Mini Project 2 report on

"Full Stack E-Commerce Shoe Website using MERN"

A mini project dissertation submitted in partial fulfilment of the requirement for the award of degree

MASTER OF COMPUTER APPLICATIONS

by

SHIVAM SINGH

1BY23MC081

Under the Guidance of

Prof Nirupama B K

Assistant Professor - II

Department of MCA

BMSIT&M

Bengaluru

Department of Master of Computer Applications

BMS Institute of Technology and Management

Bengaluru

MARCH - 2025

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Bengaluru

MARCH - 2025

Department of MCA

(Accredited by NBA, New Delhi)

CERTIFICATE

This is to certify that the dissertation titled **Full Stack E-Commerce Shoe Website using MERN** submitted in partial fulfilment of the requirements for the degree **"Master of Computer Applications"** by Visvesvaraya Technological University is based on an original study and is record of bonafide work carried out by **Shivam Singh** bearing university registration number **1BY23MC081** during the period **December 2024 to March 2025** under our supervision and guidance and that no part of the report has been submitted for the award of any other Degree/ Diploma/ Fellowship or similar title or prizes. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the Master of Computer Applications Degree.

Signature of the Guide
Prof Nirupama B K
Assistant Professor - II
Department of MCA
BMSIT&M
Bengaluru.

Signature of HoD
Dr. M.Sridevi
Assistant Professor & Head
Department of MCA
BMSIT&M
Bengaluru

DECLARATION

I **Shivam Singh**, student of MCA, BMS Institute of Technology & Management, bearing USN **1BY23MC081** hereby declare that Mini project (22MCA307) titled **Full Stack E-Commerce Shoe Website using MERN** has been carried out by me under the guide **Prof Nirupama B K** and submitted in the partial fulfilment of the requirements for the award of Degree of Master of Computer Applications by the Visvesvaraya Technological University during the academic year 2024-25. This report has not been submitted to any other Organization/University for any award of degree or certificate.

ACKNOWLEDGEMENT

I would like to utilize this opportunity to express gratitude to each person who made it possible for me to complete my project successfully. Thus, I would like to remark few people, whom I want to thank and express sincere gratitude.

I convey my truthful gratitude to BMSIT&M Management for providing a good infrastructure and educational support in lighting our career.

I would like to show my sincere gratitude to our Principal **Dr. Sanjay H A** for his kind support in completing this project.

I take this opportunity to thank our **Head of Department, Dr. M. Sridevi** who supported me with her valuable inputs on this project.

I express my deep sense of gratitude to my internal guide **Prof Nirupama B K** for his support, encouragement and valuable inputs throughout the completion of this project.

I also thank all my professors and non-teaching staff members, who contributed their help and support directly or indirectly in completing this project.

Last but not the least, I thank my parents and friends who stood with me as a moral support and encouraging me in accomplishing this project.

Shivam Singh

1BY23MC081

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Bengaluru

Department of MCA

VISION

To emerge as a leading department in computer applications, producing skilled professionals equipped to deliver sustainable solutions.

MISSION

Facilitate effective learning environment through quality education, industry interaction with orientation towards research, critical thinking and entrepreneurial skills.

Programme Educational Objectives (PEOs)

PEO 1: Develop innovative IT applications to meet industrial and societal needs.

PEO 2: Adapt themselves to evolving domain requirements.

PEO 3: Exhibit leadership skills and progress in their chosen career path.

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Bengaluru

Department of MCA

Programme Outcomes (POs)

PO1: Apply knowledge of mathematics, programming logic and coding fundamentals for solution architecture and problem solving.

PO2: Identify, review, formulate and analyse problems for primarily focusing on customer requirements using critical thinking frameworks.

PO3: Design, develop and investigate problems with an innovative approach for solutions incorporating ESG/SDG goals.

PO4: Select, adapt and apply modern computational tools such as development of algorithms with an understanding of the limitations including human biases.

PO5: Function and communicate effectively as an individual or a team leader in diverse and multidisciplinary groups using methodologies such as agile.

PO6: Use the principles of project management such as scheduling, work breakdown structure and be conversant with the principles of Finance for profitable project management.

PO7: Commit to professional ethics in managing software projects with financial aspects. Learn to use new technologies for cyber security and insulate customers from malware.

PO8: Change management skills and the ability to learn, keep up with contemporary technologies and ways of working.

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT

(An Autonomous Institution, Affiliated to VTU, Belagavi)

Bengaluru – 560064

Department of MCA

Course Outcomes (COs)

CO 1: Analyse the given requirements.

CO 2: Design a suitable system model.

CO 3: Develop the solution using appropriate tools.

CO 4: Prepare effective documentation.

CO 5: Involve in team work.

ABSTRACT

The E-Commerce Shoe Website is a complete full-stack online application created with the MERN stack (MongoDB, Express.js, React.js, and Node.js). It provides a fluid and engaging online shopping experience, allowing customers to browse a diverse range of footwear, filter goods based on numerous criteria, and make purchases securely using an integrated Stripe payment platform. The platform has a mobile-responsive design, allowing for seamless navigation across several devices while keeping an intuitive and user-friendly interface. This website contains essential e-commerce features including user authentication and authorisation, an admin dashboard for managing products and orders, and a shopping cart system that allows users to add things, place orders, and monitor their transactions in real time.

The backend is backed by a restful API that effectively manages product data, orders, and authentication. JWT authentication with bcrypt.js password encryption improve security. MongoDB with Mongoose manages data storage and retrieval, resulting in a scalable and dependable system.

Keywords:

E-Commerce, MERN Stack, Online Shopping, Product Management, User Authentication, Payment Gateway, RESTful API, MongoDB, React.js, Node.js, Secure Transactions, Admin Dashboard, Mobile-Responsive Design, Scalable Architecture, Feature-Rich Platform, Order Management, Search and Filtering, Inventory Management, JWT Authentication.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1. Project Description

As e-commerce continues to transform the retail landscape, online shopping has become a crucial aspect of modern consumer behaviour. The Full-Stack E-Commerce Shoe Website is intended to deliver a smooth and engaging purchasing experience built exclusively for footwear aficionados. This platform, built on the MERN stack (MongoDB, Express.js, React.js, and Node.js), seeks to provide a feature-rich retail experience. Users may easily explore a wide selection of footwear, narrow their search using extensive filtering tools, and make purchases safely. To protect data security and user privacy, the system includes JWT authentication and Bcrypt.js encryption, while Stripe or Razorpay will handle payment processing smoothly and securely.

An easy admin dashboard will provide shop managers all the tools they need to effectively manage inventory, handle orders, and control customer accounts. The frontend will be created using Material-UI or Tailwind CSS, resulting in a responsive, visually appealing interface that provides a consistent experience across all devices. This initiative is more than simply an e-commerce site; it wants to transform the way people buy for footwear online. Future upgrades, such as AI-powered product suggestions, user interaction tools, and a rewards program, will further personalise the shopping experience, making it more dynamic and customer-focused.

This project is more than simply an e-commerce site; it is a better digital buying experience that prioritises consumer comfort and corporate productivity. Whether it is exploring a vast assortment of footwear, choosing goods based on tastes, or completing transactions smoothly, our platform makes every step of the trip simple. The MERN stack (MongoDB, Express.js, React.js, Node.js) serves as the system's backbone, giving the flexibility and scalability needed for a seamless and future-ready e-commerce solution. With React.js powering the frontend, customers can anticipate an interactive and dynamic experience that seamlessly adjusts to multiple devices. The Node.js and Express.js backends, together with a MongoDB database, provide reliable data processing, smooth transactions, and real-time order tracking. One of the most pressing problems in e-commerce is security, which this site handles seriously. JWT (JSON Web Token) authentication enables secure user access, while Bcrypt.js encrypts passwords for an extra degree of safety. Payment processing is performed by Stripe or Razorpay, guaranteeing quick, dependable, and secure transactions without jeopardising user confidence. Additionally, store managers have access to an admin dashboard that streamlines inventory and order administration. From changing product listings to tracking sales and customer accounts, the admin panel is intended to simplify operations while keeping everything organised.

1.2. Technologies Used

To create a scalable, safe, and efficient e-commerce platform, this project employs the MERN (MongoDB, Express.js, React.js, Node.js) stack, as well as other technologies for smooth functioning and deployment. The important technologies are:

Frontend (Client Side)
React.js is a JavaScript library for creating interactive and dynamic user interfaces.
Redux is a state management library that improves data flow efficiency.
Material UI/Tailwind CSS - UI frameworks for creating responsive and visually attractive interfaces.
Axios enables HTTP requests to connect with the backend

Backend (Server Side)
Node.js is a server-side runtime for running JavaScript.
Express.js is a lightweight framework for creating RESTful APIs.
Mongoose is an Object Data Modelling (ODM) library for MongoDB interactions.
JWT (JSON Web Token) enables secure user authentication and session management.
Bcrypt.js encrypts user passwords for improved security.

Database
MongoDB is a NoSQL database for storing user accounts, product listings, orders, and transactions.
MongoDB Atlas is a cloud-based database solution that provides high availability and scalability.

Payment Integration:
Stripe is a secure payment gateway that swiftly processes transactions.
Version control tools include Git and GitHub, which enable collaboration and management of code repositories.

Deployment and DevOps:
Vercel is used to deploy both frontend and backend, guaranteeing scalable hosting.

By combining these latest technologies, our project assures a quick, secure, and scalable e-commerce platform that provides an amazing customer experience. From dynamic UI components and state management on the frontend to robust authentication, order processing, and secure transactions on the backend, each component is methodically built to provide a dependable and future-proof online retail system. With possible future upgrades such as AI-powered suggestions, multi-vendor compatibility, and customer engagement capabilities, this platform has the potential to revolutionise the online footwear purchasing experience while maintaining a high level of performance and security.

1.3. Project Objectives

The Full-Stack E-Commerce Shoe Website is intended to give footwear fans with an easy-to-use, safe, and convenient online buying experience. The major purpose of this project is to create a scalable, efficient, and feature-rich e-commerce platform that can serve both consumers and administrators while maintaining a smooth, responsive, and intuitive interface. The primary goals of this initiative are listed below:

1. Integrated and Interactive User Experience: Create an engaging and intuitive shopping experience using React.js to ensure quick navigation and dynamic UI components. Create a mobile-responsive design using Material-UI or Tailwind CSS to provide a consistent experience across PCs, tablets, and smartphones. Provide real-time product filtering and search functionality to increase user convenience and product discovery.

2. Secure Authentication and User Management: Implement JWT (JSON Web Token) authentication to provide safe user login and session management. Use Bcrypt.js to encrypt user passwords, assuring high levels of security. To prevent unauthorised actions, enable adequate access control measures for user roles (customer and admin).
Allow users to easily create accounts, edit profiles, and view their order history.

3. Effective Product and Inventory Management: Provide a feature-rich admin interface that allows shop managers to add, update, and delete goods from their inventory. Maintain a well-organised product catalogue with detailed information such as product photos, descriptions, sizes, colours, and pricing. Implement automatic stock tracking to ensure real-time updates on product availability and avoid overselling.

