

**A Mini Project Report**

**On**

**URBAN NEST - E-Commerce Platform**

Submitted in Partial fulfillment of requirements for the award of the degree  
of

**BACHELOR OF ENGINEERING**

**In**

**COMPUTER SCIENCE & ENGINEERING**

**By**

**BANDARU SOMI**

**245522733134**

*Mrs. K. Prajwala*

*Assistant Professor, Department of CSE*



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**KESHAV MEMORIAL ENGINEERING COLLEGE**

**Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)**

**D.No. 10 TC-111, Kachavanisingaram (V), Ghatkesar (M), Medchal-Malkajgiri, Telangana – 500 088**

**Website:www.kmec.in, Email-id: principal@kmec.in, M: +91 40 29560274**

**August - 2025**



# KESHAV MEMORIAL ENGINEERING COLLEGE

Approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

D.No. 10 TC-111, Kachavanisingaram (V), Ghatkesar (M), Medchal-Malkajgiri, Telangana – 500 088

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

### CERTIFICATE

This is to certify that this is a bonafide record of the project report titled “URBAN NEST” which is being presented as the Mini Project report by

**BANDARU SOMI**

**245522733134**

In partial fulfillment for the award of the degree of Bachelor of Engineering in Computer Science & Engineering affiliated to the Osmania University, Hyderabad.

**Mrs. K. Prajwala, Assistant Professor**

Guide of the Project

Department of Computer Science & Engineering,  
Keshav Memorial Engineering College,  
Hyderabad.

**Mrs. B. Ramya Sree, Assistant Professor**

Mini-Project Coordinator

Department of Computer Science & Engineering,  
Keshav Memorial Engineering College,  
Hyderabad.

**Head of the Department**

Department of Computer Science and Engineering,  
Keshav Memorial Engineering College,  
Hyderabad.

Submitted for Viva-Voce Examination held on \_\_\_\_\_

**External Examiner**

# **Vision & Mission of KMEC**

## **Vision of the College:**

- To be the fountain head in producing highly skilled, globally competent engineers. Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services.

## **Mission of the College:**

- To provide a learning environment that inculcates problem solving skills, professional, ethical responsibilities, lifelong learning through multi modal platforms, and prepares students to become successful professionals.
- To establish an industry institute Interaction to make students ready for the industry.
- To provide exposure to students on the latest hardware and software tools.
- To promote research-based projects/activities in the emerging areas of technology convergence.
- To encourage and enable students to not merely seek jobs from the industry but also to create new enterprises.
- To induce a spirit of nationalism which will enable the students to develop, understand India's challenges, and encourage them to develop effective solutions.
- To support the faculty to accelerate their learning curve to deliver excellent service to students.

## **Vision & Mission of CSE**

### **Vision of the CSE**

Producing quality graduates trained in the latest software technologies and related tools and striving to make India a world leader in software products and services. To achieve academic excellence by imparting in-depth knowledge to the students, facilitating research activities, and catering to the fast growing and ever-changing software industry demands and societal needs

### **Mission of the CSE**

- To create a faculty pool which has a deep understanding and passion for algorithmic thought process, which can then impart these skills to the students. The faculty will be chosen with extreme care and not merely based on qualification. The faculty will have to demonstrate substantial programming and solution skills during the selection process. The faculty will be nurtured to retain their cutting edge throughout their tenure.
- To impart skills in Computer Science which will not necessarily be part of the university prescribed curriculum. These skills may include programming languages, frameworks, platforms, and communication skills which will help in the development of a well-rounded Computer Science professional.
- To inculcate an ability in students to pursue Computer Science education throughout their lifetime. To exchange the use of multimodal learning platforms, including e-learning, blended learning and also remote testing and skilling.
- To ensure that the student's scope of knowledge is not limited to the conventional and traditional use of Computer Science. The student will be given exposure to different domains, different paradigms and exposed to the financial and commercial underpinning of the modern business environment. This will be enabled through the creation of an entrepreneur development cell on the campus.
- To encourage collaboration between KMEC and various organizations of repute on various latest technological and research initiatives.
- To create socially conscious and emotionally mature individuals who are aware of India's challenges and opportunities and who understand and appreciate the role and responsibility of engineers towards achieving the goal of job and wealth creation.

## PROGRAM OUTCOMES (POs)

**PO1. Engineering Knowledge:** Apply the knowledge of mathematics, science, Engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**PO2. Problem Analysis:** Identify, formulate, review research literature, and analyze complex engineering problems, reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences

**PO3. Design/Development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**PO4. Conduct Investigations of Complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**PO5. Modern Tool Usage:** Create select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**PO6. The Engineer and Society:** Apply reasoning informed by contextual knowledge to societal, health, safety. Legal und cultural issues and the consequent responsibilities relevant to professional engineering practice.

**PO7. Environment and Sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

**PO8. Ethics:** Apply ethical principles and commit professional ethics and responsibilities and norms of the engineering practice.

**PO9. Individual and Team Work:** Function effectively as an individual, and as a member or leader in diverse teams and in multidisciplinary settings.

**PO10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**PO11. Project Management and Finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**PO12. Life-Long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **PROGRAM SPECIFIC OUTCOMES (PSOs)**

**PSO1.** An ability to analyze the common business functions to design and develop appropriate Information Technology solutions for social upliftment.

**PSO2.** Shall have expertise on the evolving technologies like Mobile Apps, CRM, ERP, Big Data, etc.

## **PROGRAM EDUCATIONAL OBJECTIVES (PEOs)**

**PEO1.** Graduates will have successful careers in computer related engineering fields or will be able to successfully pursue advanced higher education degrees.

**PEO2.** Graduates will try and provide solutions to challenging problems in their profession by applying computer engineering principles.

**PEO3.** Graduates will engage in lifelong learning and professional development by rapidly adapting to a changing work environment.

**PEO4.** Graduates will communicate effectively, work collaboratively, and exhibit high levels of professionalism and ethical responsibility.

## PROJECT OUTCOMES

**P1:** Demonstrate the ability to synthesize and apply the knowledge and skills acquired in the academic program to real-world problems.

**P2:** Evaluate different solutions based on economic and technical feasibility

**P3:** Effectively plan a project and confidently perform all aspects of project management.

**P4:** Demonstrate effective coding, written, presentation and oral communication skills.

### MAPPING PROJECT OUTCOMES WITH PROGRAM OUTCOMES

PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
P1	H		M	H	M			L		M	M	M
P2	M	H	H	M		M			L	M		M
P3	M	M	H		M		L		H		H	M
P4	H	M		M	M			L	M	H	H	

**L– LOW**

**M –MEDIUM**

**H– HIGH**



### PROJECT OUTCOMES MAPPING WITH PROGRAM SPECIFIC OUTCOMES

PSO	PSO1	PSO2
P1	H	M
P2	M	
P3	H	L
P4		

### PROJECT OUTCOMES MAPPING WITH PROGRAM EDUCATIONAL OBJECTIVES

PEO	PEO1	PEO2	PEO3	PEO4
P1	H	M	H	
P2	M	H		M
P3		L		H
P4	M		M	H

## **DECLARATION**

I hereby declare that the results embodied in the dissertation entitled “**URBAN NEST**” has been carried out by me during the academic year 2024-25 as a partial fulfillment of the award of the B.E. degree in Computer Science & Engineering from Osmania University. I have not submitted this report to any other university or organization for the award of any other degree.

**BANDARU SOMI**

**245522733134**

## ACKNOWLEDGEMENT

I take this opportunity to thank all the people who have rendered their full support to our project work. I render our thanks to **Prof. P.V.N.Prasad**, Principal, who encouraged me to do the Project.

I am grateful to **Mr. Neil Gogte**, Founder & Director, and **Mr. S. Nitin**, Director, for facilitating all the amenities required for carrying out this project.

I express my sincere gratitude to **Ms. Deepa Ganu**, Academic Director, for providing an excellent environment in the college.

I am also thankful to **Dr. CH. Rathan Kumar**, **Dr. P.Bala Krishna**, Heads of the Department for providing me with time to make this project a success within the given schedule.

I am also thankful to **Mrs. B. Ramya Sree**, Project Coordinator, for supporting us from project selection till the submission to complete this project within the scheduled time.

I am also thankful to our Faculty Supervisor, **Mrs. K. Prajwala**, for her valuable guidance and encouragement given to us throughout the project work.

I would like to thank the entire CSE Department faculty, who helped us directly or indirectly in the completion of the project.

I sincerely thank our friends and family for their constant motivation during the project work.

**BANDARU SOMI**

**245522733134**

## ABSTRACT

This project presents the prototype and implementation of a full-stack e-commerce clothing shopping platform developed using the MERN stack, which integrates MongoDB, Express.js, React.js, and Node.js to create a scalable, secure, and user-friendly solution for online retail. The backend, implemented using Node.js and Express.js, is responsible for handling RESTful API development, user authentication and authorization, order processing, and product data management, while MongoDB serves as the database for securely storing user information, product catalogs, inventory details, and transaction records. The frontend, developed with React.js, offers a responsive and dynamic interface that enhances user experience by providing seamless navigation, real-time updates, and mobile-friendly design. The system enables customers to register and log in, browse and search for clothing products, add selected items to a shopping cart, and place secure online orders with proper session management. Token-based authentication using JSON Web Tokens ensures the confidentiality and integrity of user data, while password encryption further strengthens security. The platform also includes a dedicated admin panel, allowing administrators to manage the product catalog, update inventory, monitor orders, and perform CRUD operations for efficient store management. Built using modular design principles, reusable components, and scalable architecture, this system ensures maintainability and supports future feature expansion such as payment gateway integration, recommendation engines, and deployment on cloud environments for production use.

**Keywords:** MERN stack, MongoDB, Express.js, React.js, Node.js, e-commerce, online shopping, user authentication, JWT, product catalog, shopping cart, inventory management, order processing, admin panel, responsive UI.

