**Mini Project-1 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

**SHRUNGADITHYA A N, 1BY23MC083**

**SHIVAM SINGH, 1BY23MC081**

**Under the Guidance of**

**Prof Nirupama B K**

**Assistant Professor - II**

**Department of MCA**

**BMSIT&M**

**Bengaluru-560064**

****

**Department of Master of Computer Applications**

**BMS Institute of Technology and Management**

**Bengaluru**

**October-2024**

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru**

**October-2024**

**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 **Shrungadithya A N**, **Shivam Singh** bearing university registration number **1BY23MCA083, 1BY23MC081** during the period **July 2024 to October 2024** 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 Signature of HoD

**Prof Nirupama B K Dr. M.Sridevi**

Assistant Professor - II Assistant Professor & Head

Department of MCA Department of MCA

BMSIT&M BMSIT&M

Bengaluru. Bengaluru

# DECLARATION

I **Shrungadithya A N** student of MCA, BMS Institute of Technology and Management, bearing USN **1BY23MC083** hereby declare that Mini project 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. 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 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. Venkatesh A** 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.

Shrungadithya A N

1BY23MC083

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru**

**Department of MCA**

****

**VISION**

To develop quality professionals in Computer Applications who can provide sustainable solutions to the societal and industrial needs.

**MISSION**

Facilitate effective learning environment through quality education, state-of-the-art facilities, and orientation towards research and entrepreneurial skills.

**Programme Educational Objectives (PEOs)**

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

**PEO 2:** Adapt themselves to changing IT requirements through life-long learning.

**PEO 3:** Exhibit leadership skills and advance in their chosen career.

**BMS INSTITUTE OF TECHNOLOGY AND MANAGEMENT**

**(An Autonomous Institution, Affiliated to VTU, Belagavi)**

**Bengaluru**

**Department of MCA**

**Programme Outcomes (POs)**

**PO 1:** Apply knowledge of computing fundamentals, computing specialization, mathematics and domain knowledge to provide IT solutions.

**PO 2:** Identify, analyse and solve IT problems using fundamental principles of mathematics and computing sciences.

**PO 3:** Design, Develop and evaluate software solutions to meet societal and environmental concerns.

**PO 4:** Conduct investigations of complex problems using research-based knowledge and methods to provide valid conclusions.

**PO 5:** Select and apply appropriate techniques and modern tools for complex computing activities.

**PO 6:** Understand professional ethics, cyber regulations and responsibilities.

**PO 7:** Involve in life-long learning for continual development as an IT professional.

**PO 8:** Apply and demonstrate computing and management principles to manage projects in multidisciplinary environments by involving in different roles.

**PO 9:** Comprehend & write effective reports and make quality presentations.

**PO 10:** Understand the impact of IT solutions on socio-environmental issues.

**PO 11:** Work collaboratively as a member or leader in multidisciplinary teams.

**PO 12:** Identify potential business opportunities and innovate to create value for the society and seize that opportunity.

**BMS INSTITUTE OF TECHNOLOGY AND 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 full-stack web application developed using the **MERN (MongoDB, Express.js, React.js, Node.js) stack** to provide a seamless and interactive online shopping experience. Users can browse a diverse collection of footwear, search and filter products based on various attributes, and securely complete their purchases through an integrated **payment gateway (Stripe)**. The platform ensures smooth navigation with a **mobile-responsive design**, making it accessible across different devices while maintaining an intuitive and user-friendly interface.

The website incorporates essential e-commerce functionalities such as **user authentication and authorization**, an **admin dashboard for efficient product and order management**, and a **cart system** that allows users to add items, place orders, and track their purchases in real time. The backend is powered by a **RESTful API**, managing product data, orders, and user authentication through **JWT and bcrypt.js** for secure access control. The **MongoDB database**, coupled with Mongoose, ensures efficient data storage and retrieval, providing a scalable and reliable 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**

Page No.

1. **INTRODUCTION 01**
   1. Project description 01
   2. Technologies used 01
   3. Project Objectives 02
2. **LITERATURE SURVEY 03**
   1. Related Work 03
   2. Problem Definition 04
   3. Existing and Proposed System 05
   4. System study 09
      1. Feasibility 09
      2. Technical Feasibility 09
      3. Operational Feasibility 09
      4. Economic Feasibility 10
   5. System requirements 10
      1. Hardware Requirements 10
      2. Software Requirements 10
3. **SOFTWARE REQUIREMENTS SPECIFICATION 11**
   1. Functional requirements 11
   2. Non-Functional requirements 12
4. **SYSTEM DESIGN 15**
   1. System Architecture 15
   2. Context Diagram 18
5. **DETAILED DESIGN 20**
   1. Sequence diagram 20
   2. Data flow diagram 21
   3. Activity diagram 23
   4. Use Case diagram 24
6. **IMPLEMENTATION 26**
   1. Procedure 26
   2. Snippet code 26
   3. Screenshots (with Explanation) 29
7. **SOFTWARE TESTING 30**
   1. System Testing 30
   2. Test Cases 30
8. **CONCLUSION 33**
9. **FUTURE ENHANCEMENTS 34**
10. **REFERENCES 35**
11. **PLAGIARISM REPORT 36**

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| SLNO | PARTICULARS | PAGE NO |
| 1 | Boundary Value Analysis Test Cases | 30 |
| 2 | Weak Normal Equivalence Testing | 31 |
| 3 | Weak Robust Test Cases | 31 |
| 4 | Strong Robust Test Cases | 32 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| SLNO | PARTICULARS | PAGE NO |
| 1 | System Architecture | 17 |
| 2 | Context Diagram | 18 |
| 3 | Sequence Diagram | 20 |
| 4 | Data Flow Diagram | 21 |
| 5 | Activity Diagram | 23 |
| 6 | Use Case Diagram | 24 |
| 7 | Plagiarism | 36 |

### ****INTRODUCTION****

#### 1.1. Project Description

The rapid expansion of e-commerce has transformed the shopping experience by providing convenience, accessibility, and a wide range of product choices. This project, titled **"Full-Stack E-Commerce Shoe Website,"** is designed to offer an intuitive and efficient online platform dedicated to footwear shopping.

