PROJECT REPORT

ON

"STOCK TRADING WEB APP"

A Project Report Submitted in Partial fulfillments of Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted To

Submitted By

S.NO	Team Id	Student ID (Hall Ticket Number)	Student Name
1	LTVIP2024TMID05297	21HU5A0404	Deepala Gowri Akhila
2		20HU1A0402	Repalle Akshaya
3		20HU1A0410	Shaik Jasmin
4		20HU1A0413	Yangalasetty Anusha
5		21HU5A0407	Javvaji Kavya

Contents

Chapter 1

INTRODUCTION

SB Stocks is a user-friendly web app designed for stock market enthusiasts. It provides paper trading functionalities to help users practice and learn about stock trading without risking real money. Users can access real-time stock data, historical trends, and comprehensive stock listings to make informed decisions and build their investment skills.

The stock trading web app is designed to provide users with a platform to buy, sell, and manage their investments in the stock market. It offers real-time stock quotes, interactive charts, portfolio tracking, and various tools for technical and fundamental analysis. Users can create customized watchlists, set up alerts for price movements, and execute trades seamlessly.

The app aims to provide an intuitive and user-friendly experience, catering to both novice investors and experienced traders alike. Additionally, it may include features such as educational resources, community forums, and news updates to help users make informed investment decisions.

1.1 Overview A brief description about your project

The stock trading web app project involves full-stack web development to create a comprehensive platform for stock market enthusiasts to engage in trading activities. This project encompasses both frontend and backend development, along with database management and integration of external APIs for real-time data.

Introducing SB Stocks, the ultimate trading platform designed to revolutionize the way you engage with the stock market. With SB Stocks, your trading experience will be elevated to new heights of convenience and proficiency.

Our user-friendly web app empowers users to effortlessly access a wide range of stocks listed on the US Stock market, enabling you to practice and hone your trading skills through paper trading. Whether you&#39;re an experienced trader or new to the world of stocks, navigating the market has never been easier.

Imagine having comprehensive details about various stocks and their performance at your fingertips. From real-time stock prices to historical data and trends, you&#39;ll have all the information you need to make informed trading decisions. No more uncertainty – SB Stocks ensures that every aspect of your trading journey is crystal clear.

The account registration and login process are a breeze. Simply provide your necessary details, contact information, and preferred login credentials. Once registered, you can securely log in to access your trading dashboard. With SB Stocks, you can simulate a wide range of trading operations from the comfort of your own space. Practice buying and selling stocks using virtual funds, track your portfolio&#39;s performance, and analyze your trading strategies seamlessly. Real-time updates ensure that your paper trading activities are reflected immediately in your virtual portfolio.

Exploring different stocks and strategies is now hassle-free. Browse through the comprehensive list of stocks listed on the US Stock market, study their historical performance, and create virtual portfolios to test your investment strategies. Our simulation accurately replicates market conditions, giving you a realistic experience without the financial risks.

SB Stocks is here to enhance your trading experience by providing a seamless and convenient way to practice trading and sharpen your investment skills. With our user-friendly interface, comprehensive stock data, and robust portfolio management features, we ensure an enriching and educational trading experience for all users.

Get ready to embark on a new era of trading with SB Stocks – your gateway to refining your trading strategies and gaining valuable insights into the world of stock trading, all without the real-world financial risks.

1.2 Overview of Scenario based case study of your project

Scenario: Sophia is a finance enthusiast who is eager to explore the world of stock trading and investment. She wants to practice trading stocks without risking real money and improve her trading skills. Sophia decides to use SB Stocks, a MERN-based stock trading application, to simulate stock trading activities and hone her investment strategies.

User Registration and Profile Creation: Sophia visits the SB Stocks website and completes the user registration process. She provides her personal details, contact information, and preferred login credentials to create her account. Sophia&#39;s registration is successful, and she receives confirmation email with her login details.

Login and Dashboard Access: Sophia securely logs in to the SB Stocks platform using her registered email address and password. Upon successful login, she gains access to her personalized trading dashboard, where she can manage her virtual portfolio, track her trading activities, and explore different stocks listed on the US Stock market.

Explore Stock Listings: Sophia navigates to the &quot;Stock Listings&quot; section of the SB Stocks platform to explore the comprehensive list of stocks available for trading. She can browse through various categories, industries, and sectors to discover potential investment opportunities and study the performance of different stocks.

Paper Trading Simulation: Sophia engages in paper trading activities using the SB Stocks platform to simulate buying and selling stocks with virtual funds. She selects stocks from the available listings, analyzes their historical performance, and executes virtual trades to build her virtual portfolio and test her investment strategies.

Portfolio Tracking and Analysis: Sophia monitors the performance of her virtual portfolio in real-time using the SB Stocks dashboard. She tracks her investment positions, portfolio value, profit/loss, and overall performance metrics to evaluate the effectiveness of her trading strategies and make informed decisions.

Market Data and Trends: Sophia leverages the market data and trends provided by SB Stocks to gain insights into the performance of different stocks and market trends. She accesses real-time stock prices, historical data, charts, and indicators to analyze market conditions and identify potential trading opportunities.

Risk-Free Investment Strategies: Sophia explores different investment strategies and risk management techniques using the SB Stocks platform without risking real money. She practices diversification, asset allocation, and portfolio rebalancing to optimize her virtual portfolio and maximize returns while minimizing risks.

Educational Resources: SB Stocks provides Sophia with access to educational resources, tutorials, and articles to enhance her trading knowledge and skills. She learns about fundamental and technical analysis, trading strategies, market dynamics, and investment principles to become a more informed and successful trader.

Community Engagement: Sophia interacts with other users and traders within the SB Stocks community to share insights, discuss investment ideas, and learn from each other&#39;s experiences. She participates in forums, chat rooms, and discussions to network with like-minded individuals and stay updated on market developments.

Continuous Improvement: The SB Stocks development team continuously updates and improves the platform based on user feedback, market trends, and technological advancements.

They add new features, enhance performance, and ensure the platform&#39;s reliability and security to provide users like Sophia with a seamless and rewarding stock trading experience.

1.3 Purpose The use of this project. What can be achieved using this.

The stock trading web app serves several purposes and offers a range of functionalities that cater to different user needs. Here are some of the key purposes and what can be achieved using this project:

Investment Management:

- Users can track their investment portfolios, monitor performance, and make informed decisions about buying or selling stocks.
- Portfolio management tools enable users to diversify their holdings, set investment goals, and rebalance their portfolios as needed.

Trading Execution:

- The app provides a platform for users to execute trades in real-time, allowing them to buy and sell stocks directly from their accounts.
- Order management features enable users to place market orders, limit orders, stop-loss orders, and other types of trades with ease.

Market Analysis:

- Users can access a wide range of financial data, including stock quotes, historical prices, technical indicators, and company fundamentals.
- Analytical tools and charting capabilities enable users to perform technical analysis, identify trends, and conduct research on potential investment opportunities.

Education and Research:

- The app may include educational resources such as articles, tutorials, and videos to help users learn about stock market investing, trading strategies, and financial concepts.
- Research tools provide users with access to company profiles, earnings reports, analyst ratings, and other information to support their investment decisions.

Community Engagement:

- Community forums and social features allow users to interact with each other, share insights, and discuss market trends.
- Collaboration features enable users to form investment clubs, share watchlists, and collaborate on research projects.

Alerts and Notifications:

- Users can set up alerts and notifications to stay informed about significant market events, price movements, and changes in their portfolios.
- Customizable alerts help users react quickly to market developments and make timely decisions.

Risk Management:

- Risk assessment tools help users evaluate their risk tolerance, assess the potential impact of market fluctuations, and manage their exposure to risk.
- Portfolio analysis features provide insights into asset allocation, volatility, and other risk factors to help users make risk-adjusted investment decisions.

Chapter 2

LITERATURE SURVEY

2.1 Existing problem Existing approaches or method to solve this problem

One existing problem in stock trading web applications is the challenge of providing real-time and accurate market data to users. Here's a deeper look at this problem:

Real-Time Market Data:

Problem: Stock trading relies heavily on up-to-date market information. However, delivering real-time data to users in a stock trading web application can be challenging due to factors such as latency, data processing time, and network congestion. Users require timely access to stock prices, order book data, and market news to make informed trading decisions. Delays or inaccuracies in market data can lead to missed trading opportunities or financial losses.