## LIST OF FIGURES

<b>Fig. No.</b>	<b>Name of the Figure</b>	<b>Page No.</b>
1.4	Architecture Diagram of the Proposed Work	2
4.2.1	Use Case Diagram	9
4.2.2	Sequence Diagram	10
4.2.3.1	Customer State Chart Diagram	11
4.2.3.2	Admin State Chart Diagram	11
4.2.4	Class Diagram	12
4.2.5	Activity Diagram	13
4.2.6	Deployment Diagram	14
6.1	Product Page	22
6.2	Login Page	23
6.3	Cart Page	24
6.4	Checkout Page	25

## LIST OF TABLES

Table No.	Name of the Table	Page No.
5.3	Module-Wise Implementation Summary	21

# **CONTENTS**

Abstract	x
List of Figures	xi
List of Tables	xii
<b>CHAPTER-1 INTRODUCTION</b>	
1.1 Purpose of the Project	1
1.2 Scope of the Project	1
1.3 Problem with existing systems	1
1.4 Architecture Diagram	1
<b>CHAPTER-2 LITERATURE SURVEY</b>	
2.1 Overview	3
2.2 Related Work	3
<b>CHAPTER-3 SOFTWARE REQUIREMENT SPECIFICATION</b>	
3.1 Introduction to SRS	5
3.1.1 Purpose	5
3.1.2 Scope	5
3.2 Functional Requirements	6
3.3 Non-Functional Requirements	7
3.4 Software Requirements	7
3.5 Hardware Requirements	8
<b>CHAPTER-4 SYSTEM DESIGN</b>	
4.1 Introduction to UML	9
4.2 UML Diagrams	9
4.2.1 Use Case Diagram	9
4.2.2 Sequence Diagram	10
4.2.3 State Chart Diagram	11
4.2.4 Class Diagram	12
4.2.5 Activity Diagram	13
4.2.6 Deployment Diagram	14

## **CHAPTER-5    IMPLEMENTATION**

5.1	Technologies Used	16
5.2	Modules	17
5.3	Sample Code Snippets	18
5.3.1	User Registration (React + Express + MongoDB)	18
5.3.2	Product Cart Component (React State + Express API)	19
5.3.3	MongoDB Product Schema (Mongoose)	20
5.3.4	JWT Verification Middleware (Express.js)	20

## **CHAPTER-6    RESULTS**

6.1	Product Page	22
6.2	Login Page	23
6.3	Cart Page	24
6.4	Checkout Page	25

## **CHAPTER-7    CONCLUSION**

7.1	Project Findings	26
7.2	Future Scope	27

## **REFERENCES**



# **1. INTRODUCTION**

## **1.1 PURPOSE OF THE PROJECT**

The purpose of this project is to develop a full-stack e-commerce web application for online clothing shopping using the MERN stack (MongoDB, Express.js, React.js, Node.js). The goal is to provide a responsive, secure, and user-friendly platform where customers can browse clothing products, add them to a cart, and place orders. At the same time, the system allows administrators to manage product listings, inventory, and user data efficiently. By using modern technologies and a modular design, the project aims to offer a cost-effective and scalable solution for small to medium-sized clothing businesses that want to digitize their storefronts without relying on expensive third-party platforms. Security features like token-based authentication are also included to protect user information and ensure safe transactions.

## **1.2 SCOPE OF THE PROJECT**

The scope of this project is to provide a complete e-commerce solution tailored for small to medium-sized clothing businesses aiming to establish an online presence. The platform supports essential features such as user registration and login, detailed product listings with images and descriptions, shopping cart and checkout functionality, and an admin interface for managing inventory. It incorporates token-based authentication to ensure secure user access and offers a responsive frontend to deliver a smooth and accessible user experience across devices.

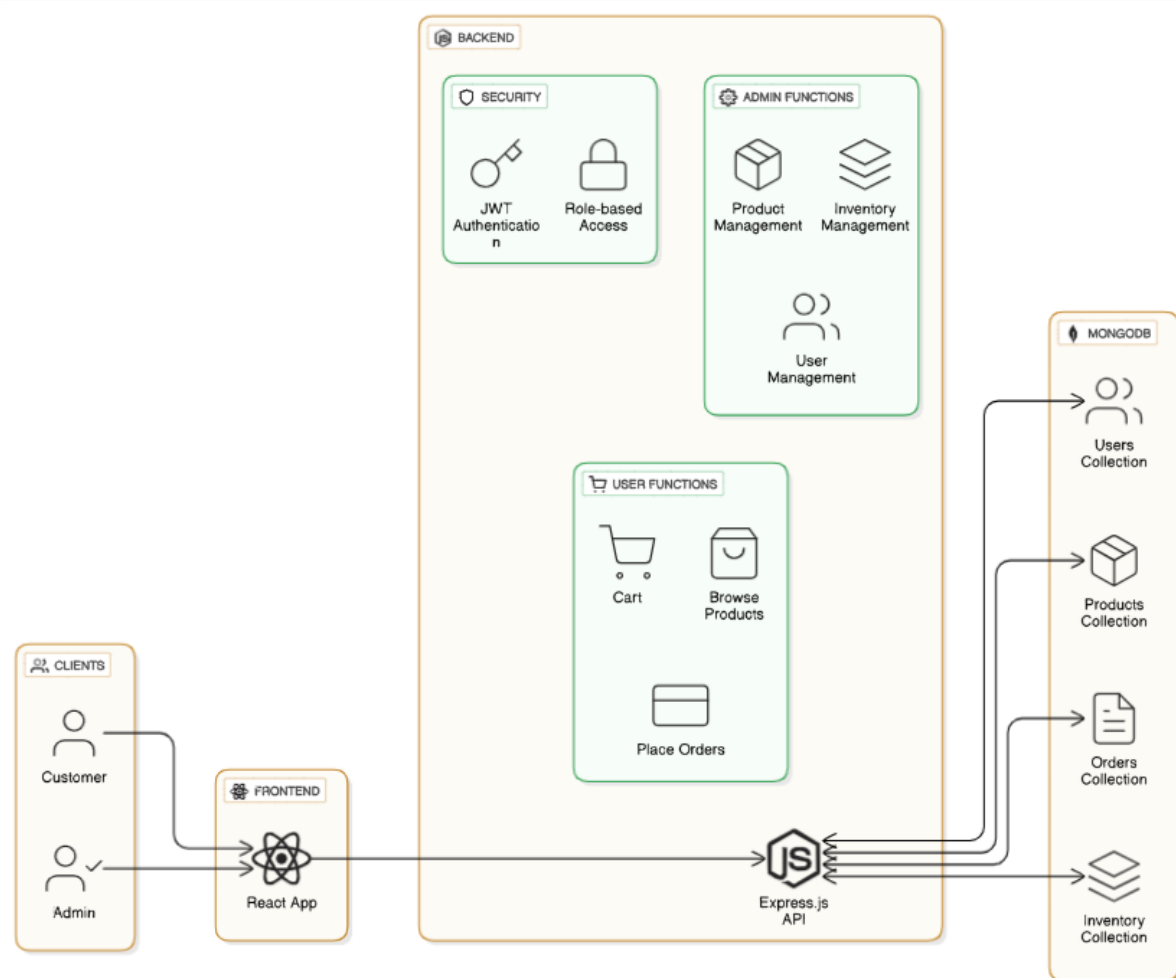
## **1.3 PROBLEMS WITH EXISTING SYSTEMS**

Most existing e-commerce platforms are expensive or hard to customize. Platforms like Shopify charge monthly fees and offer limited free features. Custom-built applications using older technologies like PHP are slower and harder to scale. Many systems are not mobile-friendly and offer a poor user experience. Security and data protection are often weak in low-cost platforms. Non-technical users also struggle to manage complex admin panels. These problems highlight the need for a simple, secure, and flexible platform. This project aims to address these issues using the MERN stack.

## **1.4 ARCHITECTURE DIAGRAM**

The architecture of Urban Nest follows a three-tier full-stack design using the MERN stack (MongoDB, Express.js, React.js, Node.js). The frontend layer, developed with React.js, provides a responsive and dynamic user interface where customers can browse products, manage their shopping cart, and place orders, while administrators can manage inventory and product listings. The backend

layer, built with Node.js and Express.js, acts as the application server, handling RESTful APIs, user authentication using JWT, business logic, and secure communication between the frontend and database. The database layer uses MongoDB to store structured and unstructured data, including user accounts, product catalogs, inventory, orders, and transaction records. The system supports role-based access control for customers and admins, follows modular and scalable design principles, and can be deployed on a cloud platform where the frontend and backend servers communicate via HTTPS, ensuring secure and efficient data flow.



**Fig. 1.4: Architecture Diagram of the Proposed Work**

## 2. LITERATURE SURVEY

### 2.1 OVERVIEW

The development of e-commerce platforms has rapidly evolved, driven by growing consumer demand for convenience, speed, and digital interaction. Over the years, many researchers have explored various technologies and frameworks to enhance the scalability, security, and usability of online shopping systems. In particular, the MERN stack—comprising MongoDB, Express.js, React.js, and Node.js—has gained popularity due to its full-stack capabilities, fast performance, and modularity. The following studies highlight how modern tools like MERN are being applied in e-commerce projects and how they compare to traditional methods.

### 2.2 RELATED WORK

According to reference [1], Patil et al. develop a robust e-commerce platform using the MERN stack, incorporating features such as a shopping cart, wish list, filters, and role-based access. Their implementation of voice search and messaging demonstrates MERN's ability to support advanced, user-centric functionality.

As noted in reference [2], De Silva et al. focus on the development workflow of a MERN-based chocolate shop. They emphasize CRUD operations, product search, and the integration of testing tools like Selenium and SonarQube, highlighting the value of DevOps practices in modern e-commerce solutions.

Reference [3] by Verma and Kale centres on a fashion e-commerce site with a strong emphasis on design and user experience. They implement a responsive interface, seamless navigation, and personalization features, aiming to enhance customer engagement and satisfaction. Their future plans also include augmented reality features.

In reference [4], Huang and Benyoucef conduct a large-scale review of 407 studies on social commerce platforms. They propose a taxonomy built around trust, technology, and social interaction, stressing the importance of community-building features for future e-commerce systems.