Developed using the **MERN (MongoDB, Express.js, React.js, Node.js) stack**, the website will incorporate essential e-commerce functionalities, including product discovery, filtering, authentication, order management, and secure payment integration.

To ensure a structured and scalable system, the platform will follow a **modular architecture**, where the **frontend (React.js)** communicates with the **backend (Node.js & Express.js) via REST APIs**, while data is stored securely in **MongoDB**. User authentication will be handled through **JWT (JSON Web Token)** and **Bcrypt.js** for encrypted password storage, ensuring data security. The payment system will leverage **Stripe or Razorpay**, facilitating seamless online transactions.

The platform will also include an **admin dashboard**, allowing store managers to efficiently manage inventory, orders, and user accounts. A **responsive user interface**, built with **Material-UI or Tailwind CSS**, will ensure smooth navigation and an optimized experience across different devices.

This project aims to provide a **fast, secure, and user-friendly e-commerce solution** tailored to the footwear industry. Potential future enhancements include **AI-driven product recommendations, customer feedback integration, and a rewards program**, further improving user engagement and satisfaction.

#### 1.2 Technologies Used

To develop a **scalable, secure, and efficient** e-commerce platform, this project leverages the **MERN (MongoDB, Express.js, React.js, Node.js) stack** along with additional tools for seamless functionality and deployment. The key technologies include:

**Frontend (Client-Side)**

* **React.js** – A JavaScript library for building interactive and dynamic user interfaces.
* **Redux** – A state management library to handle data flow efficiently.
* **Material-UI/Tailwind CSS** – UI frameworks for designing a responsive and visually appealing interface.
* **Axios** – Used for making HTTP requests to communicate with the backend.

**Backend (Server-Side)**

* **Node.js** – A runtime environment for executing JavaScript on the server.
* **Express.js** – A lightweight framework for building RESTful APIs.
* **Mongoose** – An ODM (Object Data Modeling) library for MongoDB interactions.
* **JWT (JSON Web Token)** – Ensures secure user authentication and session management.
* **Bcrypt.js** – Encrypts user passwords for enhanced security.

**Database**

* **MongoDB** – A NoSQL database used for storing user accounts, product listings, orders, and transactions.
* **MongoDB Atlas** – A cloud-based database service that ensures high availability and scalability.

**Payment Integration**

* **Stripe** – A secure payment gateway used for processing transactions efficiently.

**Version Control**

* **Git & GitHub** – Used for version control, collaboration, and managing code repositories.

**Deployment & DevOps**

* **Vercel** – Used for deploying both the **frontend and backend**, ensuring seamless and scalable hosting.

#### Project Objectives

The primary objective of this project is to develop a **full-stack e-commerce platform** specifically designed for **shoe retail**, offering a seamless and user-friendly online shopping experience. This platform will enable users to browse products, place orders securely, and make hassle-free transactions. It will also include an admin dashboard for effective store management.

The key objectives of the project include:

* **Developing an intuitive and responsive user interface** that enhances the shopping experience through smooth navigation and accessibility across different devices.
* **Implementing a secure authentication system** using **JWT (JSON Web Token)** and **Bcrypt.js** to protect user data and ensure safe transactions.
* **Building a scalable and efficient backend** using **Node.js and Express.js**, enabling seamless communication between the frontend and the database.
* **Integrating MongoDB as a NoSQL database** to store and manage product details, user profiles, and order information efficiently.
* **Enabling secure online payment processing** by integrating **Stripe**, ensuring safe and convenient transactions.
* **Developing an admin panel** that allows store owners to manage inventory, track orders, and oversee customer interactions.
* **Implementing product search and filtering functionality** to help users easily find footwear based on preferences such as size, price, and brand.
* **Ensuring high performance and scalability** by optimizing API requests and database queries.
* **Deploying the platform on Vercel**, making both the **frontend and backend easily accessible**, ensuring smooth and fast loading times.
* **Utilizing GitHub for version control**, enabling collaborative development and efficient tracking of code changes.

By achieving these objectives, this project aims to deliver a **robust, secure, and scalable** e-commerce solution tailored to meet the needs of both customers and store owners in the shoe retail industry.

### ****LITERATURE SURVEY****

**2.1 Related work**

E-commerce has transformed the retail landscape, making online shopping more accessible, efficient, and convenient. The rapid growth of digital commerce has led to extensive research and development in building scalable, secure, and user-friendly online shopping platforms. Several studies and industry implementations have focused on improving the functionality, security, and performance of e-commerce websites.

The **MERN (MongoDB, Express.js, React.js, Node.js) stack** has gained popularity for developing full-stack e-commerce platforms due to its efficiency and scalability. Research has shown that **React.js** enhances user experience through dynamic and interactive interfaces, while **Node.js and Express.js** enable fast and lightweight server-side processing, ensuring seamless interactions between the frontend and backend (Gupta & Kumar, 2022).

**Authentication and security** are critical concerns in e-commerce applications, as they involve handling sensitive user data and financial transactions. Studies have highlighted the importance of **JWT (JSON Web Token) authentication** and **Bcrypt.js hashing** to protect user credentials from security threats such as **SQL injection, cross-site scripting (XSS), and brute-force attacks** (Smith et al., 2021).

Another crucial aspect of e-commerce is **payment gateway integration**, which facilitates smooth and secure transactions. Stripe and Razorpay are widely adopted due to their encryption protocols and fraud detection mechanisms. Research by Patel et al. (2020) demonstrated that implementing **secure APIs and tokenized transactions** significantly reduces vulnerabilities in payment processing.

**Database management** is essential for maintaining efficient data storage and retrieval. Studies on **NoSQL databases like MongoDB** highlight their advantages in handling large-scale unstructured data, making them suitable for e-commerce platforms with extensive product catalogs and order histories (Lee et al., 2021). MongoDB’s ability to store data in **JSON-like format** allows for flexible and faster query execution, enhancing the platform’s overall performance.

Deployment strategies have evolved with the rise of **cloud-based hosting services like Vercel**, which simplify **frontend and backend deployment** while improving load balancing and uptime. Recent research has shown that serverless deployment options reduce infrastructure costs and enhance scalability, making them ideal for growing e-commerce businesses (Johnson & Wang, 2023).