Existing Approaches or Solutions:

1. **Data Feeds and APIs:** Many stock trading web applications integrate with financial data providers and stock exchanges to access real-time market data through data feeds and APIs (Application Programming Interfaces). These APIs provide access to a wide range of market information, including stock quotes, order book data, historical prices, and market news.
2. **Low-Latency Infrastructure:** Implementing low-latency infrastructure and network optimizations can help reduce data transmission delays and improve the responsiveness of the trading platform. This may involve using high-speed internet connections, dedicated servers, and proximity hosting services to minimize latency between the trading platform and data sources.
3. **Data Aggregation and Processing:** Stock trading web applications often aggregate data from multiple sources and perform data processing tasks to ensure accuracy and reliability. This may include data validation, error correction, and normalization to standardize data formats and eliminate inconsistencies.
4. **Caching and Streaming Technologies:** Utilizing caching mechanisms and real-time data streaming technologies can help optimize data delivery and improve user experience. Caching frequently accessed data locally on the server or client-side can reduce the need for repeated data requests, while streaming technologies enable continuous updates and notifications for real-time events.
5. **Monitoring and Alerts:** Implementing monitoring tools and alert systems can help detect data anomalies, system failures, or disruptions in data delivery. Automated alerts can notify users and administrators about potential issues, allowing for timely intervention and resolution to minimize disruptions to trading operations.

Despite these existing approaches, challenges such as network latency, data synchronization issues, and system failures can still impact the reliability and timeliness of market data in stock trading web applications. Continuous optimization and investment in technology infrastructure are necessary to address these challenges and ensure a seamless trading experience for users.

2.2 Proposed solution What is the method or solution suggested by you?

The proposed solution for developing a stock trading web application involves the following methodological approach:

1. **Technology Choice:** We'll use modern web development tools like React.js for the user interface and Node.js with Express.js for the backend.
2. **Real-Time Data Integration:** We'll connect to reliable sources of real-time stock market data to ensure that users see accurate information about stock prices and market trends.
3. **User-Friendly Design:** We'll design the app to be easy to use, with clear layouts and interactive features like charts and dashboards to help users track their investments.
4. **Trading Features:** We'll build tools for users to buy and sell stocks, manage their portfolios, and keep track of their trades, with options for different types of orders like market orders and limit orders.
5. **Security and Compliance:** We'll prioritize security measures to protect users' personal and financial information, and make sure the app follows all relevant regulations for online trading platforms.
6. **Testing and Maintenance:** We'll thoroughly test the app to make sure it works smoothly, and set up systems for ongoing maintenance and updates to keep it running smoothly over time.

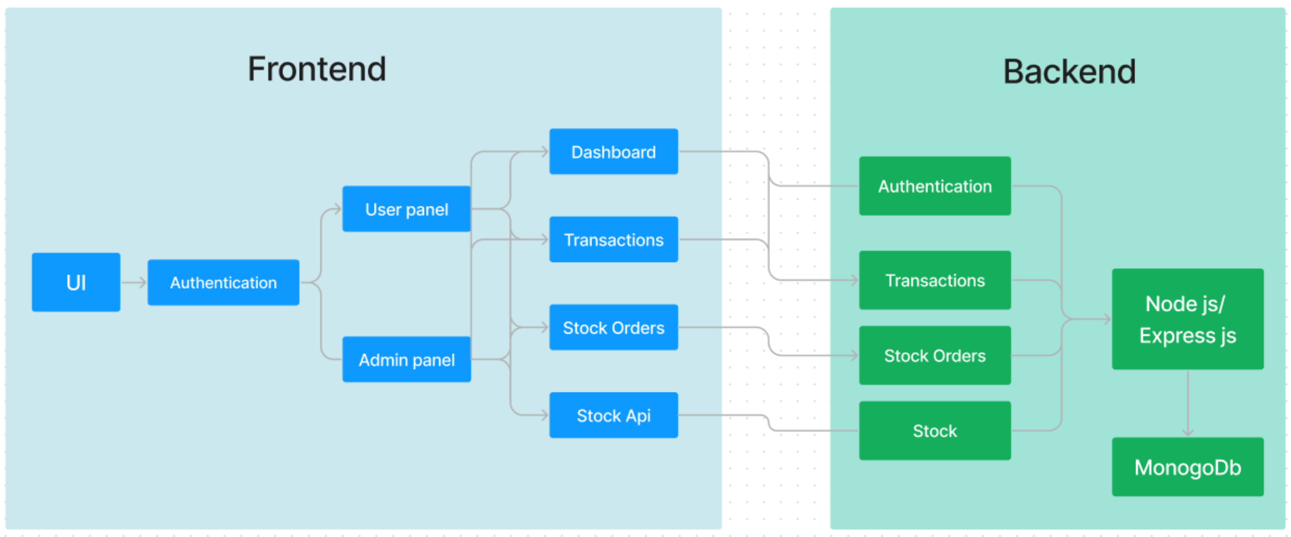
By following these steps, we aim to create a user-friendly stock trading platform that provides reliable market data and tools for users to manage their investments securely.

Chapter 3

THEORITICAL ANALYSIS

3.1 Block diagram Diagrammatic overview of the project.

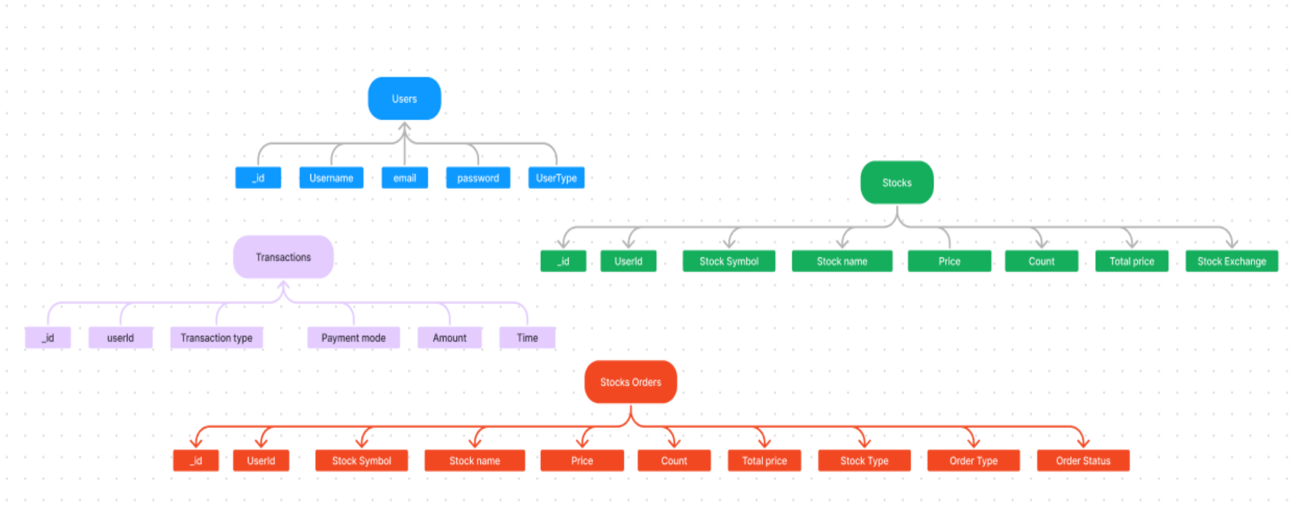
TECHINICAL ARCHITECTURE:



In this architecture diagram:

- The frontend is represented by the "Frontend" section, including user interface components such as Landing page, User home, Admin dashboard, Stock charts, stock orders and user stock portfolio.
- The backend is represented by the "Backend" section, consisting of API endpoints for Users, Deposits and Withdrawal of funds, Transactions and Stock orders. It also includes Admin Authentication and an Admin Dashboard.
- The Database section represents the database that stores collections for Users, Stocks, Orders and Transactions.

ER DIAGRAM:



The SB Stocks ER diagram serves as a blueprint for the system, outlining the key players and their interactions. It identifies entities like users, stocks, transactions, and virtual portfolios. Users can have multiple portfolios holding various stocks, while each transaction references a specific user and stock. Market data provides real-time and historical information for each stock. The beauty lies in the relationships. A single user can place many orders, and a stock can be present in numerous portfolios. This interconnectedness allows users to practice buying and selling, building virtual portfolios, and testing strategies without real-world financial risks.