4. Effective Shopping Cart and Checkout Process: Allow clients to add things to their shopping carts, change amounts, and remove products as necessary. Implement a dynamic cart system that changes totals in real time to provide an accurate checkout experience.
Allow visitor and registered user checkouts to improve accessibility and simplicity of usage.

5. Safe Payment Gateway Integration: Use Stripe or Razorpay to provide a safe and convenient online payment experience. Accept different payment options, such as credit/debit cards, UPI, and digital wallets. Implement transaction encryption and fraud protection methods to protect users' financial information.

6. Order Management and Tracking System: Allow consumers to follow their orders in real time, from placement to delivery. Allow administrators to monitor orders, amend statuses, and alert customers about delivery information. Set up automated email alerts to keep consumers updated about purchase confirmations, dispatch, and delivery updates.

2. LITERATURE SURVEY

The e-commerce sector has expanded rapidly over the last decade, with firms turning to digital platforms to improve consumer experience and market reach. A full-stack e-commerce website for selling shoes demands a strong, scalable, and efficient foundation. The MERN stack (MongoDB, Express.js, React.js, and Node.js) is widely used for creating dynamic and interactive online applications. This literature review examines existing research and technology implementations linked to e-commerce platforms, with an emphasis on MERN stack applications.

2.1. Related Work

The e-commerce sector has expanded rapidly over the last decade, with firms turning to digital platforms to improve consumer experience and market reach. A full-stack e-commerce website for selling shoes demands a strong, scalable, and efficient foundation. The MERN stack (MongoDB, Express.js, React.js, and Node.js) is widely used for creating dynamic and interactive online applications. This literature review examines existing research and technology implementations linked to e-commerce platforms, with an emphasis on MERN stack applications.
React-based E-Commerce Solutions: Research shows that React's component-based structure improves UI performance and reusability. According to studies, engaging and responsive user interfaces designed using React result in higher user engagement and conversion rates. React's virtual DOM technique reduces page reloads and improves efficiency, making it an excellent choice for e-commerce websites. Database Performance for E-Commerce: Comparisons of SQL and NoSQL databases show that MongoDB is more scalable for dynamic data processing. According to studies, conventional relational databases frequently fail to handle high amounts of read/write operations in e-commerce systems. In contrast, MongoDB's document-oriented architecture enables flexible and rapid data retrieval, lowering latency and enhancing site performance.
Authentication and Security: Studies emphasise the need of safe user authentication with JWT (JSON Web Token) and OAuth. According to research, adding multi-factor authentication (MFA) and role-based access control (RBAC) improves security on e-commerce systems. Security vulnerabilities like as SQL injection and cross-site scripting (XSS) continue to be key problems, and studies propose deploying HTTPS, data encryption, and safe API procedures to reduce these risks. Payment Gateway Integration: Research indicates that integrating services such as Stripe and PayPal improves transaction security and efficiency. According to studies on secure online transactions, people favour platforms that offer convenient and reliable payment solutions. Implementing tokenisation and AI-based fraud detection systems can help to minimise payment fraud and chargebacks.
Scalability of MERN-Based E-Commerce Applications: Studies show that when e-commerce enterprises expand, their websites must be able to manage more traffic effectively. Research indicates that using a microservices design rather than a monolithic one might considerably enhance scalability. Load balancing strategies and the usage of caching solutions such as Redis and CDN services are advised for increasing performance in high-traffic environments.

2.2. Problem Definition

One of the most typical issues with online shoe businesses is inadequate website design and navigation. Many platforms do not provide a simple and dynamic user interface, making it difficult for clients to browse, search, and filter items efficiently. A lack of optimised UI/UX might result in high bounce rates and abandoned carts, reducing business income. Furthermore, many websites are not mobile friendly, which limits accessibility for people who like to purchase on smartphones and tablets.E-commerce platforms face significant security risks since they handle sensitive user information such as login passwords and payment information. Many online retailers still use inadequate authentication techniques, leaving them vulnerable to cyber dangers including brute-force assaults, phishing, and session hijacking. E-commerce has transformed the retail industry, enabling firms to access a larger audience while giving customers with a more comfortable purchasing experience. The footwear business, in particular, has experienced an increase in online sales, as customers prefer to browse and buy shoes via digital platforms. Despite this expansion, many existing e-commerce shoe websites do not provide a seamless, safe, or efficient buying experience. Several issues, such as poor user experience, security risks, inefficient inventory management, and lack of scalability, continue to limit the efficacy of online footwear retailers.
To overcome these difficulties, it is critical to create a strong and feature-rich e-commerce platform that provides a seamless user experience while including current security measures and efficient backend management. The Full-Stack E-Commerce Shoe Website, developed using the MERN stack (MongoDB, Express.js, React.js, Node.js), attempts to address these difficulties by providing a high-performance, secure, and scalable solution customised to the demands of both customers and business owners.

2.3. Existing and Proposed System

Existing System
Many e-commerce platforms now provide online shopping options for footwear merchants, enabling users to explore and purchase items. However, despite their widespread use, these systems have some shortcomings that affect both customers and business owners. Many existing platforms lack a simple and visually appealing interface, making it difficult for customers to navigate items efficiently. Slow loading times, crowded UI, and poor mobile responsiveness all contribute to high bounce rates and low user retention. Users struggle to discover their selected shoes fast due to the limited product filtering and sorting choices.Weak authentication methods leave users vulnerable to password breaches, unauthorised access, and account hijacking. Many platforms utilise insecure password storing systems, which raises the danger of data breaches and assaults. Many existing systems have difficult checkout procedures, which leads to cart abandonment. Hidden costs, restricted payment alternatives, and the lack of real-time order monitoring all have a detrimental influence on the customer experience. Some systems do not provide an efficient shopping cart function, making it difficult for consumers to change their orders before checkout.

Proposed System

To solve the issues that present e-commerce platforms face, the proposed Full-Stack E-Commerce Shoe Website would use modern web technologies to improve user experience, security, and operational efficiency. This system will be constructed with the MERN stack (MongoDB, Express.js, React.js, and Node.js), which ensures a scalable, quick, and secure shopping experience. The website will have a modern and intuitive UI/UX, developed with React.js with Tailwind CSS, delivering a consistent experience across all platforms. Users will be able to locate their chosen items more quickly thanks to optimised search, filtering, and sorting capabilities. Fast-loading websites and easy navigation will increase engagement and lower bounce rates. To solve these issues, this project suggests developing a contemporary, full-stack e-commerce shoe shop with improved security, performance, and usability. The platform will be hosted on Vercel, which guarantees high availability, rapid speed, and little downtime.MongoDB Atlas will be utilised for cloud storage, enabling rapid data retrieval and scalability. The backend will be optimised to withstand heavy traffic, assuring seamless service even during busy shopping seasons. Store owners will have access to a specialised admin panel for simple product and order administration.Real-time inventory tracking ensures precise product availability while preventing overselling. Automated order processing enables store managers to complete requests promptly and efficiently. The checkout procedure will be streamlined with a one-click option, increasing consumer convenience. Credit and debit cards, digital wallets, and UPI will all be accepted as payment options.

2.4. System Study

System analysis is an important stage in the development of any software project since it aids in understanding the feasibility, scope, and effect of the system being built. A thorough system analysis assures that the planned system meets business objectives while remaining technically, operationally, and economically feasible. The Full-Stack E-Commerce Shoe Website is intended to deliver a smooth and efficient online buying experience, using innovative technologies to enhance security, performance, and customer happiness. This section assesses the proposed system's viability using technical, operational, and economic considerations to assure its effective implementation. The system study is a critical stage in the software development life cycle (SDLC) that ensures a complete review of the project's viability, scope, and overall effect. This phase aids in determining the feasibility of the planned system prior to its actual creation, therefore lowering risks, optimising resources, and assuring smooth execution. A well-conducted system study ensures that the final product meets corporate objectives, user needs, and industry standards while also resolving technological and financial restrictions. The Full-Stack E-Commerce Shoe Website is envisioned as a user-friendly, scalable, and feature-rich e-commerce platform for improving online buying experiences. The system combines cutting-edge technology to provide solid security, high performance, and seamless navigation, assuring customer happiness and corporate success.

2.4.1. Feasibility

A feasibility study is performed to examine whether the proposed system is viable for development and deployment. It reviews the project's technical, operational, and economic components to guarantee that it can be implemented effectively while remaining within resource limits.

The feasibility assessment for the Full-Stack E-Commerce Shoe Website assesses if necessary technology and resources are available. Operational feasibility is determining how effectively the system serves user demands and fits with existing procedures. Economic feasibility is the analysis of a project's financial viability and return on investment. Each of these factors is critical in deciding if the system should be constructed and how to optimise it for success. The project can be performed within the specified timeframe. The adoption of current web development frameworks such as MERN (MongoDB, Express.js, React.js, and Node.js) results in a more organised development process. Agile approaches provide incremental feature releases, which makes the project more manageable.

2.4.2. Technical Feasibility

The following elements contribute to the project's high technological feasibility: The project uses contemporary technologies including React.js (frontend), Node.js and Express.js (backend), MongoDB (database), and Redux (state management). These frameworks are well-documented and have widespread community support. MongoDB Atlas provides safe and scalable data storage, quickly managing user authentication, product listings, and order records. Infrastructure requirements include a Node.js/Express.js server for backend operations and a MongoDB database for product/user data storage. Cloud deployment options make infrastructure more manageable. Node.js supports secure payment gateways such as Stripe and Razorpay for smooth transactions.

2.4.3. Operational Feasibility

User Experience and Accessibility: The system has a user-friendly interface to facilitate easy navigation. Mobile responsiveness promotes accessibility across devices, which increases user engagement. Users may easily discover the goods they want thanks to advanced search and filtering features. Order and Inventory Management: A separate admin dashboard enables shop managers to effectively manage items, handle orders, and check inventories. Automated stock updates avoid overselling and supply shortages. Payment and Checkout Process: A more efficient checkout procedure lowers cart abandonment. Customers benefit from the availability of many payment alternatives. Real-time order tracking promotes openness and client happiness. Maintenance and Support: The system is designed with modular components, which makes future updates and maintenance easy. Version control (GitHub) guarantees that deployments go smoothly and bugs are fixed. User Accessibility is the system's interface is responsive and user-friendly, allowing for seamless interaction. Users may explore goods, add them to their carts, and make safe transactions. The admin panel allows for management of items, orders, and user accounts. This makes it simple to update the store dynamically. The system uses JWT-based authentication to provide secure login and authorisation. Role-based access guarantees that administrators and customers have the necessary permissions.

2.4.4. Economical Feasibility

Economic feasibility determines if the system's development and operating expenditures are justified by the anticipated benefits and revenue production. It assures that the investment in the system yields a profit. Cost Analysis in the MERN stack is completely open-source, with no licensing fees. The primary expenditures are server hosting (backend), cloud database storage (MongoDB Atlas), and payment gateway transaction fees. Return on Investment (ROI) provides a fully working e-commerce platform with the ability to generate large money through shoe sales, discount offers, and client retention programs. Scalability's future additions, like as AI-based product suggestions, customer reviews, and reward programs, can boost user engagement and profitability.

Cloud-based hosting and database administration eliminate the need for expensive on-premises servers. Automated operations, such as inventory updates and order management, reduce the cost of physical labour. Secure payment processing lowers fraud-related losses. The solution allows for 24/7 online sales, which increases income possibilities. Discounts, promotions, and loyalty programs boost client retention and repeat purchases. AI-based product suggestions (future scope) can increase sales.