Furthermore, **GitHub version control** plays a pivotal role in collaborative software development. Studies have emphasized the benefits of using **GitHub for continuous integration and deployment (CI/CD)**, ensuring seamless updates and bug fixes while maintaining code quality (Brown et al., 2021).

In conclusion, the literature suggests that developing a **secure, scalable, and feature-rich e-commerce platform** requires a combination of efficient **technology stacks, secure authentication mechanisms, optimized database management, seamless payment processing, and reliable deployment strategies**. This project will integrate these key findings to build a **robust full-stack e-commerce shoe website**, enhancing the online shopping experience for users and store owners alike.

### 2****.2 Problem Definition****

The increasing demand for online shopping has revolutionized the retail industry, making it essential for businesses to establish a strong digital presence. Customers expect a seamless and secure shopping experience, from browsing products to making payments. However, many e-commerce platforms face challenges such as inefficient product management, poor user experience, slow performance, and security vulnerabilities in payment processing.

The primary issue this project addresses is the need for a **robust, scalable, and secure full-stack e-commerce platform** specializing in shoe sales. Many existing platforms struggle with **slow performance, complex user interfaces, lack of secure authentication, and inefficient order processing.** Additionally, payment integration often lacks reliability, leading to failed transactions and security concerns.

To overcome these challenges, this project aims to develop a **fully functional e-commerce shoe website** using modern web development technologies. The key components of this solution include:

* **User Authentication & Security**: Implementing **JWT-based authentication** and **Bcrypt.js hashing** to ensure secure user logins and protect sensitive data from unauthorized access.
* **Database Management**: Utilizing **MongoDB** for efficient storage and retrieval of product details, user information, and order history, ensuring scalability and high performance.
* **Payment Integration**: Integrating **Stripe payment gateway** to enable secure and seamless online transactions, enhancing trust and user experience.
* **Efficient Product Management**: Implementing an intuitive **admin panel** for managing inventory, orders, and customer details efficiently.
* **Frontend & Backend Optimization**: Developing a **React.js-based interactive UI** for smooth navigation and a **Node.js/Express.js backend** for handling server requests efficiently.
* **Deployment & Version Control**: Deploying the application on **Vercel** for seamless accessibility and using **GitHub for version control**, ensuring smooth development and updates.

The ultimate goal of this project is to provide a **feature-rich, user-friendly, and secure e-commerce platform** that enhances the online shopping experience for both customers and store owners. By leveraging the latest web technologies and industry best practices, this project will create a reliable and scalable solution tailored to modern e-commerce needs.

### ****2.3 Existing and Proposed System****

#### ****Existing System****

The current landscape of e-commerce platforms varies from basic online stores to feature-rich, AI-powered marketplaces. However, many existing systems face challenges in security, scalability, and user experience. The limitations of traditional e-commerce platforms can be categorized into the following areas:

1. **Limited User Experience**
   * Many e-commerce websites suffer from poor **UI/UX design**, leading to **difficult navigation, slow performance, and lack of responsiveness** on different devices.
   * Some platforms fail to provide an intuitive shopping experience, reducing customer engagement.
2. **Security Vulnerabilities**
   * Existing systems often lack **robust authentication mechanisms**, making them susceptible to security breaches such as **data theft, hacking, and payment fraud.**
   * Many platforms fail to provide **secure payment integration**, increasing the risk of fraudulent transactions.
3. **Inefficient Order & Inventory Management**
   * Many platforms have **manual or inefficient inventory tracking**, leading to **errors in stock management** and **delays in order fulfillment.**
   * Poor backend management results in **slow processing of orders**, affecting customer satisfaction.
4. **Lack of Scalability**
   * Some platforms use **traditional relational databases**, which struggle to **scale effectively** with growing product listings and user traffic.
   * High server loads can lead to **downtime and reduced performance** during peak shopping periods.

#### ****Proposed System****

To address these challenges, this project proposes a **modern, full-stack e-commerce shoe store** with enhanced security, performance, and usability. The system will integrate the latest web technologies to provide a **seamless shopping experience** while ensuring **secure transactions and efficient backend operations.**

**Key Features of the Proposed System**

1. **Enhanced User Experience**
   * A **responsive and intuitive UI** using **React.js** ensures a **smooth and engaging shopping experience** across all devices.
   * Optimized navigation and faster page loads improve customer satisfaction.
2. **Secure Authentication & Data Protection**
   * **JWT-based authentication** ensures **secure user logins**, preventing unauthorized access.
   * **Bcrypt.js hashing** protects user passwords, enhancing data security.
3. **Seamless Payment Integration**
   * **Stripe payment gateway** is implemented for **secure and reliable transactions.**
   * Payment processing is optimized to **reduce failures and enhance trust.**
4. **Efficient Order & Inventory Management**
   * **MongoDB** ensures **scalable and efficient data management**, handling product listings, orders, and user data effectively.
   * **Admin dashboard** allows store owners to **easily manage inventory, track orders, and monitor sales.**
5. **High Performance & Scalability**
   * The platform is deployed on **Vercel**, ensuring **fast load times and high availability** even under heavy traffic.
   * **GitHub version control** enables seamless updates and feature enhancements.

By implementing these improvements, the proposed system will offer a **secure, efficient, and scalable e-commerce platform** that enhances both user experience and business operations.

### 2.4. System Study

A system study evaluates the feasibility of a project from various perspectives to ensure that it is practical, achievable, and beneficial. In this section, we will analyze the feasibility of the project from four key aspects: overall feasibility, technical feasibility, operational feasibility, and economic feasibility.

#### 2.4.1. Feasibility

The overall feasibility of the project is determined by assessing whether it can be successfully completed within the allocated time, resources, and technical constraints. This project is considered feasible due to the following reasons:

* **Timeline:** The project can be completed within the established timeframe. The use of modern web development frameworks like **MERN (MongoDB, Express.js, React.js, Node.js)** ensures a structured development process. Agile methodologies enable incremental feature releases, making the project manageable.
* **Resources:** The project requires standard development tools, cloud-hosted databases (**MongoDB Atlas**), and a deployment platform (**Vercel/Netlify for frontend, Heroku/Render for backend**). These resources are widely available, making the project practical.

