Naan Mudhalvan Project

MONGODB With MERN STACK

Project Title: Grocery Web App

Submitted By :

DIVAYANSHU TIWARI - 2124212430012
YOGESH KUMARAN – 212421243039
BHARANIDHARAN - 212421243009
DELLI GANESH - 212421243010

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

SREE SASTHA INSTITUTE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY CHENNAI – 600025

NOV/DEC 2024

BONAFIDE CERTIFICATE

Certified that this report titled "Grocery Web App" for the Naan mudhalvan project is a Bonafide work of (DIVAYANSHU TIWARI, YOGESHKUMARAN, BHARANIDHARAN, DELLI GANESH) in MERN stack by Mongo DB -NM1042 who carried out the work under my supervision.

Certified further that to the best of my knowledge, the work reported here does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

FACULTY MENTOR	**HEAD OF DEPARTMENT**	**SPOC**

Submitted for the University Practical Examination held on ________________

Internal Examiner **External Examiner**

Index

Abstract

This project presents an advanced e-commerce web application developed using the MERN stack (MongoDB, Express.js, React, Node.js). The application aims to provide a seamless and efficient online shopping experience for users, featuring a wide range of functionalities that cater to both customers and administrators. Key features include secure user authentication and authorization with JWT, role-based access control, and a comprehensive product catalog with detailed descriptions, pricing, and images. The application also offers advanced product filtering and search capabilities, a user-friendly shopping cart system, and a secure checkout process with payment gateway integration. Additionally, it includes real-time order tracking and history for users, an admin panel for managing orders, and integration with Cloudinary for efficient image storage. The backend is hosted on Render, while the frontend is deployed on Netlify, ensuring robust performance and scalability. MongoDB Atlas is utilized for scalable and secure database management. This project is licensed under the MIT License, highlighting its open- source nature and encouraging collaboration. Overall, the advanced e-commerce web application showcases a full-stack development approach, integrating modern technologies to deliver a robust, scalable, and user-friendly platform, demonstrating the developer’s proficiency in handling complex web applications and creating a seamless online shopping experience.

CHAPTER 1. Project Overview

EKO-MART is a MERN-based e-commerce web application designed to streamline the online shopping experience for both customers and administrators. This platform allows customers to browse product listings, filter items based on various criteria, and make purchases effortlessly. On the other hand, administrators can manage product listings, handle customer inquiries, and track orders efficiently.

The app features an admin panel to ensure a smooth user experience, offering oversight and governance over user activities and platform security. Admins can validate user accounts, monitor and approve product listings, and resolve any issues to maintain a high-quality and secure platform for everyone involved.

The main goal of **EKO-MART** is to create a centralized space where customers can easily find and purchase products, and administrators can efficiently manage their inventory and interactions with customers. By focusing on simplifying the shopping process and incorporating essential features like real-time updates, advanced filtering, and secure user authentication, the app enhances the shopping experience for all users.

EKO-MART also aims to contribute to the growth of the online retail market by providing a modern and easy-to-use solution for product management and online transactions.

Objectives

The primary objectives of the Grocery Web App are to:

1. Streamline Online Shopping: Provide a user-friendly platform for customers to browse, filter,and purchase products effortlessly.
2. Efficient Product Management: Enable administrators to manage product listings, handle customer inquiries, and track orders efficiently.
3. Admin Panel Oversight: Ensure smooth user experience and platform security through an admin panel that allows for account validation, listing approvals, and issue resolution.
4. Centralized Shopping Space: Create a centralized space where customers can easily find and purchase products, and administrators can manage inventory and customer interactions.
5. Real-Time Updates: Incorporate real-time updates to keep product listings and order statuses current.

CHAPTER 2. Technology Stack & System Requirements

The **EKO-MART** is built using the MERN stack, a powerful combination of technologies that enable the development of dynamic, scalable web applications. The core components of the stack include:

- **MongoDB**: A NoSQL database used for storing product information, user data, orders, and other relevant details.
- **Express.js**: A flexible backend framework that powers the API and handles server-side logic for the application.
- **React.js**: A frontend framework used to build a dynamic, responsive user interface for seamless user interaction.
- **Node.js**: A runtime environment that executes backend JavaScript and handles server requests efficiently.