3.2 Hardware / Software designing Hardware and software requirements

To develop a full-stack banking management app using AngularJS, Node.js, and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

Node.js and npm: Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager/

MongoDB: Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions: https://docs.mongodb.com/manual/installation/

Express.js: Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

- Installation: Open your command prompt or terminal and run the following command: **npm install express**

React.js: React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications. To install React.js, a JavaScript library for building user interfaces, follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

HTML, CSS, and JavaScript: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

Database Connectivity: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

Front-end Framework: Utilize Angular to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

Version Control: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

Development Environment: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

- Visual Studio Code: Download from https://code.visualstudio.com/download
- Sublime Text: Download from https://www.sublimetext.com/download
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

To Connect the Database with Node JS go through the below provided link:

- Link: https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

To run the existing Stocks Trading App project downloaded from google drive:

Download the code from the google drive https://drive.google.com/drive/folders/1ajrYknSAJs6lwQ9ObHVJ1uG2PiodBi69?usp=sharing

Install Dependencies:

- Navigate into the cloned repository directory:

cd Stocks-Trading-MERN

- Install the required dependencies by running the following command:

npm install

Start the Development Server:

- To start the development server, execute the following command:

npm run start

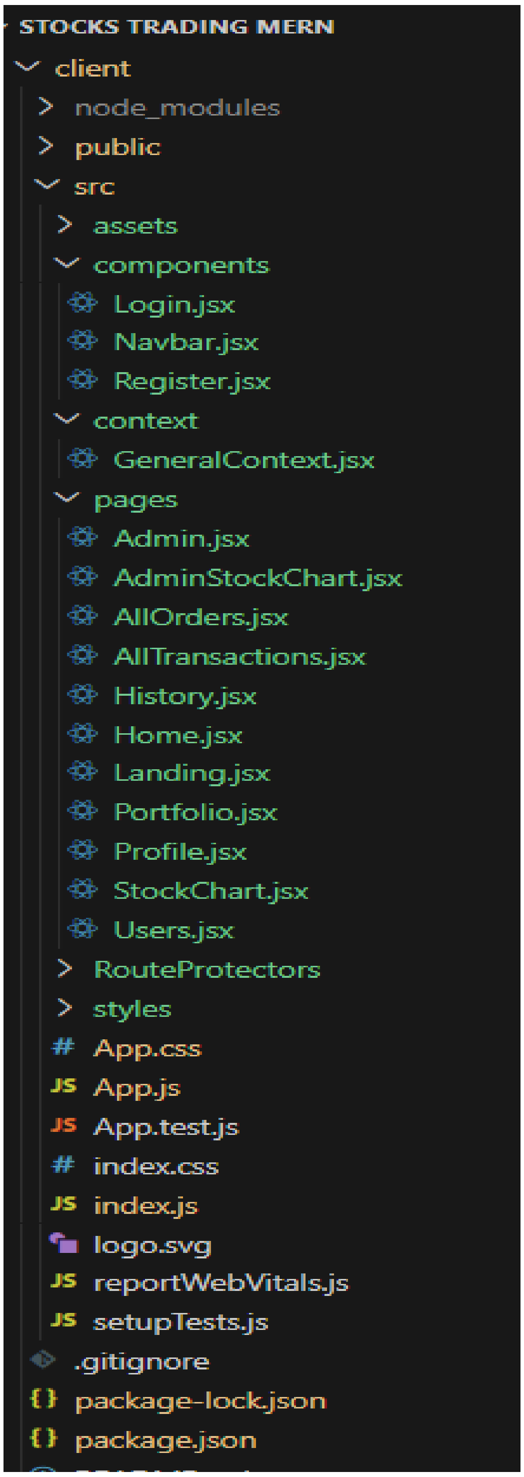
- The stocks app will be accessible at http://localhost:3000 by default. You can change the port configuration in the .env file if needed.

Access the App:

- Open your web browser and navigate to http://localhost:3000.
- You should see the stocks trading app's landing page, indicating that the installation and setup were successful.

You have successfully installed and set up the stocks trading app on your local machine. You can now proceed with further customization, development, and testing as needed.

3.3 Project structure:



```
STOCKS TRADING MERN
  client
    node_modules
    public
    src
      assets
      components
        Login.jsx
        Navbar.jsx
        Register.jsx
      context
        GeneralContext.jsx
      pages
        Admin.jsx
        AdminStockChart.jsx
        AllOrders.jsx
        AllTransactions.jsx
        History.jsx
        Home.jsx
        Landing.jsx
        Portfolio.jsx
        Profile.jsx
        StockChart.jsx
        Users.jsx
      RouteProtectors
      styles
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
      setupTests.js
    .gitignore
    package-lock.json
    package.json
```

The frontend structure assumes a React app and follows a modular approach. Here's a brief explanation of the main directories and files:

- **src/components:** This directory has minor components such as Login, Register, etc.,

- **src/pages:** The pages folder contains all the pages of the application like landing page, home page, etc.,

3.4 Application FLOW of your project

User Flow:

- Users start by registering for an account.
- After registration, they can log in with their credentials.
- Once logged in, they can check stocks, portfolio, orders, account balance and transactions.
- Users can deposit/withdraw money in the profile section using various payment modes.
- They can buy/sell the stocks they wish to do so.
- After buying/selling stocks, they will be reflected to the orders and portfolio.

Admin Flow:

- Admins start by logging in with their credentials.
- Once logged in, they are directed to the admin Dashboard.
- Bank admin can access all the data including stocks, users, transactions, and orders.

Project Flow:

Milestone 1: Project Setup and Configuration:

Let’s do project setup and configure it as per our requirements.

1. Create project folders and files:

   Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

   - Client folders.
   - Server folders

2. **Install required tools and software:**

   For the backend to function well, we use the libraries mentioned in the prerequisites. Those libraries includes

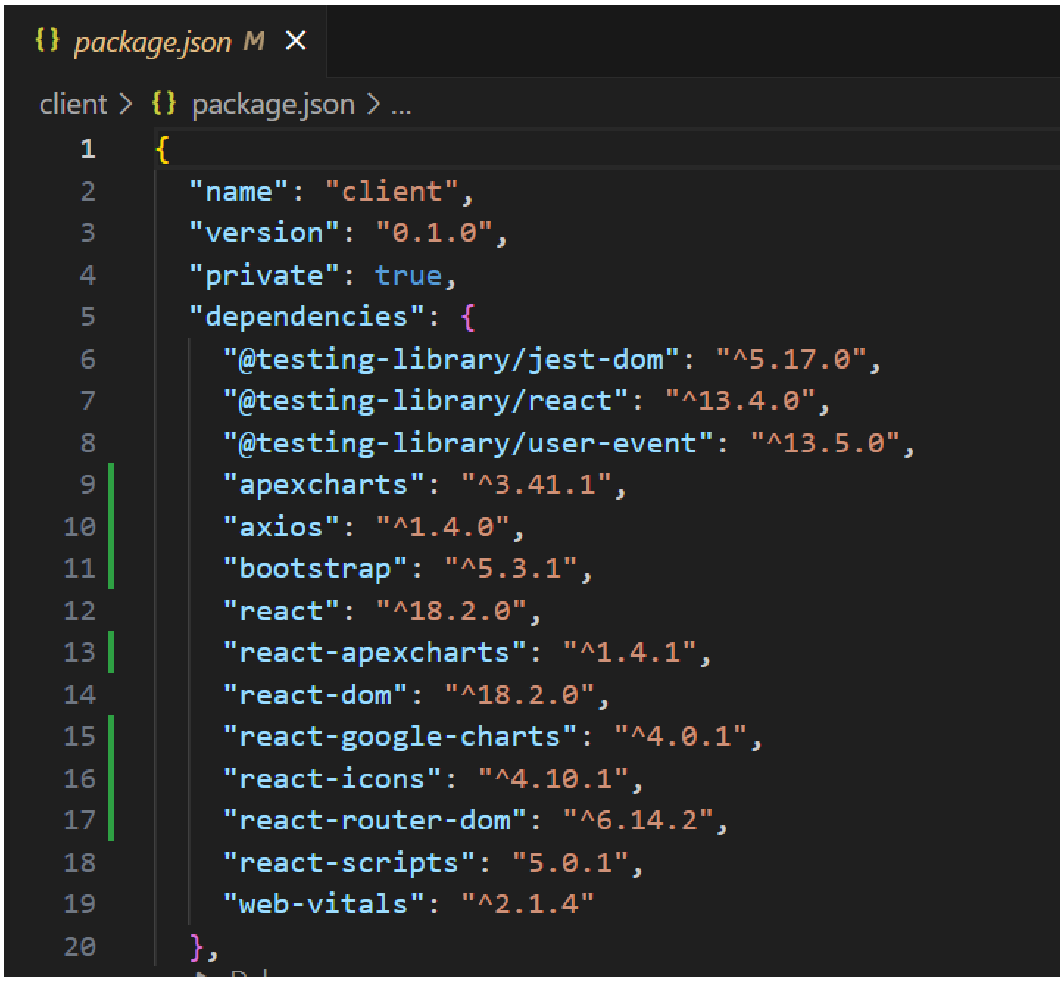
   - Node.js.
   - MongoDB.
   - Bcrypt
   - Body-parser

