research, various Machine Learning (ML) techniques have been used for stock price analysis, including models like Support Vector Machines (SVM), Decision Trees, and Random Forests.
- Among these, deep learning models, especially Long Short-Term Memory (LSTM) networks, are promising because they can handle sequential data and learn long-term patterns.
- Studies show that combining LSTM models with technical indicators such as RSI, MACD, Bollinger Bands, and moving averages can reveal meaningful patterns in historical data and greatly improve prediction accuracy.
- These methods focus on identifying patterns within numerical data, without depending on external sources like news or social media, which makes them perfect for systems based on historical price data alone.

1.3 Introduction to the Project

This project combines Machine Learning and mobile app development to create a smart and user-friendly solution for stock market analysis. The LSTM-based model provides insights through a simple and easy-to-use Android interface, making stock analysis accessible for both new and experienced investors.

The app does more than just show data; it helps users stay connected to the market. With features like watchlists, real-time updates, and trend charts, users can track stocks and make informed decisions anytime, anywhere. The goal is to turn complex market data into easy-to-understand insights.

Built for the future, the app continuously learns and improves as more data becomes available. It is designed to be simple, scalable, and ready for new features, ensuring users always have a reliable and up-to-date financial tool in their pocket.

- The core of the system is the LSTM-based engine, trained on years of historical stock data to provide more accurate insights than traditional methods.

- To improve the quality of these insights, we've added popular technical indicators like RSI, MACD, Bollinger Bands, SMA, and EMA, helping users better understand market trends and momentum.
- The backend, built with Flask (Python), handles everything from processing data and analyzing it to managing users and communication with the frontend.
- On the frontend, we've used Kotlin to create a fast and responsive Android app focused on performance, clarity, and ease of use.
- Key features include user login, profile customization, watchlists, live stock tracking, and market trend charts all presented in a clean, modern mobile design.

1.4 Proposed Logic, Algorithms, and Architecture

➤ Model and Indicators

The core of our system is the LSTM model, which is a great choice for stock price analysis because it can process sequences and learn patterns over time. However, we are not just using raw prices. To make the insights smarter and more reliable, we've added a set of well-known technical indicators that give deeper insights into market behavior:

- **RSI (Relative Strength Index):** Helps identify whether a stock is overbought or oversold, making it easier to spot potential reversal points.
- **MACD (Moving Average Convergence Divergence):** Tracks momentum and highlights changes in trend direction, giving useful buy/sell signals.
- **Bollinger Bands:** Measure market volatility by plotting bands above and below a moving average. Price movements near these bands can hint at upcoming breakouts or reversals.
- **SMA/EMA (Simple and Exponential Moving Averages):** Smooth out price data to highlight trends over different timeframes, helping the model (and the user) focus on meaningful patterns rather than short-term noise.

➤ Architecture Overview

The architecture of this system is modular, scalable, and designed for real-world use cases. Here's a breakdown of how the key components interact:

- **Frontend:** The user-friendly Android app, built with Kotlin, offers a sleek, responsive interface that prioritizes both performance and simplicity.
- **Backend:** The backend is powered by a Flask-based API server, written in Python, which handles everything from generating predictions using the ML model to managing user authentication and data.
- **Database:** A secure SQL-based database stores important data such as user information, stock watchlists, historical predictions, and system logs ensuring that everything is organized and easy to access.
- **Deployment:** The backend can be hosted on cloud platforms like AWS or Heroku, while the mobile app is distributed through APKs or app stores, making it easy for users to access.

➤ Business Model and Technical Integration

For the stock market analysis platform, our business model is designed to be both scalable and adaptable. Here's how it works:

- **Freemium Model:** We offer basic market analysis tools for free, with premium options for advanced features like real-time alerts and detailed stock insights.
- **Subscription Model:** Users can subscribe for access to premium content, including advanced stock predictions and historical data analysis.
- **Broker API Integration:** We plan to integrate broker APIs, allowing users to execute trades directly from the app, streamlining their trading experience.
- **Data Analytics as a Service:** Businesses can access customized stock market analytics through a subscription-based service, leveraging our predictive models and market data.

1.5 Scope of the Project

This project aims to bridge the gap between complex stock market analysis and everyday investors by offering a mobile-based platform powered by Machine Learning. From a technical perspective, the focus is on analyzing stock trends using LSTM models, which are enhanced with key technical indicators like RSI, MACD, and Bollinger Bands. The system is designed to be scalable and secure, built with technologies like Flask for the backend, Kotlin for the Android app, and JWT tokens for secure user authentication. Real-time stock data is fetched using APIs like `yfinance`, ensuring that users always have the latest information.

The app is designed for a wide range of users from beginners just getting started in the stock market to experienced traders seeking smarter tools. It combines accurate stock analysis with a clean, user-friendly interface that makes it easy to use while providing valuable insights. In the future, this project has the potential to evolve into a full personal finance assistant, including features like portfolio tracking, educational modules, and trading integrations all available directly from a smartphone.

Chapter 2
Software Requirement Specification

2.1 Overall Description

This section provides a clear, high-level overview of our software system, explaining its purpose, how it works, and how different parts of the system interact. It also sets the context for the detailed requirements you'll find later in the document.

2.1.1 Product Perspective

Our stock market prediction app is a standalone mobile application built for Android. Using machine learning, the app predicts future stock prices based on both historical and real-time data. It connects to a Flask-based backend, which processes the data and returns predictions via a secure API.

- The app collects historical stock data and uses trained machine learning models to generate predictions.
- It combines real-time insights with simple, easy-to-understand visuals to help users grasp market trends.
- The backend and frontend are designed to work independently, making the system modular and scalable.
- The prediction models are regularly updated to improve their accuracy.
- The app enables users to track stocks, receive personalized predictions, and make more informed investment decisions.

2.1.1.1 System Interfaces

System interfaces define how different parts of the app interact with one another. Our app uses APIs, secure backend services, and database connections to ensure everything works smoothly.

- We use the `yfinance` API to fetch accurate and timely stock data.
- Flask API endpoints handle user authentication, data requests, and predictions.
- The database stores and retrieves user-related stock data efficiently.
- We use JWT tokens to secure API communications.
- Maintenance tools, like error logs and system health checks, ensure everything runs reliably.

2.1.1.2 User Interfaces

We've designed the app's user interface to be clean, intuitive, and mobile-friendly. It's easy for users to navigate through the app and access its features.

- Users can log in, view predictions, and manage a personalized watchlist.
- The app displays real-time stock charts, which update with a single tap.
- The layout adjusts to fit different screen sizes, ensuring compatibility with all Android devices.
- Interactive charts and tooltips help users better understand the data.
- Push notifications alert users to important stock changes or updates.

2.1.1.3 Hardware Interfaces

The hardware requirements depend on whether you're using the app as an end user or managing it as a developer or server admin.

- End users need an Android device with at least 2 GB of RAM and Android 10 or higher.
- Developers and admins should have a machine with a decent CPU and memory for efficient development.
- The backend is hosted on cloud-based servers, ensuring reliable performance and availability.
- Hardware requirements are optimized to make the system easy to deploy and scale.

2.1.1.4 Software Interfaces

We've chosen modern and reliable software tools to build the entire system, from the backend to the frontend and everything in between.

- The backend is built using Python 3.8+, with Flask as the web framework and Keras for developing and deploying machine learning models.
- The frontend of the mobile application is developed using Kotlin, and the application is built and tested using Android Studio to ensure a responsive and modern user experience.
- For database management, MySQL is used in the production environment to support scalability and high performance.
- Data processing and transformation are handled using libraries such as NumPy and Pandas.
- Matplotlib and Seaborn are used to generate visual reports, plots, and charts for effective data visualization and trend analysis.

2.1.1.5 Communications Interfaces

Secure and smooth communication is essential for our system. We make sure that all data shared between the app and the server is protected and handled appropriately.

- All communication is encrypted using HTTPS to ensure user privacy.
- We use JSON to send and receive data, as it's lightweight and easy to process.
- Our APIs follow REST architecture, making them easy to maintain and extend. The app supports cross-origin API calls to enable the frontend to work across different domains.
- System logs help track errors and detect any unauthorized activity.

2.1.1.6 Memory Constraints

Memory usage is optimized to ensure smooth performance across devices and services.

- The mobile app runs well on devices with at least 2 GB RAM.
- Development systems should have 8 GB RAM or higher; Jupyter notebooks require 16 GB.
- The backend API performs efficiently with 8 GB RAM.
- Production servers need 16 GB RAM for handling concurrent users and data.
- Lightweight ML models and regular cleanup of logs/files help minimize memory consumption.

2.1.1.7 Operations

Here's what daily use looks like for users and administrators.

- Users can sign in, sign up, view live stock prices for all tickers, and see future price predictions.

- Users can create and manage a personalized watchlist of stocks and stay updated with the latest stock market news directly from the app.
- Admins can update prediction models, upload new stock data, and monitor the system's performance through a dedicated admin dashboard.
- Logs are used to track model performance, data changes, and user activity to ensure smooth operation and facilitate troubleshooting.
- The system sends notifications for new predictions, model updates, and important announcements to keep users informed.
- Regular backups are conducted to ensure that critical data, such as user information and stock predictions, are securely stored and protected.

2.1.1.8 Project Functions

At its core, our app is focused on delivering accurate and real-time stock price predictions.

- Users can input a stock ticker, and the system will return the predicted stock price based on trained machine learning models.
- The app displays both historical stock trends and future price predictions through interactive graphs.
- The platform is designed to be extensible, allowing the future addition of features such as news integration.
- All user interactions, including stock searches and watchlist updates, are logged to help improve the app's performance.
- The app includes a secure sign-in and sign-up process for personalized stock tracking and prediction.

2.1.1.9 User Characteristics

We’ve designed the system with a wide range of users in mind, including beginners in stock trading.

- Users can get started with just a basic understanding of finance and mobile apps.
- Helpful tutorials and tooltips walk users through every feature.
- Accessibility options are included for users with visual or motor impairments.
- Built-in help and FAQs enhance the user experience.
- The app is designed to be user-friendly for both beginners and experienced users, with easy-to-understand features and tools for all levels.

2.1.1.10 Constraints

Every system has its limitations. Here are some known constraints we’re working to improve.

- Currently, the app only works on Android devices running version 10 or above.
- A stable internet connection is required for live predictions and data fetching.
- The accuracy of predictions depends on how complete and clean the data is.

2.1.1.11 Assumptions and Dependencies

There are a few assumptions we’re making for the system to function as expected.

