A Full-Stack Stock Trading Platform Using MERN Stack with JWT Authentication

Prof. O. M. Patil[1], Prof. Nagraj P. Kamble[2], Sayali Doke [3], Shravani Kadwade [4], Vyankatesh Shinde [5]

[1,2,3,4,5] *Department of Information Technology M. S. Bidve Engineering College, Latur, Maharashtra, India.*

***Abstract*—This page presents the design and development of a web-based Stock Trading Platform implemented using the MERN stack (MongoDB, Express.js, React.js, and Node.js) to provide a secure and interactive virtual trading environment for beginners and stock market enthusiasts. The platform enables users to register and authenticate securely, access real-time stock market data through third-party APIs, and perform virtual buy and sell operations using a simulated balance without financial risk. It offers features such as order history tracking, portfolio performance analysis, and graphical insights through interactive dashboards, helping users understand market trends and trading strategies effectively. The system emphasises scalability, responsiveness, and data security through JWT-based authentication and encrypted password storage, ensuring a reliable user experience across devices. By bridging the gap between theoretical knowledge and practical exposure, the proposed platform serves as an effective learning and practice tool for enhancing stock market awareness and decision-making skills.**

***Index Terms*—Stock Trading Platform, MERN Stack, Virtual Trading, Web Application, Real-Time Stock Data**

I. INTRODUCTION

Stock trading is a major component of the global financial system, allowing individuals to invest capital and generate returns. However, real-world stock trading involves significant financial risk and requires practical knowledge of market behaviour. Beginners often find it difficult to participate due to a lack of experience and fear of losses.

Virtual trading platforms address this challenge by providing simulated environments where users can practice trading without real monetary involvement. This project focuses on developing a web-based stock trading platform using the MERN stack that allows users to trade stocks virtually, analyse market movements, and monitor portfolio performance. The system aims to bridge the gap between theoretical knowledge and real-world trading experience.

II. LITERATURE REVIEW

Several studies highlight that modern full-stack web technologies such as React.js, Node.js, and MongoDB are widely used to develop interactive and scalable financial applications. These technologies support real-time data processing, asynchronous communication, and responsive user interfaces, which are essential for stock trading platforms that rely on live market data and frequent user interactions [2], [3].

Research on virtual or paper trading platforms emphasizes their role in improving financial literacy and practical understanding of stock market operations. Such systems allow beginners to experiment with trading strategies, analyze market trends, and manage portfolios without financial risk, thereby enhancing learning outcomes compared to traditional theoretical methods [1].

Studies in financial technology also discuss the growing use of data analytics and performance tracking in trading platforms. Features such as order history analysis, portfolio visualization, and graphical insights help users evaluate their trading behaviour and make informed decisions, which aligns with the design goals of educational trading systems [3], [12].

Recent academic work explores the integration of machine learning and artificial intelligence in trading systems to predict stock prices and automate decision-making. Techniques such as deep reinforcement learning and LSTM networks have shown promising results in optimising trading strategies based on historical data and market patterns [9], [10].

Literature helps conclude that while many existing platforms focus on professional trading, there is a growing demand for user-friendly, educational trading applications that combine real-time data, security, and analytics, validating the need for the proposed stock trading platform [1], [3].

III. PROBLEM STATEMENT

Despite the availability of numerous stock trading applications, beginners often face challenges such as complex interfaces, a lack of proper guidance, and exposure to financial risk. Many existing platforms are designed primarily for professional traders and require real monetary investment, which discourages learners from experimenting and understanding trading strategies freely. This creates a gap between theoretical learning and practical experience in stock market education.

The problem addressed by this project is the lack of a secure, beginner-friendly, and risk-free trading environment that closely mimics real stock market conditions. There is a need for a platform that allows users to practice buying and selling stocks, analyse performance, and understand market trends using real-time data without financial loss. The proposed system aims to solve this problem by offering a virtual trading platform with simulated funds and real-time market insights. The platform empowers beginners to learn through trial and error in a safe environment.

IV. PROPOSED SYSTEM

The proposed system is a web-based Stock Trading Platform developed using the MERN stack to provide a complete virtual trading experience. It enables users to register and authenticate securely using JWT-based authentication and encrypted passwords. Once logged in, users can access real-time stock prices, view market trends, and perform virtual buy and sell operations using a simulated balance.