In addition to the MERN stack, several other technologies are utilized to enhancethe user experience and application security:

- **Bootstrap, Material UI, and Ant Design**: CSS frameworks used to design responsive and aesthetically pleasing user interfaces.
- **JSON Web Token (JWT)**: For secure user authentication and session management, ensuring that user data is protected.
- **Cloudinary**: For efficient image storage and management.

System Requirements:

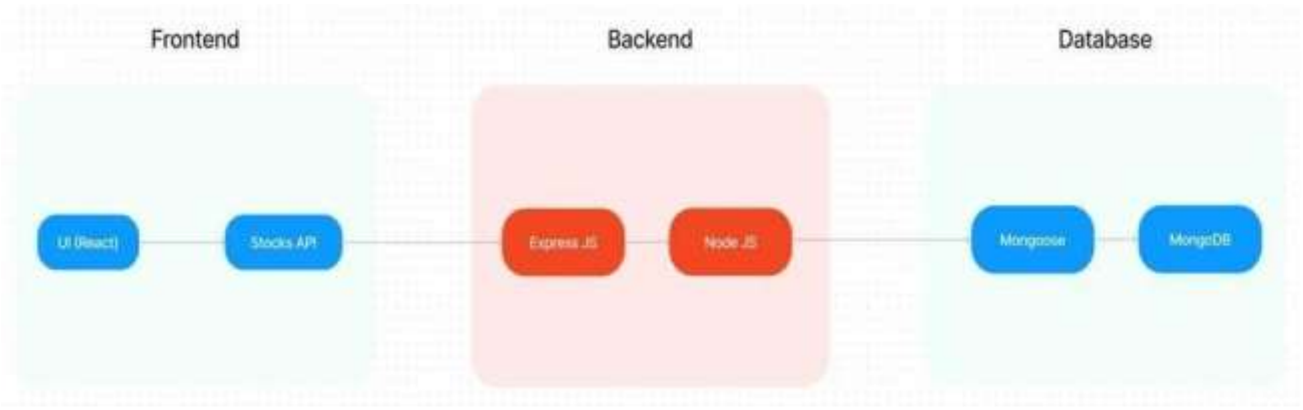
To run the Grocery Web App, the following hardware and software are required:

- **Hardware**:
  - Windows 8 or higher machine with a stable internet connection (30 Mbps recommended).
- **Software**:
  - **Node.js**: The latest version of Node.js should be installed.
  - **MongoDB Community Server**: Required for database management.
  - Two web browsers (for testing purposes) are recommended for cross-browser compatibility.

CHAPTER 3. Project Architecture

The application follows a modular architecture, split between frontend (client-side) and backend (server-side) components:

1. **Frontend (Client-Side)**:
   - Built with React.js for dynamic rendering, creating an intuitive and interactive experience for users.
   - CSS frameworks like Bootstrap, Material UI, and Ant Design are used tostyle components, ensuring consistency in design.
   - Communicates with backend services using API endpoints, enabling real-time data handling and smooth user interactions.
2. **Backend (Server-Side)**:
   - Express.js handles the core API and server functions, managing routes and ensuring that each request is processed efficiently.
   - MongoDB serves as the database, storing information related to users, properties, and booking transactions.
   - JWT tokens secure user authentication and authorization, preventing unauthorized access.
3. **Database (MongoDB)**:
   - MongoDB collections store all app-related data, including user profiles,property details, inquiries, and bookings.
   - The database is designed for optimal retrieval, supporting fast queries and handling real-time updates efficiently.