2.5. System Requirements

2.5.1. Hardware Requirements:

- **Processor:** Intel i5 or higher (or AMD equivalent)
- **RAM:** Minimum **8GB** (recommended 16GB for smooth development)
- **Storage:** At least **100GB SSD** for storing development and project files.

2.5.2. Software Requirements:

- **Operating System:** Windows 10, macOS, or Linux.
- **Development Tools:**
  - **VS Code** (IDE for development)
  - **Postman** (for testing APIs)
- **Programming Languages & Frameworks:**
  - Frontend: React.js, Redux, Tailwind CSS/Material-UI
  - Backend: Node.js, Express.js
  - Database: MongoDB with Mongoose ORM
  - Authentication: JSON Web Tokens (JWT) and Bcrypt.js
  - Payment Gateway: Stripe or Razorpay for secure payments
- **Deployment Platforms:**
  - **Frontend:** Vercel or Netlify
  - **Backend:** Heroku or Render
  - **Database:** MongoDB Atlas (Cloud-based database hosting)

3. SOFTWARE REQUIREMENTS SPECIFICATION

3.1. Functional Requirements

Functional requirements define the specific behaviors, functionalities, and capabilities that the system must possess to meet user needs. For the **Full-Stack E-Commerce Shoe Website**, the functional requirements can be categorized as follows:

1. User Management- User Registration & Login: Users must be able to register an account and log in securely using email and password authentication. There should be separate access levels for customers and administrators. The system must implement password hashing and encryption to ensure security.

2. Product Management- Product Catalog: The website must display a list of available shoes with details such as name, price, size, color, brand, and stock availability. Product Categories & Filters: Users should be able to filter and search for shoes based on different attributes like price, size, color, and brand. Customers must be able to leave reviews and rate products. Product Availability: The system should update stock availability in real-time based on purchases.

3. Shopping Cart and Checkout

Users should be able to add or remove items from the cart before purchasing. The system should remember cart contents even after the user logs out. Before checkout, users must be able to review their selected items, prices, and quantities. The system must integrate a payment gateway (Strip) to facilitate transactions securely. Upon successful payment, users should receive an order confirmation via email.

4. Order Management- Users should be able to view their past purchases. Customers should receive tracking details for their shipments. The system must support return and refund requests, managed by the admin panel.

5. Admin Panel- Admins should be able to add, update, or remove products from the catalog. Admins should manage customer orders, update order statuses, and process returns. Admins should be able to monitor and manage user accounts. The system should generate reports on sales, revenue, and customer activity.

6. Customer Support - Customers should be able to submit queries via a contact form. The system should support real-time customer assistance. A dedicated FAQ page must be available to address common concerns.

3.2. Non-Functional Requirements

Non-functional requirements specify the quality attributes of the system, ensuring smooth operation and usability.

1. Performance Response Time: The system should load product listings in under 2 seconds. The architecture must support increased traffic and transactions.

2. Security Data Encryption: User data, including payment details, must be encrypted. Secure login mechanisms such as JWT authentication should be implemented.

3. Usability User-Friendly Interface: The website should be intuitive and responsive across devices. The platform must function seamlessly on all major web browsers.

4. Reliability and Availability Uptime Guarantee: The system should maintain a minimum uptime of 99.5%. The system should gracefully handle and log errors without affecting user experience.

5. Maintainability and Extensibility: The system must follow a modular design for easy updates and enhancements. The platform should allow easy integration with third-party services such as shipping providers and analytical tools.

4. SYSTEM DESIGN

4.1. System Architecture

The system architecture defines the key stages involved in the E-Commerce Shoe Website, from data input to processing and output. The architecture ensures a seamless user experience, secure transactions, and efficient management of the online store.

1. User Interaction (Frontend)

Users interact with the system through a responsive and dynamic React.js frontend. The frontend allows users to Browse and search for products, Add products to cart and proceed to checkout, Manage their accounts (Login/Register), Track orders and make payments

2. API Layer (Backend Communication)

The frontend communicates with the Node.js/Express.js backend through REST APIs. Key functionalities handled include User authentication (JWT-based login/register), Product retrieval, filtering, and management, Shopping cart operations, Secure payment handling (Stripe/Razorpay integration), Order placement and tracking

3. Database Management

The backend interacts with MongoDB using Mongoose ORM. It stores structured data including User information (hashed passwords using bcrypt.js), Product details (name, price, stock, images, etc.), Order history and transactions, Payment details (securely processed via Stripe/Razorpay)

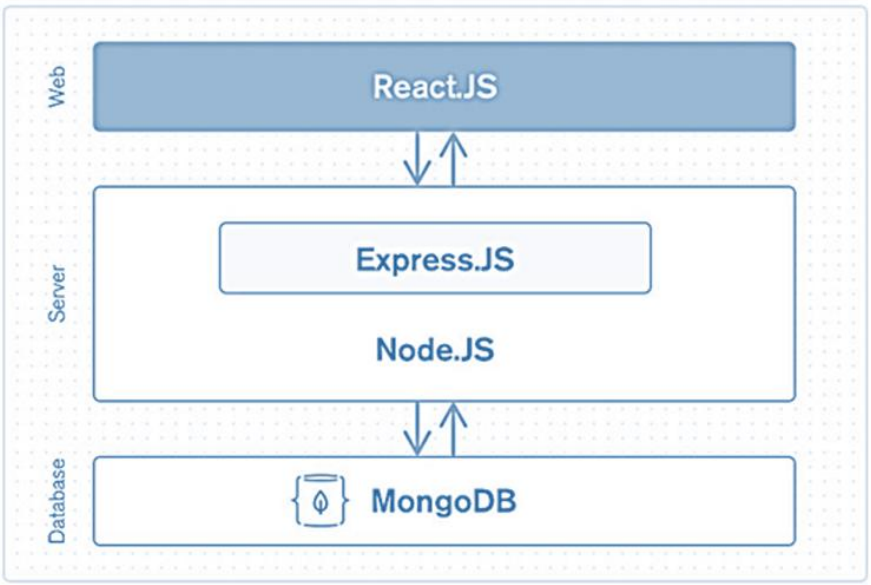

Figure 4.1.1 System Architecture

4.2. Context Diagram

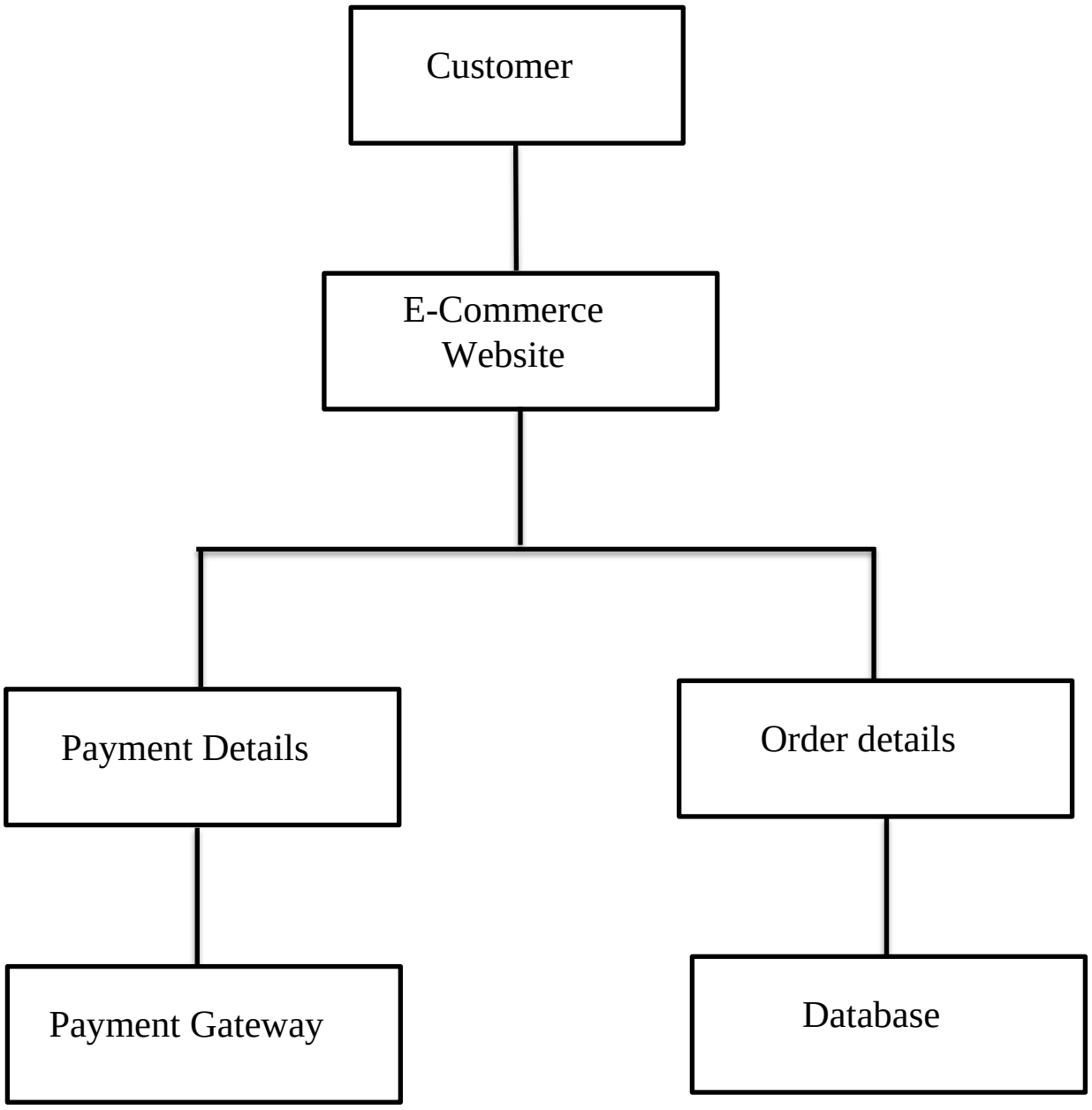

Figure 4.2.1 Context diagram

1. Purpose of the Context Diagram

The context diagram provides a high-level overview of the e-commerce system and its interactions with external entities.

2. System Boundary

The diagram delineates system components, including the frontend, backend, and database, while distinguishing them from external entities.

3. External Entities

- Users: Customers browsing, purchasing, and managing orders
- Admin: Manages products, orders, and user roles
- Payment Gateway (Stripe/Razorpay): Processes online transactions
- Database (MongoDB Atlas): Stores all application data

4. Data Flow

- Users interact with the frontend to search, filter, and purchase shoes.
- Requests are sent to the backend API for processing.
- The backend interacts with the database to retrieve and store data.
- The payment gateway handles secure transactions.
- The system provides real-time feedback to users regarding their orders and payments.

5. User Interactions

- Guest Users: Browse products, view details, and register/login.
- Registered Users: Add to cart, place orders, and manage profiles.
- Admin: Manages inventory, orders, and users.

6. System Outputs

- Order Confirmations: Users receive order details and tracking information.
- Payment Status Updates: Securely processed transactions and receipts.
- Product Recommendations: Personalized product suggestions based on user behavior.