Esmaeili and Hashemi, in reference [5], analyse 81 research studies to examine trends in social commerce growth. Their findings highlight the significance of interface design, user behaviour, and platform adoption, with trust and engagement identified as core factors in successful platforms.

According to reference [6], Fernando et al. build a business management system using the MERN stack that includes modules for inventory, order tracking, and user/admin roles. Their project

showcases full end-to-end development and demonstrates MERN's scalability for enterprise-grade applications.

Reference [7] features Rafi's development of a multi-vendor e-commerce platform. Key components include secure user authentication, seller dashboards, and role-based access control. The project emphasizes a modular, scalable architecture suited for large, secure online marketplaces.

Budhe et al., in reference [8], create a real-time auction system using MERN technologies. With features like live bidding and dynamic product updates, their work illustrates the stack's strength in building interactive, event-driven applications in the e-commerce space.

As outlined in reference [9], Linga et al. present a MERN-based product display portal optimized for responsive design and mobile compatibility. The platform incorporates clean layouts and advanced filtering options to improve overall user experience.

Finally, in reference [10], Shrinidhi et al. introduce a digital storefront tailored for artisans. The MERN-based system emphasizes localization, accessibility, and personalized seller pages, offering an inclusive platform aimed at supporting small businesses in underrepresented regions.

The reviewed literature highlights the growing relevance of the MERN stack in developing scalable, user-friendly, and secure e-commerce platforms. From modular backend design to engaging front-end interfaces and social commerce integration, each study contributes unique insights into building effective online retail systems. These findings provided a strong foundation and direction for the design and development of our proposed system. The next chapter outlines the detailed software requirements specification for the project.

## **3. SOFTWARE REQUIREMENT SPECIFICATION**

### **3.1 INTRODUCTION TO SRS**

The Software Requirement Specification (SRS) for Urban Nest provides a comprehensive overview of the system's functionalities, features, and performance expectations. It acts as a formal agreement between stakeholders and the development team, ensuring that the final platform fulfills both user and technical requirements. Urban Nest is a web-based e-commerce application designed to facilitate seamless online clothing shopping, featuring dynamic product browsing, secure user authentication, cart and order management, and integrated payment processing. This document details the system's behavior under normal and exceptional conditions, addressing both functional and non-functional requirements.

Urban Nest is designed for two primary user roles: customers and administrators. Customers can register, browse products, manage carts, and place orders, while administrators manage product listings, track orders, and oversee inventory. The system prioritizes user experience, performance, security, and scalability, ensuring a reliable shopping experience for end users.

#### **3.1.1 Purpose**

The purpose of this project is to develop a full-stack web application for an e-commerce clothing shopping platform using the MERN stack (MongoDB, Express.js, React.js, Node.js). The system enables users to browse clothing items, register and log in, add products to a cart, and place orders. An administrative backend is included for managing product inventory, customer data, and orders. Security, scalability, and responsiveness are emphasized to provide a reliable shopping experience for users and easy backend operations for business owners.

#### **3.1.2 Scope**

This project aims to serve small to medium-sized clothing businesses by offering a complete e-commerce platform that eliminates the need for expensive third-party solutions. The platform supports:

- User registration, login, and authentication.
- Cart management and order placement.
- Admin dashboard for product/inventory control.
- Secure JWT token-based authentication.
- Responsive UI for desktop and mobile.

## 3.2 FUNCTIONAL REQUIREMENTS

- **User Registration & Login**

The platform provides secure user registration and login functionality. During sign-up, user passwords are hashed before being stored in the database, ensuring that sensitive credentials are not saved in plain text. Upon successful login, the system generates a JSON Web Token (JWT) that is used to manage the session. This token-based authentication ensures secure and persistent access to protected routes and user-specific data throughout the session.

- **Product Browsing**

Users can browse a wide range of clothing products available on the platform. The system supports filtering by category, size, or price, allowing users to narrow down their search and find items more efficiently. Each product is presented with detailed information on a separate product page, including images, descriptions, price, and other relevant attributes to help users make informed purchase decisions.

- **Shopping Cart**

The shopping cart allows users to manage selected items before finalizing a purchase. Users can add or remove products from the cart and update the quantity of each item as needed. The cart dynamically displays a summary that includes the list of items, individual prices, and the total cost, providing a clear and organized view of the pending order.

- **Order Placement**

Once the user is satisfied with their cart, they can proceed to place an order. The system captures the cart items and stores the order details securely in the MongoDB database. This includes product information, quantities, total price, and user identification. Users can later view their order history, which provides a record of all previously placed orders along with status updates where applicable.

- **Admin Dashboard**

The admin dashboard enables administrators to manage the platform's product inventory and customer orders. Admins can add new products, edit existing listings, or delete items that are no longer available. The dashboard also allows inventory levels to be monitored and updated. In addition, administrators can view customer orders, track order statuses, and oversee the fulfillment process to ensure smooth business operations.

### 3.3 NON-FUNCTIONAL REQUIREMENTS

#### ➤ Performance

The system is designed to deliver high performance under typical usage conditions. API responses are optimized to occur within 500 milliseconds under normal load, ensuring smooth and fast user interactions. Additionally, the application employs techniques such as lazy loading and image compression to improve the speed and efficiency of image delivery, which enhances the overall responsiveness and user experience of the platform.

#### ➤ Security

To maintain robust security, all user passwords are encrypted using the Bcrypt hashing algorithm before storage, preventing unauthorized access in case of a data breach. The system uses secure, token-based session management via JSON Web Tokens (JWT) to protect sensitive routes and user data. Also, HTTPS is enforced for all data transmissions between the client and server to safeguard against man-in-the-middle attacks and ensure encrypted communication.

#### ➤ Usability

The platform emphasizes user-friendly design by providing a clean, well-organized user interface that supports intuitive navigation across features. Whether accessing the platform via desktop, tablet, or mobile device, users benefit from a responsive layout that adapts seamlessly to various screen sizes, ensuring accessibility and ease of use for a wide range of users.

#### ➤ Scalability

The system architecture is built with scalability in mind, utilizing modular components in both the backend and frontend. This allows for easy integration of additional features in the future, such as payment gateways or third-party services. The database is optimized using indexing techniques to improve query speed and handle growing amounts of data efficiently, making the platform suitable for expanding user and product bases over time.

### 3.4 SOFTWARE REQUIREMENTS

- Frontend: React.js, Axios, React Router.
- Backend: Node.js, Express.js, MongoDB, Mongoose, JWT, Bcrypt.
- Database: MongoDB (Cloud via MongoDB Atlas or Local).
- Tools: Postman (API testing), Git & GitHub (version control), VS Code.

### **3.5 HARDWARE REQUIREMENTS**

- **Processor:** Quad-core CPU such as Intel i5 or AMD Ryzen 5
- **RAM:** Minimum 8 GB
- **Storage:** 20 GB of free space on an SSD
- **Internet:** Stable broadband, 10 Mbps



## 4. SYSTEM DESIGN

### 4.1 INTRODUCTION TO UML

Unified Modeling Language (UML) is used to visually represent the design and structure of a software system. It helps in understanding the system flow, user interactions, and component relationships. For this project, UML diagrams are used to depict functional and structural aspects of the e-commerce platform, including how users interact with the system and how data flows internally.

### 4.2 UML DIAGRAMS

#### 4.2.1 USE CASE DIAGRAM

A use case diagram is a visual tool that shows how different users interact with a system and what functions the system provides to them. It focuses on the actions a system can perform, called use cases, and the people or external systems that use it, called actors. For an e-commerce clothing shopping platform, the use case diagram would display how customers can register, log in, browse products, add items to a cart, and place orders, while administrators can manage the product catalog, update inventory, and monitor orders. The diagram includes a system boundary that contains all the functions, with actors placed outside and connected to their respective use cases, making it easy to understand the system's overall functionality and requirements for developers and stakeholders.