CHAPTER 4. Installation and Setup

Prerequisites

1. Install **Node.js** and **npm**.
2. Install **MongoDB Community Server**.

Step-by-Step Setup

1. **Clone the Repository**

   https://github.com/DivayanshuTiwari2912/Naan-Mudhalvan-Grocery-Web-Application.git

2. **Backend Setup**:
   a. Navigate to the backend folder and install dependencies:
      - cd../Grocery Web App /backend
      - npm install

   b. Create an .env file with MongoDB connection and JWT key.

   c. Modify the MongoDB connection string in connect.js in config folder

   d. Start the backend server:
      - npm start

3. **Frontend Setup**:
   a. Navigate to the frontend folder and install dependencies:
      - cd../Grocery Web App/frontend
      - npm install

   b. Start the frontend server:
      - npm run dev

4. **Access the App**:
   a. Frontend: http://localhost:3000

   b. Backend: http://localhost:3000

CHAPTER 5. FOLDER STRUCTURE

The folder structure of the Grocery Web App is organized to separate frontendandbackend components, allowing for a modular and maintainable codebase.

Grocery Web app/

frontend/

```
├── public/
│   └── index.html
├── src/
│   ├── components/
│   │   ├── Header.js
│   │   ├── Footer.js
│   │   ├── ProductList.js
│   │   └── Cart.js
│   ├── pages/
│   │   ├── HomePage.js
│   │   ├── ProductPage.js
│   │   └── CheckoutPage.js
│   ├── context/
│   │   └── AuthContext.js
│   ├── services/
│   │   └── api.js
│   ├── App.js
│   ├── index.js
│   └── package.json
```

```
con backend/
├── config/
│   └── db.js
├── controllers/
│   ├── authController.js
│   ├── productController.js
│   └── orderController.js
├── models/
│   ├── User.js
│   ├── Product.js
│   └── Order.js
├── routes/
│   ├── authRoutes.js
│   ├── productRoutes.js
│   └── orderRoutes.js
├── middleware/
│   └── authMiddleware.js
├── utils/
│   └── errorHandler.js
├── .env
├── server.js
└── package.json
└──  README.md              # Documentation and instructions for the project
```

CHAPTER 6. Workflow and Usage

User Roles and Functionalities

The **EKO-MART** includes three primary user roles: Customer, Administrator, and Vendor. Each role has specific responsibilities and functionalities to ensure smooth operation of the platform.

1. **Customer**:
   - **Account Creation and Login**: Customers create an account and log in using their email and password.
   - **Dashboard Access**: Upon login, they are shown a list of available productsin their dashboard.
   - **Product Details**: By clicking on a product, customers can view detailed information about the product and its vendor. They can add products to their cart and proceed to checkout.
   - **Order Status**: Customers can check the status of their orders in the “Orders” section, where the status will initially be displayed as “pending.” The vendor will later update the status (shipped, delivered, etc.).
2. **Vendor**:
   - **Admin Approval**: Vendors must first receive approval from the admin to create a vendor account.
   - **CRUD Operations for Products**: Once approved, vendors can perform all CRUD (Create, Read, Update, Delete) operations for their product listings. They can add, edit, or remove products as needed.
   - **Order Management**: Vendors can update the status of orders (e.g., processing, shipped, delivered) to reflect current conditions.
3. **Administrator**:
   - **Vendor Approval**: Admins review and approve legitimate users as vendors. Only after approval can a vendor add and manage products.
   - **User Monitoring**: Admins monitor all user activities on the platform, ensuring compliance with platform rules.
   - **Enforcing Policies**: Admins implement and enforce platform policies, terms of service, and privacy regulations to ensure a secure and trustworthy environment for all users.