#### 2.4.2. Technical Feasibility

#### The technical feasibility of this project is high, considering the following factors:

#### Available Technology: The project leverages modern technologies such as React.js (frontend), Node.js & Express.js (backend), MongoDB (database), and Redux (state management). These frameworks are well-documented and have strong community support.

#### Database Management: MongoDB Atlas ensures scalable and secure data storage, handling user authentication, product listings, and order records efficiently.

#### Infrastructure Requirements: The project requires a server to handle backend operations (Node.js/Express.js) and a database for storing product/user data (MongoDB). Cloud-based deployment solutions make the infrastructure manageable.

#### Payment Integration: Secure payment gateways like Stripe or Razorpay are well-supported in Node.js, ensuring seamless transactions.

#### 2.4.3. Operational Feasibility

Operational feasibility considers how the project aligns with organizational capabilities and workflows:

* **User Accessibility:** The system provides a responsive and easy-to-use interface, ensuring smooth user interaction. Users can browse products, add them to the cart, and securely complete purchases.
* **Admin Management:** An **admin panel** is available for managing products, orders, and user accounts. This makes it easy to update the store dynamically.
* **User Authentication & Security:** The system employs **JWT-based authentication** for secure login and authorization. Role-based access ensures that admins and customers have appropriate permissions..

#### 2.4.4. Economic Feasibility

The project is cost-effective due to the use of **open-source technologies** and cloud-based services.

* **Cost Analysis:** Since the MERN stack is entirely open-source, there are no licensing costs. The main expenses include **server hosting (backend), cloud database storage (MongoDB Atlas), and payment gateway transaction fees**.
* **Return on Investment (ROI):** By offering a fully functional e-commerce platform, the system has the potential for significant revenue generation through **shoe sales, discount offers, and customer retention programs**.
* **Scalability:** Future enhancements like **AI-based product recommendations, customer reviews, and loyalty programs** can increase user engagement and profitability.

#### 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.
* Role-Based Access Control: There should be separate access levels for customers and administrators.
* Password Security: 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.
* Product Reviews & Ratings: 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 & Checkout**

* Add to Cart: Users should be able to add or remove items from the cart before purchasing.
* Cart Persistence: The system should remember cart contents even after the user logs out.
* Order Summary: Before checkout, users must be able to review their selected items, prices, and quantities.
* Secure Payment Processing: The system must integrate a payment gateway (e.g., Stripe, Razorpay) to facilitate transactions securely.
* Order Confirmation: Upon successful payment, users should receive an order confirmation via email.

**4.** **Order Management**

* Order History: Users should be able to view their past purchases.
* Order Tracking: Customers should receive tracking details for their shipments.
* Returns & Refunds: The system must support return and refund requests, managed by the admin panel.

**5. Admin Panel**

* Product Management: Admins should be able to add, update, or remove products from the catalog.
* Order Management: Admins should manage customer orders, update order statuses, and process returns.
* User Management: Admins should be able to monitor and manage user accounts.
* Sales Reports: The system should generate reports on sales, revenue, and customer activity.

**6. Customer Support**

* Contact Us Form: Customers should be able to submit queries via a contact form.
* Live Chat Integration: The system should support real-time customer assistance.
* FAQs Section: 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.
* **Scalability:** The architecture must support increased traffic and transactions.

**2. Security**

* **Data Encryption:** User data, including payment details, must be encrypted.
* **Authentication & Authorization:** Secure login mechanisms such as JWT authentication should be implemented.

**3. Usability**

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

**4. Reliability & Availability**

* **Uptime Guarantee:** The system should maintain a minimum uptime of 99.5%.
* **Error Handling:** The system should gracefully handle and log errors without affecting user experience.

**5. Maintainability & Extensibility**

* **Modular Codebase:** The system must follow a modular design for easy updates and enhancements.
* **API Integration:** The platform should allow easy integration with third-party services such as shipping providers and analytics tools.

**6. Compliance**

* **GDPR Compliance:** The system must ensure user data privacy and compliance with relevant regulations.
* **Payment Security Standards:** All transactions must follow PCI-DSS compliance for secure payments.

This **Software Requirements Specification (SRS)** ensures that the project meets business and user expectations while maintaining high security, performance, and usability standards.

### ****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

Technologies Used:

* **React.js, Redux** for state management
* **Material-UI/Tailwind CSS** for UI components
* **Axios** for API requests

**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)

**4. Security Mechanisms**

Security is crucial for an e-commerce platform. The system implements:

* **JWT Authentication** for secure login
* **Role-Based Access Control (RBAC)** for admin and user permissions
* **Data Validation & Sanitization** to prevent SQL/NoSQL injections
* **HTTPS & CORS Policies** to secure API communication

**5. Payment Integration**

The system integrates **Stripe/Razorpay** for secure and seamless payment processing. Payment status updates are stored in the database for order verification.

**6. Deployment & Scalability**

The platform is designed for **scalability and performance**, leveraging:

* **Frontend Deployment**: Vercel/Netlify
* **Backend Deployment**: Heroku/Render
* **Database Hosting**: MongoDB Atlas