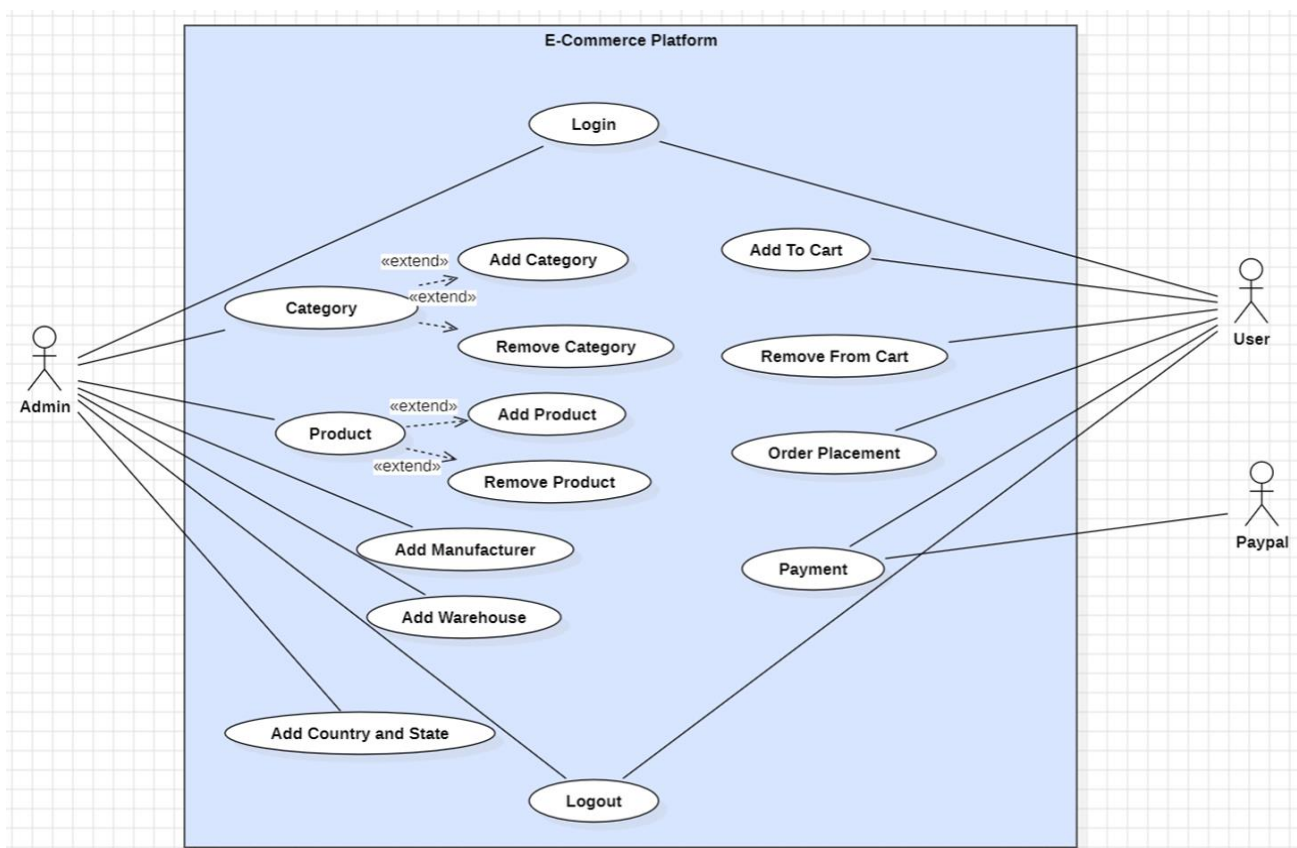
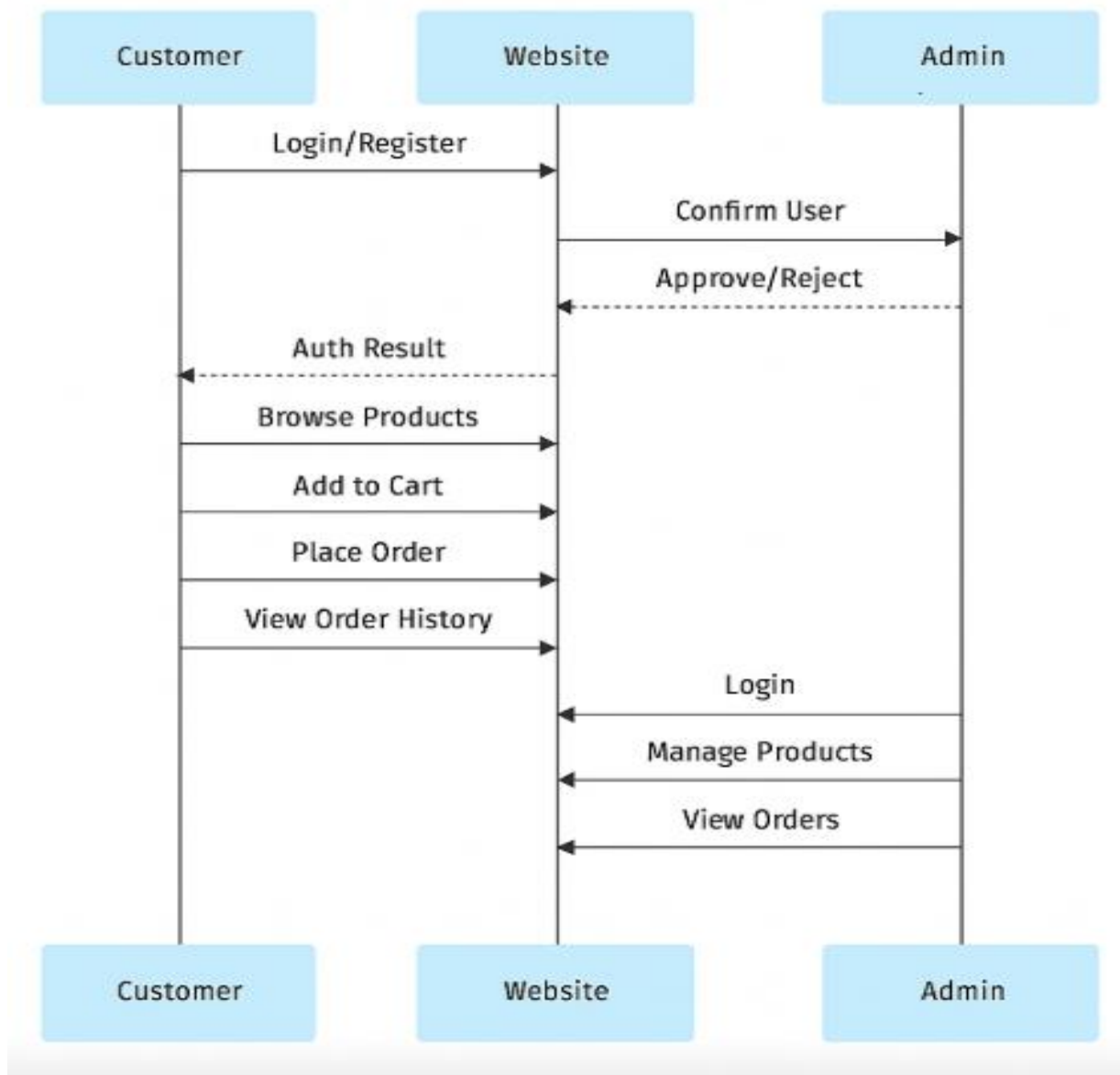


Fig. 4.2.1: Use Case Diagram

### 4.2.2 SEQUENCE DIAGRAM

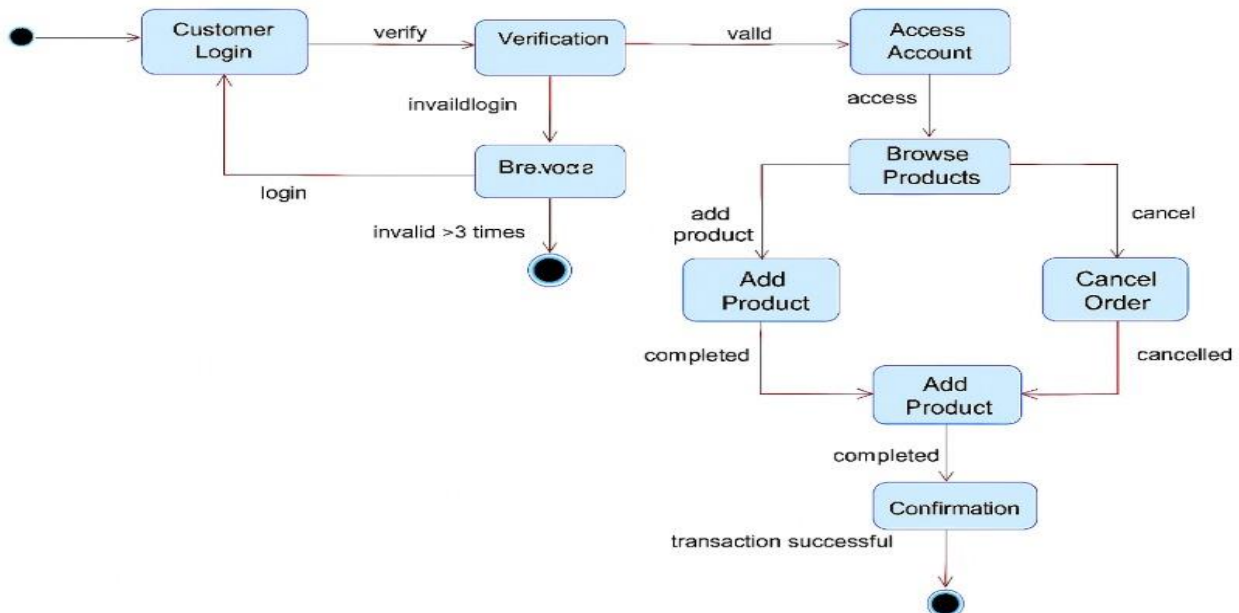
A sequence diagram shows the step-by-step flow of interactions between users and system components over time. In an e-commerce platform, it can depict how a customer logs in, browses products, adds items to the cart, and places an order, with messages flowing between the customer, frontend, backend, and database. It helps visualize the order and timing of system operations.



**Fig. 4.2.2: Sequence Diagram**

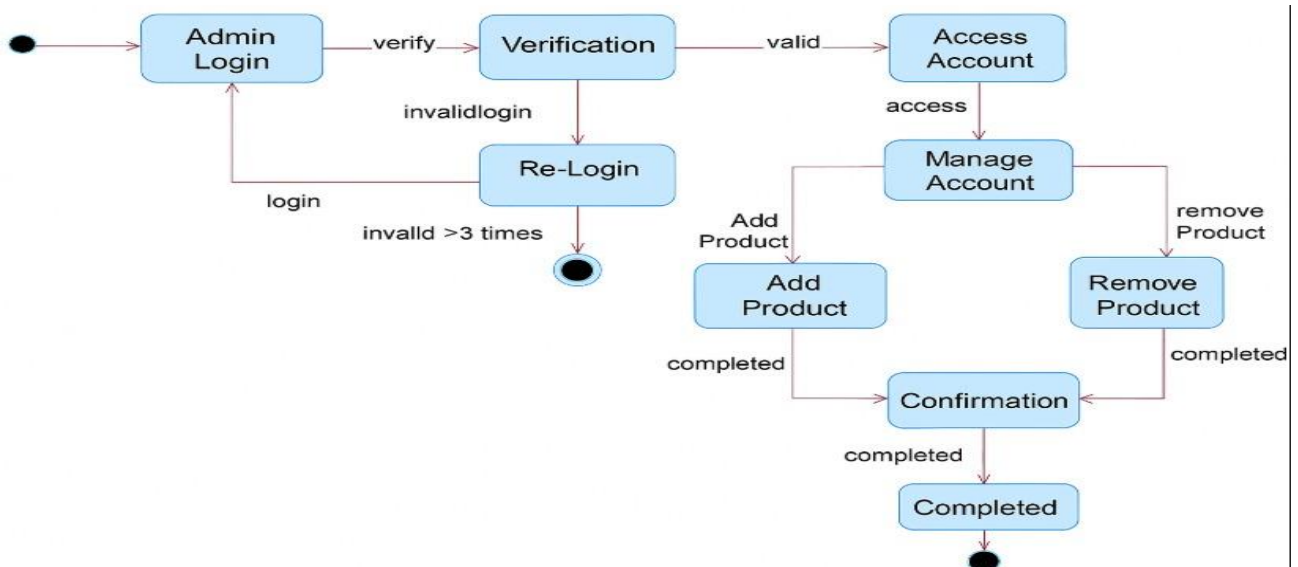
### 4.2.3 STATE CHART DIAGRAM

A state chart diagram shows the different states an object or system goes through during its lifecycle and how it transitions between these states based on events or actions. It focuses on the system's behavior rather than its structure.