Also, for the frontend we use the libraries such as

- React Js.
- Material UI
- Bootstrap

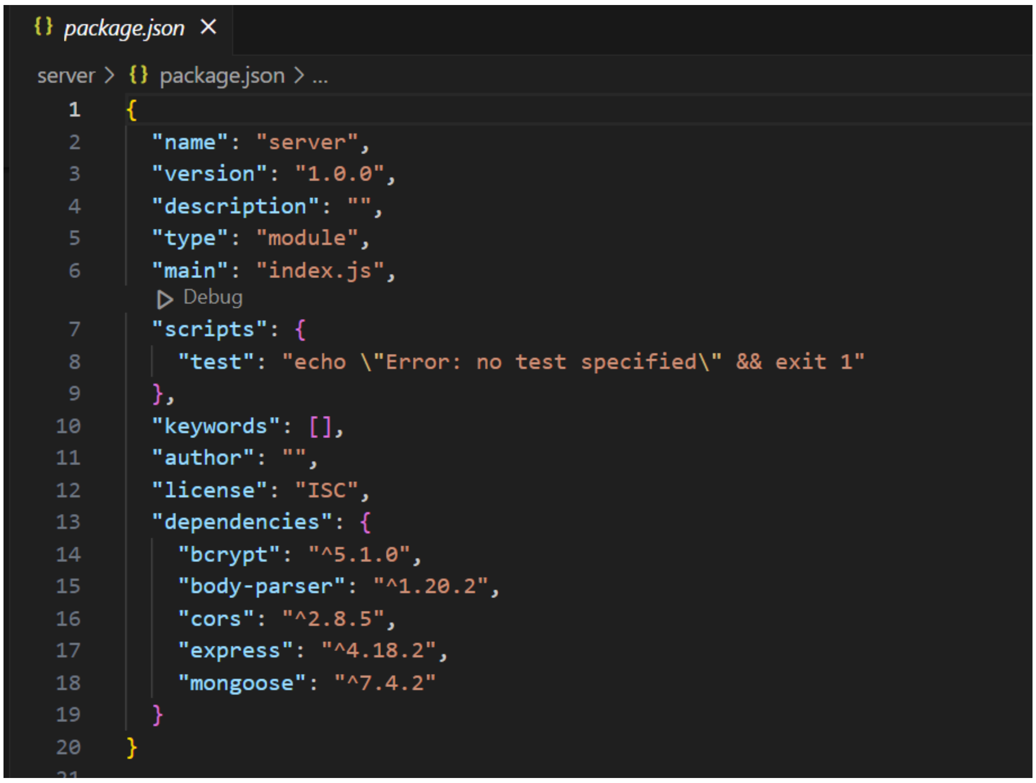
- Axios

After the installation of all the libraries, the package.json files for the frontend looks like the one mentioned below.



```
{} package.json M ×
client > {} package.json > ...
{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "apexcharts": "^3.41.1",
    "axios": "^1.4.0",
    "bootstrap": "^5.3.1",
    "react": "^18.2.0",
    "react-apexcharts": "^1.4.1",
    "react-dom": "^18.2.0",
    "react-google-charts": "^4.0.1",
    "react-icons": "^4.10.1",
    "react-router-dom": "^6.14.2",
    "react-scripts": "5.0.1",
    "web-vitals": "^2.1.4"
  },
```

After the installation of all the libraries, the package.json files for the backend looks like the one mentioned below.



```
{} package.json ×
server > {} package.json > ...
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "type": "module",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "mongoose": "^7.4.2"
  }
}
```

Milestone 2: Backend Development:

- **Set Up Project Structure:**
  - Create a new directory for your project and set up a package.json file using npm init command.
  - Install necessary dependencies such as Express.js, Mongoose, and other required packages.
- **Create Express.js Server:**
  - Set up an Express.js server to handle HTTP requests and serve API endpoints.
  - Configure middleware such as body-parser for parsing request bodies and cors for handling cross-origin requests.
- **Define API Routes:**
  - Create separate route files for different API functionalities such as authentication, stock actions, and transactions.
  - Implement route handlers using Express.js to handle requests and interact with the database.
- **Implement Data Models:**
  - Define Mongoose schemas for the different data entities like Bank, users, transactions, deposits and loans.
  - Create corresponding Mongoose models to interact with the MongoDB database.
  - Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.
- **User Authentication:**
  - Implement user authentication using strategies like JSON Web Tokens (JWT) or session-based authentication.
  - Create routes and middleware for user registration, login, and logout.
  - Set up authentication middleware to protect routes that require user authentication.
- **Handle new transactions:**
  - Allow users to make transactions to other users using the user's account id.
  - Update the transactions and account balance dynamically in real-time.
- **Admin Functionality:**
  - Implement routes and controllers specific to admin functionalities such as fetching all the data regarding users, transactions, stocks and orders.
- **Error Handling:**
  - Implement error handling middleware to catch and handle any errors that occur during the API requests.
  - Return appropriate error responses with relevant error messages and HTTP status codes.

Reference video for backend code:

https://drive.google.com/file/d/10WoAJ1KXlsc_J1FkDsuQodsW5tDBYu78/view?usp=sharing

Milestone 3: Database Development:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- Create a database and define the necessary collections for users, transactions, stocks and orders.

- Also let’s see the detailed description for the schemas used in the database.

1. **User Schema:**

   - Schema: userSchema
   - Model: ‘User’
   - The User schema represents the user data and includes fields such as username, email, balance, and password.
   - It is used to store user information for registration and authentication purposes.
   - The email field is marked as unique to ensure that each user has a unique email address**.**
   - The balance field represents the amount in users account and it gets updated with transactions.

2. **Transactions Schema:**

   - Schema: transactionsSchema
   - Model: ‘Transactions’
   - The Transactions schema represents the transactions data such as user details, transaction type, amount, time, etc.,

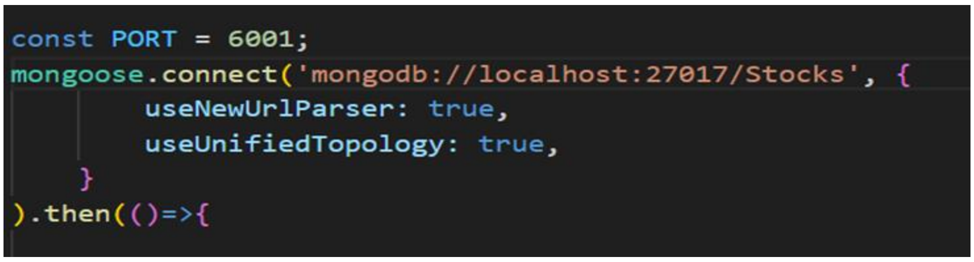
3. **Stocks Schema:**

   - Schema: stocksSchema
   - Model: ‘Stocks’
   - The Stocks schema represents the stock data such as user details, stock details, stock exchange, stock price, etc.,

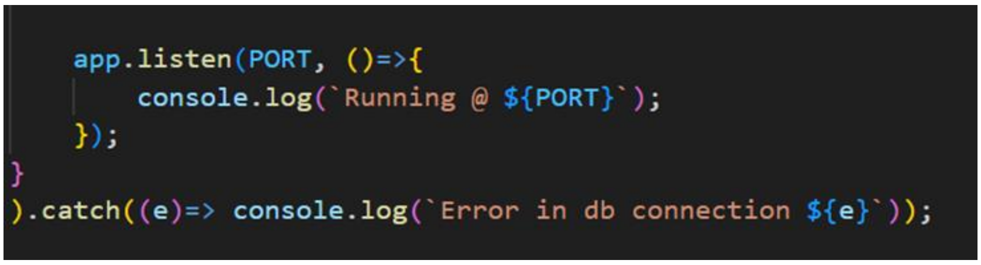
4. **Orders Schema:**

   - Schema: ordersSchema
   - Model: ‘Orders’
   - The Orders schema represents the stock order data and includes fields such as user details, stock details, total amount, order type, etc.,