In addition to trading functionality, the system provides detailed order history and performance analytics through interactive charts and dashboards. These features help users track their trading activities, analyse profits and losses, and improve decision-making skills. The platform is designed to be responsive and accessible across devices, ensuring ease of use and scalability for future enhancements.

Following are the ER-Diagram and Data Flow Diagrams :

A. Data Flow Diagram:

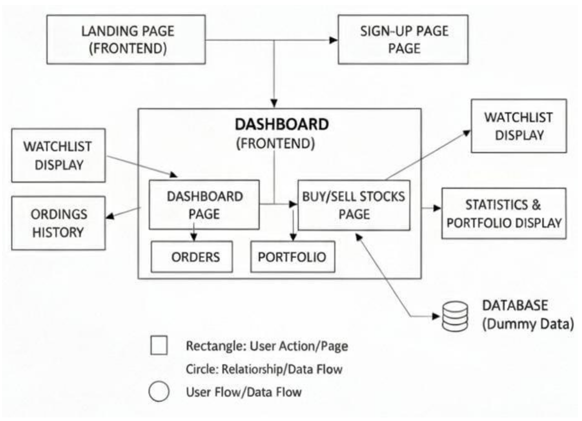

Fig.: Data Flow Diagram

V. SYSTEM ARCHITECTURE

The architecture is divided into three logical layers :

- Presentation Layer
- Application Layer
- Data Layer

Each layer performs a distinct function and communicates with adjacent layers to process user requests and generate appropriate responses.

A. Presentation Layer

The presentation layer is responsible for user interaction and interface rendering. It is developed using React.js along with Tailwind CSS or Bootstrap to create a responsive and interactive user interface. This layer displays pages such as homepage, signup, login, dashboard, pricing, and support sections, ensuring smooth navigation and an intuitive user experience.

The presentation layer communicates with the backend through RESTful APIs to fetch real-time stock data, user information, and transaction history. By separating UI logic from business logic, the system ensures better maintainability and scalability while enhancing user engagement through dynamic components and charts.

B. Application Layer
The application layer handles the core business logic of the system and is implemented using Node.js and Express.js. This layer processes user requests, manages authentication, validates transactions, and interacts with external stock market APIs to retrieve live data. It acts as a bridge between the presentation layer and the database.

This layer also enforces security rules, such as JWT validation and role-based access control, to ensure that only authorised users can perform trading operations. By modularising services such as user management, trading logic, and analytics, the application layer ensures efficient request handling and system scalability.

C. Data Layer
The data layer uses MongoDB as a NoSQL database to store user profiles, transaction records, portfolio details, and order history. MongoDB's flexible schema allows efficient handling of dynamic trading data and user activities. The database ensures quick data retrieval and storage for real-time performance.

This layer supports data consistency and integrity by maintaining structured collections for users, trades, and stock records. Secure access to the database is ensured through environment variables and controlled API endpoints, preventing unauthorised data manipulation.

VI. METHODOLOGY

A. Requirement Analysis
The development process began with a detailed requirement analysis to identify user needs and system objectives. Functional requirements included user authentication, real-time stock data display, virtual trading, and performance tracking. Non- functional requirements focused on security, scalability, and responsiveness.

This phase ensured that the system design aligns with the learning goals of beginners while maintaining realistic market simulations. Clear requirements helped in selecting appropriate technologies and defining system workflows effectively.

B. System Design
System design involved defining the overall architecture, data flow, and interaction between different layers. ER diagrams and flowcharts were used to visualise database relationships and system processes. This phase helped in identifying key modules such as authentication, trading engine, and analytics.

A modular design approach was adopted to allow future expansion and easier maintenance. Clear separation of concerns ensured efficient communication between frontend, backend, and database components.

C. Frontend Development
Frontend development focused on building a responsive and interactive user interface using React.js. Components were designed for login, signup, dashboard, and trading screens. Tailwind CSS or Bootstrap was used to enhance UI consistency and responsiveness across devices.

Dynamic rendering and state management enabled real-time updates of stock prices and portfolio values. User-friendly layouts and navigation improved usability and overall user experience.

D. Application Logic Implementation
The backend logic was implemented using Node.js and Express.js to handle API requests and business rules. Trading logic ensured proper validation of buy and sell operations based on virtual balance and stock availability. Middleware was used to manage authentication and request validation.