**Fig.4.2.3.1: Customer State Chart Diagram**

Fig.4.3.3.1 Illustrates the customer state chart illustrates the various stages a customer goes through in an e-commerce system, including registration/login, browsing products, adding items to the cart, placing orders, and viewing order history. It highlights the transitions between actions and the flow of customer interactions with the platform.



**Fig.4.2.3.2: Admin State Chart Diagram**

Fig.4.3.3.1 Illustrates the admin state chart shows the states and transitions for administrative tasks, starting from admin login, managing account details, adding or removing products, and confirming each action. It focuses on product management and ends once the requested administrative operation is completed.

#### 4.2.4 CLASS DIAGRAM

A class diagram represents the structure of the system by showing classes, their attributes, methods, and relationships. For this project, classes could include User, Product, Cart, Order, and Admin, with relationships like a User having a Cart, a Cart containing Products, and an Order linked to both User and Product. It helps in understanding how data and logic are organized in the system.

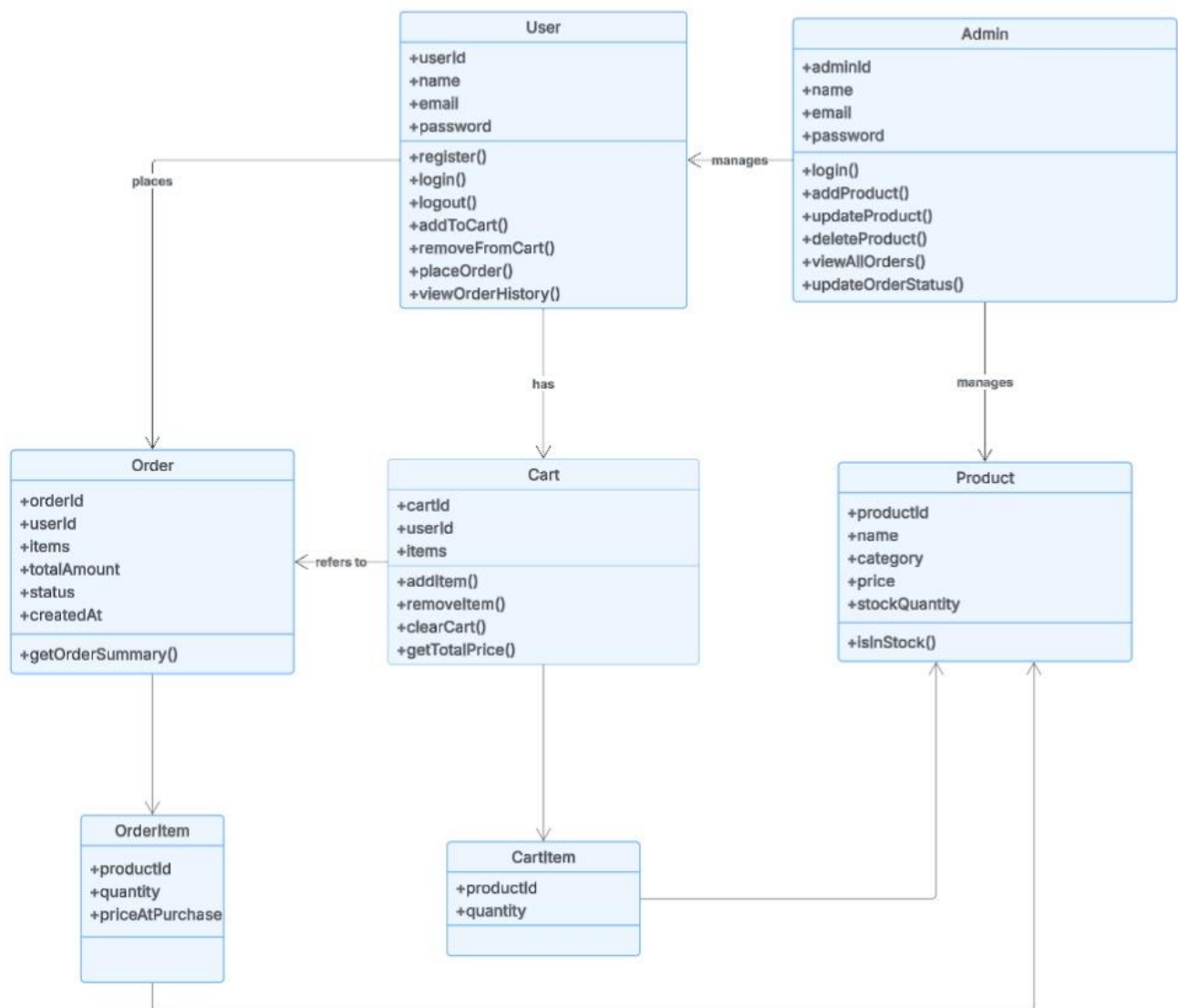
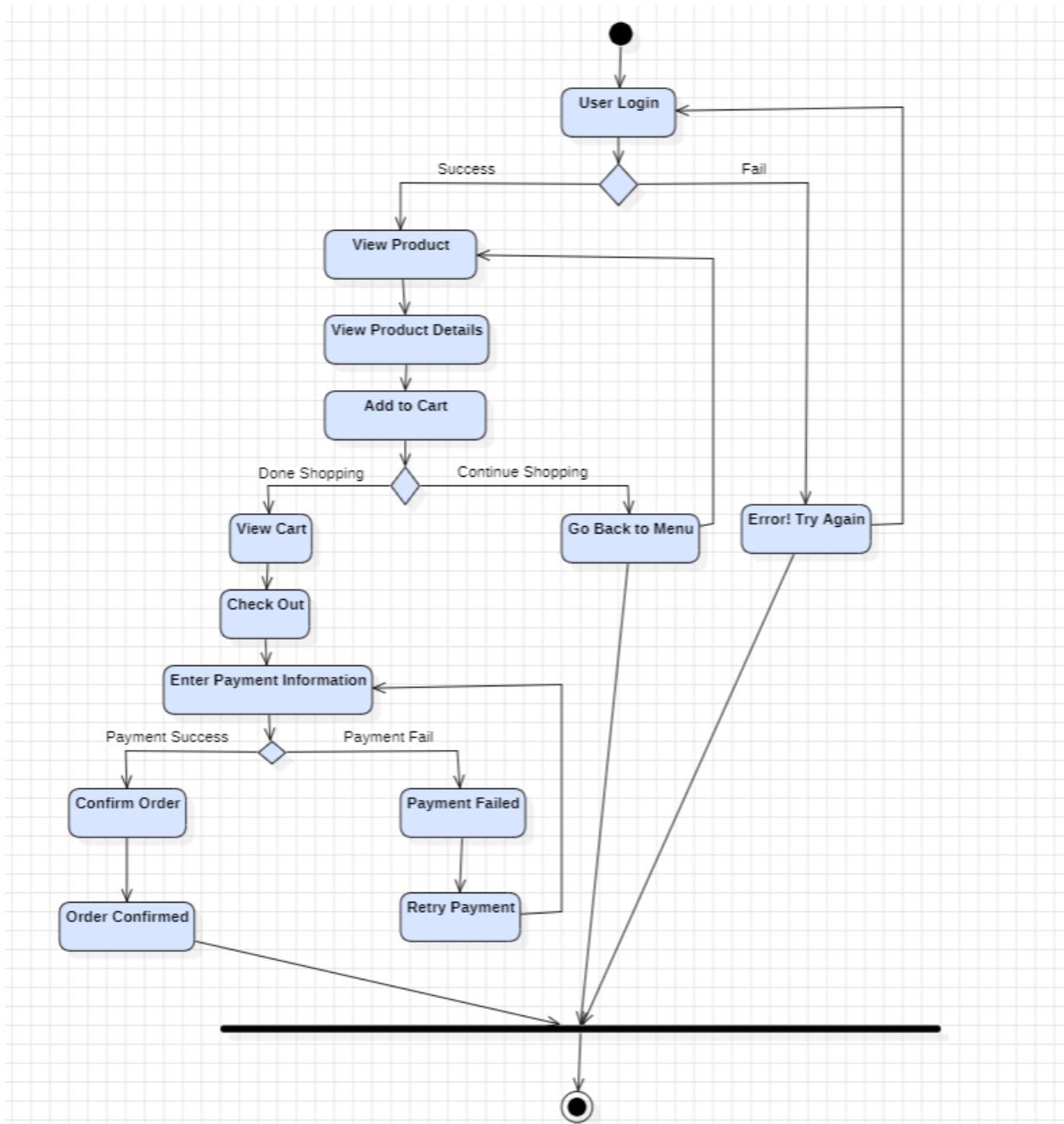