For the connection of database use the code given below

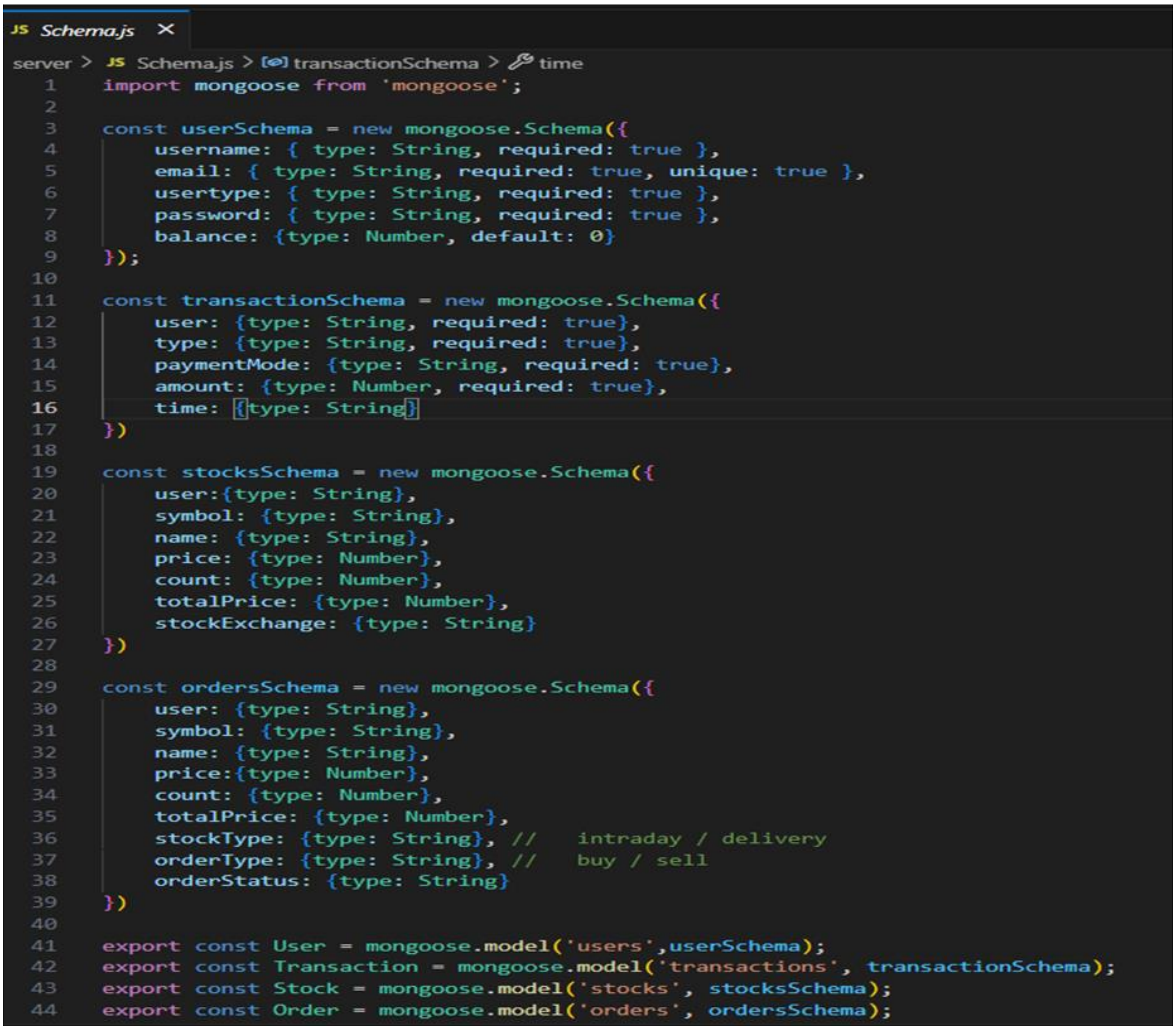


```
const PORT = 6001;
mongoose.connect('mongodb://localhost:27017/Stocks', {
        useNewUrlParser: true,
        useUnifiedTopology: true,
    }
).then(()=>{
```



```
    app.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });
}
).catch((e)=> console.log(`Error in db connection ${e}`));
```

Change the URL string according to the MongoDB connection string from the Atlas(cloud) / MongoDB Compass(local). The schemas for the database are given below



```javascript
import mongoose from 'mongoose';

const userSchema = new mongoose.Schema({
    username: { type: String, required: true },
    email: { type: String, required: true, unique: true },
    usertype: { type: String, required: true },
    password: { type: String, required: true },
    balance: {type: Number, default: 0}
});

const transactionSchema = new mongoose.Schema({
    user: {type: String, required: true},
    type: {type: String, required: true},
    paymentMode: {type: String, required: true},
    amount: {type: Number, required: true},
    time: {type: String}
})

const stocksSchema = new mongoose.Schema({
    user:{type: String},
    symbol: {type: String},
    name: {type: String},
    price: {type: Number},
    count: {type: Number},
    totalPrice: {type: Number},
    stockExchange: {type: String}
})

const ordersSchema = new mongoose.Schema({
    user: {type: String},
    symbol: {type: String},
    name: {type: String},
    price:{type: Number},
    count: {type: Number},
    totalPrice: {type: Number},
    stockType: {type: String}, //   intraday / delivery
    orderType: {type: String}, //   buy / sell
    orderStatus: {type: String}
})

export const User = mongoose.model('users',userSchema);
export const Transaction = mongoose.model('transactions', transactionSchema);
export const Stock = mongoose.model('stocks', stocksSchema);
export const Order = mongoose.model('orders', ordersSchema);
```

Milestone 4: Frontend Development:

1. **Setup React Application:**

   Bringing SB Stocks to life involves a three-step development process. First, a solid foundation is built using React.js. This includes creating the initial application structure, installing necessary libraries, and organizing the project files for efficient development. Next, the user interface (UI) comes to life. To start the development process for the frontend, follow the below steps.

   - Install required libraries.
   - Create the structure directories.

2. **Design UI components:**

   Reusable components will be created for all the interactive elements you'll see on screen, from stock listings and charts to buttons and user profiles. Next, we'll implement a layout and styling scheme to define the overall look and feel of the application. This ensures a visually-appealing and intuitive interface. Finally, a navigation system will be integrated, allowing you to effortlessly explore different sections of SB Stocks, like viewing specific stocks or managing your virtual portfolio.

3. **Implement frontend logic:**

In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

- Integration with API endpoints.
- Implement data binding.

Reference video for frontend code:
https://drive.google.com/file/d/1s_P6w8jIKgaQrPso96L_CEC7LUCcPhQk/view?usp=sharing

Milestone 5: Project Implementation:

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it.

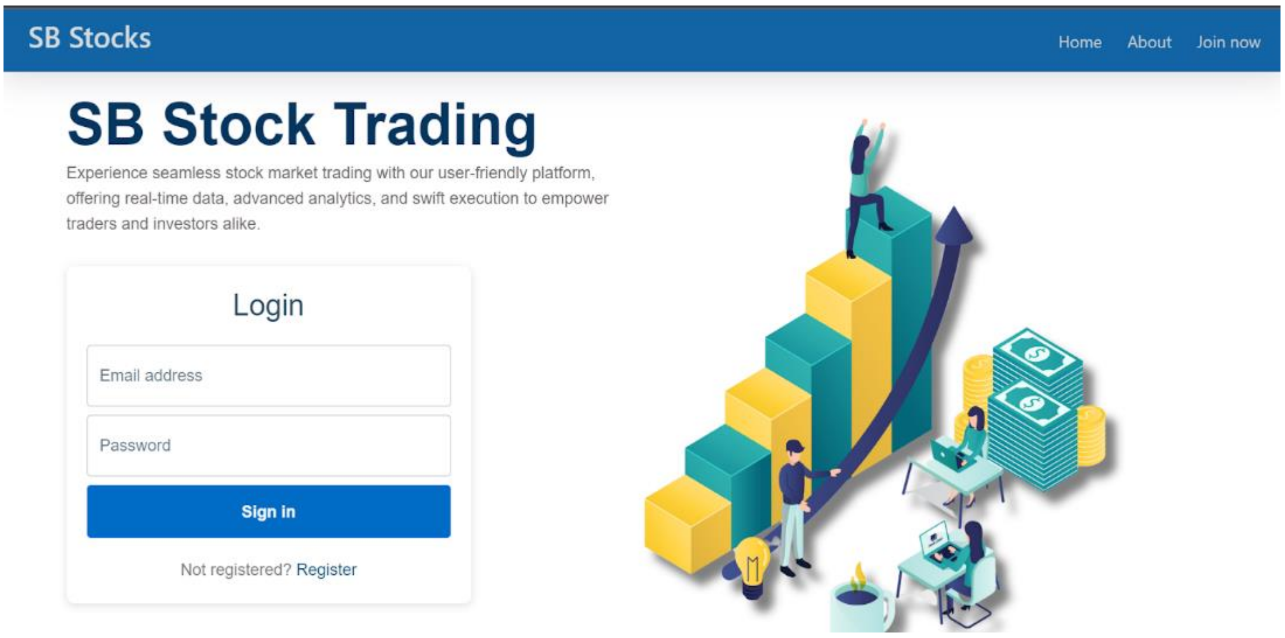
Chapter 4

RESULT

4.1 Final findings (Output) of the project along with screenshots.

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

Landing page:



SB Stock Trading

Experience seamless stock market trading with our user-friendly platform, offering real-time data, advanced analytics, and swift execution to empower traders and investors alike.