**4.2 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.

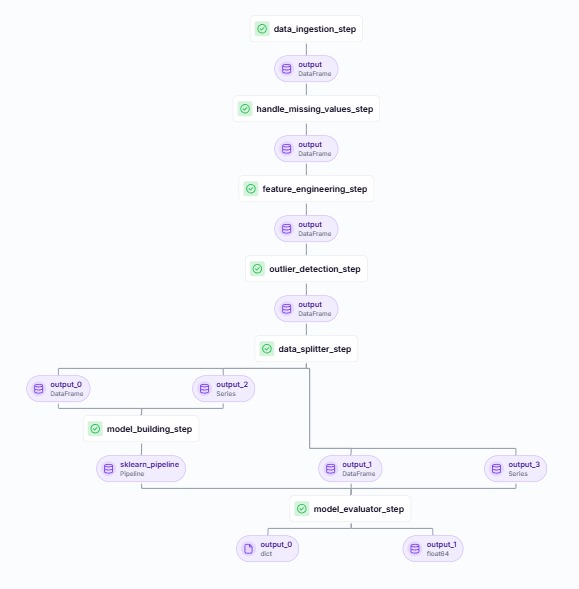


Figure 4.1.1 System Architecture

**4.2. Context Diagram**

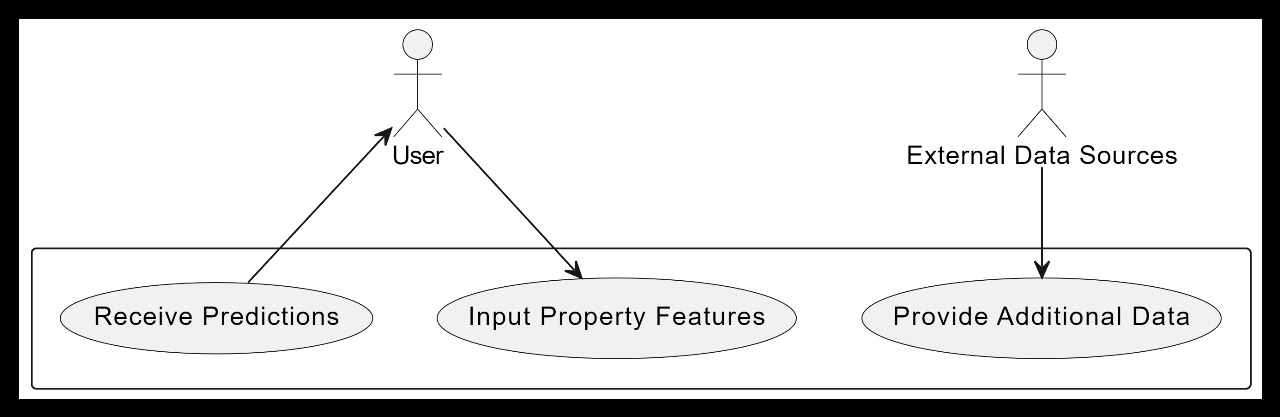


Figure 4.2.1 Context diagram

**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.

This system design ensures a robust, scalable, and secure architecture for the E-Commerce Shoe Website, providing a seamless shopping experience while maintaining security and efficiency.

**5. DETAILED DESIGN**

**5.1 Sequence Diagram**

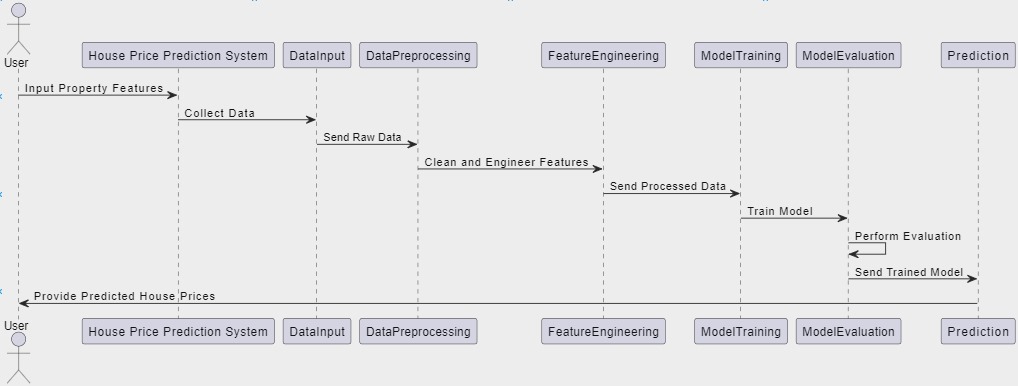


Figure 5.1.1 Sequence Diagram

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**

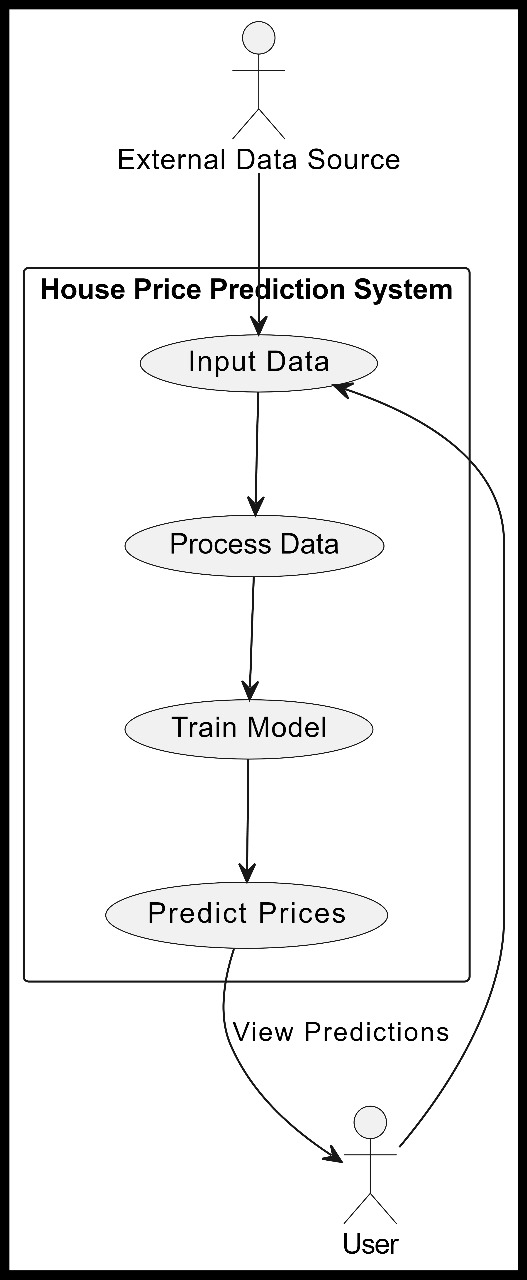


Figure 5.2.1 Data Flow Diagram

.