CHAPTER 7. Testing

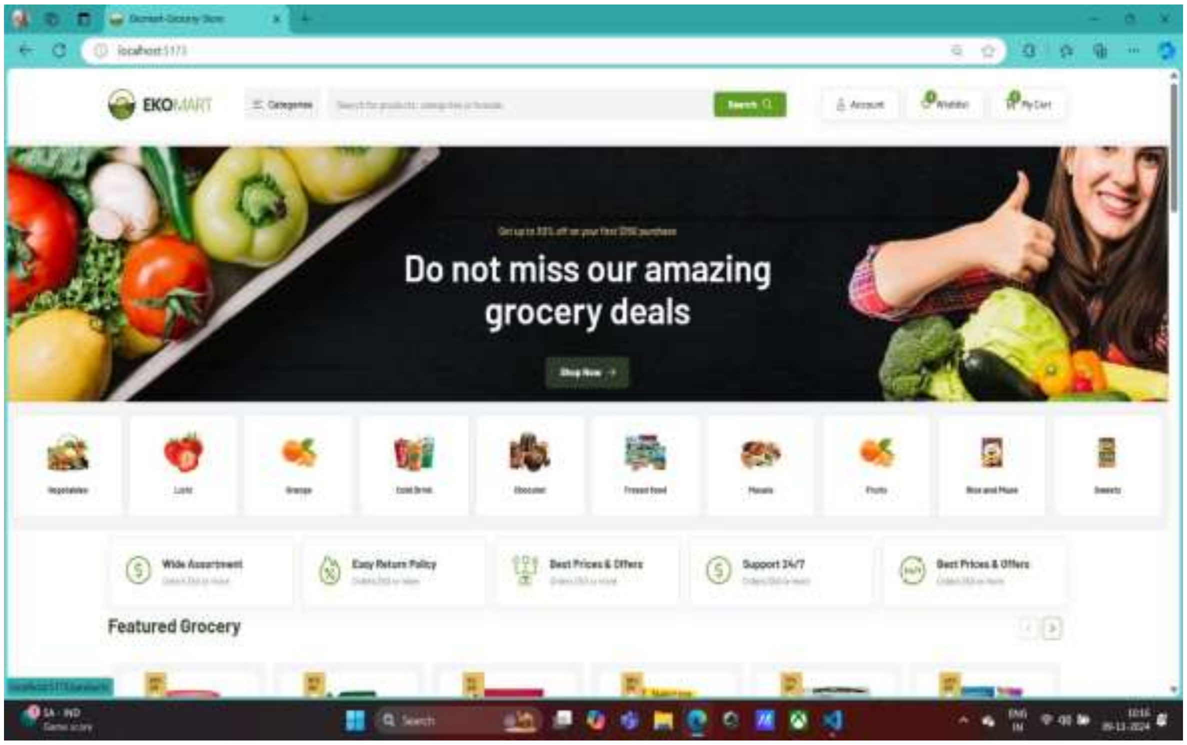
Manual testing was conducted to ensure that **EKO-MART** functions as expected and provides aseamless experience for users. The focus of manual testing included:

1. **User Registration and Login**:
   - Tested the registration and login functionality for both customers and vendors,ensuring that the correct validation messages appear for invalid inputs (e.g., empty fields, incorrect credentials).
   - Verified that only authorized users (with admin approval) could register asvendors.
2. **Product Listings**:
   - Verified that vendors could add new product listings, edit existing ones, anddelete products successfully.
   - Checked if product details such as name, price, category, and availability werecorrectly displayed on the frontend.
3. **Search and Filter Functionality**:
   - Tested the search and filter options to ensure that products could be filtered byvarious criteria like category, price range, and availability.
   - Ensured that applying filters resulted in relevant product listings being displayed.
4. **Order and Cart System**:
   - Tested the cart and order system by adding products to the cart, proceeding tocheckout, and confirming that orders were placed successfully.
   - Verified that the order status correctly changes from “pending” to “shipped” or“delivered” by the vendor.
5. **Admin Functionality**:
   - Checked if the admin could approve vendor accounts and ensure that onlyauthorized vendors could add or manage product listings.
   - Verified the admin’s ability to monitor user activities and review vendorregistrations.
6. **Responsiveness**:
   - Tested the user interface on various devices (desktop, tablet, mobile) to ensurethat the layout adapts appropriately to different screen sizes and that key functionalities remain accessible.
7. **Error Handling**:
   - Tested invalid inputs and error scenarios (e.g., incorrect product details, unauthorized access) to ensure that the system provides appropriate errormessages and does not crash or behave unexpectedly.
8. **Real-Time Updates**:
   - Verified that product availability and order statuses were updated in real-time andreflected correctly on both the customer’s and vendor’s dashboards.