Login

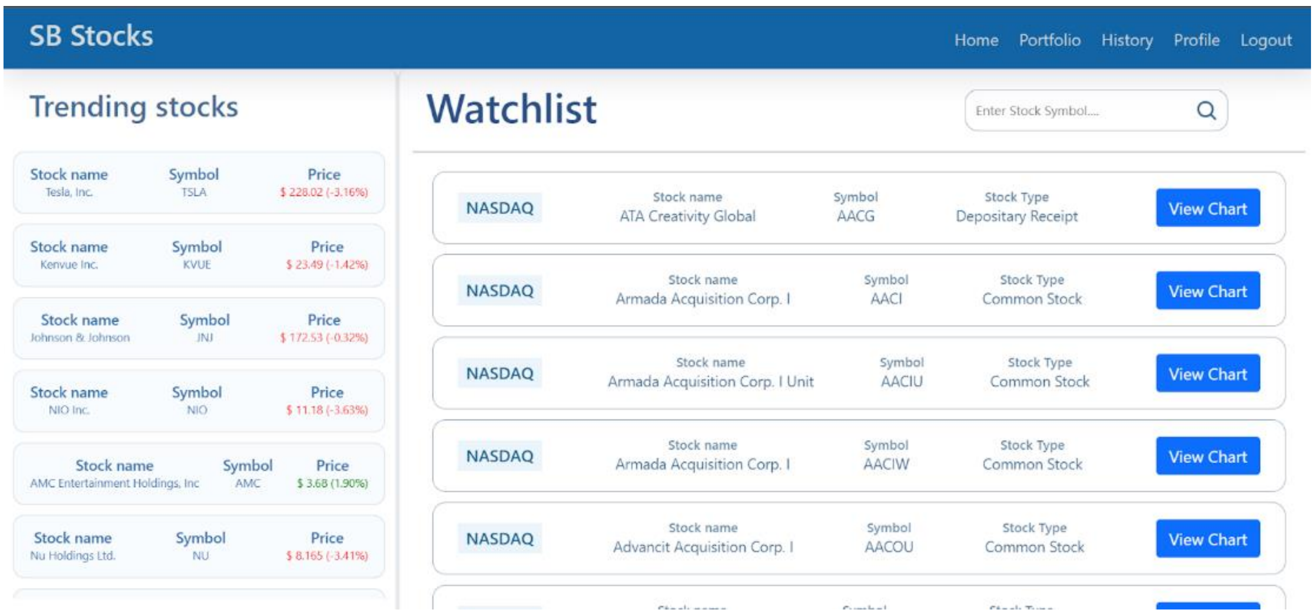
Email address

Password

Sign in

Not registered? Register

Home page:



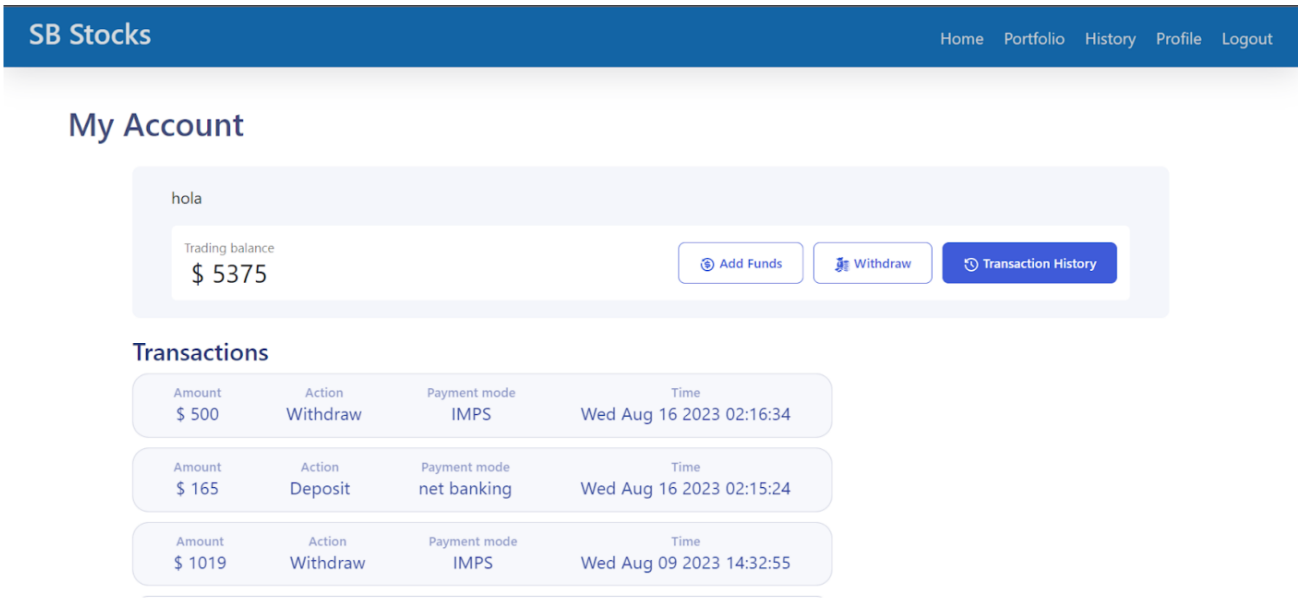
Trending stocks

Stock name	Symbol	Price
Tesla, Inc.	TSLA	$ 228.02 (-3.16%)
Kenvue Inc.	KVUE	$ 23.49 (-1.42%)
Johnson & Johnson	JNJ	$ 172.53 (-0.32%)
NIO Inc.	NIO	$ 11.18 (-3.63%)
AMC Entertainment Holdings, Inc	AMC	$ 3.68 (1.90%)
Nu Holdings Ltd.	NU	$ 8.165 (-3.41%)

Watchlist

	Stock name	Symbol	Stock Type	
NASDAQ	ATA Creativity Global	AACG	Depositary Receipt	View Chart
NASDAQ	Armada Acquisition Corp. I	AACI	Common Stock	View Chart
NASDAQ	Armada Acquisition Corp. I Unit	AACIU	Common Stock	View Chart
NASDAQ	Armada Acquisition Corp. I	AACIW	Common Stock	View Chart
NASDAQ	Advancit Acquisition Corp. I	AACOU	Common Stock	View Chart

Profile page (User):