Fig. 4.2.4: Class Diagram

### 4.2.5 ACTIVITY DIAGRAM

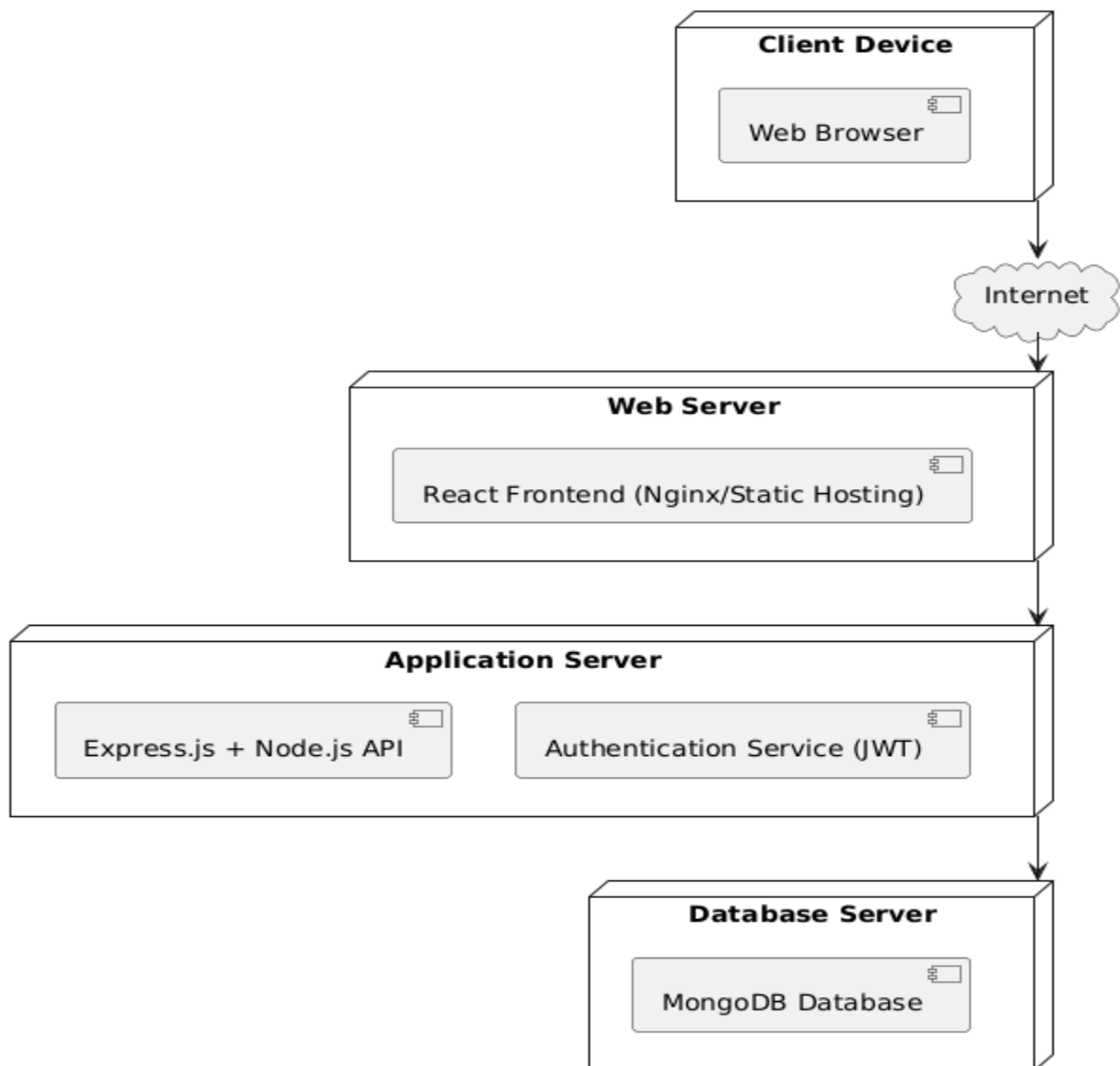
An activity diagram shows the flow of activities or actions in a process, often like a flowchart. For the e-commerce platform, it can represent the steps a customer takes from logging in to placing an order, including decision points like payment success or failure. It helps map out the logic and workflow of system operations.



**Fig. 4.2.5: Activity Diagram**

#### 4.2.6 DEPLOYMENT DIAGRAM

A deployment diagram shows the physical architecture of a system, including hardware nodes (like servers or client devices) and how software components are deployed on them. For an e-commerce clothing platform, it can show the user's browser or mobile app connecting to a web server (Node.js and Express.js), which communicates with a MongoDB database server, all possibly hosted on a cloud platform. It helps visualize the system's infrastructure and deployment setup.



**Fig. 4.2.6: Deployment Diagram**

## 5. IMPLEMENTATION

The Urban Nest e-commerce platform has been implemented using the MERN stack (MongoDB, Express.js, React.js, and Node.js), providing a secure, scalable, and responsive environment for online clothing shopping. The system is divided into two primary components: the Frontend (client-side) and the Backend (server-side). These components communicate seamlessly through RESTful APIs, and security is ensured using JWT-based authentication for user login, registration, and role-based access control.

The backend of Urban Nest is developed with Node.js and Express.js, serving as the central hub for all business logic and database interactions. The backend connects to a MongoDB database, which is designed with separate collections for users, products, carts, and orders. The Users collection stores account details including usernames, hashed passwords, and roles (customer or admin). The Products collection holds all product-related data such as name, price, category, stock availability, description, and associated images. The Cart and Orders collections are responsible for tracking users' selected items and purchase history, enabling a smooth order management process. Authentication and authorization are implemented using JSON Web Tokens (JWT). During login, users receive a JWT token that is stored on the client side and sent with each subsequent request to verify identity and permissions. Passwords are encrypted using bcrypt to enhance security. Role-based access control ensures that administrative actions, such as adding, updating, or removing products, are restricted to admin users only, while regular customers can browse, add items to the cart, and place orders.

The frontend of Urban Nest is built using React.js, which provides a dynamic and responsive user interface. The application allows users to browse the product catalog, view detailed product descriptions, and add items to their shopping cart. Customers can securely register and log in, after which they can manage their cart and place orders. For administrators, additional features such as product management (adding, editing, and deleting products) are integrated into the frontend, with access restricted via the role-based authentication system. Communication between the frontend and backend is handled through RESTful APIs, ensuring smooth interaction for operations such as fetching products, processing orders, and managing carts. The APIs are designed to be modular and scalable, making the system easy to extend in the future. Overall, the MERN implementation of Urban Nest ensures a seamless, secure, and engaging shopping experience for users, with clear separation of concerns between client-side and server-side logic.

## 5.1 TECHNOLOGIES USED

The following technologies were used to develop the e-commerce clothing web application:

- **Frontend:**

- **React.js** – for building the user interface
- **Axios** – for making HTTP requests
- **React Router** – for navigation
- **CSS3 / Tailwind CSS / Bootstrap** – for styling

- **Backend:**

- **Node.js** – as the runtime environment
- **Express.js** – to build RESTful APIs
- **JSON Web Token (JWT)** – for user authentication and route protection
- **bcrypt.js** – for password hashing

- **Database:**

- **MongoDB Atlas** – cloud-hosted NoSQL database
- **Mongoose** – for modeling and interacting with MongoDB collections

- **Tools:**

- **Postman** – to test APIs
- **Git & GitHub** – for version control
- **VS Code** – for development

- **Axios:** A promise-based HTTP client for making API requests in JavaScript applications
- **Tailwind CSS:** A utility-first CSS framework for rapidly building custom user interfaces
- **Bootstrap:** A popular CSS framework for developing responsive and mobile-first websites
- **Bcrypt.js:** A JavaScript library used to hash and compare passwords securely
- **MongoDB Atlas:** A cloud-based database service for hosting and managing MongoDB databases
- **Postman:** A tool for testing and debugging APIs by sending HTTP requests and viewing responses



## 5.2 MODULES

- **Client-Server Separation:**

The platform adopts a clear client-server architecture, with the React frontend managing the user interface and the Express backend responsible for handling business logic and database operations. This separation enhances modularity, simplifies maintenance, and allows independent development of frontend and backend components.

- **RESTful API Design:**

The backend APIs follow REST principles, using standard HTTP methods for operations. Key endpoints include GET /products for retrieving product data, POST /products for adding new products, PUT /products/:id for updating existing products, and DELETE /products/:id for product removal. This design ensures consistent, scalable, and predictable communication between the client and server.

- **Schema Modeling:**

Data structure and validation are handled through Mongoose schema modeling. Collections such as products and users are defined with specific fields, data types, and validation rules, ensuring data consistency and integrity in MongoDB. This approach enforces a structured format for all stored information.

- **Middleware Usage:**

The system incorporates essential middlewares like `express.json()` for parsing JSON request bodies and `cors()` for enabling cross-origin requests. Custom middleware is also implemented for error handling, enhancing security and simplifying API request processing. This setup ensures smooth request-response cycles across the application.

- **CRUD Operations:**

The application implements full CRUD functionality using `async/await` syntax for asynchronous operations. All database interactions are managed through dedicated controller functions in the backend, ensuring code clarity, separation of concerns, and efficient handling of product and user data.

- **Frontend Integration:**

On the frontend, React components interact with the backend through API calls made using `Axios`. Application state is managed with `useState` and `useEffect` hooks, while navigation and routing are handled via `React Router`. This setup ensures a smooth, responsive, and user-friendly experience.

## 5.3 SAMPLE CODE

The frontend and backend of the Online Shopping Website work together to deliver secure, dynamic, and user-friendly functionality. The frontend is built using React.js for interactive UI components, with Axios used to fetch and send data to the backend. The backend, built with Express.js and MongoDB, manages user data, product details, and cart functionality. JWT-based authentication secures all protected operations, such as managing carts and placing orders.