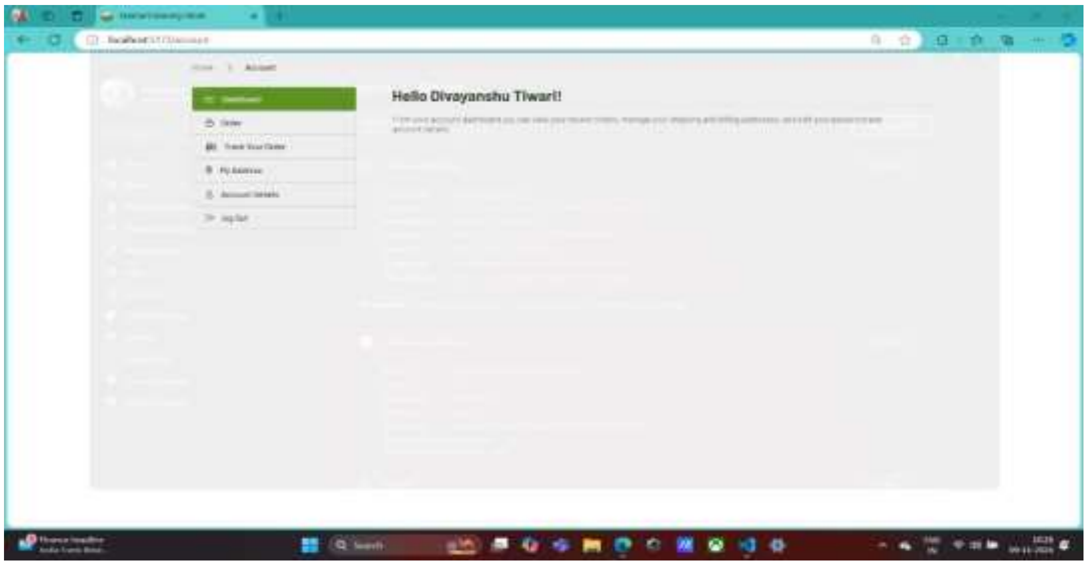
CHAPTER 8. Project Implementation & Execution