SB Stocks

Home Portfolio History Profile Logout

My Account

hola

Trading balance

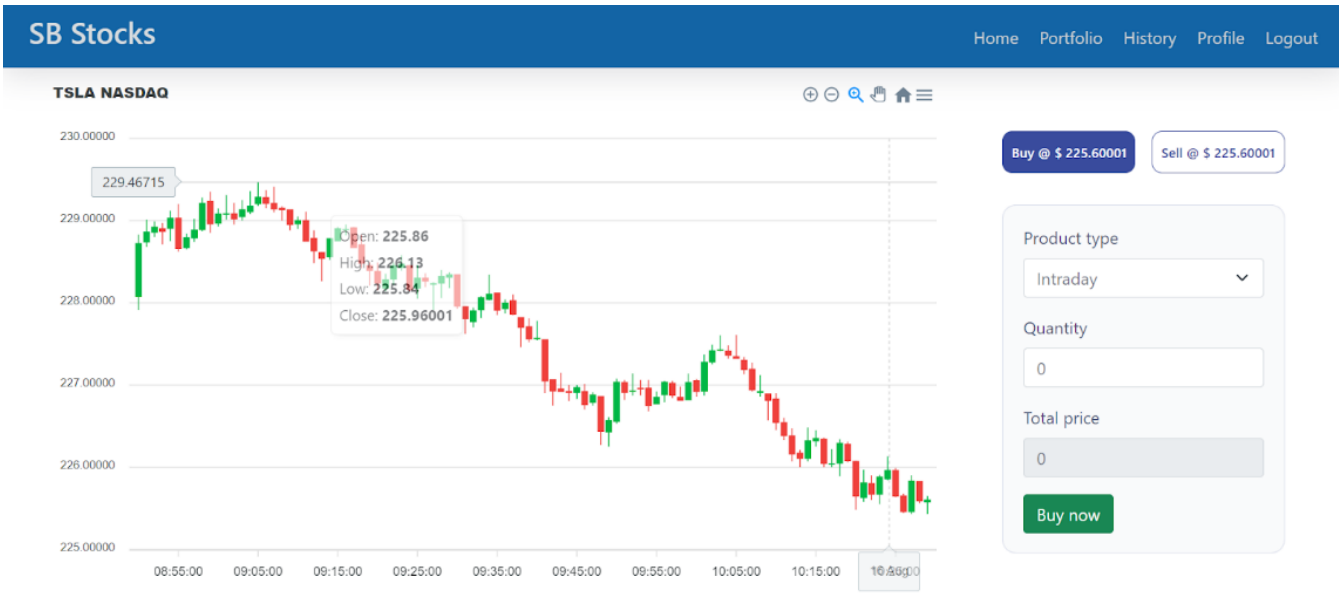
$ 5375

Add Funds | Withdraw | Transaction History

Transactions

Amount	Action	Payment mode	Time
$ 500	Withdraw	IMPS	Wed Aug 16 2023 02:16:34
$ 165	Deposit	net banking	Wed Aug 16 2023 02:15:24
$ 1019	Withdraw	IMPS	Wed Aug 09 2023 14:32:55

Stocks Chart:


SB Stocks

Home Portfolio History Profile Logout

TSLA NASDAQ

Open: 225.86
High: 226.13
Low: 225.84
Close: 225.96001

Buy @ $ 225.60001 | Sell @ $ 225.60001

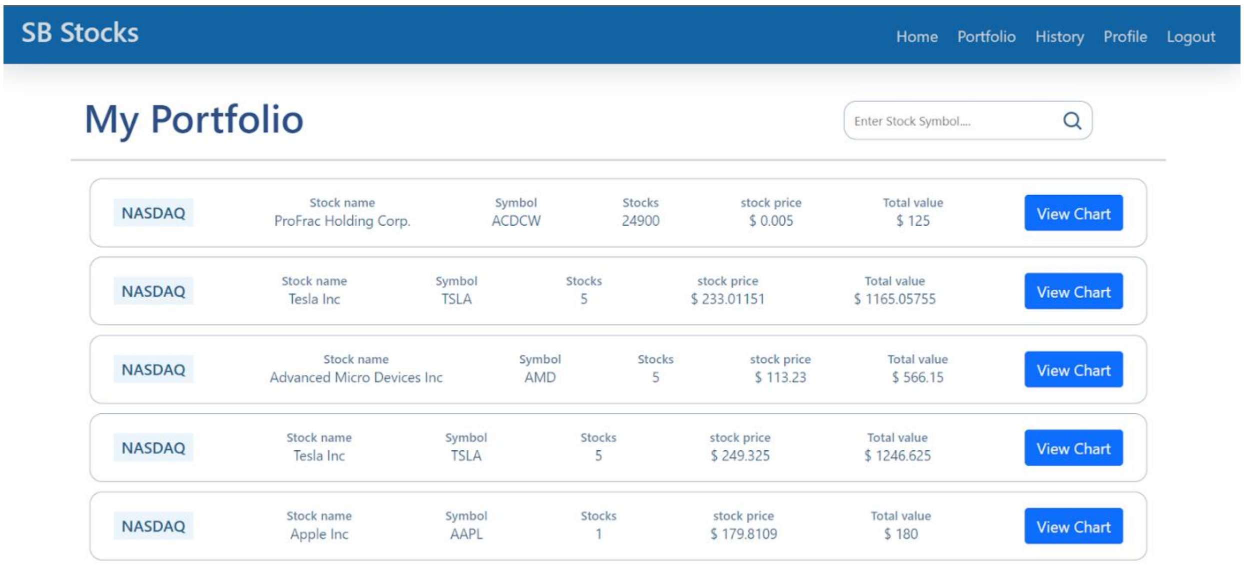
Product type: Intraday

Quantity: 0

Total price: 0

Buy now

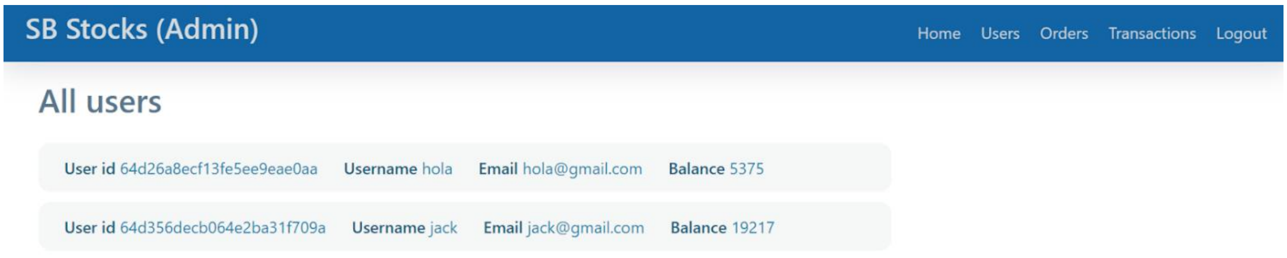
Portfolio:



My Portfolio

	Stock name	Symbol	Stocks	stock price	Total value	
NASDAQ	ProFrac Holding Corp.	ACDCW	24900	$ 0.005	$ 125	View Chart
NASDAQ	Tesla Inc	TSLA	5	$ 233.01151	$ 1165.05755	View Chart
NASDAQ	Advanced Micro Devices Inc	AMD	5	$ 113.23	$ 566.15	View Chart
NASDAQ	Tesla Inc	TSLA	5	$ 249.325	$ 1246.625	View Chart
NASDAQ	Apple Inc	AAPL	1	$ 179.8109	$ 180	View Chart

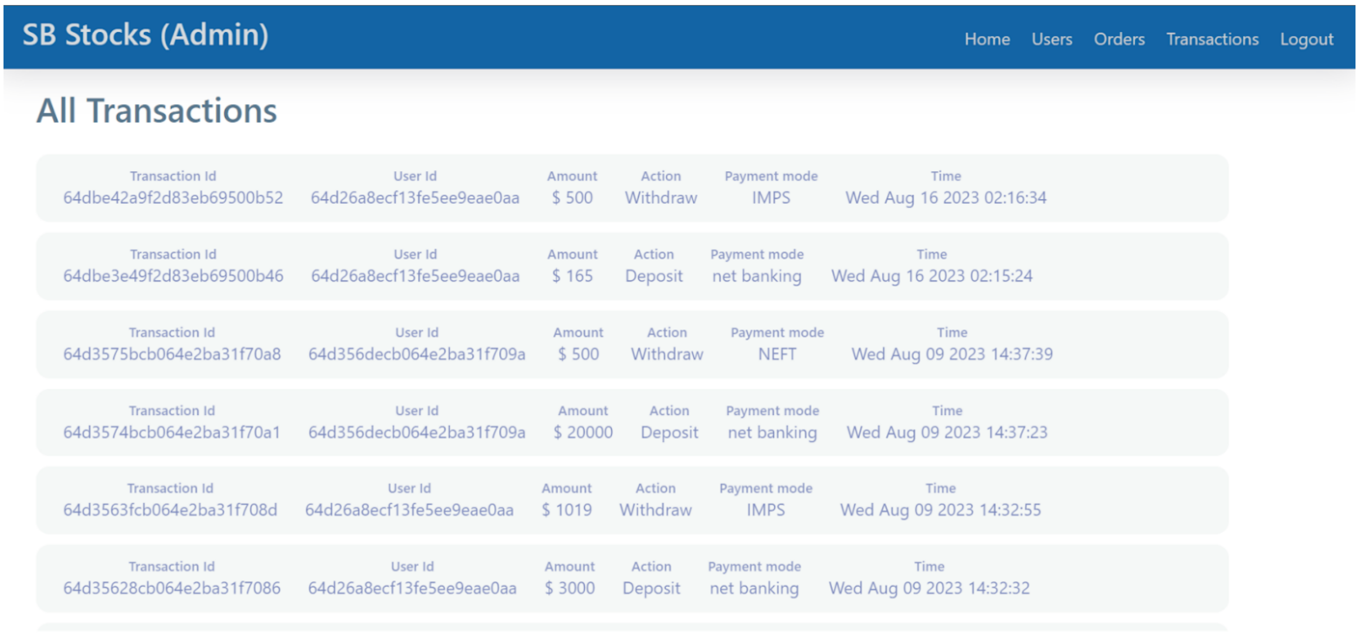
All users (Admin):