- The `yfinance` API remains available for fetching real-time and historical stock data.
- Admins will regularly maintain the backend and keep the ML models updated.
- Backend hosting will be reliable and minimize downtime.
- Users will have access to mobile data or Wi-Fi to use the app’s features.

Chapter 3
System Design Specification

3.1 System Architecture

Our stock price prediction app follows a client-server architecture, built with two primary components: a mobile application (Android) and a Flask-based backend server. The server acts as the brain of the system, handling communication between the frontend and the machine learning model.
To make things more structured, we have broken down the architecture into three logical layers:

- **Presentation Layer:** This is the mobile app developed using Kotlin. It's what users interact with directly whether it's logging in, searching for stock tickers, viewing current prices, or requesting future price predictions.
- **Application Layer:** This is where the core logic lives. Our Flask backend receives requests from the app, handles user authentication, fetches data from financial APIs, runs predictions using the trained LSTM model, and communicates with the database.
- **Data Layer:** The structural MS SQL database is used to manage user profiles, prediction logs, and stock data. It ensures that everything is stored securely and can be accessed quickly when needed.

This layered design helps keep the system modular and easy to maintain. It also makes it more scalable and flexible if we ever need to add new features or connect to additional services.

3.2 Module Decomposition Description

To better organize functionality, we've divided the app system into distinct modules, each handling specific responsibilities:

- **Authentication Module:** This module handles user sign-up and login, ensuring secure access using JWT (JSON Web Token).
- **Stock Data Module:** It gets real-time and past stock data from sources like `yfinance`, processes it, and sends the cleaned data to the mobile app.
- **Prediction Engine:** This is where our trained LSTM model works. It processes data, scales it, and predicts future stock prices based on past trends.
- **User Interface Module:** This module in the mobile app manages everything the user interacts with, like searching for stocks, navigating between screens, viewing predictions, and managing the watchlist.
- **Database Interface Module:** This acts as a bridge between the app/server and the database. It stores user information, prediction requests, and watchlist data.

Each module is designed to work independently but fits seamlessly with the others. This approach makes development, debugging, and future updates much smoother.

3.3 High-Level Design Diagrams

The following diagrams help visualize the system's structure, workflows, and how data moves between different components.

3.3.1 Use Case Diagram

This diagram shows how users (both regular users and admins) interact with the system.

Actors:

- **User:** Typically an investor or trader who wants to explore stock market data and get predictions.

- **Admin:** Responsible for managing backend data, updating models, and ensuring the system runs smoothly.

Use Cases:

- **Collect Data (Admin):** Uploads stock data using CSV files, APIs, or manual input.
- **Compute Results (Admin):** Runs the ML model to generate predictions.
- **System Update (Admin):** Periodically updates models and software components.
- **View Trade Exchange (User):** Views current market data in real-time.
- **Companies Stock (User):** Looks up price history and details of listed companies.
- **View Predicted Outcome (User):** Sees future stock price predictions.

Relationships:

- Users interact with trade exchange data, company stock info, and predictions.
- Admins manage backend operations like data collection and system maintenance.

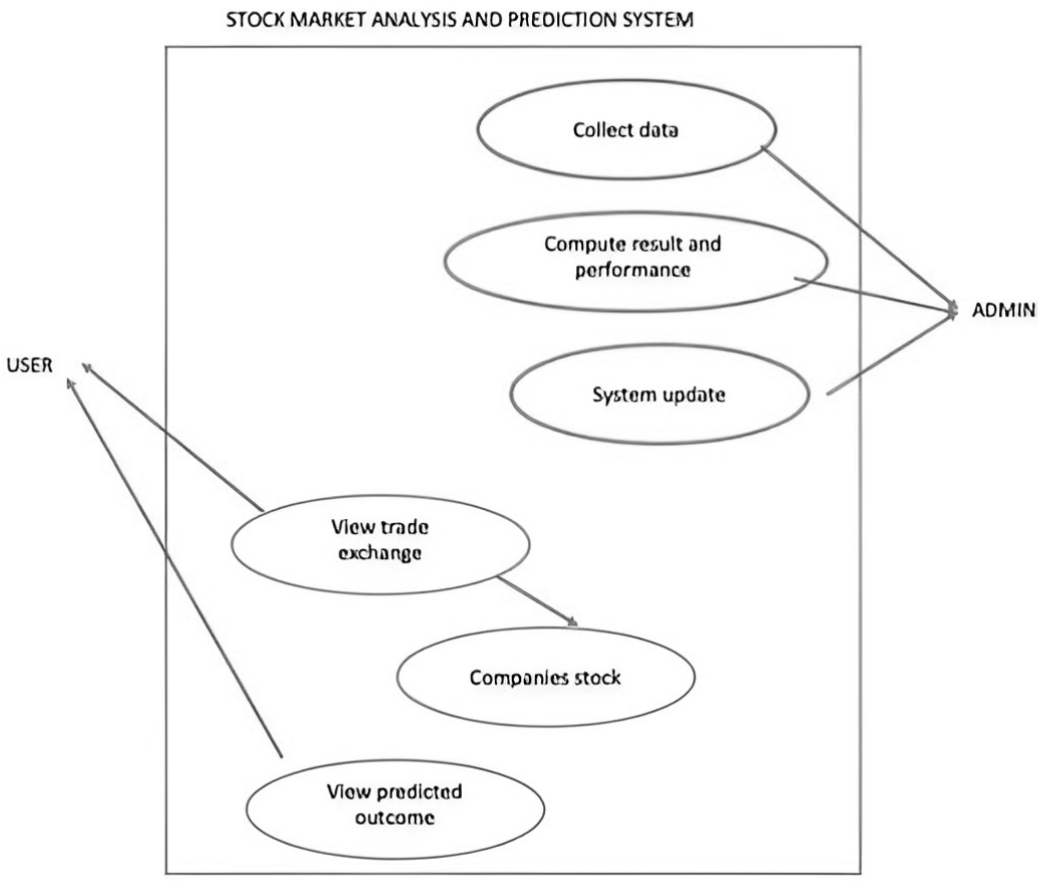


Figure 3.1: Use Case Diagram

3.3.2 Data-Flow Diagrams

These diagrams show how data flows through the system across various levels.

DFD Level 0:

- Shows the basic interaction between external users (User and Admin) and the overall Stock Prediction System.

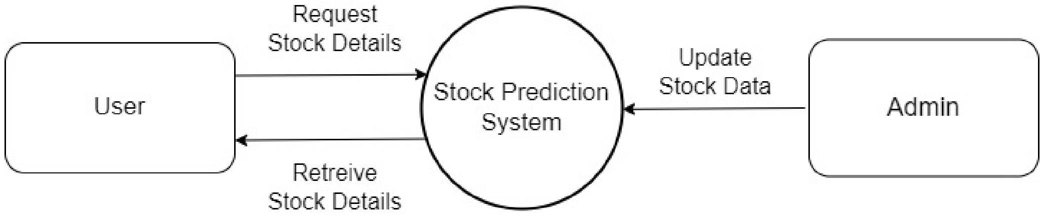


Figure 3.2: DFD - Level 0

DFD Level 1:

- Breaks down how internal processes work:
  - **Users:** Request stock-related data.
  - **ML Model:** Responds with analysis and predictions.
  - **User Watchlist:** Saves tickers and related data selected by the user.

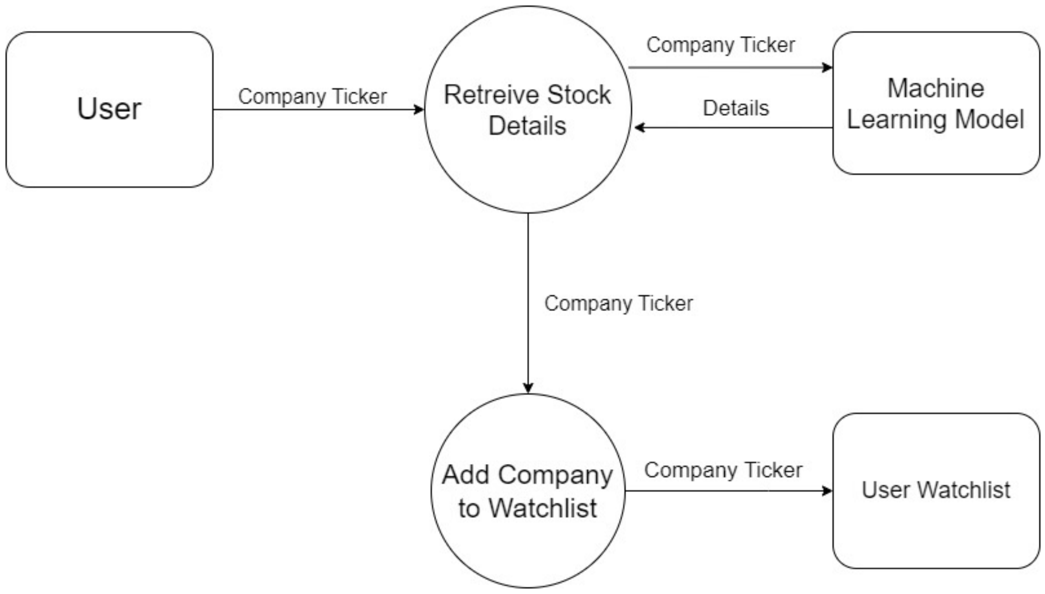


Figure 3.3: DFD - Level 1

DFD Level 2:

- The diagram shows how the system handles user interaction and stock data flow.
- The user enters a company ticker to retrieve stock details.
- These details are fetched from the machine learning model which performs the following steps internally:
  - Accepts stock data from a dataset.
  - Preprocesses the data to handle noise and missing values.
  - Extracts important features relevant for price prediction.
  - Performs analysis and predicts future stock prices.
- The user can then add the company to their watchlist for quick access later.

- The watchlist stores the selected company tickers for personalized tracking.