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

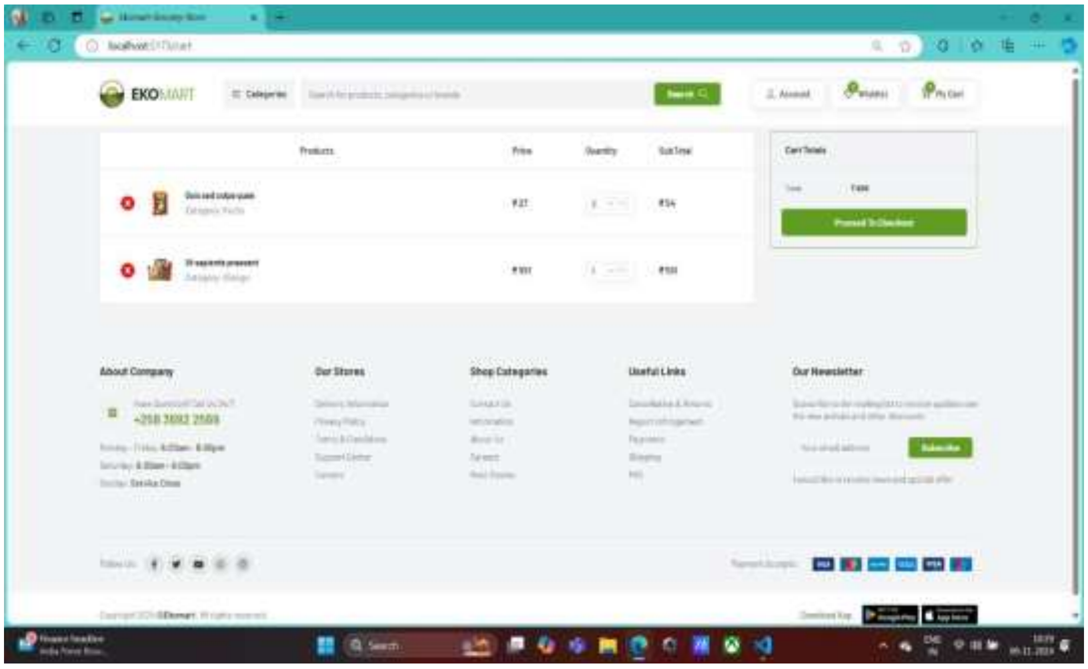
- **Opening page:**



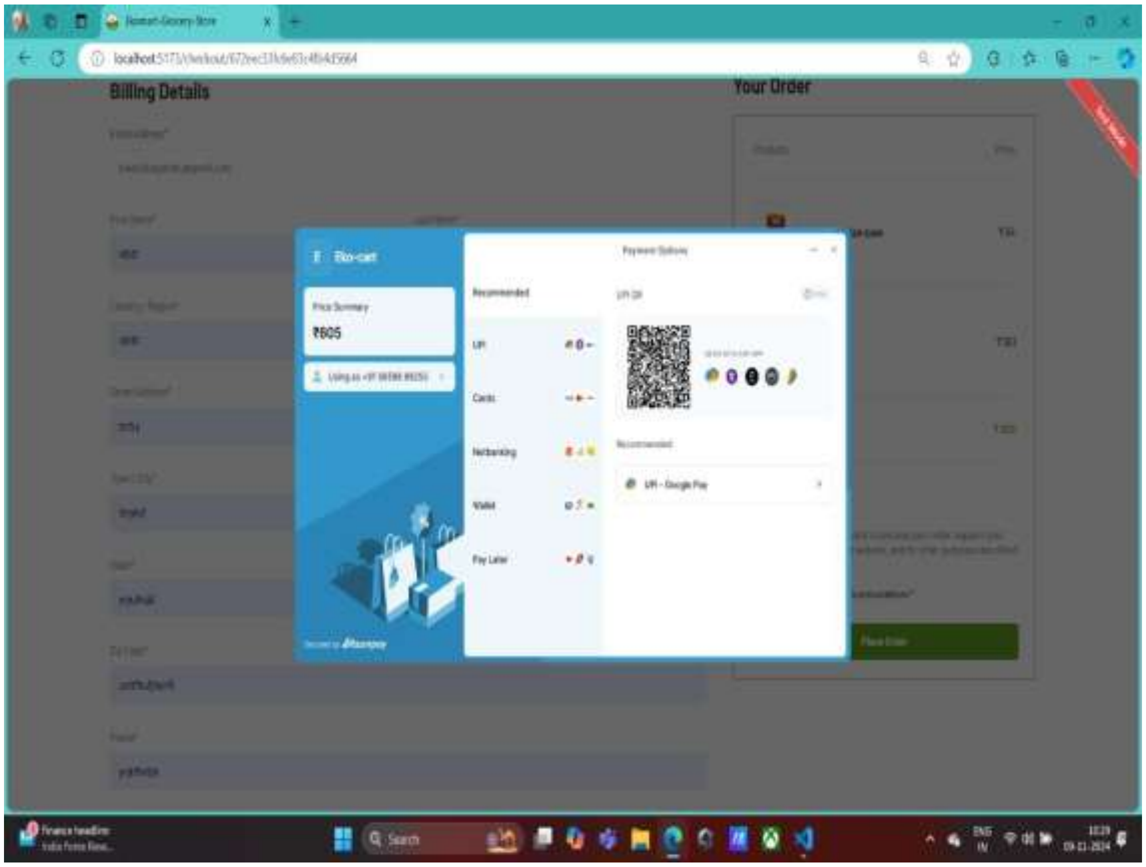
- **Login page:**



- **Cart:**



- **Checkout Page:**



CHAPTER 9. Challenges Faced

1. **User Role Management**:
   - Developing a secure role-based access control system to ensure users only access their permitted features.
   - Handling edge cases where users might attempt to access features meant for other roles, requiring extensive testing and debugging.
   - Implementing seamless transitions between roles (e.g., from Customer to Vendor) without affecting existing user data.

2. **Database Design**:
   - Designing relationships between collections (e.g., Users, Products, and Orders) to minimize redundancy and improve scalability.
   - Balancing schema flexibility with query performance to accommodate varyingproduct details and order statuses.
   - Managing database migrations and updates as the application evolved during development.

3. **Frontend Design**:
   - Creating a responsive design that works efficiently across multiple devices and screen sizes.
   - Ensuring the UI is intuitive and provides clear feedback to users during actions like adding products to the cart or checking order statuses.
   - Handling performance issues when rendering large datasets (e.g., product listings) on the client side.