### 5.3.1 User Registration (React + Express + MongoDB)

The user registration module enables new customers to create an account on the platform by providing basic details such as username, email, and password. The data is securely stored in MongoDB, with passwords encrypted using hashing techniques for enhanced security. Once registered, users receive authenticated access via a JSON Web Token (JWT), allowing them to log in, browse products, add items to their cart, and place orders.

```
const signup = async () => {

  let dataObj;

  await fetch('http://localhost:4000/signup', {

    method: 'POST',

    headers: {

      Accept: 'application/form-data',

      'Content-Type': 'application/json',

    },

    body: JSON.stringify(formData),

  })

  .then((resp) => resp.json())

  .then((data) => {
```

```

    dataObj = data;

  });

  if (dataObj.success) {

    localStorage.setItem('auth-token', dataObj.token);

    window.location.replace("/");

  } else {

    alert(dataObj.errors);

  }

}

```

### 5.3.2 Product Cart Component (React State + Express API)

The product cart component provides customers with the ability to add, view, update, and remove items. It uses React state to track selected products and quantities dynamically, while the total price is recalculated in real time. For logged-in users, the cart is synced with the backend via REST APIs, ensuring persistence across sessions and devices. Guest users can maintain a session-based cart using local storage. This component acts as a bridge between product browsing and checkout, delivering seamless shopping flow.

```

const Cart = ({ items }) => {

  const total = items.reduce((sum, item) => sum + item.price * item.quantity, 0);

  return (

    <div>

      {items.map(item => (

        <div key={item.id}>

          <p>{item.name} - ${item.price} x {item.quantity}</p>

        </div>

```

```
    )}]
```

```
<h3>Total: ${total}</h3>
```

```
</div>
```

```
);
```

```
};
```

### 5.3.3 MongoDB Product Schema (Mongoose)

The product schema defines the structure for storing product information in MongoDB, ensuring efficient retrieval and management of product data across the application. Each product document contains key fields such as product name, description, category, price, stock quantity, images, and timestamps for creation and updates. The schema is designed to support fast queries, scalability, and seamless integration with the frontend for displaying product details, managing inventory, and enabling cart and checkout operations.

```
const mongoose = require('mongoose');
const productSchema = new mongoose.Schema({
  name: { type: String, required: true },
  description: { type: String },
  category: { type: String, required: true },
  price: { type: Number, required: true },
  stock: { type: Number, required: true },
  images: [{ type: String }], // URLs for product images
}, { timestamps: true }); // Adds createdAt & updatedAt fields
module.exports = mongoose.model('Product', productSchema);
```

### 5.3.4 JWT Verification Middleware (Express.js)

The JWT verification middleware is responsible for securing protected routes by validating JSON Web Tokens (JWT) attached to incoming requests. It authenticates users by decoding the token, verifying its signature, and extracting the user ID and role before granting access to specific API endpoints. This middleware ensures that only authenticated and authorized users can perform actions such as viewing orders, managing the cart, or accessing admin-level functionalities, thereby enhancing the overall security of the application.

```

const jwt = require('jsonwebtoken');

const verifyToken = (req, res, next) => {
  const token = req.headers['authorization'];
  if (!token) return res.status(401).json({ message: 'Access Denied' });

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.user = decoded; // Attach user info to request
    next();
  } catch (err) {
    res.status(403).json({ message: 'Invalid or Expired Token' });
  }
};

module.exports = verifyToken;

```

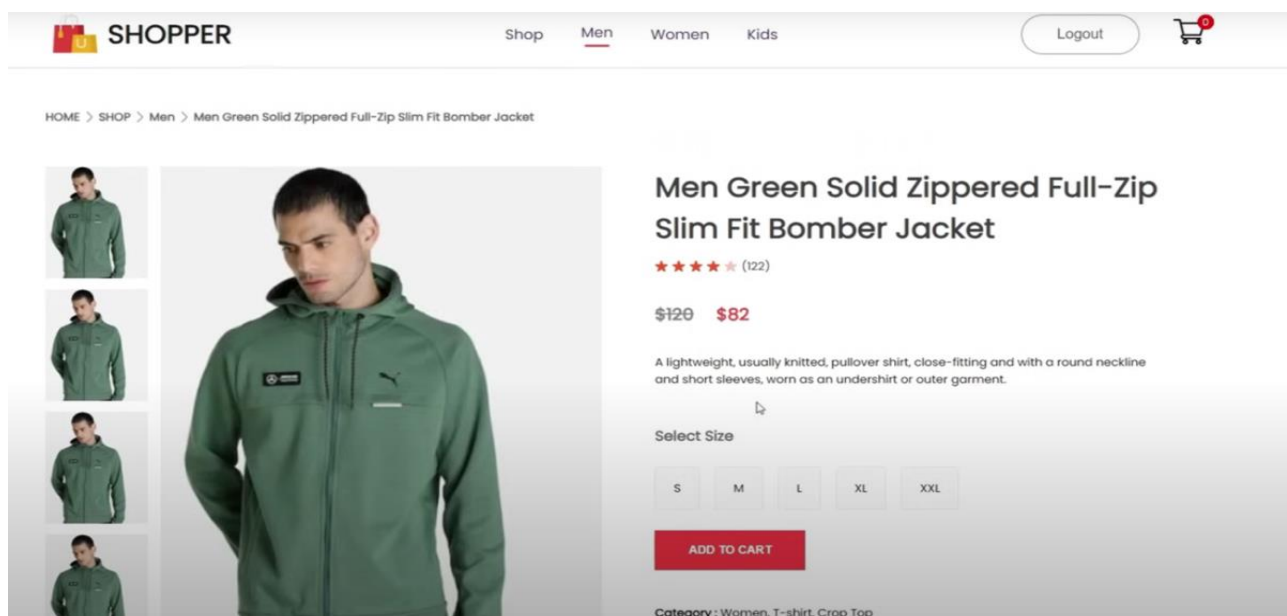
**Table 5.3: Module-Wise Implementation Summary**

Module	Tool/Libraries Used	Description
Authentication	JWT, Bcrypt, React, Axios	Secure login/signup with session persistence.
Product Management	Express.js, MongoDB, Mongoose	Store and manage product details and inventory.
Shopping Cart	React State, Axios, Express.js, MongoDB	Add, update, and sync cart items across sessions.
Order & Checkout	Express.js, MongoDB	Handle payments, order creation, and stock updates.
Frontend UI	React.js, Tailwind CSS, React Router DOM	Responsive interface with smooth navigation.
Admin Dashboard	Express.js, MongoDB, JWT Role-Based Access	Admins manage products, orders, and sales securely.

## 6. RESULTS

The Urban Nest e-commerce platform was thoroughly tested with real-time input data from customers, administrators, and backend services. Inputs were primarily received through the React-based frontend, where customers created accounts, browsed products, and managed their shopping carts. Product details, including names, categories, images, prices, and stock levels, were managed via the admin dashboard and stored in MongoDB through Express.js API endpoints. Axios handled all data communication between the frontend and backend, ensuring smooth synchronization. For payment processing, test credentials and tokens were utilized with Stripe's sandbox environment to simulate real purchase transactions. JWT-authenticated sessions ensured secure user logins, role-based access for admins, and protected data flow between all system components. Additionally, cart updates, order creation, and inventory adjustments were rigorously tested to validate the end-to-end functionality of the platform.

### 6.1 PRODUCT PAGE

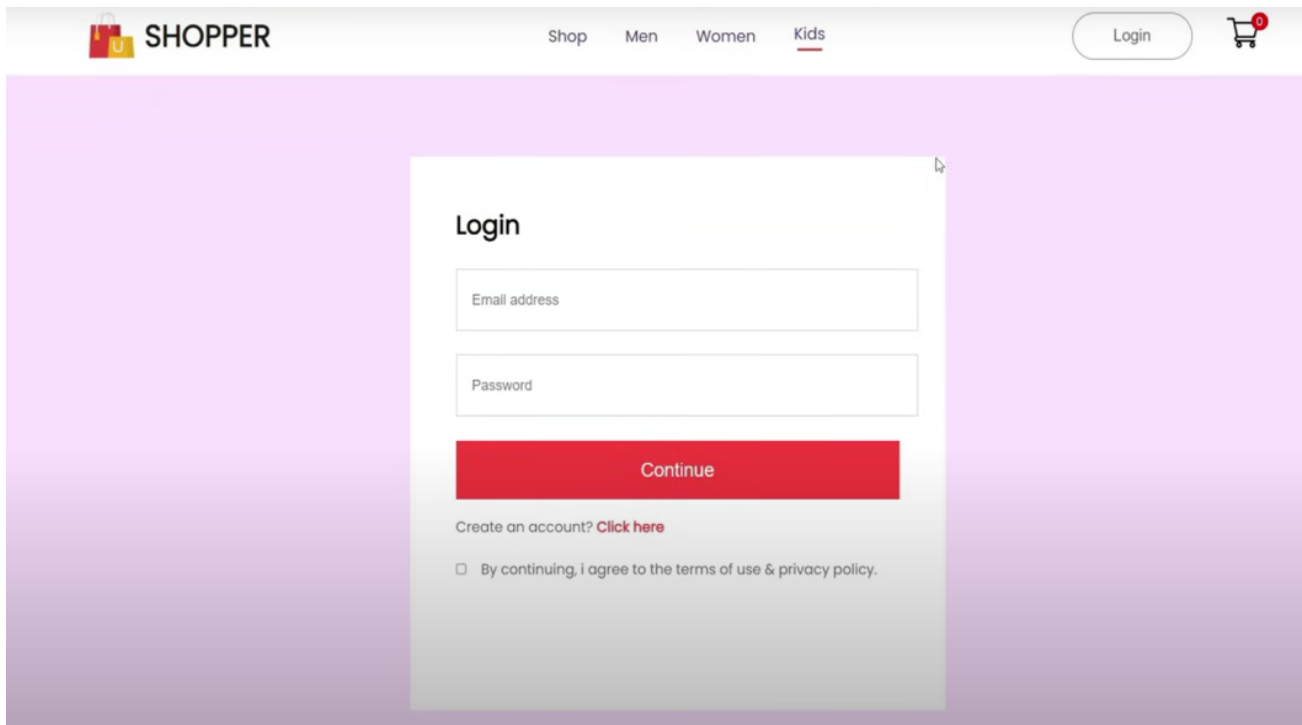


**Fig. 6.1: Product Page**

The product page provides a comprehensive view of each clothing item, presenting high-quality images, detailed descriptions, pricing, category classification, and real-time stock status. Data for the page is dynamically fetched from the backend via RESTful API calls, ensuring that information such as availability, price changes, and stock updates is always current. Interactive features allow users to seamlessly add products to their **cart** for purchase or to their wishlist for future reference, with instant

feedback provided through UI updates. The page layout is designed to be responsive, ensuring optimal viewing and interaction on both desktop and mobile devices. Additionally, related products or recommendations can be displayed to enhance user engagement and encourage further browsing.