Integration with third-party stock APIs enabled real-time data fetching and market updates. Error handling and logging mechanisms were implemented to maintain system reliability.

E. Data Management
MongoDB was used for efficient storage and retrieval

of user and trading data. Collections were designed to manage users, orders, and transaction history effectively. Indexing techniques improved query performance for large datasets.

Data consistency was maintained through validation rules and controlled update operations. Backup strategies ensured data safety and reliability.

F. Security Measures

Security was a key focus of the system, with JWT-based authentication ensuring secure session management. Passwords were encrypted using Bcrypt to protect user credentials. API endpoints were secured to prevent unauthorized access.

Additional measures included input validation, secure environment variables, and protection against common web vulnerabilities. These measures ensured data integrity and user trust in the platform. Role-Based Access Control (RBAC): The system can implement role-based access control to restrict functionalities based on user roles, such as normal users and administrators.

VII. IMPLEMENTATION DETAILS

A. Programming Languages

The system uses JavaScript as the primary programming language for both frontend and backend development. React.js handles client-side rendering, while Node.js manages server-side logic. This unified language approach improves development efficiency and maintainability.

JavaScript enables asynchronous data handling, which is essential for real-time stock updates and responsive user interactions. Its wide ecosystem supports integration with APIs and databases effectively.

B. Development Tools

Development tools include Visual Studio Code for coding. Postman for API testing, and git for version control. These tools facilitated efficient development, debugging, and collaboration during the project lifecycle.

Browser developer tools were used for frontend debugging, while MongoDB Compass assisted in database visualisation and management.

C. Execution Environment

The application runs in a web-based environment supported by modern browsers. The backend server operates on Node.js runtime, while MongoDB serves as the database engine. The system can be deployed on cloud platforms for scalability.

Local and production environments were configured using environment variables to ensure secure and flexible execution.

D. Transaction Handling Using JavaScript

Transaction handling is managed through JavaScript functions that validate user actions and update virtual balances accordingly. Buy and sell requests are processed atomically to ensure data consistency and accuracy.

Asynchronous operations handle real-time data fetching and transaction updates, ensuring a smooth user experience and reliable trading simulations.

VIII. RESULTS AND DISCUSSION

The implemented Stock Trading Platform successfully provides a realistic and interactive virtual trading environment. Users can register securely, access live stock data, and perform virtual trades with ease. The dashboard displays order history and performance analytics effectively.
The system demonstrates good responsiveness and scalability due to the MERN stack architecture. Real-time updates and interactive charts enhance learning outcomes, making the platform suitable for beginners and enthusiasts alike. This ensures that users are not just watching the market, but are actively preparing for future financial success in a safe, controlled space.

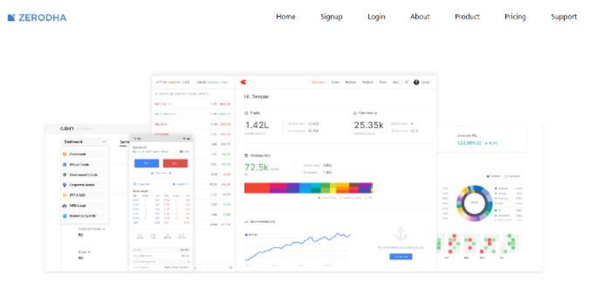

Fig.: Landing Page

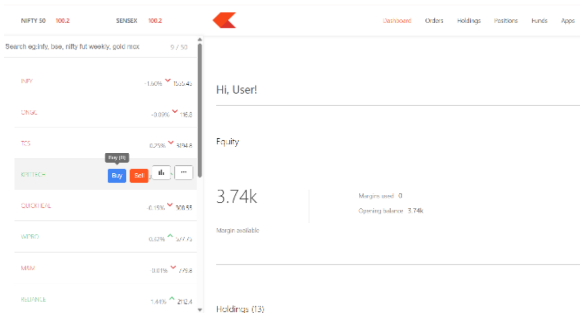

Fig.: Dashboard

IX. LIMITATIONS AND FUTURE ENHANCEMENTS

The current platform is primarily limited by its dependence on third-party APIs, which can lead to data latency or service interruptions. It also lacks real-market complexities such as brokerage fees, slippage, and advanced technical indicators like RSI or MACD. Additionally, the system requires constant internet connectivity and does not account for the psychological factors, such as panic selling, that influence real-world trading.