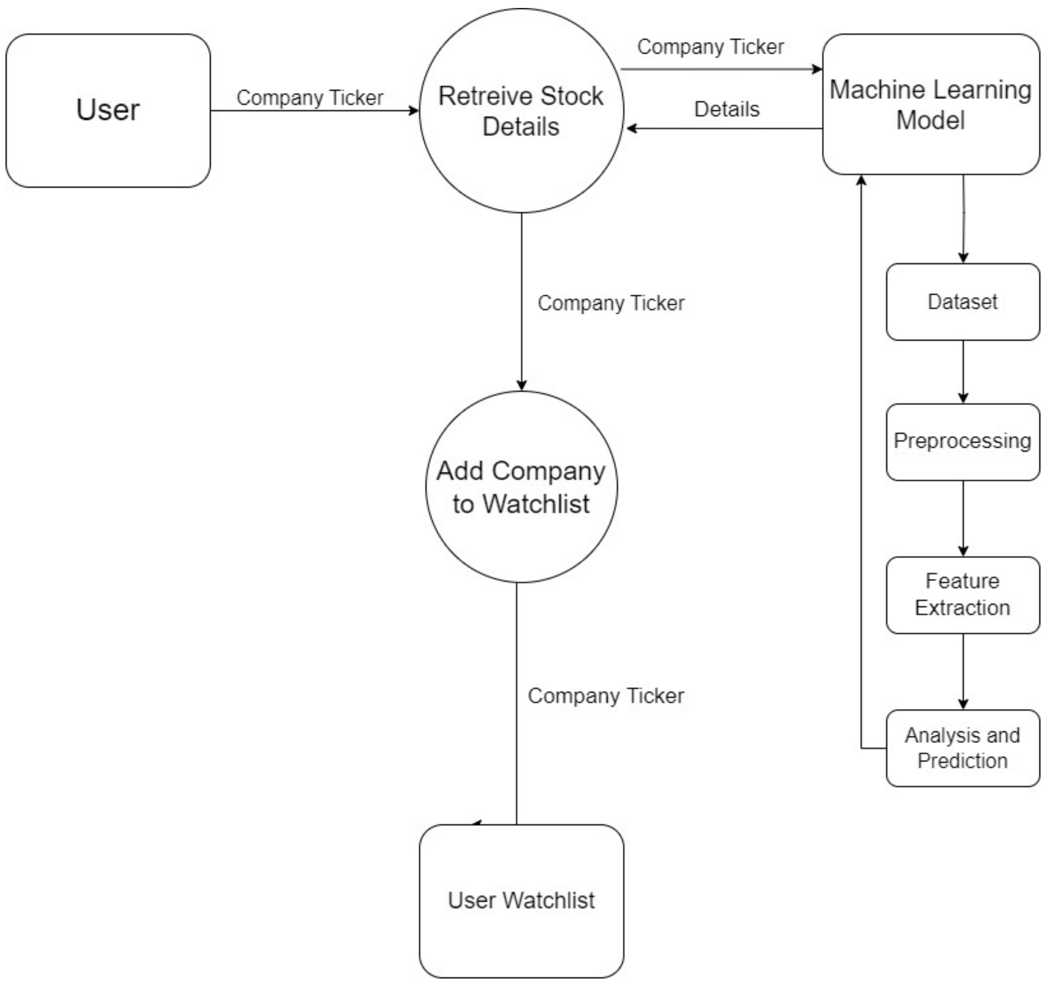


Figure 3.4: DFD - Level 2

3.3.3 Class Diagram

This diagram describes the object-oriented structure of the system.

- **Ticker:** Handles fetching of stock prices based on stock symbols and works with `StockManager`.
- **StockManager:** Updates stock information and runs the prediction model.
- **Stock:** Represents stock entities with basic info like symbol, price, highs, and lows.
- **StockHolding:** Tracks how many shares a user owns, along with cost and valuation.

- **Portfolio:** Contains all stocks held by a user.
- **WatchList:** Holds stocks that the user is monitoring but hasn't purchased yet.
- **PortfolioView / WatchListView / StockView:** UI components for rendering respective views in the app.

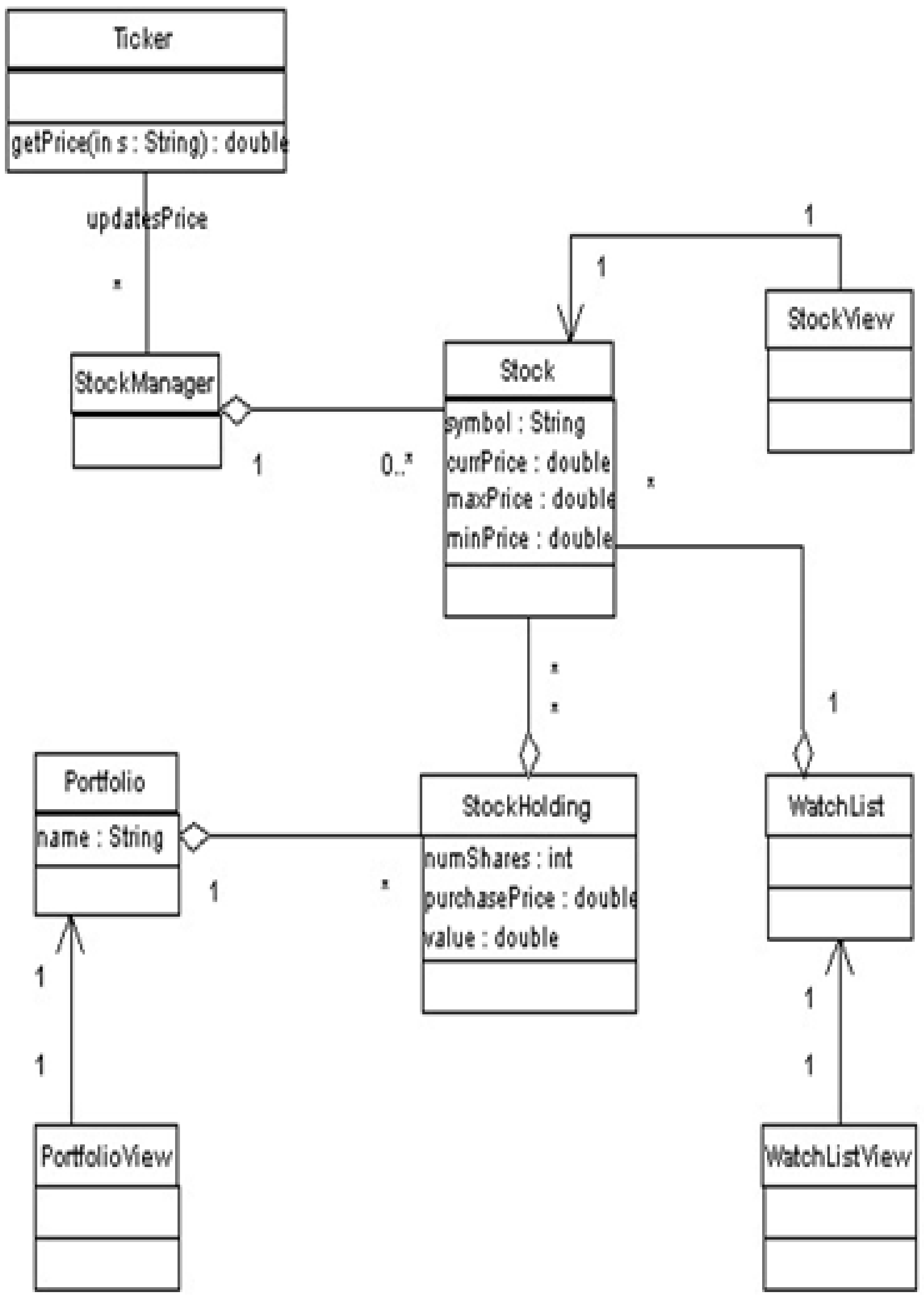


Figure 3.5: Class Diagram

3.3.4 Entity-Relationship (ER) Diagram

This diagram maps out how our database entities relate to each other.

Entities:

- **User:** Includes fields like UserID, Name, Password, and Account Creation Date.
- **Watchlist:** Contains StockID, UserID, StockName, and Ticker info.

Relationships:

- It's a many-to-many relationship: users can follow multiple stocks, and the same stock can be followed by many users.

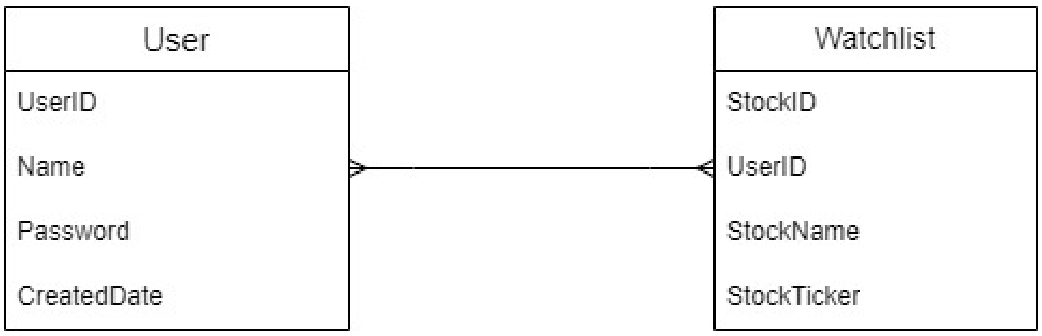


Figure 3.6: Entity Relationship Diagram

3.3.5 Sequence Diagram

This sequence diagram shows how different parts of the system talk to each other during a typical prediction use case.

Actors:

- **User:** Initiates tasks like loading stock data and requesting predictions.
- **System:** Processes data, performs ML tasks, and prepares responses.
- **Database:** Handles data storage and retrieval.

Message Flow:

- **Load Data:** User uploads or selects data.

- **Pre-process Data:** System cleans and prepares it.
- **Store:** Data is saved in the database.
- **Data Analysis:** ML model generates prediction.
- **View Prediction:** Prediction results are sent to the user.
- **Visualizer:** Graphs and charts are displayed.
- **Save Stock:** If user opts in, the selected stock is stored in the database.

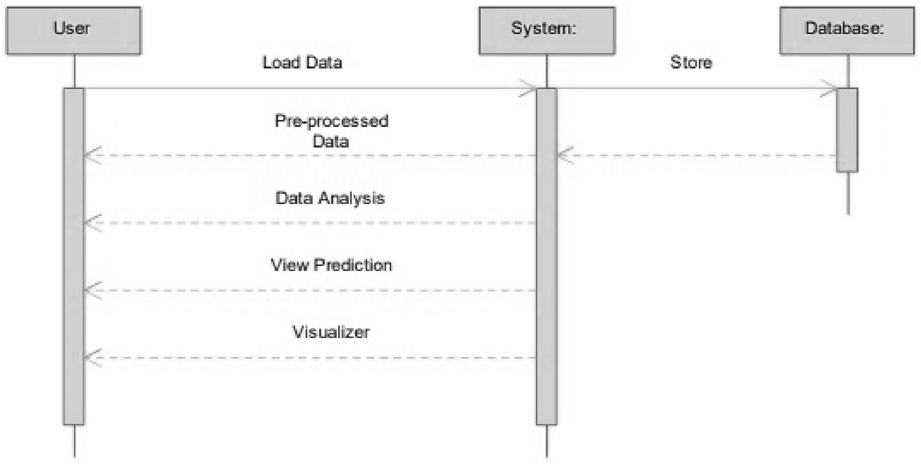


Figure 3.7: Sequence Diagram

Chapter 4
Methodology and Team

4.1 Introduction to Waterfall Framework

The Waterfall Model is a traditional, linear project management approach commonly used in software development. It is characterized by distinct, sequential phases—requirements gathering, design, implementation, testing, and maintenance—where each phase must be completed before moving to the next. This model is most effective when the project requirements are clearly defined and unlikely to change, offering a structured and systematic approach to development.