7. Future Enhancements

- AI-based Recommendation System: Personalize user experience.
- Multi-Vendor Marketplace: Expand platform capabilities.
- Progressive Web App (PWA): Enable offline functionality and push notifications.

5. DETAILED DESIGN

5.1 Sequence Diagram

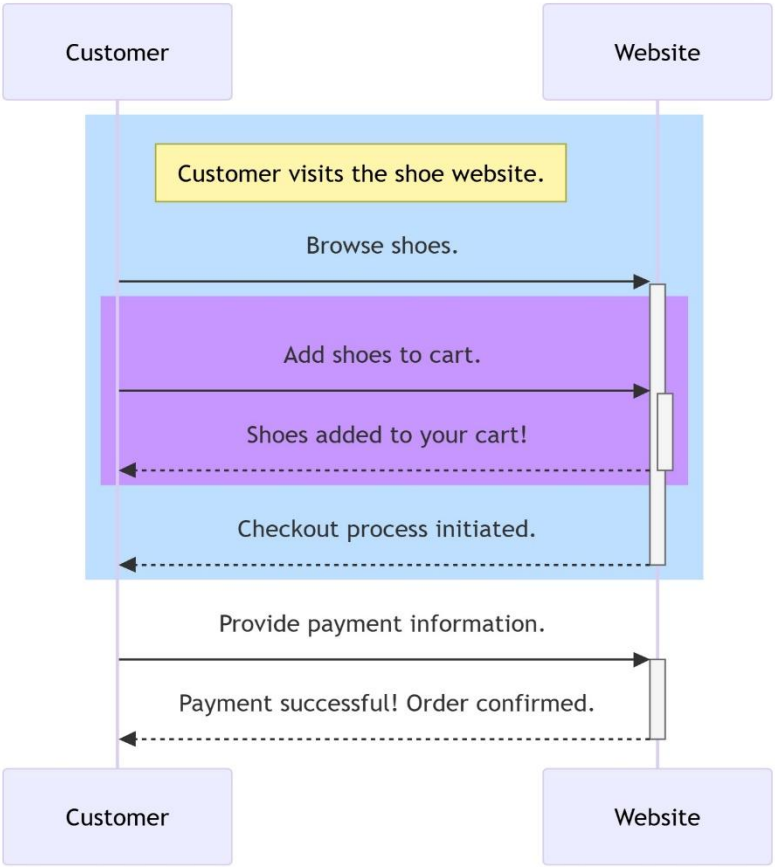

Figure 5.1.1 Sequence Diagram

1. **User Interaction Flow:** The sequence diagram outlines the step-by-step interaction between users (shoppers) and the e-commerce shoe website, showcasing how users browse, select, and purchase products.
2. **Initialization:** It begins with the user accessing the website through a responsive user interface, setting the context for further interactions.
3. **Product Selection:** Users browse the catalog, view shoe details, filter by brand, size, and price, and add selected products to their shopping cart.
4. **Cart Management:** Users can review their cart, modify quantities, or remove items before proceeding to checkout.
5. **Checkout Process:** Upon initiating checkout, the system validates user details, shipping address, and payment information.

6. **Order Confirmation:** After successful payment, the order is processed, and a confirmation email is sent to the user with order details.
7. **Inventory Update:** The system updates inventory levels to reflect purchased products.
8. **Delivery Tracking:** Users can track their order status, ensuring transparency in the shipping process.
9. **Feedback & Reviews:** After receiving the product, users can submit feedback and rate their purchased items.
10. **Error Handling:** The system handles errors such as payment failures, stock unavailability, and incorrect user details, ensuring smooth user experience.

5.2 Data flow diagram

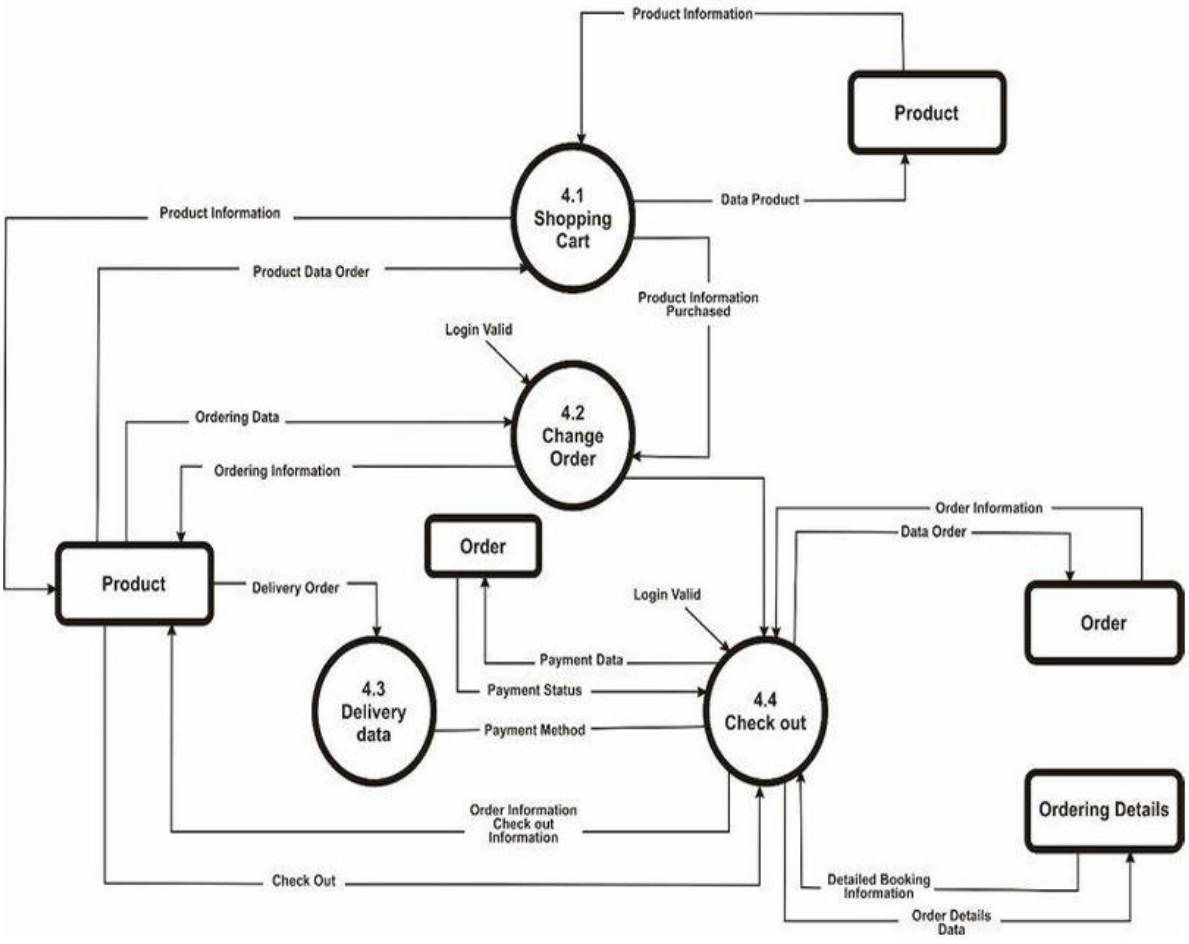

Figure 5.2.1 Data Flow Diagram

1. **Data Sources:** The DFD illustrates data sources like the product database, user database, and payment gateway, providing necessary information for transactions.
2. **User Interaction:** Users interact with the system to browse products, add to cart, and place orders, representing the core functionalities of the e-commerce platform.
3. **Cart & Order Processing:** The cart module processes user selections, calculates total costs, applies discounts (if any), and prepares for checkout.
4. **Payment Gateway Integration:** Secure payment processing is handled through third-party gateways, ensuring encrypted transactions.
5. **Inventory Management:** The system updates stock levels dynamically after each successful purchase.
6. **Order Fulfillment:** The warehouse or logistics team is notified for order dispatch, ensuring timely delivery.
7. **User Account Management:** The system manages user authentication, order history, and profile details, enhancing personalized shopping experiences.
8. **Feedback & Reviews:** Users can submit product ratings and reviews, which are stored in the database to improve recommendations.

5.3 Activity Diagram

1. **User Login/Guest Checkout:** The process begins when a user logs in or proceeds as a guest to browse available shoes.
2. **Product Search & Selection:** Users explore categories, use filters, and select shoes to add to their cart.
3. **Cart Management:** Users review their selections, modify quantities, or remove items before proceeding.
4. **Checkout & Payment:** The system collects shipping details and processes secure payments.
5. **Order Confirmation:** Once the payment is validated, the order is confirmed, and users receive a confirmation email.
6. **Inventory Update:** The stock count is updated to reflect purchases, preventing overselling.
7. **Order Fulfillment & Delivery:** The system coordinates with logistics to process and deliver the order.
8. **Order Tracking:** Users can track their shipments in real-time, ensuring a seamless shopping experience.