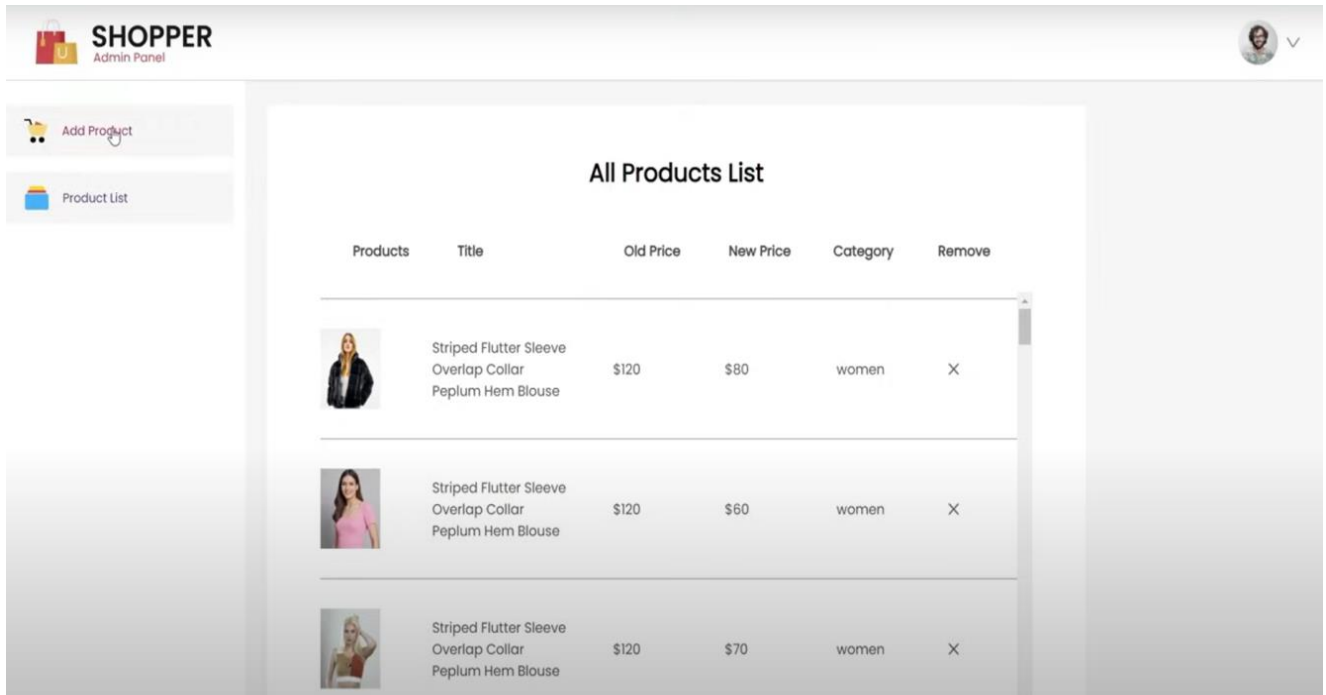
## 6.2 LOGIN PAGE



**Fig. 6.2: Login Page**

The login page serves as the secure entry point for registered users to access their accounts. Users are prompted to enter their registered email address and password, which are transmitted securely to the backend via HTTPS. On the backend, the system verifies the credentials against encrypted passwords stored in the MongoDB database using bcrypt for hashing. If authentication is successful, a JWT (JSON Web Token) is generated containing the user's unique ID and role, with an expiration time for security. This token is sent back to the client and stored securely (e.g., in HTTP-only cookies or local storage) for subsequent requests. The presence of this token allows the user to maintain a session without re-entering credentials, granting access to personalized features such as managing their cart, tracking and viewing order history, and updating profile details. Any invalid credentials or expired tokens trigger secure error responses, ensuring the integrity and confidentiality of user data.

## 6.3 CART PAGE




**Fig.6.3: Cart Page**

The cart page serves as a centralized space where users can view and manage all products they have added for purchase. Each item is displayed with its name, thumbnail image, selected size or variant (if applicable), quantity, individual price, and subtotal, along with the real-time calculated total cost of all items. Users can conveniently update product quantities, remove items, or clear the entire cart with immediate visual feedback. For logged-in users, the cart is dynamically synced with the backend, ensuring persistence across devices and sessions through database storage. Any changes made on the cart page trigger instant updates via API calls, keeping the UI and backend perfectly aligned. A proceed to checkout option is prominently displayed, guiding users toward the payment and order confirmation process. The page is fully responsive, offering a seamless experience across desktop, tablet, and mobile devices.



## 6.4 CHECKOUT PAGE

The screenshot displays a web application's checkout page. At the top, a navigation bar includes a 'SHOPPER' logo, category links (Shop, Men, Women, Kids), a 'Logout' button, and a shopping cart icon with a red notification badge. Below the navigation bar, a table lists the items in the cart. The table has columns for 'Products', 'Title', 'Price', 'Quantity', 'Total', and 'Remove'. One item is listed: 'Men Green Solid Zippered Full-Zip Slim Fit Bomber Jacket' with a price of \$82 and a quantity of 1. Below the cart table, the 'Cart Totals' section shows a subtotal of \$82, a shipping fee of 'Free', and a total of \$82. To the right of the totals, there is a field for a promo code and a 'Submit' button. At the bottom left, a red button labeled 'PROCEED TO CHECKOUT' is visible.

Products	Title	Price	Quantity	Total	Remove
	Men Green Solid Zippered Full-Zip Slim Fit Bomber Jacket	\$82	1	\$82	×

**Cart Totals**

Subtotal	\$82
Shipping Fee	Free
<b>Total</b>	<b>\$82</b>

**PROCEED TO CHECKOUT**

If you have a promo code, Enter it here

**Submit**

**Fig. 6.4: Checkout Page**

The checkout page is the final step in the purchasing process, allowing users to review their complete cart summary, including product details, quantities, prices, and the final total amount payable. Users can then enter or select their shipping address, choose delivery options. Before order processing, the system communicates with the backend to validate stock availability in real time, ensuring that all items are still in stock. Once the order placement is successfully authorized, the order is automatically created in the backend, storing details such as order items, shipping information, and timestamps in the database for future tracking. A confirmation page is then displayed to the user, and a confirmation email or receipt is sent for their records. The page is fully optimized for security, usability, and mobile responsiveness, ensuring a smooth and trustworthy checkout experience.

## 7. CONCLUSION

This project successfully demonstrates the development of an e-commerce clothing shopping platform using the MERN stack. The application allows users to browse products, manage a shopping cart, and place orders, while admin users can add, update, and delete products. It provides a secure, responsive, and user-friendly experience across devices. React.js was used to build a dynamic frontend, while Node.js and Express.js handled backend logic and API routing. MongoDB ensured flexible and scalable data storage. The use of JWT for authentication adds a layer of security to protect user data. Overall, the project meets the goals of building a functional, scalable, and maintainable full-stack web application. It also provided hands-on experience in using modern technologies and design principles for real-world applications.

### 7.1 PROJECT FINDINGS

- The MERN stack proved effective for full-stack development with a consistent JavaScript-based workflow.
- React.js enabled fast and interactive user interfaces with reusable components.
- Node.js and Express.js efficiently handled API requests, routing, and middleware integration.
- MongoDB offered flexible and scalable data storage for user and product data.
- The implemented features covered all core e-commerce functionalities such as authentication, product listing, cart operations, and order processing.
- Admin-side functionality was successfully implemented for inventory and product management.
- The application met key goals for security, usability, and maintainability.

## 7.2 FUTURE SCOPE

The Urban Nest e-commerce platform has been designed as a scalable and extensible system that can evolve with changing business and user needs. While the current implementation provides core functionalities like user registration, product browsing, cart management, and secure order processing, there is significant potential to enhance the platform with advanced features, integrations, and optimizations. The following points outline the key areas of future development:

- **Mobile Application Development:** Launching Android and iOS apps to provide a more accessible and seamless shopping experience on mobile devices.
- **AI-Powered Product Recommendations:** Integrating machine learning models to suggest personalized products based on user browsing and purchase history.
- **Advanced Search and Filtering:** Implementing AI-driven or voice-enabled search, along with more granular filters like price range sliders and category tags.
- **Multi-Vendor Marketplace Expansion:** Allowing third-party sellers to list and manage products, turning Urban Nest into a marketplace platform.
- **Improved Analytics Dashboard:** Offering admins advanced sales analytics, inventory predictions, and customer behavior insights.
- **Enhanced Security Measures:** Adding two-factor authentication, fraud detection, and more robust data encryption techniques.

## REFERENCES

- [1] Patil, R., et al. (2025). Harnessing the MERN stack for scalable e-commerce website design: A full-stack approach with MongoDB, Node.js, Express.js, and React.js. *Journal of Innovations in Science & Technology (JIST)*.
- [2] De Silva, P., et al. (2022). Efficiency of an e-commerce web application with MERN stack and modern tools. *International Journal of Engineering and Management Research (IJEMR)*.
- [3] Verma, A., & Kale, S. (2025). Revolutionizing online fashion: A MERN stack-based e-commerce platform. *International Journal of Scientific Research in Engineering and Management (IJSREM)*.
- [4] Huang, Z., & Benyoucef, M. (2018). Social commerce: A systematic review and data synthesis. *Decision Support Systems, Elsevier*.
- [5] Esmaeili, A., & Hashemi, S. (2019). A systematic review on social commerce. *Journal of Strategic Marketing, Taylor & Francis*.
- [6] Fernando, D. E., et al. (2022). Computerized system to manage business functions of e-commerce web application using MERN stack technology. *International Journal of Engineering and Management Research (IJEMR)*.
- [7] Rafi, A. H. (2024). MERN stack-based multi-seller e-commerce site. *Global Journal of Computer Science and Technology (GJCST)*.
- [8] Budhe, P., et al. (2023). E-commerce with auction – Web application using MERN technology. *SSGM Journal of Science and Engineering*.
- [9] Linga, N., et al. (2024). E-commerce product showcase using MERN stack. *Advancements in Communication and Systems*.
- [10] Shrinidhi, N., et al. (2023). Crafting a digital presence: An in-depth exploration of MERN stack solutions for Indian artisans. *International Journal of Innovative Science and Research Technology (IJSRT)*.