The Waterfall Model used in the Stock Market Analysis App follows a step-by-step process. It is particularly effective for projects requiring clear planning and reliable results. Each stage—such as gathering requirements, designing, building, testing, and launching—is executed one after the other in a fixed order. This method ensures that each phase is completed thoroughly before moving on to the next, minimizing risks and errors. With clear goals and proper documentation at each phase, tracking progress, verifying work, and addressing issues early is simplified. This approach leads to a well-organized app that is accurate and easy to maintain.

In this approach, the software development process is divided into distinct, separate phases. The process includes phases such as Requirement Analysis, System Design, Coding, Testing, Deployment, and Maintenance. Each phase is dependent on the successful completion of the previous one, and typically, there is no backtracking or revisiting earlier phases once they are finished. This clear, structured approach ensures that each phase's deliverables are well-defined and measurable.

Following is a diagrammatic representation of different phases of waterfall model.

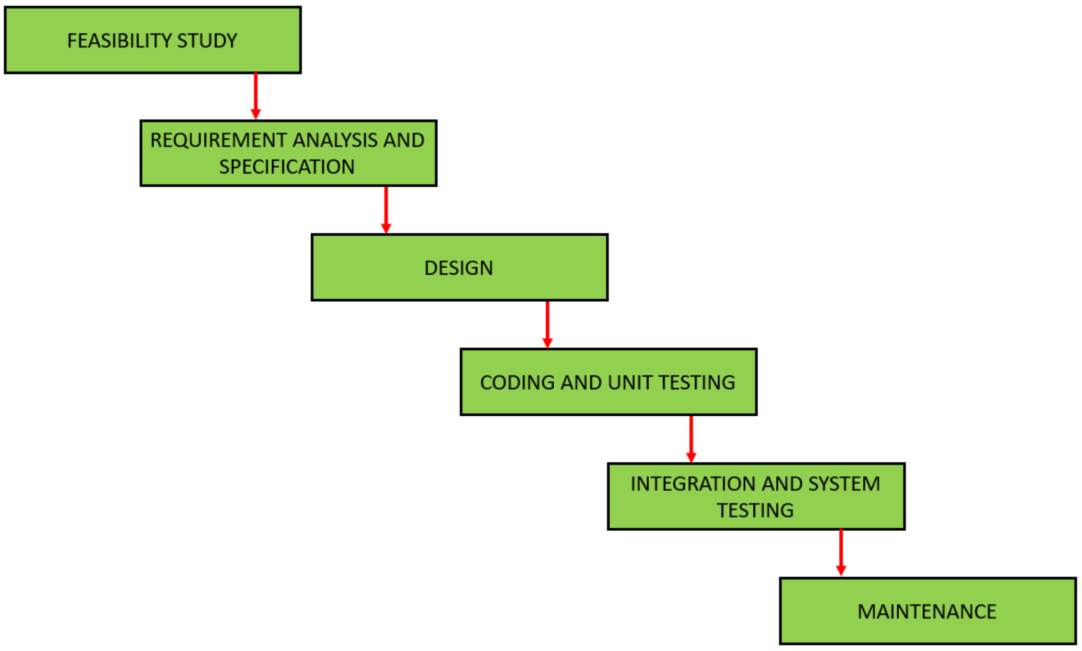

FEASIBILITY STUDY
REQUIREMENT ANALYSIS AND SPECIFICATION
DESIGN
CODING AND UNIT TESTING
INTEGRATION AND SYSTEM TESTING
MAINTENANCE

Figure 4.1: WaterFall model

1. **Requirement Gathering and Analysis:**

   During the requirement gathering and analysis phase, we focused on identifying both functional and non functional requirements. We defined user needs such as stock market analysis, app responsiveness, and authentication. Data sources, including Yahoo Finance, were finalized for real time data. Functional goals included prediction accuracy and mobile compatibility. We also performed a detailed analysis of user expectations, ensuring that all requirements were aligned with the needs of our target audience.

2. **System Design:**

   During the system design phase, we focused on creating the system architecture and planning the interaction between system components. Diagrams such as ER models, DFDs, and system architecture diagrams were developed. The design outlined how the frontend, backend, and machine learning modules would interact. We placed special emphasis on modularity and scalability. Additionally, we reviewed potential risks and designed the system to accommodate future expansion needs.

3. **Implementation:**
   During the implementation phase, we began by training different models such as LSTM, SVR, and Random Forest. We created backend APIs using Flask, and a mobile app interface was designed. Each part was developed in line with the earlier design. The codebase was managed using version control to ensure collaboration. Testing scripts were also written during this phase. The implementation phase also involved integrating various tools and technologies, ensuring that the system components worked together seamlessly.

4. **Integration and Testing:**
   During the integration and testing phase, we combined all components to ensure they functioned correctly as a whole. We used unit testing and integration testing to verify proper module interactions. The ML models were evaluated using RMSE, MAE, and R2 metrics. We also validated API calls, user authentication, and real time updates. Testing helped identify and resolve bugs, ensuring system stability. This phase also included load testing to ensure the system could handle multiple users without performance degradation.

5. **Deployment of System:**
   In the deployment phase, we focused on releasing the complete system into a live environment. The trained LSTM model was hosted via a Flask backend, which communicated with the mobile app. The app was published and linked with real time prediction APIs. Authentication was handled using Firebase or JWT tokens. The system became accessible for user testing and feedback. During deployment, monitoring tools were put in place to track the system's performance and address any initial issues that might arise.

6. **Maintenance:**
   During the maintenance phase, we ensure the system continues to perform effectively after deployment. This includes retraining models with new stock data and optimizing backend performance. We address bugs and security issues as they arise. User feedback helps us identify new features to add. Regular updates keep the app functional and up to date with market trends.

All these phases are cascaded into each other, with progress flowing steadily downwards (like a waterfall) through the phases. The next phase begins only after the defined set of goals for the previous phase are achieved and signed off, which is why it's called the "Waterfall Model." In this model, the phases do not overlap.

Waterfall Model Pros & Cons

Advantages of Waterfall Model:

- It's a clear, structured process that's easy to manage with a sequential flow.
- The stages are well documented, making future reference easier.
- It's ideal for projects with clearly defined requirements and no expected changes.
- Works well for smaller projects or when the scope is fixed.
- Each phase has specific deliverables, which makes tracking the project's progress simpler.

Disadvantages of Waterfall Model:

- The changes can be difficult to implement once development has started.
- It's not ideal for projects that need iterative feedback or flexibility.
- Since customer feedback is limited, issues might not be discovered later on.
- It's harder to manage when requirements change or the scope evolvement during the project.
- Late stage errors can be costly to fix because of the lack of early testing.

4.2 Team Members, Roles & Responsibilities

Our team comprised three members, each contributing their unique expertise to different facets of the stock market prediction system. Below are the assigned roles and detailed contributions for each team member:

- **Aaditya Sharma (21ESKCA002) – Backend and API Integration:**
  In this role, Aaditya Sharma is responsible for building the backend system of the application using Flask. His main task is to integrate the trained machine learning models with a web based interface, allowing users to access real time stock predictions. He sets up secure API routes, manages user authentication with tokens, and ensures that the backend handles prediction requests and data display smoothly. His work enables users to interact with the app and receive accurate stock price efficiently.

- **Ansh Tyagi (21ESKCA017) – Android App Development:**
  In this role, Ansh Tyagi is responsible for developing the Android application, focusing on both the design and functionality of the app. He creates an intuitive user interface to display predicted stock prices and other relevant information in a clean and organized manner. Ansh also ensures that the app communicates seamlessly with the backend, ensuring users receive up to date predictions. He makes efforts to optimize the app's performance across various screen sizes, keeping the design simple and user friendly.

- **Bhavya Gupta (21ESKCA032) – Data Analysis and Machine Learning:**
  In this role, Bhavya Gupta is responsible for handling the machine learning aspect of the project. He trains several models, including LSTM, SVR, and Random Forest, to predict stock prices based on historical data. Bhavya carefully tunes these models and ensures the data is split appropriately to avoid overfitting, which could skew predictions. Special features are engineered to help the models understand price trends and market patterns. After testing the models with various accuracy metrics, the LSTM model proved to be the most effective. This model is then saved and connected to the backend, allowing it to provide real time predictions within the mobile app.

Chapter 5
Centering System Testing

5.1 Functionality Testing

Functionality testing has been carried out throughout the development process to ensure the Android application behaved as expected under typical usage scenarios. Testing focused on core features related to user interaction, data retrieval, and system integration.

Throughout the development process, the core features of the Android app were regularly tested to ensure smooth operation in real-world usage. Testing primarily focused on user interactions, data retrieval, and the integration of different system components.

- **User Authentication:** We thoroughly tested the login and registration processes to make sure they communicate correctly with the backend. When valid credentials were entered, users could log in seamlessly. Invalid inputs, on the other hand, triggered clear and helpful error messages. We also verified that session handling and logout features worked without any issues.
- **Data Retrieval & Display:** We tested how the app fetches and displays stock prediction data from the backend. The data shown in the app consistently matched what we expected, and in cases where data wasn't available or there were connection issues, the app handled it gracefully with fallback responses.
- **Form Validation & Input Handling:** We paid close attention to how the app handles user input. All fields were tested for proper format and completeness. If someone tried to submit incomplete or incorrect information, the app provided clear instructions to resolve the issue easily.
- **API Integration:** Every API call, whether for logging in, fetching data, or

showing predictions, was tested to make sure it was sending the right requests and handling the responses properly. We also checked for edge cases like slow responses, server errors, or unexpected data formats to ensure the app stayed stable and responsive.

- **Error Handling:** To make the app more reliable, we tested how it responds to different types of failures like entering wrong data or when the server is unavailable. In each case, the app showed user-friendly messages and didn't crash or freeze, which is important for a smooth user experience.

5.2 Performance Testing

While formal, large-scale load and stress testing using specialized tools was not conducted within the scope of this project, the system's performance was observed under typical single-user conditions during functional testing.