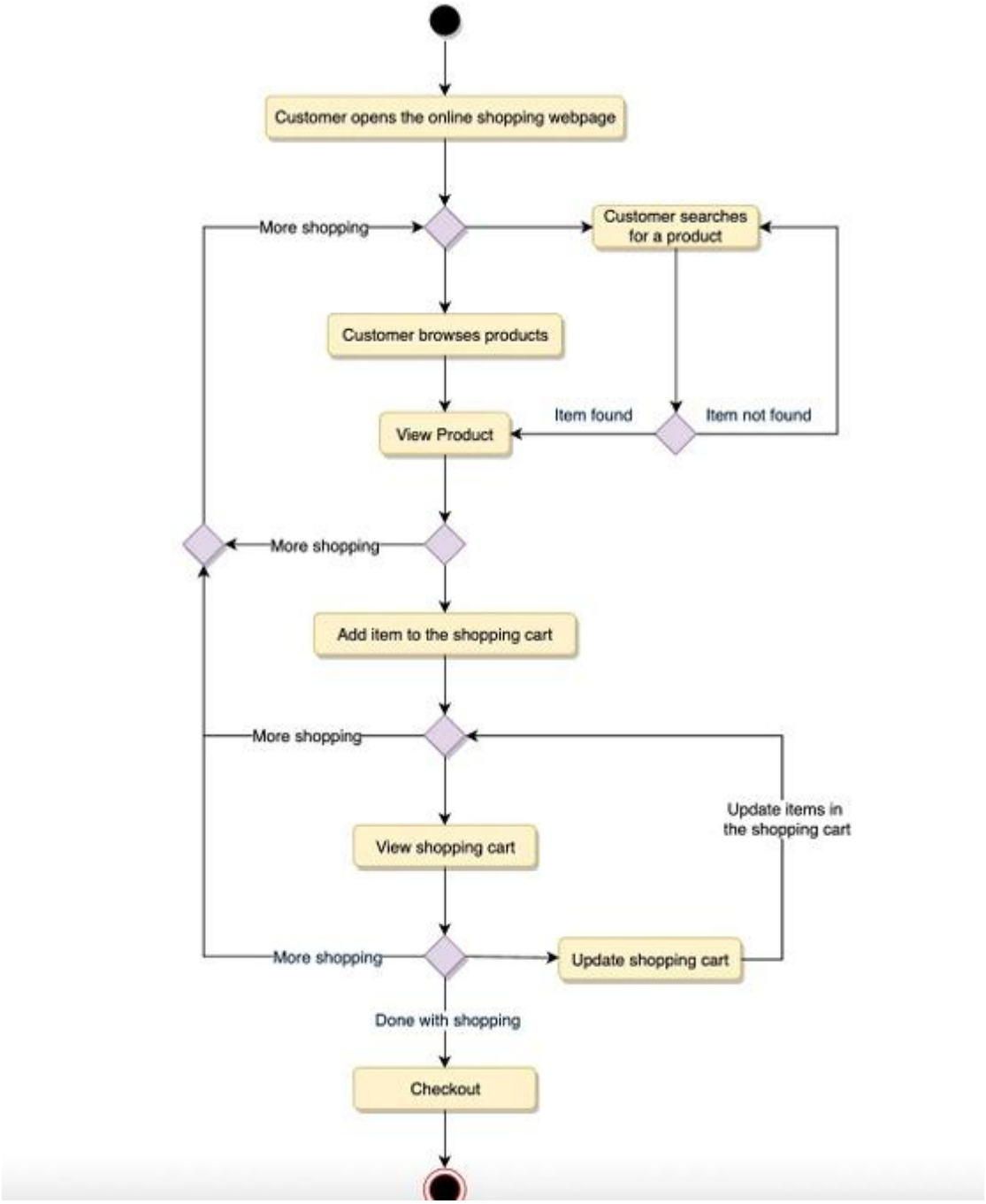

Figure 5.3.1 Activity Diagram

5.4 Use Case diagram

Figure 5.4.1 Use Case Diagram

1. Actors: The primary actors in the system are Shoppers, Admins, and Logistics Teams, who interact with the system to facilitate smooth e-commerce operations.
2. Browse Products: Users can search for shoes using filters like brand, size, color, and price.
3. Add to Cart: Shoppers can add selected products to their cart and manage the items before proceeding to checkout.
4. Complete Purchase: Users can enter payment and shipping details to complete their order securely.
5. Track Orders: After purchasing, users can track the shipment status until delivery.
6. Provide Reviews: Users can leave reviews and ratings for products they have purchased, helping future shoppers make informed decisions.
7. Manage Inventory: Admins update stock levels, add new product listings, and remove unavailable items to ensure accurate product availability.
8. Process Orders: The logistics team processes and dispatches orders efficiently, ensuring timely deliveries.

6. IMPLEMENTATION

6.1. Procedure

1. System Setup: Install and configure the necessary development tools, frameworks, and database systems required for implementation.
2. Backend Development: Develop server-side logic to handle user authentication, product management, order processing, and payment integration.
3. Frontend Development: Create a responsive user interface that enables smooth navigation and interaction with the e-commerce platform.
4. Database Integration: Set up and optimize the database to store product details, user accounts, transaction history, and reviews.
5. API Development: Develop and integrate RESTful APIs to facilitate communication between the frontend and backend systems.
6. Payment Gateway Integration: Implement secure payment processing to handle transactions via credit cards, digital wallets, and other payment methods.
7. Testing and Debugging: Perform unit testing, integration testing, and user acceptance testing to identify and fix any bugs or inconsistencies.
8. Deployment: Deploy the system on a cloud platform or hosting service, ensuring scalability and security measures are in place.
9. Performance Optimization: Monitor system performance, optimize database queries, and enhance user experience based on analytics data.
10. Maintenance and Updates: Continuously update and improve the platform based on user feedback, new technologies, and market trends.

6.2. Snippet Code

Index .js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';
import { RouterProvider } from 'react-router-dom';
import router from './routes';
import { Provider  } from 'react-redux';
import { store } from './store/store';
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  // <React.StrictMode>
    <Provider store={store}>
      <RouterProvider router={router}/>
    </Provider>
   </React.StrictMode>
);
reportWebVitals();
```

Home .js

```
import React from 'react'
import CategoryList from '../components/CategoryList'
import BannerProduct from '../components/BannerProduct'
import HorizontalCardProduct from '../components/HorizontalCardProduct'
import VerticalCardProduct from '../components/VerticalCardProduct'

const Home = () => {
  return (
    <div>
      <CategoryList/>
      <BannerProduct/>

      <HorizontalCardProduct category={"ankle length shoes"} heading={"Top's ankle length shoes"}/>
      <HorizontalCardProduct category={"casual shoes"} heading={"Popular's casual shoes"}/>

      <VerticalCardProduct category={"Formal shoes"} heading={"Formal shoes"}/>
      <VerticalCardProduct category={"men's boot"} heading={"men's boot"}/>
      <VerticalCardProduct category={"Running shoes"} heading={"Running shoes"}/>
      <VerticalCardProduct category={"ballet flats"} heading={"ballet flats"}/>

      <VerticalCardProduct category={"boots"} heading={"boots"}/>
```

```
      <VerticalCardProduct category={"heels"} heading={"heels"}/>
      {/* <VerticalCardProduct category={"refrigerator"} heading={"Refrigerator"}/>
      <VerticalCardProduct category={"trimmers"} heading={"Trimmers"}/> */}
    </div>
  )
}

export default Home
```

Login page

```
import React, { useContext, useState } from 'react'
import loginIcons from '../assest/signin.gif'
import { FaEye } from "react-icons/fa";
import { FaEyeSlash } from "react-icons/fa";
import { Link, useNavigate } from 'react-router-dom';
import SummaryApi from '../common';
import { toast } from 'react-toastify';
import Context from '../context';

const Login = () => {
    const [showPassword,setShowPassword] = useState(false)
    const [data,setData] = useState({
        email : "",
        password : ""
    })
    const navigate = useNavigate()
    const { fetchUserDetails, fetchUserAddToCart } = useContext(Context)

    const handleOnChange = (e) =>{
        const { name , value } = e.target

        setData((preve)=>{
            return{
                ...preve,
                [name] : value
            }
        })
    }


    console.log("data login",data)

 \<Link to={'/forgot-password'} className='block w-fit ml-auto hover:underline hover:text-red-600'>
                        Forgot password ?
                    </Link>
```

```
                </div>

                <button className='bg-red-600 hover:bg-red-700 text-white px-6 py-2 w-full max-w-[150px] rounded-full hover:scale-110 transition-all mx-auto block mt-6'>Login</button>

            </form>

            <p className='my-5'>Don't have account ? <Link to={"/sign-up"} className=' text-red-600 hover:text-red-700 hover:underline'>Sign up</Link></p>
        </div>

        </div>
   </section>
  )
}

export default Login
```

Db.js

```
const mongoose = require("mongoose")

async function connectDB(){
   try{
      await mongoose.connect(process.env.MONGODB_URI)
   }catch(err){
      console.log(err)
   }
}