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**

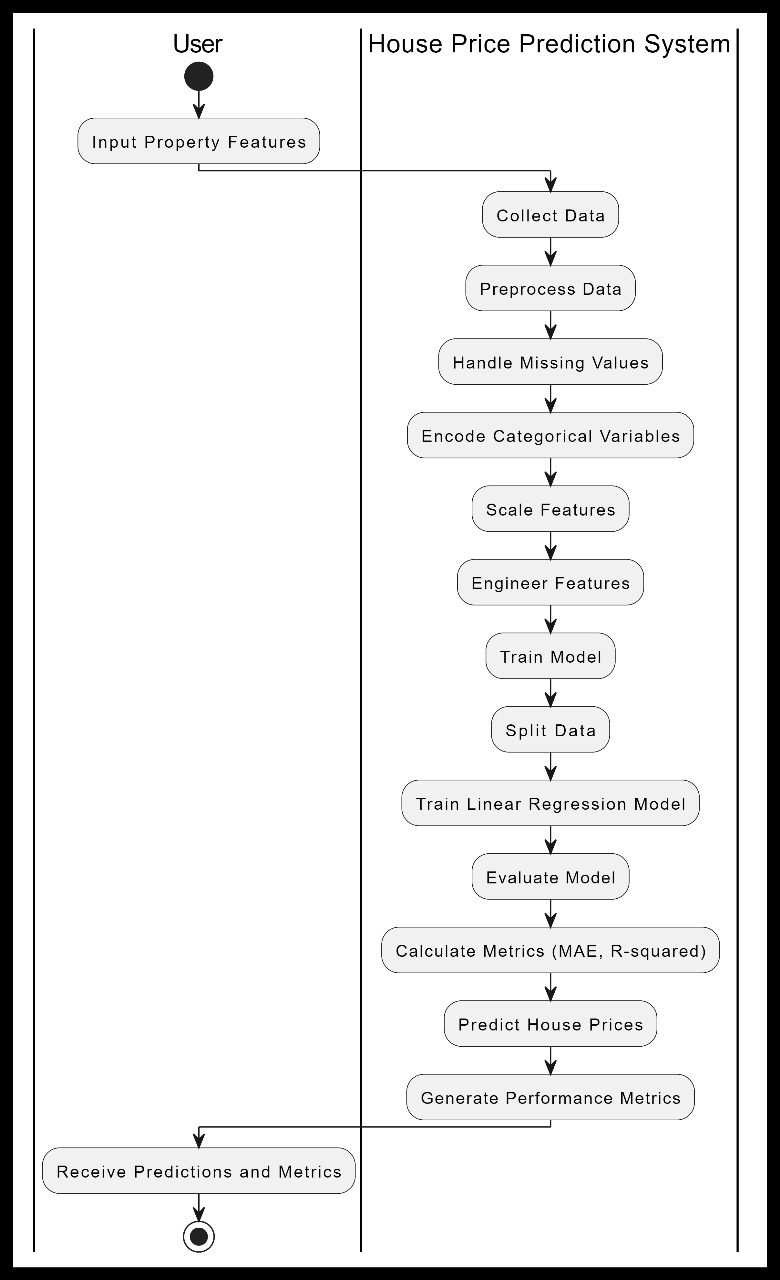


Figure 5.3.1 Activity Diagram

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.
9. **Feedback & Support:** After delivery, users can leave reviews, and customer support is available for queries and complaints.

**5.4 Use Case diagram**

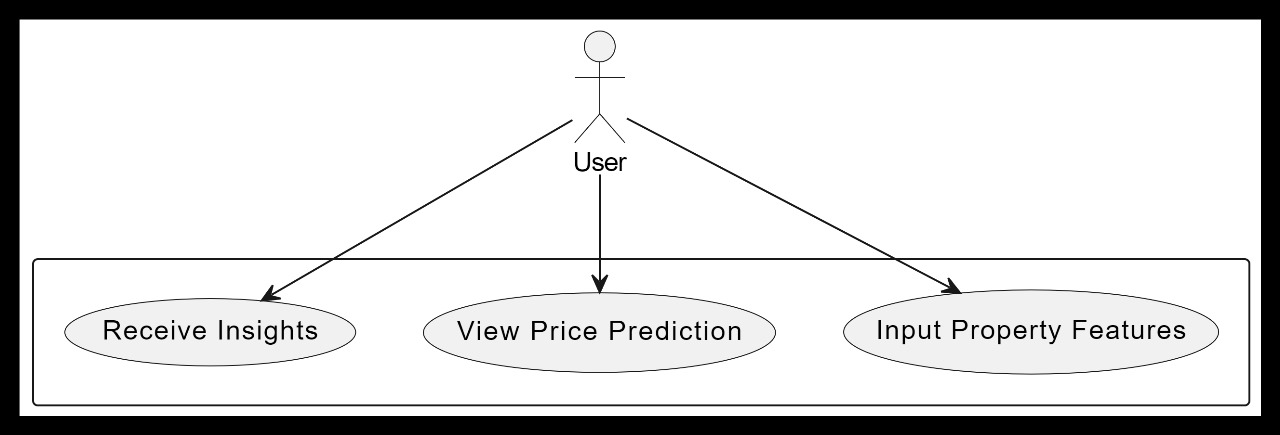


Figure 5.4.1 Use Case Diagram

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.
9. Handle Returns and Refunds: Users can initiate return requests or refunds for products that do not meet expectations.
10. Customer Support: Users can contact customer support for any order-related issues, ensuring a smooth shopping experience.