- **Responsiveness:** We checked how quickly the main screens like the Home screen and Explore screen loaded during regular use. The response times were consistently within a comfortable range, making the app feel smooth and interactive.
- **Data Retrieval:** The app's ability to fetch and display prediction results and stock information was tested using sample data. We found that the responses from the backend were quick and the information displayed on the screen without noticeable delays.
- **Resource Usage:** During manual testing, the app showed no unusual spikes in memory or CPU usage on the client side. It ran smoothly on standard Android devices, and all features worked well without causing slowdowns.

For broader deployment with many users, deeper performance tuning and scalability testing would be considered.

5.3 Usability Testing

Usability of the Android application was evaluated iteratively during development through informal testing sessions, developer walkthroughs, and by following established mobile UI/UX design standards.

- **Navigation:** The app was designed with a simple and intuitive navigation structure. Users could easily access key features like checking predicted stock prices, exploring stock details, or managing their profile. The bottom navigation bar and smooth screen transitions made the app easy to get around without confusion.
- **Interface Clarity:** We made sure buttons, input fields, and data displays were clean, readable, and easy to interact with. Important actions like logging in or requesting a prediction were clearly labeled and stood out visually. We also added error messages and loading indicators to keep users informed during interactions.
- **Responsiveness:** The app was tested on different Android devices and screen sizes to make sure it looked good and worked properly everywhere. On most standard devices, everything ran smoothly. For older or lower-end phones, we may explore some layout adjustments in the future to improve the experience further.
- **Accessibility:** While full accessibility testing tools like TalkBack were not used, we have followed some best practices. The app shows adherence to these practices, with touch targets kept large enough, readable font sizes used, and good color contrast maintained. There is still room for improvement, and full accessibility compliance will be a goal for future updates.

Chapter 6
Test Execution Summary

This chapter presents a detailed summary of the test execution activities performed during the verification phase of the project. It outlines the outcomes of individual test cases designed to validate both functional and non-functional requirements of the application. Comprehensive testing was essential to ensure that the developed system aligns with the defined specifications and is capable of handling real-world scenarios with accuracy and efficiency.

The primary objective of this testing phase was to ensure that all critical workflows, particularly user authentication, registration, and dashboard functionalities, behave as expected under normal and edge-case scenarios. These test cases focused not only on typical user behavior but also on boundary conditions, erroneous inputs, and security-related validations such as unauthorized access attempts and session management under load.

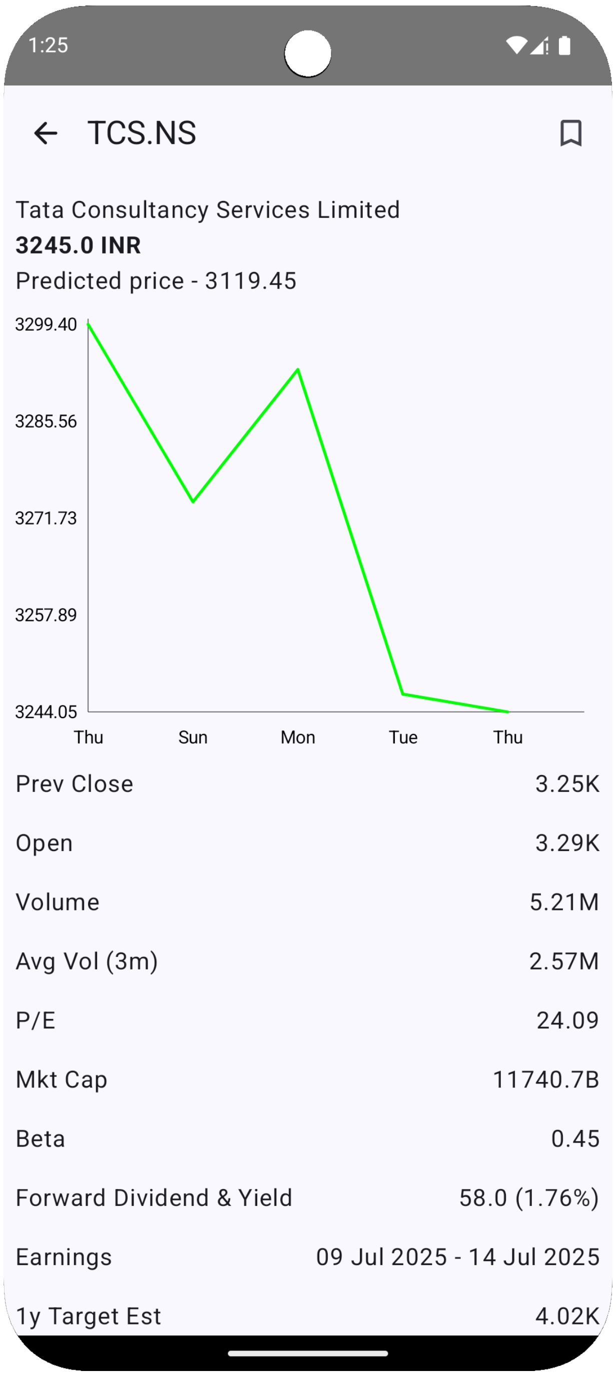
Each test case listed below includes a unique identifier, a brief description of the intended test scenario, the execution result, and the specific resources or inputs required. Test cases were designed to cover a broad spectrum of user actions including form submissions, navigation, data updates, and backend validations. Special emphasis was placed on scenarios involving invalid data input, network delays, and server-side exceptions to ensure the system could gracefully handle adverse conditions.

All test cases were executed in a controlled environment using valid and invalid inputs to simulate real-world user interactions. The testing process involved both manual and automated test procedures, with continuous monitoring of logs and system behavior. Various tools were employed for functional testing, integration testing, and load testing to confirm the reliability and responsiveness of the application. The results indicate a successful execution of all scenarios with no critical failures observed during the process.

S.No	Test Case ID	Test Case Description	Test Case Status	Resources Consumed
1	TC001	Verify successful login with valid registered email and correct password.	Passed	Valid email, Valid password
2	TC002	Verify login failure with incorrect password.	Passed	Valid email, Invalid password
3	TC003	Verify login failure with unregistered email.	Passed	Invalid email, Any password
4	TC004	Verify registration with new valid email and password.	Passed	New email, Valid password
5	TC005	Verify error message on registration with already registered email.	Passed	Registered email, Any password
6	TC006	Verify password field hides characters during typing.	Passed	Any password input
7	TC007	Verify user can log out successfully and is redirected to the login screen.	Passed	Logged-in session, Logout action
8	TC008	Verify stock prediction is displayed after entering valid stock symbol.	Passed	Valid stock symbol
9	TC009	Verify real-time stock details are fetched and displayed correctly.	Passed	Valid stock symbol
10	TC010	Verify app handles no internet connection gracefully.	Passed	Network disabled
11	TC011	Verify app displays loading indicator during data fetch.	Passed	Valid symbol, Fetch action
12	TC012	Verify form validation prevents submission of empty login fields.	Passed	Empty email/password fields
13	TC013	Verify stock details UI adjusts correctly on different screen sizes.	Passed	Device with small and large screen
14	TC014	Verify navigation between Home and Prediction screens works properly.	Passed	Tap on navigation buttons
15	TC015	Verify app retains session while switching between apps.	Passed	Logged-in state, App switch
16	TC016	Verify app terminates session after logout and blocks back navigation.	Passed	Logout, Back button pressed
17	TC017	Verify app launches to login screen when not authenticated.	Passed	No active session
18	TC018	Verify user input is cleared after successful submission.	Passed	Stock search form, Submit action

Table 6.1: Test Case Execution Summary

Chapter 7
Project Screen Shots



1:25

TCS.NS

Tata Consultancy Services Limited

3245.0 INR

Predicted price - 3119.45

Prev Close	3.25K
Open	3.29K
Volume	5.21M
Avg Vol (3m)	2.57M
P/E	24.09
Mkt Cap	11740.7B
Beta	0.45
Forward Dividend & Yield	58.0 (1.76%)
Earnings	09 Jul 2025 - 14 Jul 2025
1y Target Est	4.02K