module.exports = connectDB
```

6.3 Screenshot

Home page

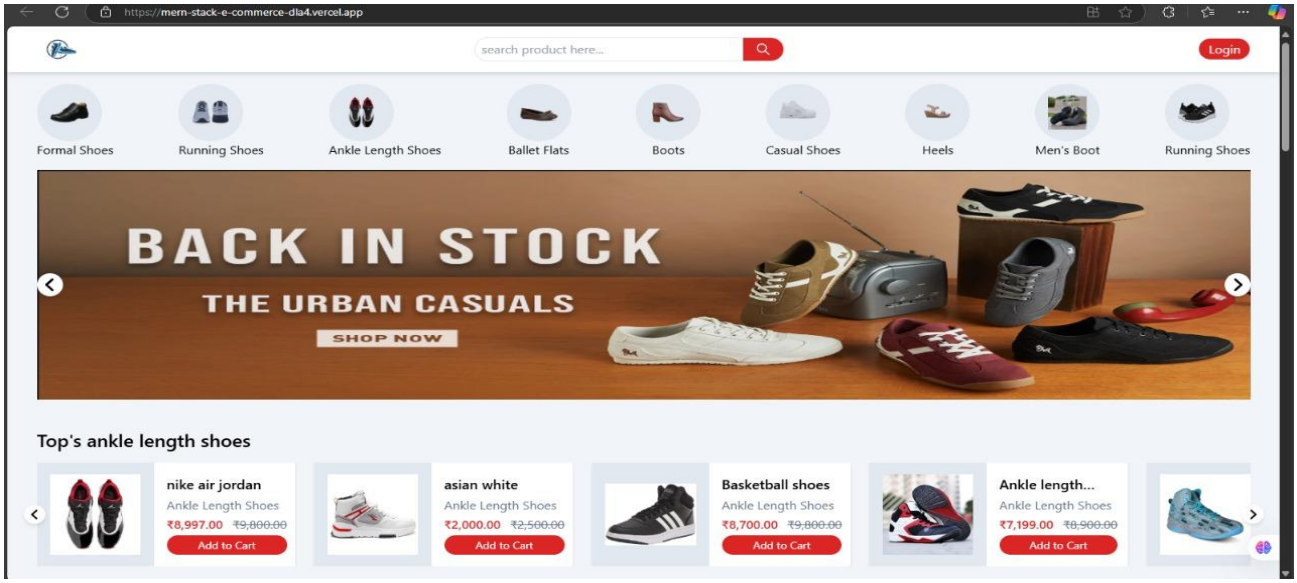

Figure 6.3.1 Home page

Home Page: Showcases featured products and highlights key offerings**.**

Login page

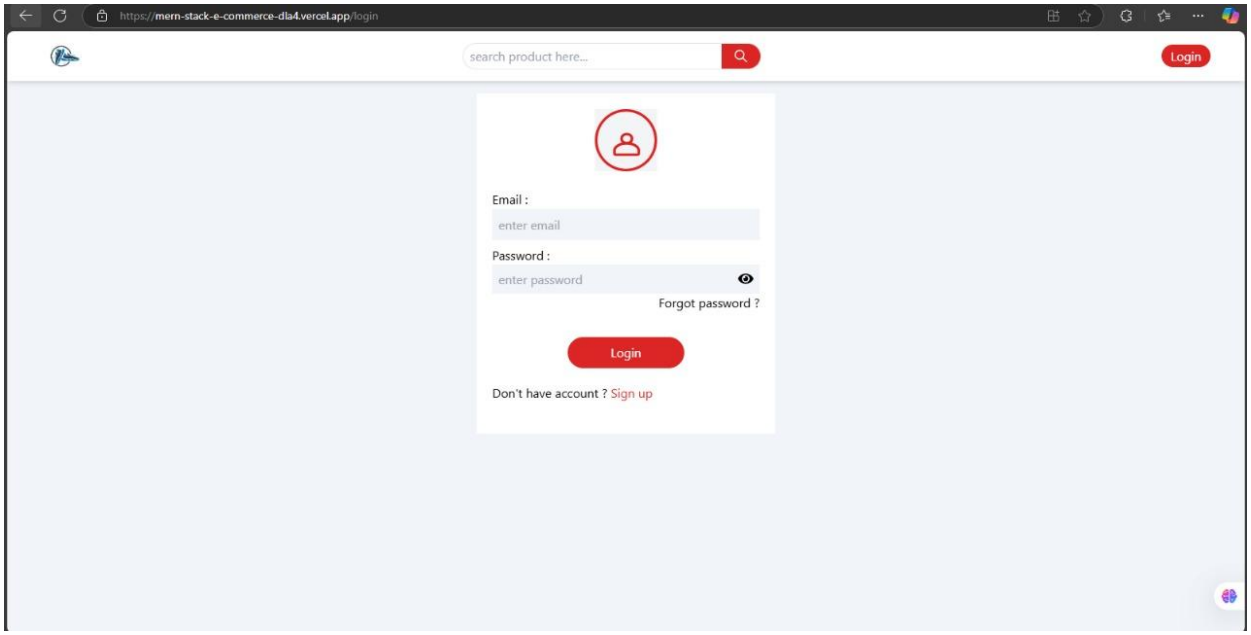

Figure 6.3.2 Login Page

Login Page: Allows users to securely sign in to their accounts.

Sign Up

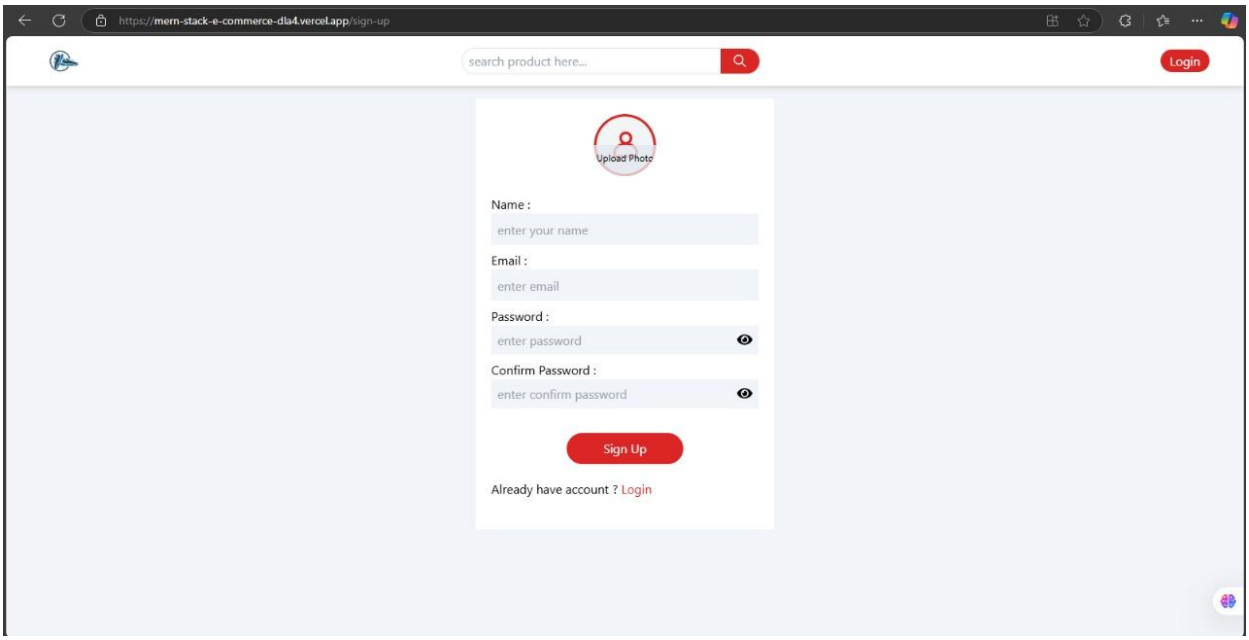

Figure 6.3.3 Sign Up

Signup Page: Enables new users to register and create an account.

Product page

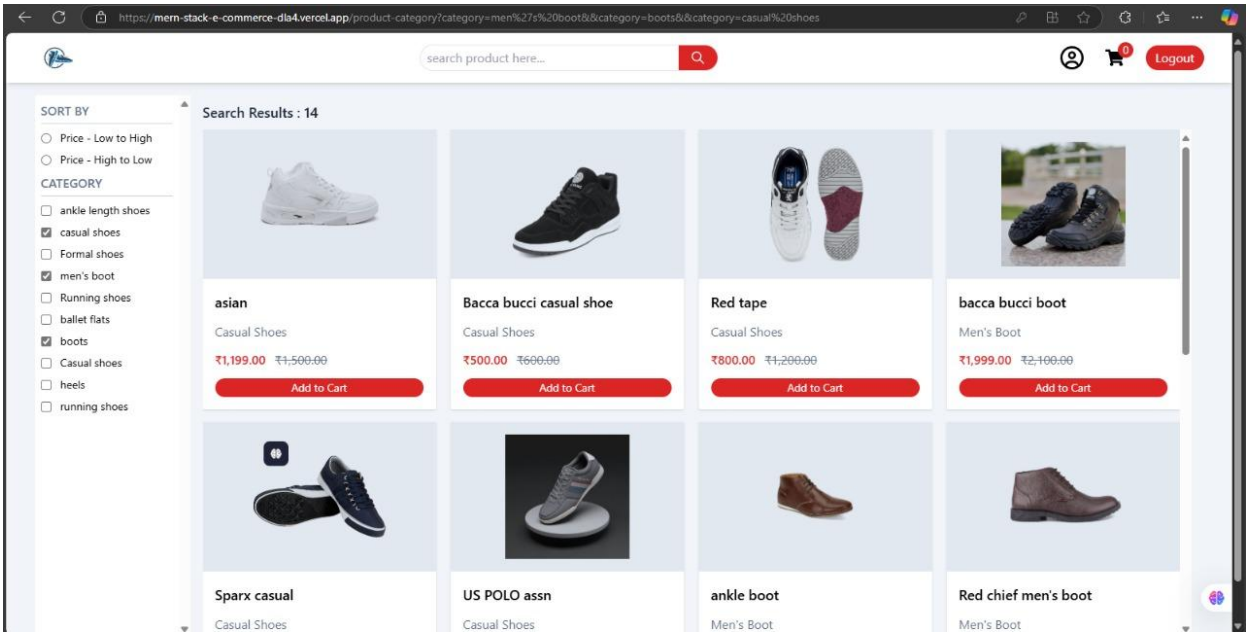

Figure 6.3.4 Product Page

Product Page: Displays detailed information about a specific product.

Products

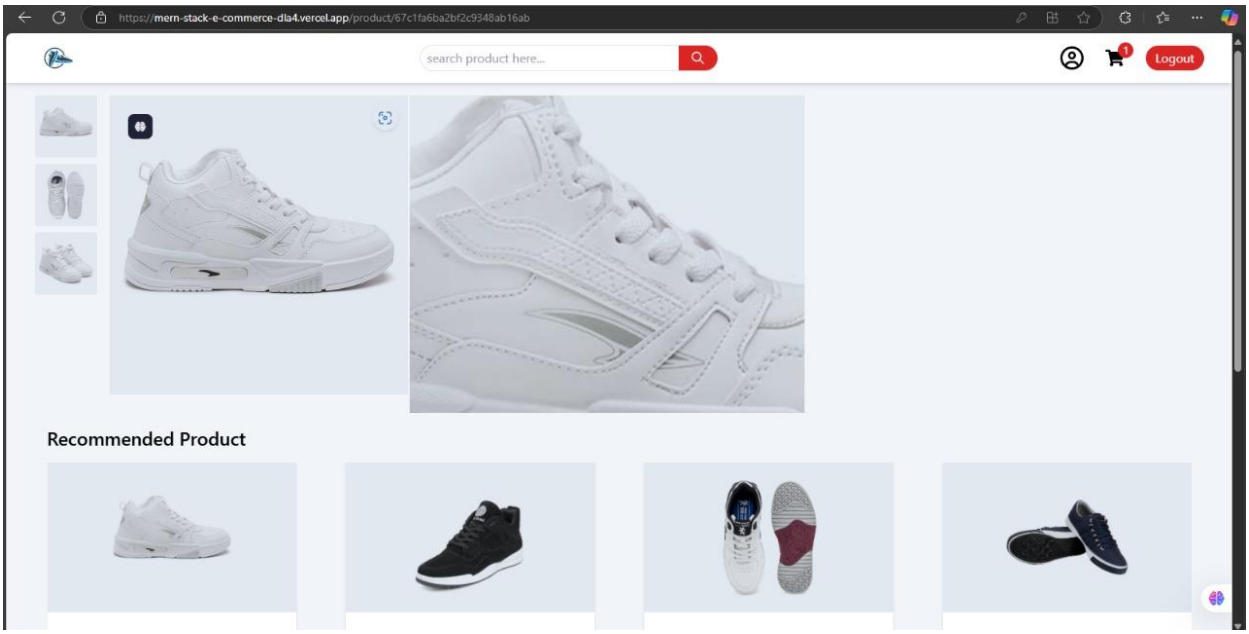

Figure 6.3.5 Product

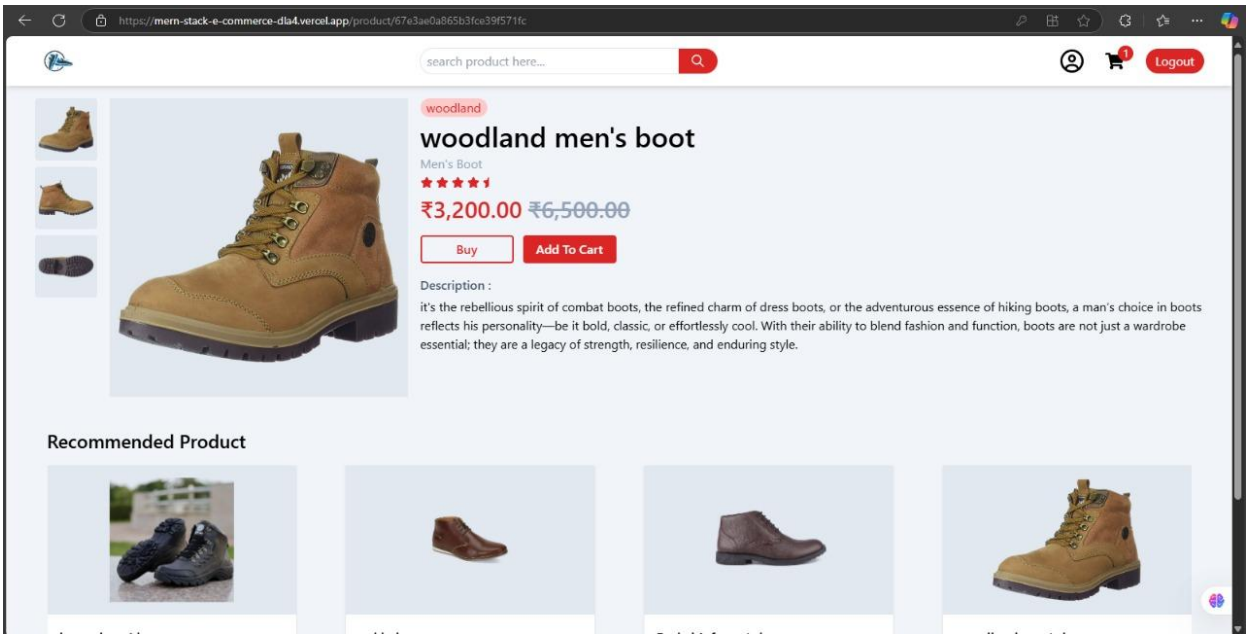

Figure 6.3.6 Product

Product Page: Displays detailed information about a specific product.

Cart page

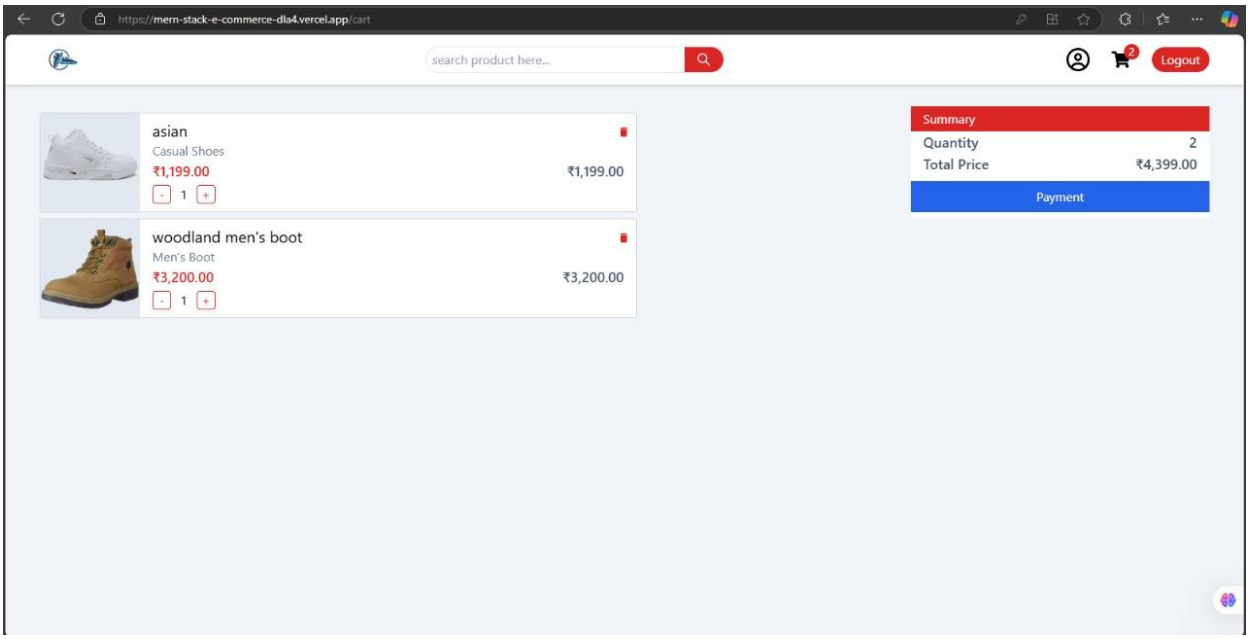

Figure 6.3.7 Cart page

Cart Page: Shows selected items and allows users to manage their cart.

Payment page

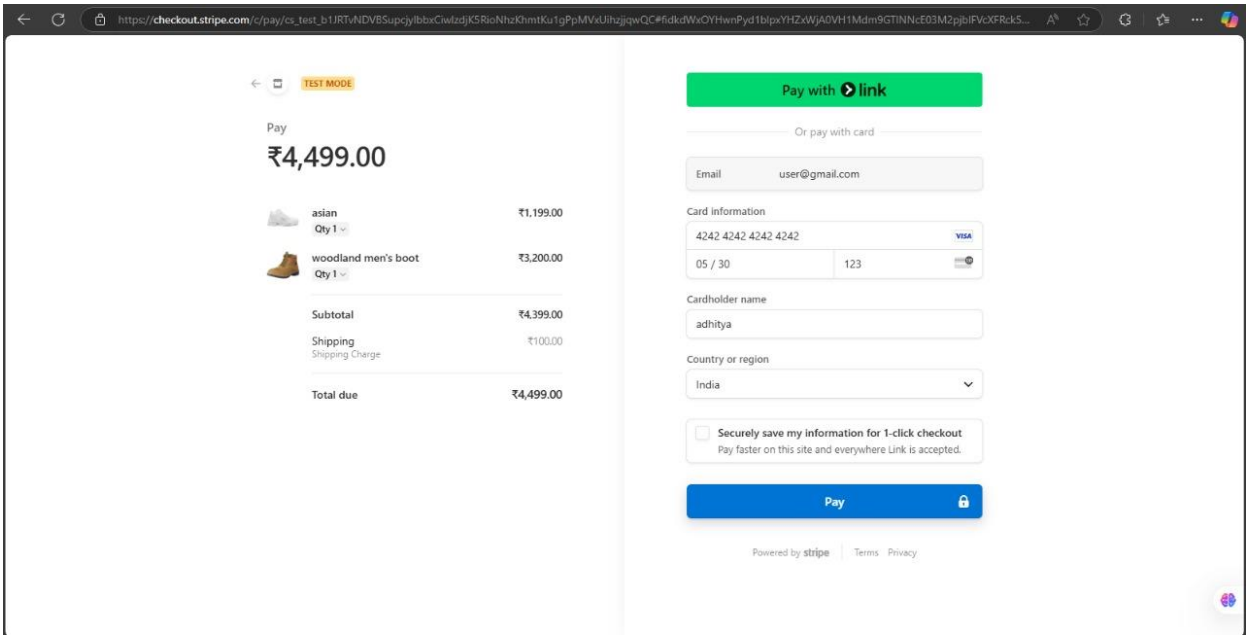

Figure 6.3.8 Payment Page

Payment Page: Facilitates secure payment processing for orders.

Orders page

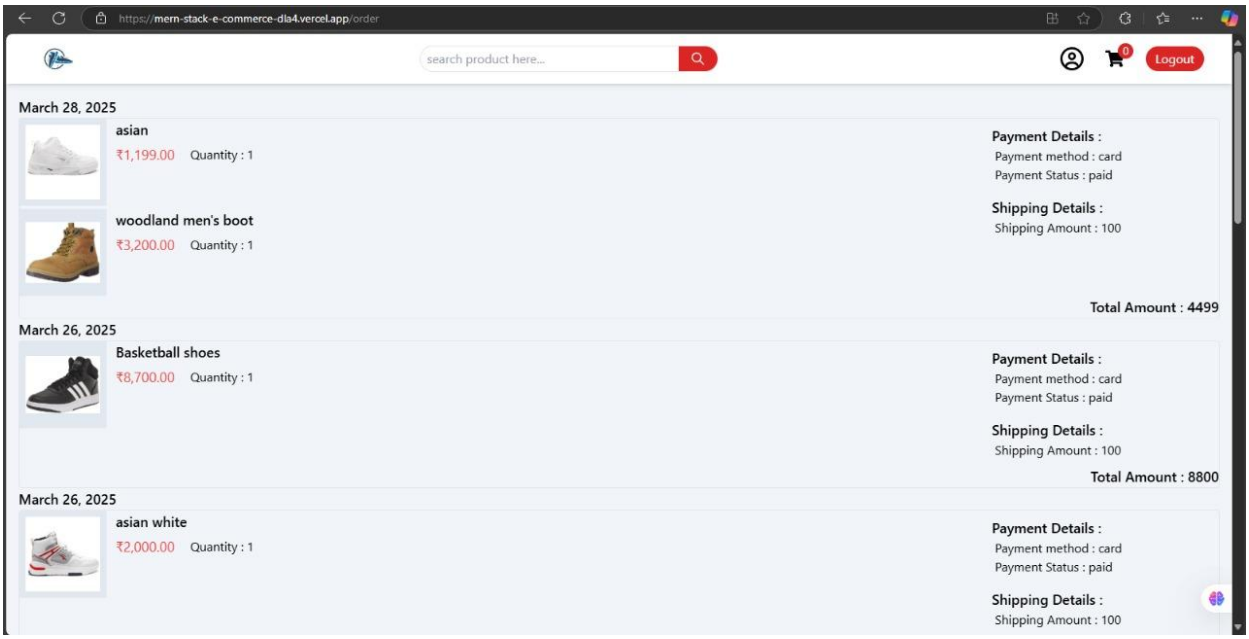

Figure 6.3.9 Order Page

Orders Page: Lists past and current orders with status updates**.**

Admin page

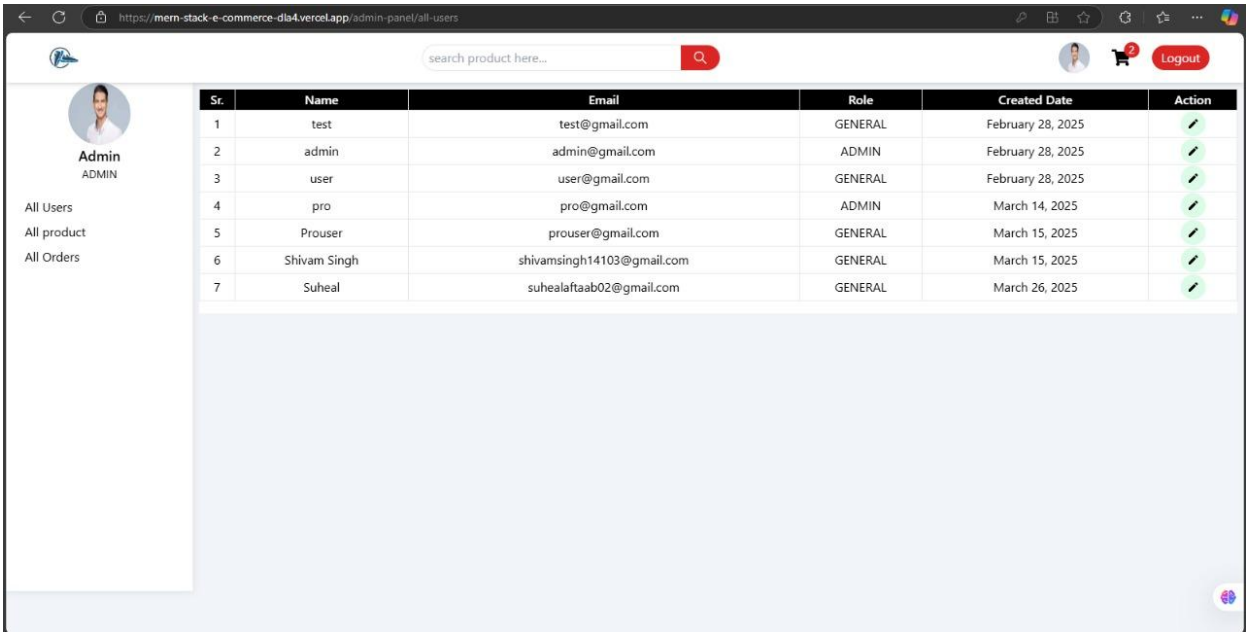

Figure 6.3.10 Admin Page

Admin Page: Provides management controls for products, users, and orders**.**

Upload Product

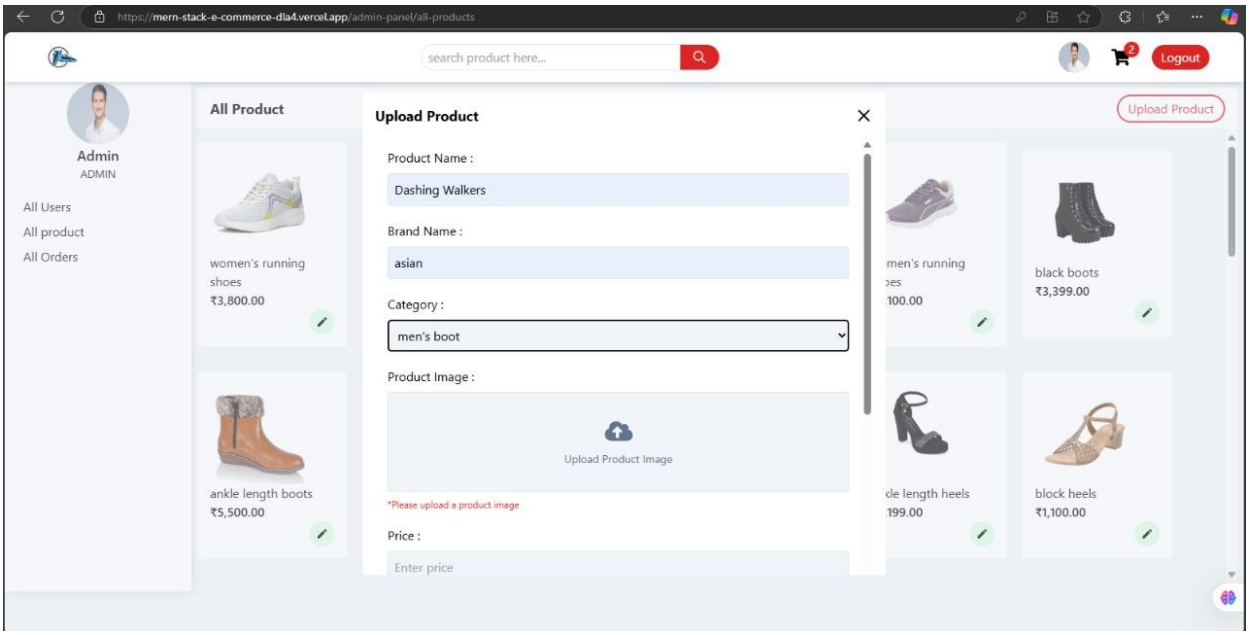

Figure 6.3.11 Upload Product

Upload Page: Enables admins to add or update products in the store.

All Orders page

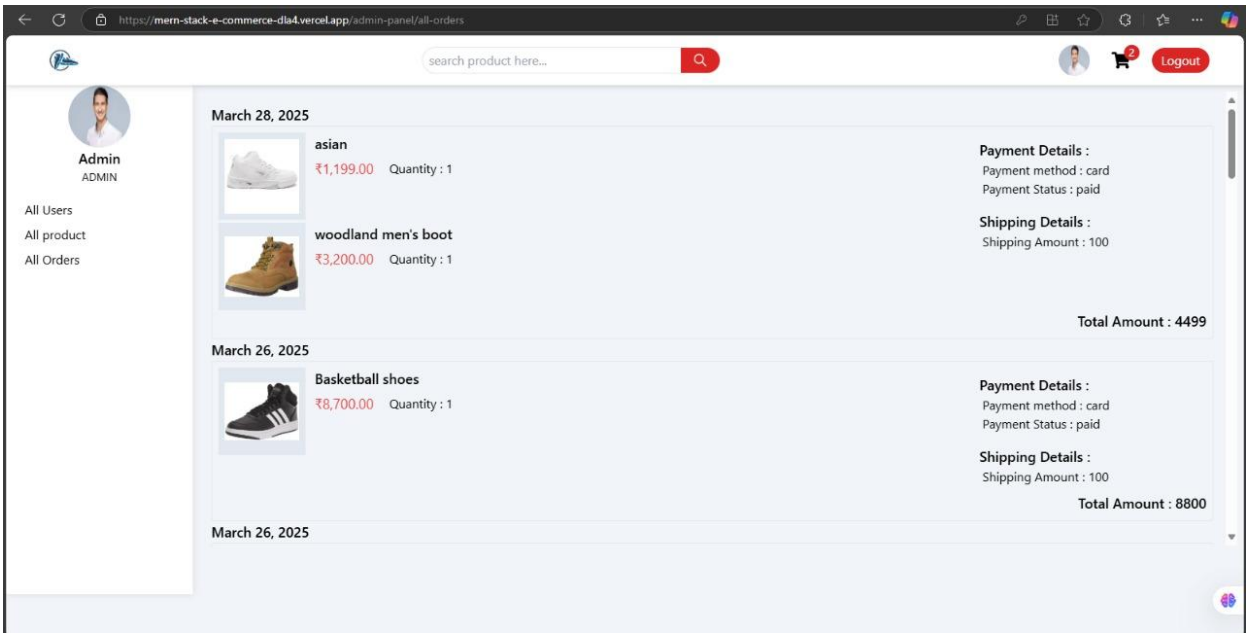

Figure 6.3.12 All Orders Page

All Products Page: Displays the complete catalog of available products.

7. SOFTWARE TESTING

7.1. System Testing

System testing ensures that all components of the **E-Commerce Shoe Website** function correctly, both individually and as a whole. It involves testing the **frontend, backend, database, and API interactions** to validate functional and non-functional requirements.

7.1.1 Functional Testing

- Ensures that each feature, such as authentication, product search, cart management, and payment, works as expected.

7.1.2 Unit Testing

- Tests individual components such as the login system, API endpoints, and UI elements using tools like Jest and Mocha.

7.1.3 Integration Testing

- Verifies interactions between modules, such as user authentication with JWT, database CRUD operations, and payment processing with Stripe/Razorpay.

7.1.4 Security Testing

- Ensures protection against common vulnerabilities such as **SQL Injection, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF)**.

7.2 Test Cases

7.2.1 User Authentication

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-001	User Registration	Enter valid details and submit	User account is created successfully	passed
TC-002	User Registration with Invalid Data	Enter invalid email or weak password	Error message displayed	passed
TC-003	User Login	Enter valid credentials and submit	User logs in successfully	passed