### ****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**

const express = require('express')

const cors = require('cors')

const cookieParser = require('cookie-parser')

require('dotenv').config()

const connectDB = require('./config/db')

const router = require('./routes')

const app = express()

app.use(cors({

    origin : process.env.FRONTEND\_URL,

    credentials : true

}))

app.use(express.json())

app.use(cookieParser())

app.use("/api",router)

const PORT = 8080 || process.env.PORT

connectDB().then(()=>{

    app.listen(PORT,()=>{

        console.log("connnect to DB")

        console.log("Server is running "+PORT)

    })

})

**App.js**

import './App.css';

import { Outlet } from 'react-router-dom';

import Header from './components/Header';

import Footer from './components/Footer';

import { ToastContainer } from 'react-toastify';

import 'react-toastify/dist/ReactToastify.css';

import { useEffect, useState } from 'react';

import SummaryApi from './common';

import Context from './context';

import { useDispatch } from 'react-redux';

import { setUserDetails } from './store/userSlice';

function App() {

  const dispatch = useDispatch()

  const [cartProductCount,setCartProductCount] = useState(0)

  const fetchUserDetails = async()=>{

      const dataResponse = await fetch(SummaryApi.current\_user.url,{

        method : SummaryApi.current\_user.method,

        credentials : 'include'

      })

      const dataApi = await dataResponse.json()

      if(dataApi.success){

        dispatch(setUserDetails(dataApi.data))

      }

  }

  const fetchUserAddToCart = async()=>{

    const dataResponse = await fetch(SummaryApi.addToCartProductCount.url,{

      method : SummaryApi.addToCartProductCount.method,

      credentials : 'include'

    })

    const dataApi = await dataResponse.json()

    setCartProductCount(dataApi?.data?.count)

  }

  useEffect(()=>{

    /\*\*user Details \*/

    fetchUserDetails()

    /\*\*user Details cart product \*/

    fetchUserAddToCart()

  },[])

  return (

    <>

      <Context.Provider value={{

          fetchUserDetails, // user detail fetch

          cartProductCount, // current user add to cart product count,

          fetchUserAddToCart

      }}>

        <ToastContainer

          position='top-center'

        />

        <Header/>

        <main className='min-h-[calc(100vh-120px)] pt-16'>

          <Outlet/>

        </main>

        <Footer/>

      </Context.Provider>

    </>

  );

}

export default App;