Figure 7.1: TCS.NS Dashboard Price Page

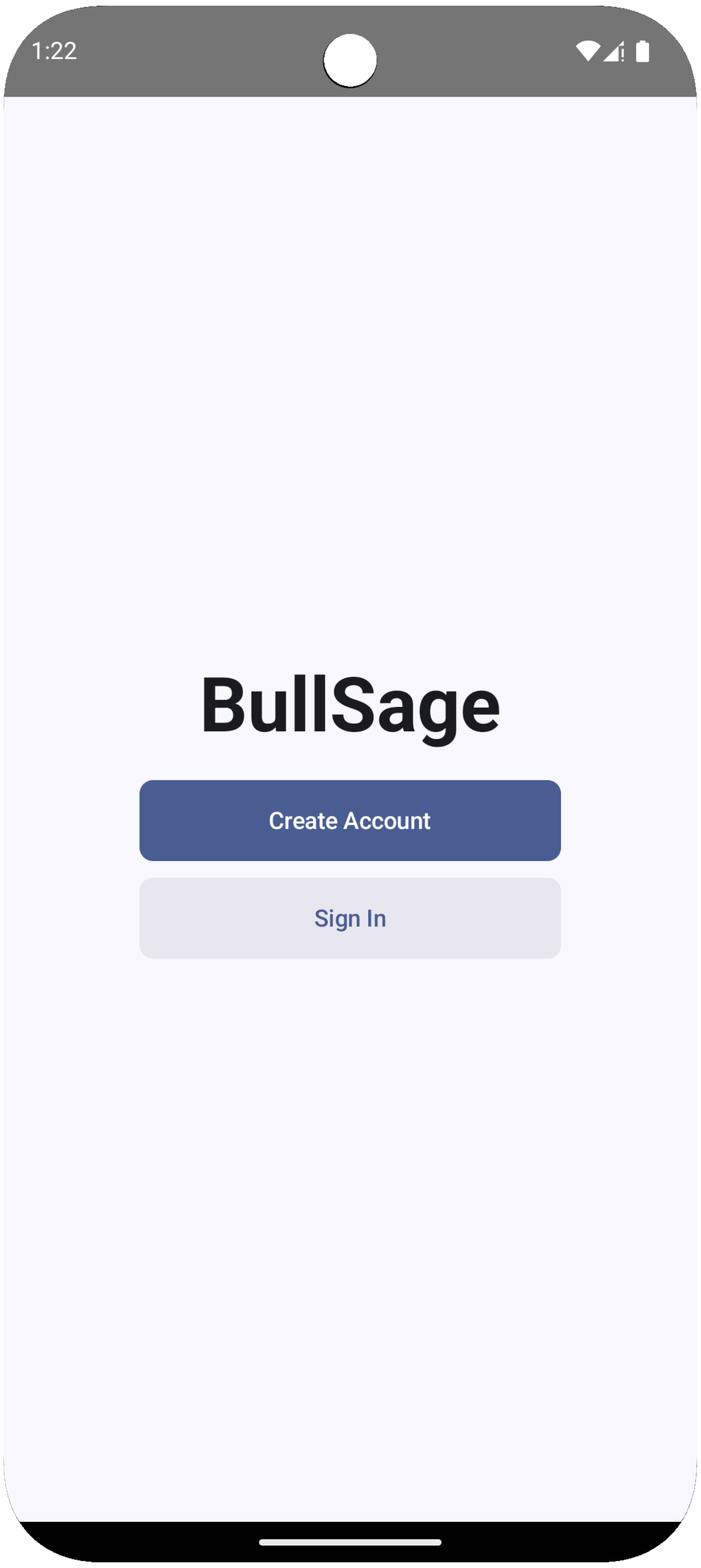
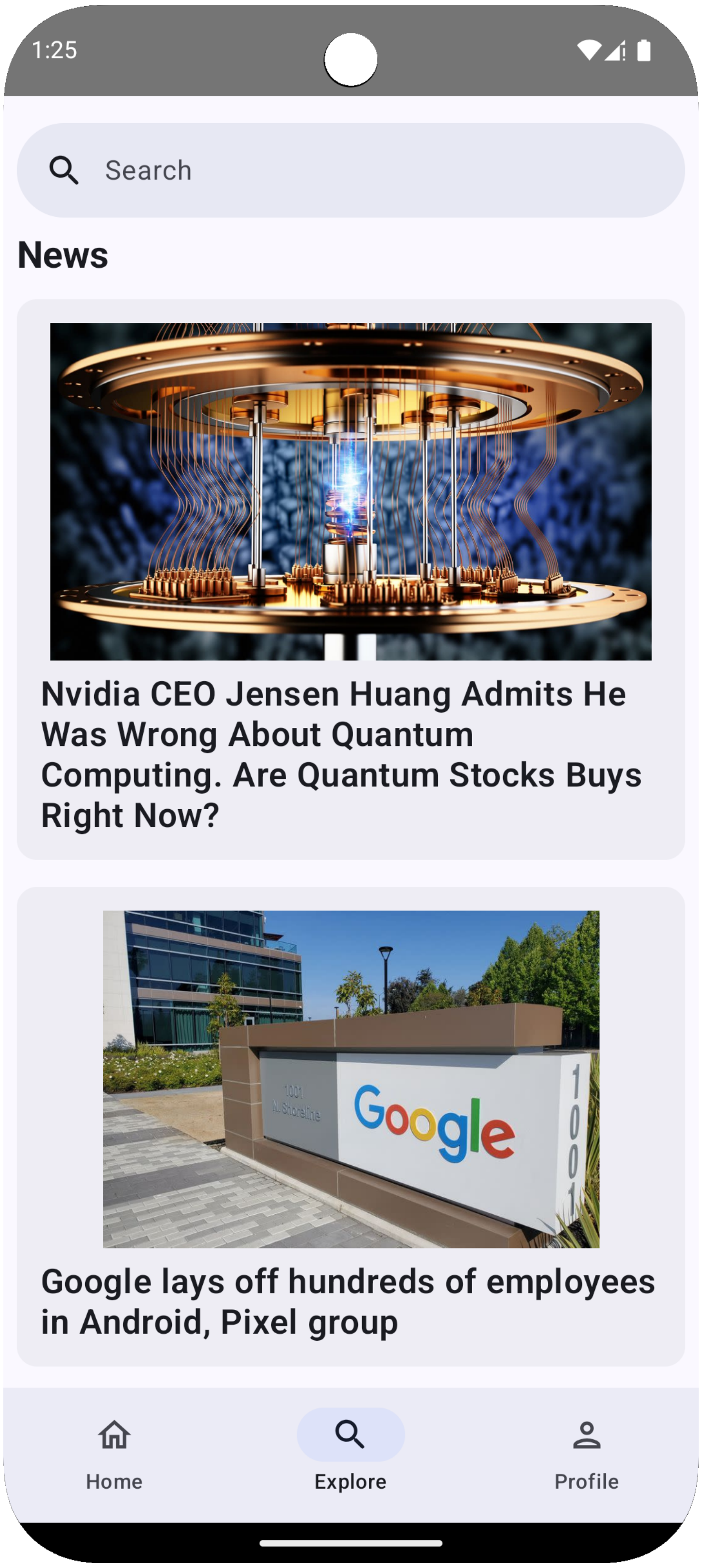


Figure 7.2: Main Page


1:25

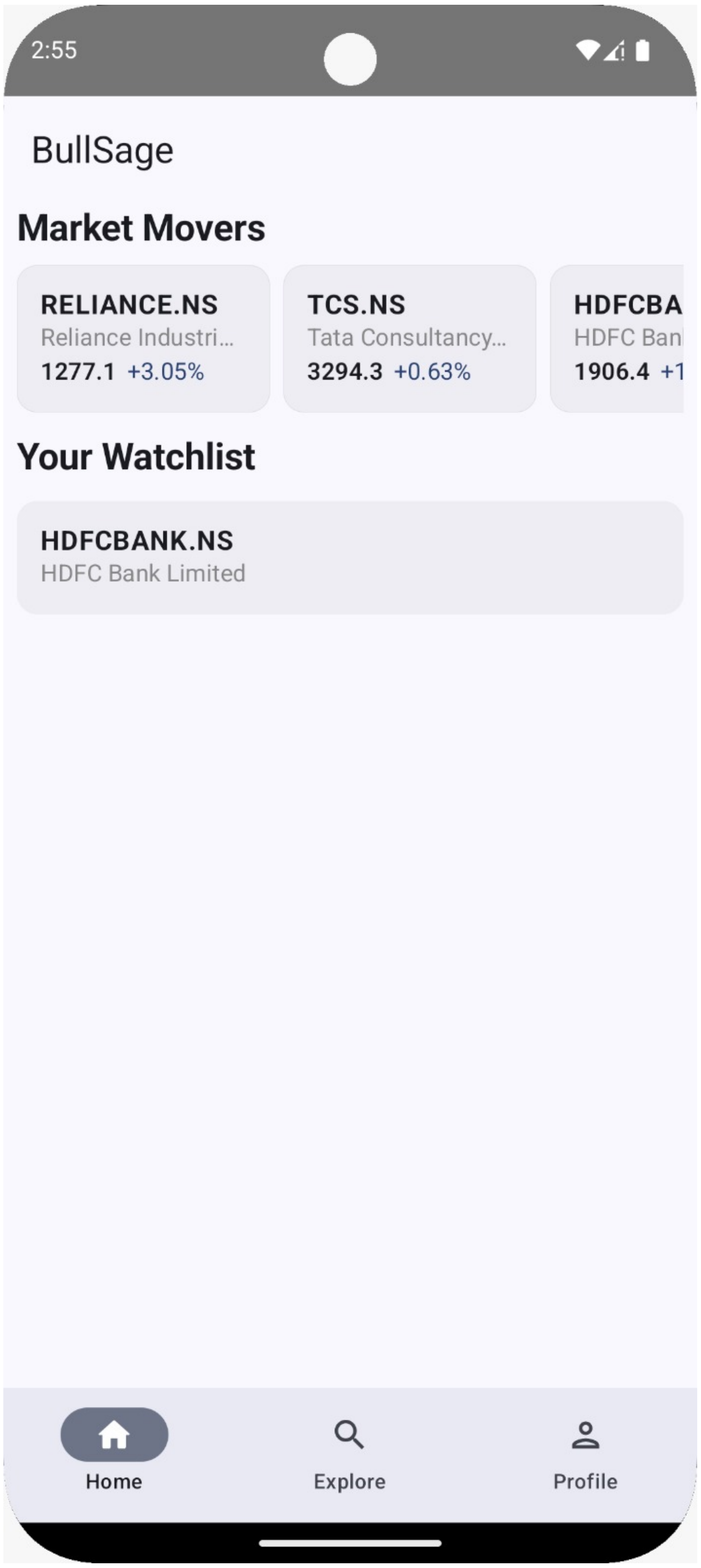
Search

News

Nvidia CEO Jensen Huang Admits He Was Wrong About Quantum Computing. Are Quantum Stocks Buys Right Now?