4. **Backend Optimization**:
   - Building APIs that handle high traffic efficiently, especially for product searchesand order processing.
   - Ensuring proper error handling and debugging for asynchronous operations involving MongoDB queries.
   - Optimizing the backend to support future scalability as more users and products are added.

5. **Testing and Debugging**:
   - Performing end-to-end testing for features like order processing and vendor approvals.
   - Identifying and resolving bugs in edge cases (e.g., simultaneous order placements).

CHAPTER 10. Future Enhancements

1. Mobile Compatibility:
   - Develop a dedicated mobile app for iOS and Android using frameworks like React Native or Flutter to expand accessibility. Optimize the user interface for mobile screens, ensuring easy navigation and usability.
   - Add offline functionality to allow renters and owners to access saved data without an internet connection.
2. Advanced Filtering:
   - Allow renters to search by more specific criteria, such as property size, floor plan, or specific amenities (e.g., pet-friendly properties, parking spaces).
   - Incorporate geolocation-based filtering to show properties near the user's current location. Enable dynamic filtering to refine search results in real-time as users adjust criteria.
3. Payment Integration:
   - Implement secure payment gateways like Stripe or PayPal for renters to pay booking deposits directly through the platform.
   - Add support for multiple currencies to cater to international users. Provide automated invoice generation and payment confirmation emails for completed transactions.
4. In-App Chat:
   - Develop a chat feature with real-time messaging between renters and property owners using WebSocket or libraries like Socket.io. Allow file sharing (e.g., lease agreements, property photos) directly through the chat.
   - Incorporate chat moderation features to filter inappropriate messages and maintain platform quality.
5. Machine Learning Recommendations:
   - Add AI-based recommendations to suggest properties based on user preferences, search history, and location. Use machine learning to predict property demand trends and suggest optimal rental pricing for owners.

CHAPTER 11. Conclusion

EKO-MART successfully bridges the gap between customers and vendors, offering a seamless platform for product listing, browsing, and purchasing. By leveraging the MERN stack, the application ensures high performance, scalability, and user-friendly interactions. With features like secure authentication, real-time updates, and role-based access control, the app provides a robust and secure environment for both users and admins.

Through comprehensive planning, implementation, and testing, the app delivers a streamlined shopping experience that reduces friction for both parties. The inclusion of advanced search filters and communication tools enhances user satisfaction, while the admin dashboard ensures platform integrity and security. The project exemplifies how modern web technologies can be used to solve real-world problems, providing value to users and improving the overall shopping process.

Future enhancements, such as mobile compatibility and payment integration, are expected to further expand the app’s functionality, making it even more accessible and efficient. Overall, **EKO-MART** is a successful implementation that lays the groundworkfor future innovation in e-commerce services.

CHAPTER 12.Reference

1. **React.js Official Documentation**
   - https://reactjs.org/
     Comprehensive documentation for building user interfaces using React.
2. **Node.js Official Documentation**
   - https://nodejs.org/en/docs/
     Detailed guides for building server-side applications with Node.js.
3. **Express.js Guide**
   - https://expressjs.com/
     Documentation for the Express framework, essential for handling backend routing andmiddleware.
4. **MongoDB Manual**
   - https://www.mongodb.com/docs/manual/
     Database design principles and best practices for managing collections and queries.
5. **Mongoose.js Documentation**
   - https://mongoosejs.com/docs/

     ORM for MongoDB to simplify schema creation and CRUD operations.
6. **Material UI**
   - https://mui.com/
     A React component library for creating modern and responsive UIs.
7. **Ant Design**
   - https://ant.design/
     A design system and component library for React-based applications.
8. **Bootstrap**
   - https://getbootstrap.com/
     A popular CSS framework for responsive design and styling.
9. **JSON Web Tokens (JWT)**
   - https://jwt.io/introduction/
     Overview and use cases of JWT for secure authentication.
10. **W3Schools (HTML, CSS, JavaScript Basics)**
    - https://www.w3schools.com/
      Beginner-friendly resources for learning web development basics.