TC-004	Incorrect Login	Enter incorrect credentials	Error message displayed	passed
TC-005	Logout Functionality	Click logout	User is logged out and redirected to the homepage	passed

Table 7.2.1

7.2.2 Product Search and Filtering

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-006	Search for a Product	Enter product name in the search bar	Display relevant results	passed
TC-007	Apply Filters	Select filters (size, price, brand, color)	Display filtered products	passed
TC-008	Search for a Non-existent Product	Enter random text	Show "No results found" message	passed

Table 7.2.2

7.2.3 Cart Management

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-009	Add Item to Cart	Click "Add to Cart" button on a product	Product appears in cart	passed
TC-010	Remove Item from Cart	Click "Remove" button in cart	Product is removed from cart	passed
TC-011	Update Quantity in Cart	Change quantity using dropdown	Cart updates quantity correctly	passed

Table 7.2.3

7.2.4 Checkout and Payment Processing

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-012	Proceed to Checkout	Click checkout with products in cart	Redirect to payment page	passed
TC-013	Complete Payment	Enter valid payment details and submit	Payment is processed, order confirmation displayed	passed
TC-014	Payment Failure	Enter invalid card details	Payment fails, error message displayed	passed

Table 7.2.4

7.2.5 Order Tracking

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-015	View Order Status	Navigate to order history	Display current status (Processing, Shipped, Delivered)	passed
TC-016	Cancel Order	Click "Cancel Order" for eligible orders	Order is canceled successfully	passed

Table 7.2.5

7.2.6 Admin Panel Testing

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-017	Add a New Product	Fill out product form and submit	Product appears in the product list	passed
TC-018	Edit a Product	Modify product details and save	Updated details reflected in UI	passed
TC-019	Delete a Product	Click "Delete" on a product	Product is removed from database	passed

Table 7.2.6

7.2.7 Security Testing

Test Case ID	Test Scenario	Test Steps	Expected Output	Status
TC-020	SQL Injection	Attempt to enter SQL code in login fields	Input is sanitized, SQL attack prevented	passed
TC-021	Cross-Site Scripting (XSS)	Enter <script>alert("XSS")</script> in input fields	Script is not executed, data is escaped	passed
TC-022	Unauthorized Admin Access	Try accessing admin panel without login	Access is denied, redirect to login page	passed

Table 7.2.7

8. CONCLUSION

The "Full Stack E-Commerce Shoe Website" project successfully demonstrates the development of a robust, scalable, and user-friendly online shopping platform using the MERN stack. By incorporating modern web technologies, secure authentication, efficient product management, and seamless payment integration, this project provides a comprehensive solution for both customers and administrators.