Google lays off hundreds of employees in Android, Pixel group

Figure 7.3: News Page


2:55

BullSage

Market Movers

RELIANCE.NS
Reliance Industri...
1277.1 +3.05%

TCS.NS
Tata Consultancy...
3294.3 +0.63%

HDFCBA
HDFC Ban
1906.4 +1

Your Watchlist

HDFCBANK.NS
HDFC Bank Limited

Figure 7.4: Home Page

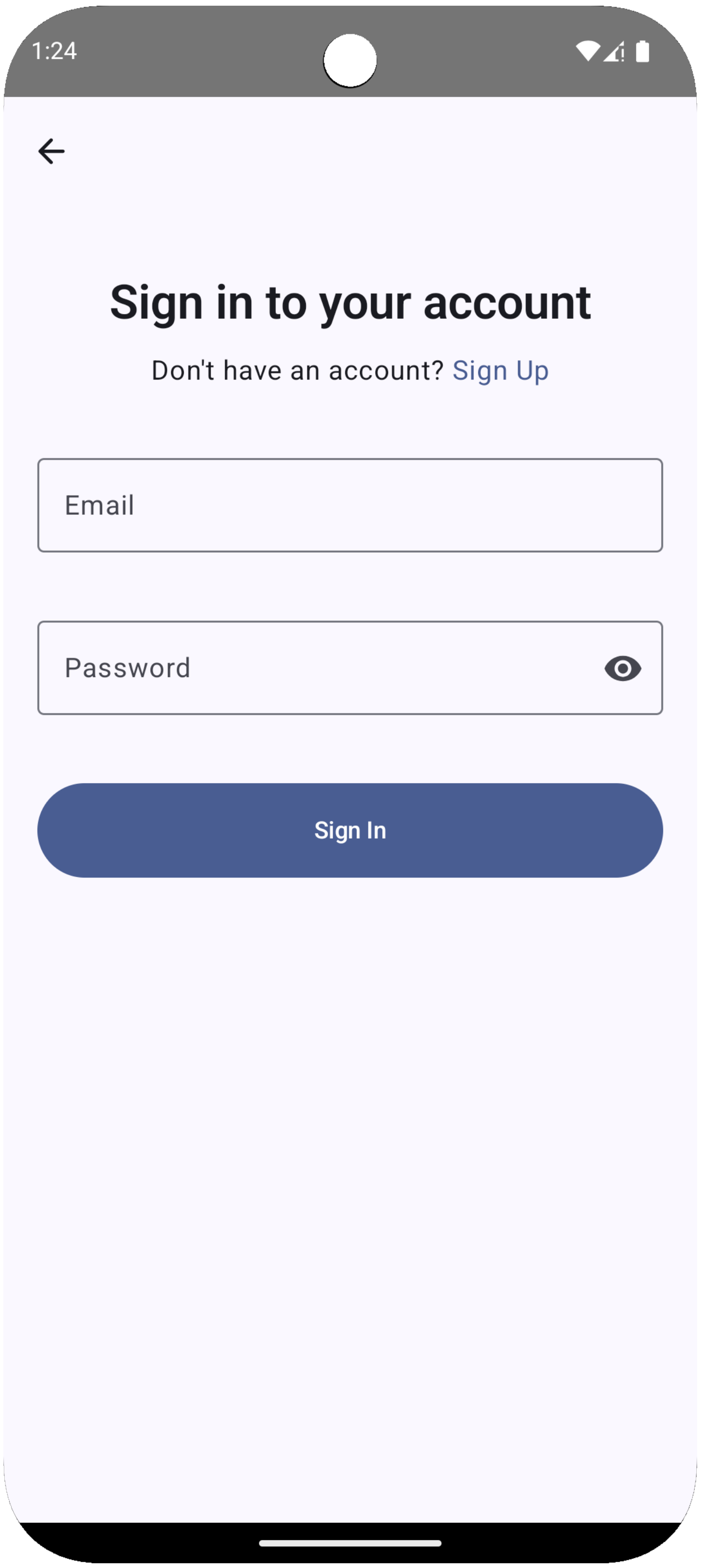


Figure 7.5: Sign In Page

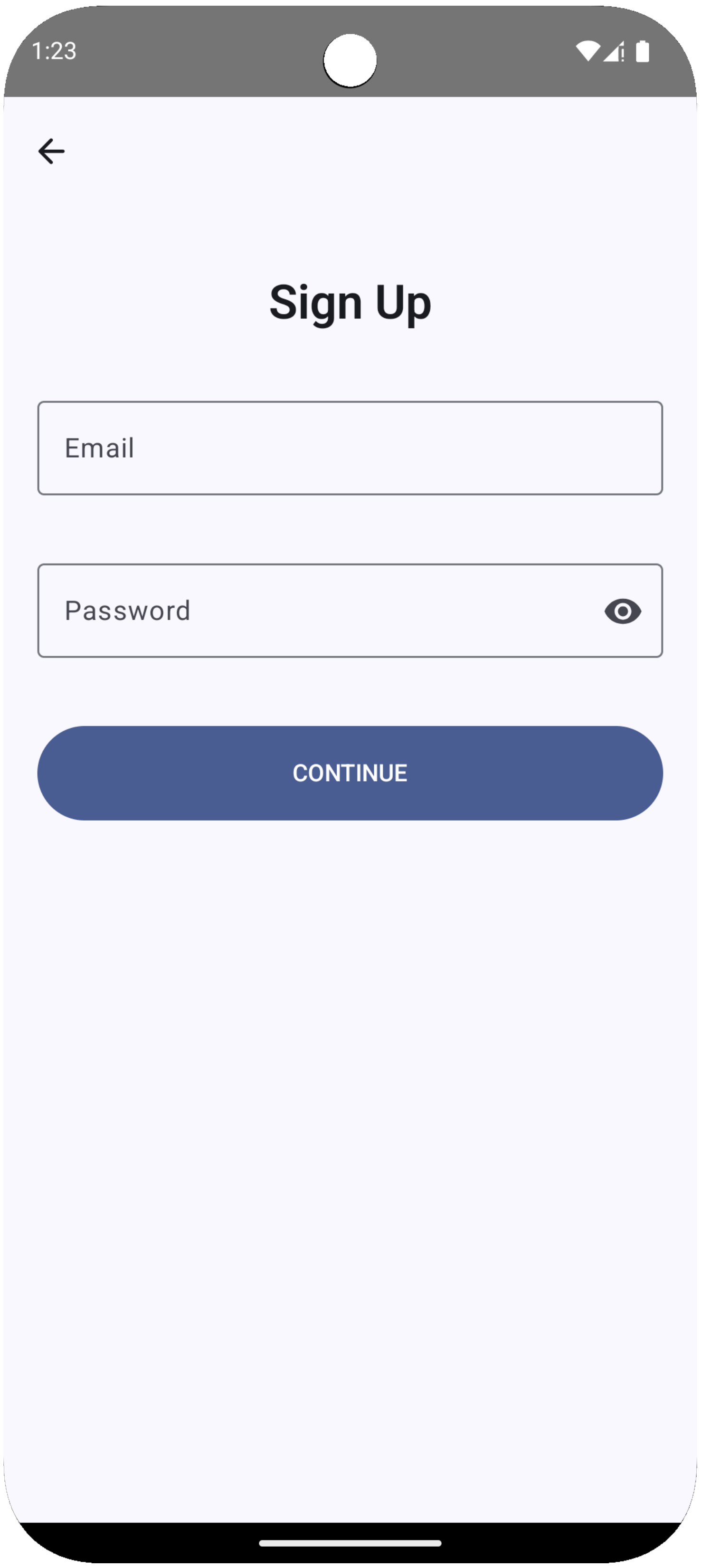


Figure 7.6: Sign Up Page

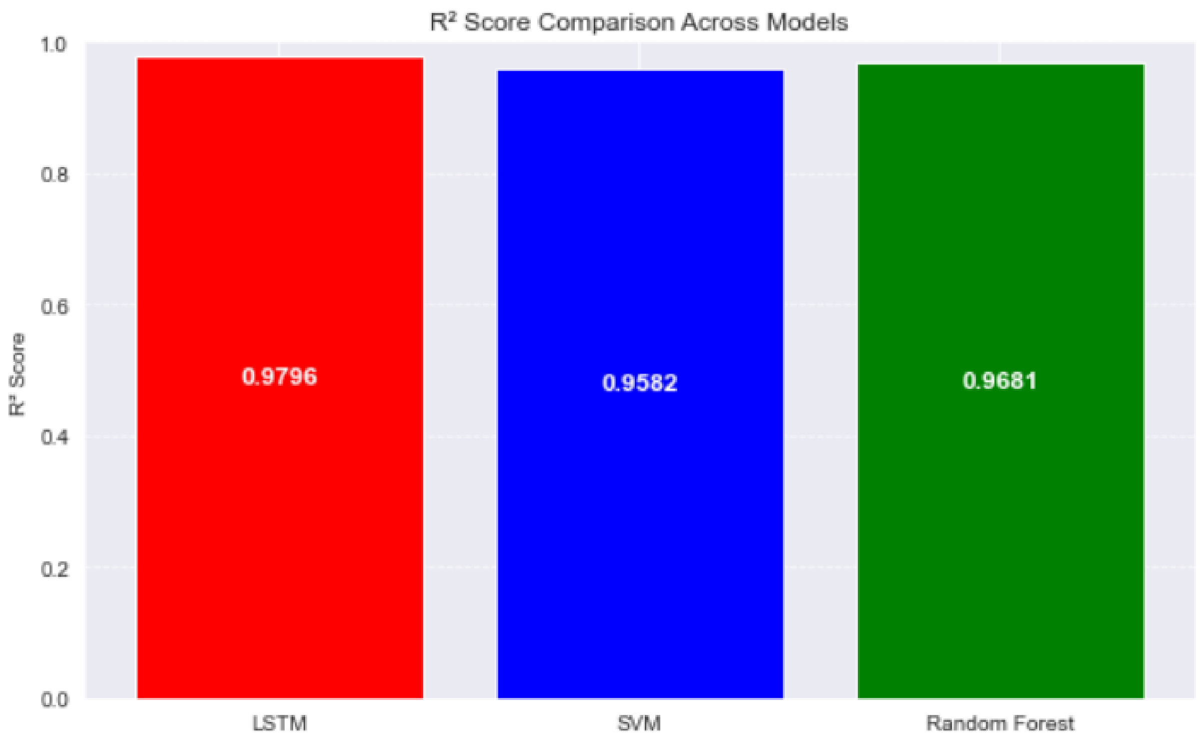
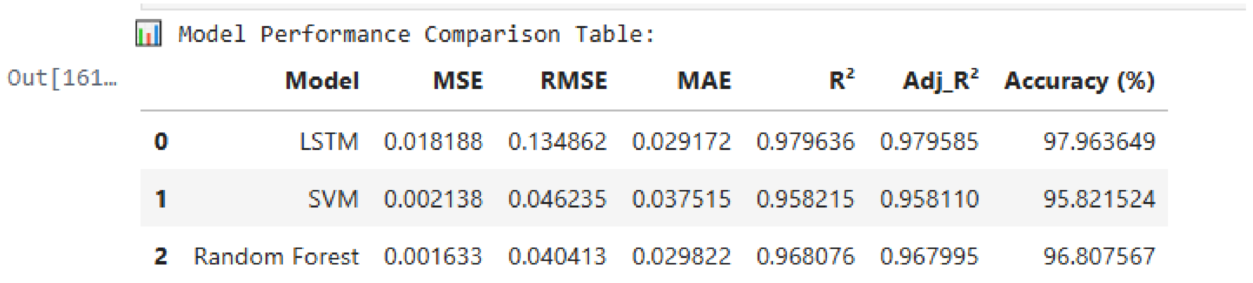


Figure 7.7: R2 Score Comparion Across Model



```
📊 Model Performance Comparison Table:
Out[161…
```

	Model	MSE	RMSE	MAE	R^2	Adj_R^2	Accuracy (%)
0	LSTM	0.018188	0.134862	0.029172	0.979636	0.979585	97.963649
1	SVM	0.002138	0.046235	0.037515	0.958215	0.958110	95.821524
2	Random Forest	0.001633	0.040413	0.029822	0.968076	0.967995	96.807567

Figure 7.8: Final Result Table

Chapter 8
Conclusion and Future Scope

8.1 Conclusion

We have successfully developed a mobile application that helps users predict stock prices using machine learning. The system brings together three key components — an Android-based frontend, a Flask-powered backend, and an LSTM-based machine learning model. Our aim was to build a practical tool that delivers accurate stock predictions through a clean and simple and easy-to-use design.