All users

User id	Username	Email	Balance
64d26a8ecf13fe5ee9eae0aa	hola	hola@gmail.com	5375
64d356decb064e2ba31f709a	jack	jack@gmail.com	19217

All transactions (Admin):



All Transactions

Transaction Id	User Id	Amount	Action	Payment mode	Time
64dbe42a9f2d83eb69500b52	64d26a8ecf13fe5ee9eae0aa	$ 500	Withdraw	IMPS	Wed Aug 16 2023 02:16:34
64dbe3e49f2d83eb69500b46	64d26a8ecf13fe5ee9eae0aa	$ 165	Deposit	net banking	Wed Aug 16 2023 02:15:24
64d3575bcb064e2ba31f70a8	64d356decb064e2ba31f709a	$ 500	Withdraw	NEFT	Wed Aug 09 2023 14:37:39
64d3574bcb064e2ba31f70a1	64d356decb064e2ba31f709a	$ 20000	Deposit	net banking	Wed Aug 09 2023 14:37:23
64d3563fcb064e2ba31f708d	64d26a8ecf13fe5ee9eae0aa	$ 1019	Withdraw	IMPS	Wed Aug 09 2023 14:32:55
64d35628cb064e2ba31f7086	64d26a8ecf13fe5ee9eae0aa	$ 3000	Deposit	net banking	Wed Aug 09 2023 14:32:32

Chapter 5
ADVANTAGES & DISADVANTAGES

5.1 List of advantages and disadvantages of the proposed solution

Advantages:

1. **Accessibility:** Users can access the stock trading platform from any device with an internet connection, enabling trading on-the-go.
2. **Real-Time Information:** Integration with APIs allows users to access real-time stock market data, enabling informed decision-making.
3. **User-Friendly Interface:** A well-designed user interface provides an intuitive trading experience, making it easy for users to navigate and execute trades.
4. **Customization:** Users can customize their trading experience, such as setting up watchlists, alerts, and notifications based on their preferences.
5. **Automation:** Implementing features like algorithmic trading or automatic portfolio rebalancing can help users automate their investment strategies.
6. **Scalability:** The application can scale to accommodate a growing user base and handle increased trading volumes without significant infrastructure changes.
7. **Cost-Effective:** Compared to traditional brokerage firms, online stock trading platforms often have lower fees and commissions, making it more cost-effective for users.

Disadvantages:

1. **Security Risks:** Stock trading platforms deal with sensitive financial data, making them attractive targets for cyberattacks. Security breaches can lead to financial losses and damage to the platform's reputation.
2. **Regulatory Compliance:** Compliance with financial regulations and licensing requirements can be complex and time-consuming, particularly in different jurisdictions.
3. **Technical Complexity:** Developing and maintaining a full-stack stock trading web app requires expertise in frontend and backend development, database management, security, and financial markets.
4. **Market Volatility:** Stock markets are inherently volatile, and fluctuations in stock prices can impact users' investment portfolios and overall trading activity.
5. **Liquidity Concerns:** Illiquid stocks or low trading volumes in certain markets may result in delays or difficulties in executing trades at desired prices.

6. **Dependency on External APIs:** Integration with external APIs for market data may introduce dependencies and potential points of failure if the API provider experiences downtime or changes its terms of service.
7. **Competition:** The online brokerage industry is highly competitive, with many established players and new entrants vying for market share. Differentiating the platform and attracting users can be challenging.

Chapter 6

APPLICATIONS

6.1 The areas where this solution can be applied

1. **Retail Investing:**
    - Individuals can use the platform to buy and sell stocks, ETFs, mutual funds, and other financial instruments directly through their web browser or mobile device.
2. **Portfolio Management:**
    - Investors can use the platform to manage their investment portfolios, track performance, set investment goals, and rebalance their portfolios as needed.
3. **Research and Analysis:**
    - Traders and investors can access real-time market data, news, research reports, and technical analysis tools to make informed trading decisions.
4. **Algorithmic Trading:**
    - Advanced users can implement automated trading strategies using algorithms and trading bots, leveraging historical data and technical indicators to execute trades automatically.
5. **Financial Education:**
    - The platform can serve as an educational resource, providing tutorials, webinars, and other learning materials to help users understand financial markets and investment strategies.
6. **Financial Planning:**
    - Users can use the platform to set financial goals, create savings plans, and track progress towards their goals over time.
7. **International Trading:**
    - The platform can support trading in multiple markets and currencies, allowing users to invest in international stocks and diversify their portfolios.
8. **Institutional Trading:**
    - Institutional investors, such as hedge funds, asset managers, and pension funds, can use the platform for trading and managing large portfolios of assets.
9. **Financial Advisory Services:**
    - Financial advisors can use the platform to provide investment advice, portfolio management services, and personalized financial planning to their clients.

10. **Compliance and Reporting:**

    - The platform can generate regulatory reports, tax statements, and other compliance documents required by financial authorities.

11. **Risk Management:**

    - Users can assess and manage investment risks using risk management tools, including portfolio diversification, stop-loss orders, and risk tolerance assessments.

12. **Social Trading:**

    - The platform can facilitate social trading features, allowing users to follow and copy the trades of experienced investors or trading communities.

Chapter 7

CONCLUSION

7.1 Conclusion summarizing the entire work and findings.

In conclusion, a stock trading web app offers a modern and accessible platform for individuals and institutions to engage with financial markets. By providing real-time data, intuitive interfaces, and customizable features, these platforms empower users to make informed investment decisions and execute trades efficiently. However, the development and maintenance of such platforms come with challenges, including security risks, regulatory compliance, and technical complexity. Despite these considerations, a well-executed stock trading web app has the potential to revolutionize the way users interact with financial markets, offering cost-effective solutions and scalability to accommodate growing user bases. Ultimately, success in this space requires a careful balance of innovation, security, and user-centric design to deliver value to both novice investors and seasoned professionals alike. The future of stock trading web apps is characterized by innovation, accessibility, and user-centric design. By embracing emerging technologies, regulatory changes, and evolving market trends, these platforms will continue to empower investors, foster financial inclusion, and shape the future of finance.

Chapter 8

FUTURE SCOPE

Enhancements that can be made in the future.

The future scope for stock trading web apps is expansive, marked by technological advancements, evolving user preferences, and regulatory changes. Here are some key areas of potential growth and innovation:

1. **AI and Machine Learning Integration:** Incorporating AI and machine learning algorithms will enable predictive analytics, sentiment analysis, and algorithmic trading strategies. These technologies can help users make informed investment decisions based on data-driven insights and market trends.
2. **Blockchain Technology:** Utilizing blockchain technology for transparent and secure transactions has the potential to revolutionize the trading ecosystem. Smart contracts, decentralized exchanges, and tokenization of assets can streamline processes, reduce costs, and enhance trust among market participants.
3. **Mobile Trading Dominance:** With the increasing prevalence of smartphones and mobile apps, the future of stock trading will likely be mobile-first. Enhanced mobile trading platforms with intuitive interfaces, real-time notifications, and advanced charting tools will cater to the preferences of modern investors who value convenience and accessibility.
4. **Social Trading Platforms:** Social trading platforms will continue to gain popularity, allowing users to follow, interact with, and replicate the trades of successful investors. These platforms foster community engagement, knowledge sharing, and collaborative decision-making, empowering users of all experience levels to participate in the market.
5. **Regulatory Innovation:** Regulatory frameworks will evolve to accommodate technological advancements and ensure investor protection. Regulatory sandboxes, digital identity verification, and standardized reporting requirements will foster innovation while maintaining market integrity and compliance with regulatory standards.
6. **Global Expansion and Access to Emerging Markets:** Stock trading web apps will expand their reach to emerging markets, providing access to a broader range of investment opportunities. Cross-border trading, international market access, and localized platforms will cater to the growing demand for global investment diversification.

7. **Green and Sustainable Investing:** The rise of environmental, social, and governance (ESG) investing will drive demand for platforms that offer sustainable investment options and ESG scoring metrics. Investors increasingly prioritize companies with strong ESG profiles, and stock trading web apps will facilitate the identification and evaluation of these opportunities.
8. **Partnerships and Ecosystem Integration:** Collaboration with fintech startups, financial institutions, and technology providers will drive innovation and expand the capabilities of stock trading web apps. Integration with banking, insurance, and wealth management services will create comprehensive financial ecosystems that offer seamless user experiences and personalized services.