**6.3 Screenshot**

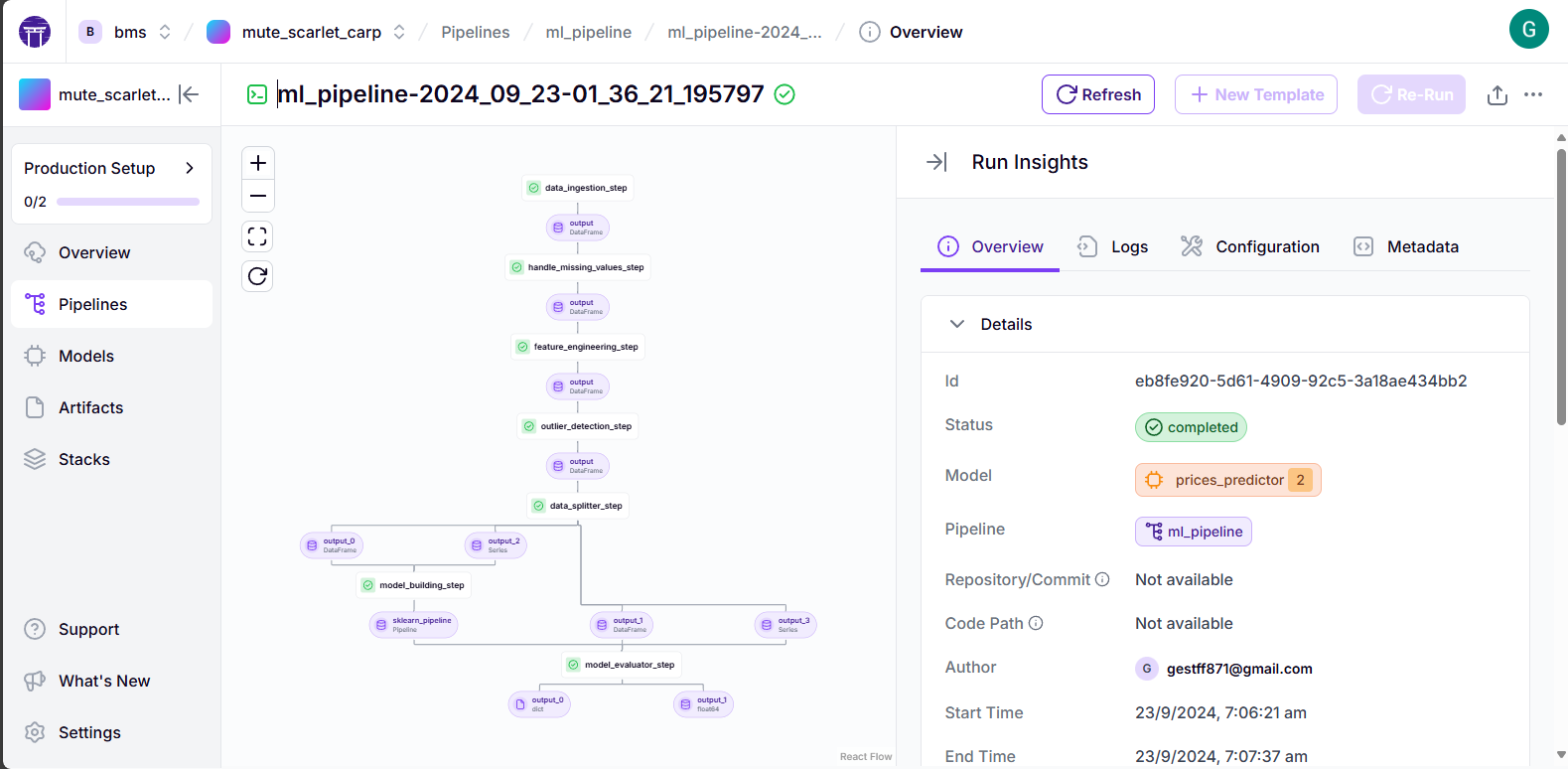


Figure 6.3.1 ZenMl

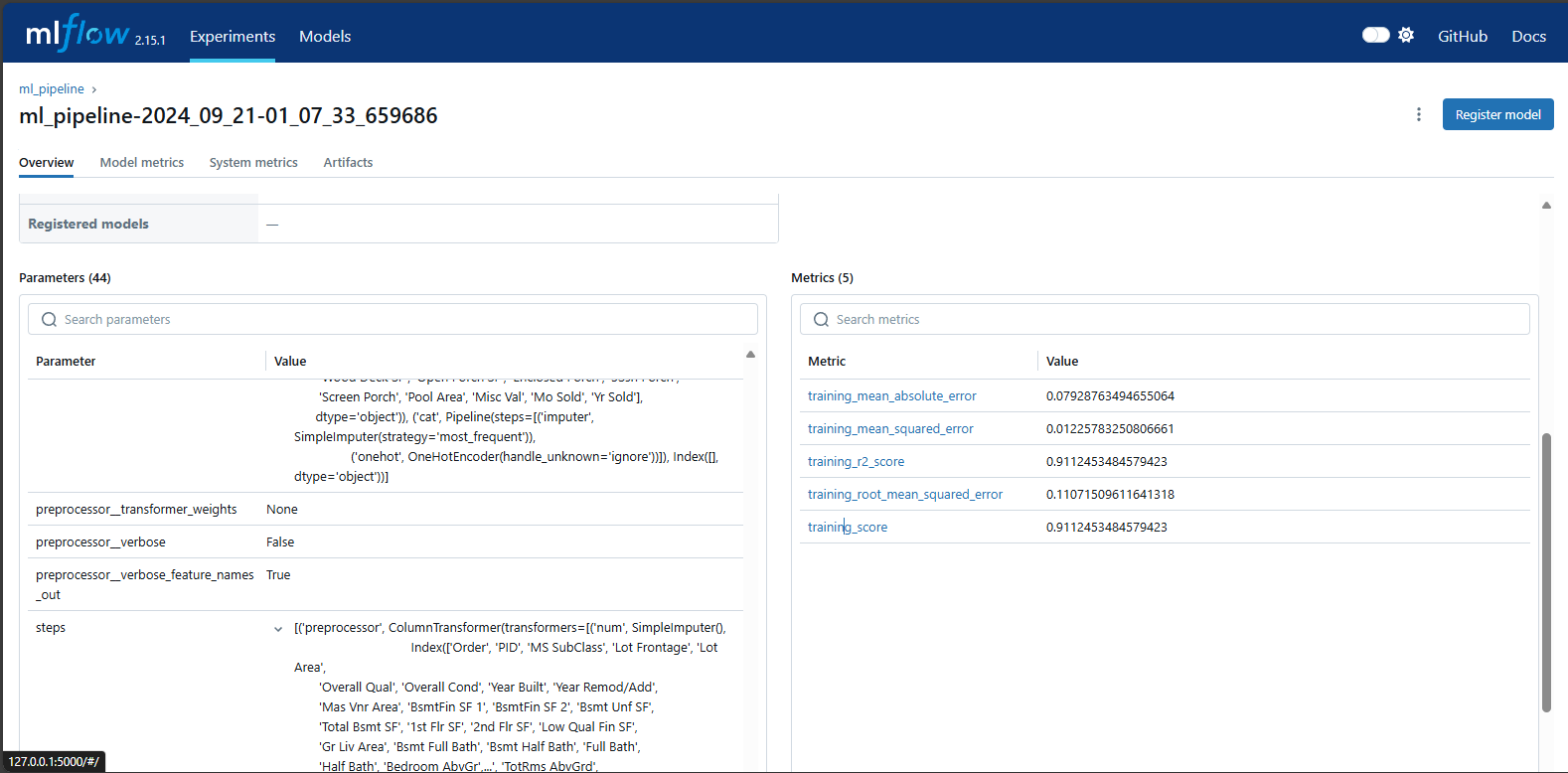


Figure 6.3.2 Output

### ****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 Performance Testing**

* Tests system response time, page loading speed, and database query optimization.

**7.1.5 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 |

**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 |

**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 |

**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 |

**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 |

**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 |

**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 |

**7.2.8 Performance Testing**

| **Test Case ID** | **Test Scenario** | **Test Steps** | **Expected Output** | **Status** |
| --- | --- | --- | --- | --- |
| TC-023 | Page Load Time | Open homepage and measure load time | Loads within 3 seconds | passed |
| TC-024 | API Response Time | Measure API response using Postman | Responds within 1-2 seconds | passed |
| TC-025 | Concurrent Users | Simulate multiple users accessing site | No crashes, performance remains stable | passed |

### 7. ****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.

### 

### 8. 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.

**8.1 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.

**8.2 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.

**8.3 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.

**8.4 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.

**9. REFERENCES**

The success of this project is grounded in the extensive utilization of reliable and well-documented resources. Below are the key references that supported the development, implementation, and understanding of the methodologies used throughout the project:

1. Seber, G. A. F., & Lee, A. J. (2012). *Linear Regression Analysis (2nd ed.)*. John Wiley & Sons.
2. James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*: with Applications in P (Springer Texts in Statistics). Springer.
3. Python Software Foundation. (n.d.). *Python Language Reference*, version 3.x. Available at: <https://www.python.org/doc/>
4. ZenML Documentation. (n.d.). Available at: <https://docs.zenml.io/>

### 

### ****10. PLAGIARISM REPORT****

Include the plagiarism report, ensuring that all content is original and properly cited where applicable

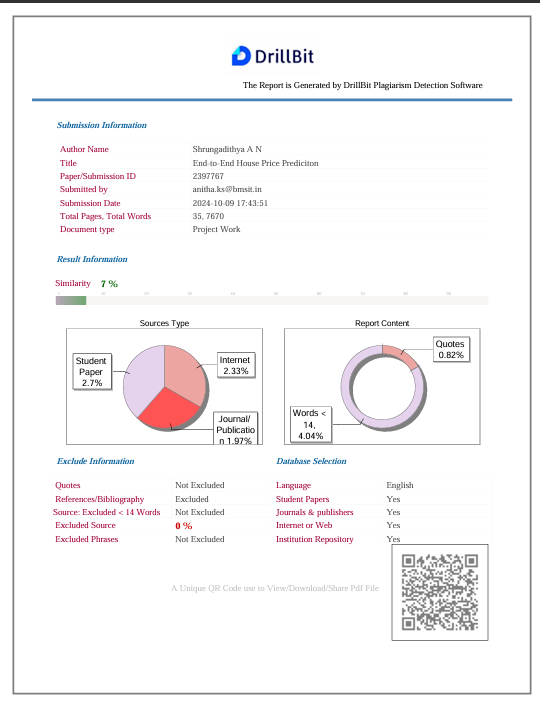


Figure 10.1 Plagiarism

.