The machine learning model was trained using historical stock data, allowing it to learn patterns and trends in the market. When a user searches for a stock, the app sends the request to the backend, where the Flask server processes the data, runs the prediction through the model, and returns the results to the app almost instantly.

To keep the predictions up-to-date and relevant, we integrated Yahoo Finance for real-time and historical stock data. User authentication details and activity logs are securely stored in a database, ensuring both security and personalization throughout the user experience.

This project demonstrates how machine learning, mobile development, and web technologies can come together to create a practical and accessible solution. It allows users, even those with limited technical knowledge, to gain valuable valuable information from the app.

In conclusion, we've achieved the core goal of this project — building a smart, fast, and easy-to-use stock prediction app. It lays a strong foundation for future enhancements, such as integrating more advanced analytics, adding personalized suggestions, or supporting multiple languages. This project proves that machine learning can be made accessible, practical, and valuable for everyday users.

8.2 Future Scope

While the app is fully functional and achieves its primary goal, there are numerous areas for future improvement:

- **Expansion to iOS and Web:** Right now, the app is available only on Android. In the future, we plan to develop iOS and web versions so that more users can access it on their preferred devices.
- **Incorporating Sentiment Analysis:** We are considering adding a feature that analyzes realtime news and social media posts to understand public sentiment about specific stocks. This could help improve the accuracy of our stock predictions.
- **Portfolio Tracking:** We want to make it easier for users to track their investments. By adding a portfolio tracking feature, users will be able to save the stocks they've purchased and monitor their profit or loss over time.
- **Smart Alerts:** In the future, we plan to introduce smart notifications. The app could alert users when a stock sees a sudden price change or when there's an important market update.

By adding these features, the app can become a complete tool for stock market prediction and personal finance. It will not only show predictions but also help users make better and smarter decisions about their investments.

Chapter 9
UN Sustainable Development Goals

9.1 Mapping with UN Sustainable Development Goals

There are 17 goals defined by the United Nations Department of Economic and Social Affairs for Sustainable Development. These goals form part of the 2030 Agenda for Sustainable Development, which was adopted by 193 member states at the UN General Assembly Summit in September 2015 and came into effect on 1 January 2016.

As part of our responsibility toward global development, we have mapped our project to the relevant SDG. The table below outlines how our stock market prediction app contributes to Goal 8: Decent Work and Economic Growth.

SDG Title: Goal 8 – Decent Work and Economic Growth		
S. No	**Product Objective**	**Proposed Solution and Outcome**
1	To develop a machine learning-based system that predicts stock prices using historical trends and technical indicators.	We built an LSTM-based prediction model and integrated it into our app. It provides accurate stock price analysis to help users make smarter investment choices.
2	To provide users with access to real-time data for timely and accurate predictions.	We connected the app to the Yahoo Finance API, enabling real-time stock data and alerts for better tracking and decision-making.
3	To enable personalized stock tracking for more effective investment monitoring.	We added a ”Watchlist” feature that lets users easily keep an eye on their favorite stocks.
4	To support better decision-making by keeping users updated with financial news.	A built-in news section displays the latest headlines related to market trends and stock performance.
5	To ensure the app is accessible and easy to use for everyone.	The app was designed with a clean, user-friendly interface for Android that delivers a smooth and intuitive experience.

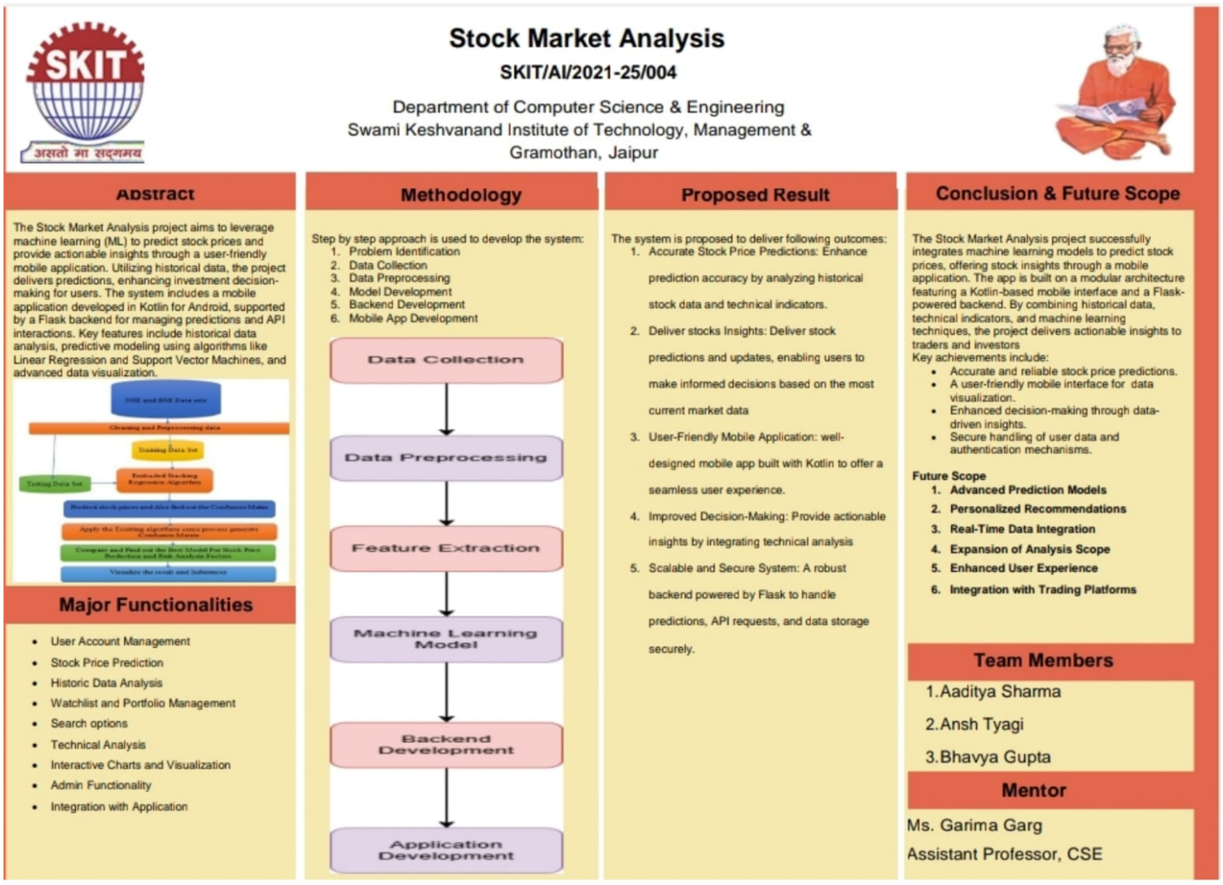
Table 9.1: Mapping of the Project with UN SDG Goal 8

GitHub Link

Link:

https://github.com/Aadityainf/BullSage

Project Poster



Stock Market Analysis

SKIT/AI/2021-25/004

Department of Computer Science & Engineering
Swami Keshvanand Institute of Technology, Management &
Gramothan, Jaipur

Abstract

The Stock Market Analysis project aims to leverage machine learning (ML) to predict stock prices and provide actionable insights through a user-friendly mobile application. Utilizing historical data, the project delivers predictions, enhancing investment decision-making for users. The system includes a mobile application developed in Kotlin for Android, supported by a Flask backend for managing predictions and API interactions. Key features include historical data analysis, predictive modeling using algorithms like Linear Regression and Support Vector Machines, and advanced data visualization.

NSE and BSE Data sets
Cleaning and Preprocessing data
Training Data Set
Testing Data Set
Embedded Stacking Regression Algorithm
Predict stock prices and Also find out the Confusion Matrix
Apply the Existing algorithms same process generate Confusion Matrix
Compare and Find out the Best Model For Stock Price Prediction and Risk Analysis Factors
Visualize the result and Inferences

Major Functionalities

- User Account Management
- Stock Price Prediction
- Historic Data Analysis
- Watchlist and Portfolio Management
- Search options
- Technical Analysis
- Interactive Charts and Visualization
- Admin Functionality
- Integration with Application

Methodology

Step by step approach is used to develop the system:
1. Problem Identification
2. Data Collection
3. Data Preprocessing
4. Model Development
5. Backend Development
6. Mobile App Development

Proposed Result

The system is proposed to deliver following outcomes:
1. Accurate Stock Price Predictions: Enhance prediction accuracy by analyzing historical stock data and technical indicators.
2. Deliver stocks Insights: Deliver stock predictions and updates, enabling users to make informed decisions based on the most current market data
3. User-Friendly Mobile Application: well-designed mobile app built with Kotlin to offer a seamless user experience.
4. Improved Decision-Making: Provide actionable insights by integrating technical analysis
5. Scalable and Secure System: A robust backend powered by Flask to handle predictions, API requests, and data storage securely.

Conclusion & Future Scope

The Stock Market Analysis project successfully integrates machine learning models to predict stock prices, offering stock insights through a mobile application. The app is built on a modular architecture featuring a Kotlin-based mobile interface and a Flask-powered backend. By combining historical data, technical indicators, and machine learning techniques, the project delivers actionable insights to traders and investors
Key achievements include:
- Accurate and reliable stock price predictions.
- A user-friendly mobile interface for data visualization.
- Enhanced decision-making through data-driven insights.
- Secure handling of user data and authentication mechanisms.

Future Scope
1. **Advanced Prediction Models**
2. **Personalized Recommendations**
3. **Real-Time Data Integration**
4. **Expansion of Analysis Scope**
5. **Enhanced User Experience**
6. **Integration with Trading Platforms**

Team Members

1.Aaditya Sharma

2.Ansh Tyagi

3.Bhavya Gupta

Mentor

Ms. Garima Garg
Assistant Professor, CSE

Figure 9.1: Poster

References

[1] Nelson, D. M., Pereira, A. C., & De Oliveira, R. A. (2017, May). Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)* (pp. 1419-1426). IEEE.

[2] Meesad, P., & Rasel, R. I. (2013, May). Predicting stock market price using support vector regression. In *2013 International Conference on Informatics, Electronics and Vision (ICIEV)* (pp. 1-6). IEEE.

[3] Choy, Y. T. (2021). A Mobile Application For Stock Price Prediction (Doctoral dissertation, UTAR).

[4] Nayak, A., Pai, M. M., & Pai, R. M. (2016). Prediction models for Indian stock market. *Procedia Computer Science*, 89, 441-449.

[5] Bagastio, K., Oetama, R. S., & Ramadhan, A. (2023). Development of stock price prediction system using Flask framework and LSTM algorithm. *Development*, 7(3), 2631.