Through a structured approach, including system design, feature implementation, testing, and deployment, the project ensures a high-quality user experience. The admin panel streamlines product management, while features like search, filtering, and order tracking enhance usability for customers. The integration of Stripe ensures a secure and reliable payment gateway, improving transaction safety.

Extensive software testing, covering functional, security, performance, and integration aspects, validates the system's reliability and efficiency. The implementation of best coding practices and scalable architecture ensures long-term maintainability and adaptability for future enhancements.

9. FUTURE ENHANCEMENTS

To further improve the **Full Stack E-Commerce Shoe Website**, several enhancements can be implemented to enhance user experience, increase engagement, and optimize business operations.

9.1 AI-Driven Product Recommendations

- Implement **machine learning algorithms** to analyze user behavior, past purchases, and browsing history.
- Provide **personalized recommendations** to users, increasing conversion rates and customer satisfaction.
- Utilize **collaborative filtering and content-based filtering** techniques to suggest relevant products.

9.2 Customer Reviews & Ratings

- Enable users to **rate and review** purchased products, helping other customers make informed decisions.
- Implement a **verified purchase badge** to ensure authenticity and credibility of reviews.
- Use **sentiment analysis** to categorize and display the most helpful reviews.

9.3 Multiple Payment Options

- Add support for **UPI, digital wallets (Google Pay, PayPal, Apple Pay), and EMI options** for greater flexibility.
- Implement **Buy Now, Pay Later (BNPL)** features to enhance affordability for customers.

9.4 Advanced Order Tracking

- Integrate with **third-party logistics APIs** to provide real-time order tracking.
- Send **automated SMS/email notifications** to update users on order status changes.

10. BIBLIOGRAPHY

React.js Documentation – Official React documentation for building interactive user interfaces. (react.dev)

Redux Toolkit – State management library for efficient data handling in React applications. (redux-toolkit.js.org)

Node.js & Express.js Documentation – Backend framework for handling API requests and server-side logic. (nodejs.org, expressjs.com)

MongoDB Documentation – NoSQL database for storing product, user, and order data. (mongodb.com)

JWT Authentication Guide – Secure authentication and authorization using JSON Web Tokens. (jwt.io)

Tailwind CSS Documentation – Utility-first CSS framework for modern UI design. (tailwindcss.com)

Cloudinary API Documentation – Image management and optimization for eCommerce platforms. (cloudinary.com)

PayPal Developer Guide – Secure online payment integration for seamless transactions. (developer.paypal.com)

bcrypt.js Documentation – Password hashing for secure authentication. (npmjs.com/package/bcryptjs)

MERN Stack Best Practices – Comprehensive guide on building scalable applications using MongoDB, Express, React, and Node. (dev.to, medium.com)