To evolve, the future scope includes integrating AI-driven price predictions and advanced charting tools to support deeper market analysis. We plan to develop a mobile application and introduce social trading features, allowing users to follow and learn from experienced traders. These updates will make the platform more accessible and provide a more comprehensive learning environment.

Finally, the system can be expanded to support global markets and real-market simulations, including transaction taxes and execution delays. Future versions may also offer integration with real brokerage APIs for live trading. By adding enhanced security features like two-factor authentication, the platform will transition from a basic simulator into a robust, professional-grade trading tool.

X. CONCLUSION

The Stock Trading Platform successfully achieves its objective of providing a secure, scalable, and user-friendly virtual trading environment. By leveraging the MERN stack and real-time stock APIs, the system offers a practical learning tool for understanding stock market dynamics.

The platform bridges the gap between theoretical knowledge and practical experience, allowing users to develop trading skills without financial risk.

The modular architecture adopted in the platform ensures ease of maintenance, scalability, and future enhancement, making it adaptable to evolving market and technological requirements.

The integration of real-time stock market APIs enhances the realism of the trading environment and improves user engagement by providing up-to-date market information.

Through performance analytics and order history tracking, the system enables users to develop data-driven decision-making skills, which are essential in real-world trading scenarios.

The successful implementation of this project confirms that web-based simulation systems can effectively bridge the gap between theoretical financial knowledge and practical market exposure.

Overall, the Stock Trading Platform stands as a robust, scalable, and educationally valuable system, capable of supporting learners, educators, and stock market enthusiasts in understanding trading concepts and market dynamics.

XI. ACKNOWLEDGMENT

I would like to express my sincere gratitude to my project supervisor and the faculty members for their invaluable guidance and constant support throughout the development of this Stock Trading Platform. I am also thankful to my peers for their constructive feedback and to the open-source community for providing the tools and resources that made this project possible. Finally, I appreciate the support of my family and friends, whose encouragement helped me stay focused on achieving the goals of this project and enhancing my technical expertise in the MERN stack.

REFERENCES

[1] P. Rakibe et al., Virtual Trade-X (The Paper Trading Web-Application), International Journal of Novel Research and Development (IJNRD), Vol. 9, Issue April 2024. Available: https://www.ijnrd.org/papers/IJNRD2404602.pd f

[2] S. Dilakshan, Authentication & JWT in

MERN Stack, Medium article (2025). Available: https://medium.com/@sivanathandilakshan/auth entication-jwt-in-mern-stack-946925e382da

[3] Stock Portfolio Tracker using MERN Stack, GeeksforGeeks tutorial. Available: https://www.geeksforgeeks.org/mern/stock-portfolio-tracker-using-mern-stack/

[4] MongoDB Documentation, “MongoDB Manual,” Available: https://www.mongodb.com/docs/

[5] REST API Tutorial, “RESTful Web Services,” Available: https://restfulapi.net/

[6] GitHub, “GitHub Documentation,” Available: https://docs.github.com/

[7] N. Provos and D. Mazieres, “A Future-Adaptable Password Scheme,” Available: https://www.usenix.org/legacy/events/usenix99/provos.html

[8] JSON Web Token, “JSON Web Token (JWT) Introduction,” Available: https://jwt.io/introduction

[9] Node.js Documentation, “Node.js v20 Documentation,” Available: https://nodejs.org/en/docs

[10] React Documentation, “React – A JavaScript library for building user interfaces,” Avai Available:

[11] Express.js Documentation, “Express Web Framework,” Available: https://expressjs.com/

[12] OWASP Foundation, “OWASP Top Ten Web Application Security Risks,” Available: https://owasp.org/www-project-top-ten/

[13] S. Suresh & D. Shankar, *A Survey on Online Stock Trading Systems and Market Simulation, International Journal of Computer Applications (2024).

[14] M. Patel, *Real-Time Data Integration in Stock Market Analysis using Web APIs, International Journal of Software Engineering and Applications (2025).

[15] H. Khasnabish & A. Kumar, *Web-Based Interactive Financial Learning Systems: Design and Evaluation, Journal of Web Systems and Technologies (2023). Available: [https://www.sciencedirect.com/science/article/pii/S2352711023000456] (https://www.sciencedirect.com/science/article/pii/S2352711023000456)