Full Stack Development with MERN

Project Documentation format

1. Introduction

ShopEZ is a full-stack e-commerce application built using the MERN stack (MongoDB, Express, React, Node.js) as part of the Smart Bridge project. This application provides a seamless shopping experience for users, allowing them to browse, filter, and purchase products, as well as manage orders. It includes both user and admin functionalities.

- **Project Title:** ShopEZ Website
- **Team Members**:

Team Members	REGISTRATION NO	Roles
M.S.K. Chaithanya Raj	22481A12G0	Project Planning & Documentation
Shaik Anwar	22481A12F2	Documentation
Naziya Tahseen		Frontend & Backend, Demonstration
Shakeera Shaik		Assisted in Frontend

2. Project Overview

- **Purpose**:
  The primary purpose of ShopEZ is to create a scalable, responsive, and user-centric e-commerce solution that caters to the needs of all stakeholders—customers, sellers, and administrators. By leveraging modern web technologies, ShopEZ aims to bridge the gap between functionality and user experience, ensuring a smooth shopping process, efficient seller operations, and simplified administrative management. The project also serves as a practical implementation of modular development and real-time interactivity within a modern online marketplace.

- **Features:**
  - Register as a new user or log in with existing credentials.
  - Filter products by categories, gender, popularity, price (low to high or high to low).
  - Add products to the cart and proceed to checkout.
  - Provide details such as address, city, pincode, mobile number, notesto place an order.
  - View orders from the user profile page.
  - Admin login page
  - View the total number of orders and their details, including the number of orders placed by each user.
  - Product Management: Add, update, or remove products. Add products with categories, images, brands, etc.
  - View all placed orders, update their status (e.g., Order Placed, In Transit, Delivered), and cancel orders if necessary.

3. Architecture

- **Frontend:**

  The frontend of our e-commerce platform is built using **React.js**, ensuring a dynamic and responsive user interface. With features like product browsing, cart management, and user authentication, it provides a seamless shopping experience. The integration of **Redux Toolkit** allows for efficient state management, while **Tailwind CSS** offers a modern and customizable design.

- **Backend:**

  Our backend leverages **Node.js** and **Express.js** to create a robust and scalable server-side environment. It handles RESTful API endpoints for product management, user authentication, and order processing. Security is enhanced through **JSON Web Tokens (JWT)** for authentication and **bcrypt** for password hashing, ensuring a secure platform for both users and administrators.

- **Database:**

  We utilize **MongoDB**, a NoSQL database, to manage our data efficiently. Its flexible schema design allows for easy storage and retrieval of product details, user information, and order histories. MongoDB's scalability ensures that our platform can handle growth and increasing data volumes effectively.

4. Setup Instructions

- **Prerequisites:**
  - Node.js v16+
  - MongoDB Atlas Account
  - Git
  - Code Editor (VS Code)
- **Installation**
  - Clone the GitHub repository
  - Navigate to client and server folders
  - Install dependencies

```
npm install
```

Client

```
cd client
npm install
```

Server

```
cd server
npm install
```

5. Folder Structure

Client (React Frontend):

- src/assets: Static files (images, icons)
- src/components: UI components (admin, auth, shopping)
- src/config: API URLs and app constants
- src/lib: API utilities and custom hooks
- src/pages: Routes for user/admin flows
- src/store: Redux slices
- App.jsx, main.jsx: Root React components

Server (Node.js Backend):

- controllers: Logic for routes
- models: MongoDB schemas
- routes: API endpoint definitions
- helpers: Utility functions (e.g., token creation)
- server.js: Entry point to connect DB and start the server

6. Running the Application

- Provide commands to start the frontend and backend servers locally.
  - **Frontend:** npm start in the client directory

```
cd client
npm install
npm start
```

  - **Backend**: npm start in the server directory.

```
cd server
npm install
npm run dev
```

7. API Documentation

Endpoint	Method	Description
/api/auth/register	POST	Register a new user
/api/auth/login	POST	Log in user and return JWT token
/api/products	GET	Get list of all products
/api/products/:id	GET	Get details of a single product

/api/cart/add	POST	Add item to the user's cart
/api/cart/remove	DELETE	Remove item from the user's cart
/api/checkout	POST	Initiate payment via paypal
/api/admin/users	GET	Admin-only: View all registered users

8. Authentication

- JWT-based login system
- After successful login, token is stored in localStorage
- Each protected route requires Authorization: Bearer <token> header
- Middleware checks user roles for admin, seller, or customer
- Passwords stored as bcrypt hashes.

9. User Interface

The UI is designed to offer ease of navigation and accessibility:

- **Navbar** – Dynamic based on user login status
- **Homepage** – Lists all clothes by category
- **Search Bar** – Filter by name
- **Cart** – Preview, remove items, add items
- **Checkout** – Address input, order summary, paypal popup
- **Seller Dashboard** – Add/edit/delete products
- **Admin Panel** – View users, monitor orders, system usage.

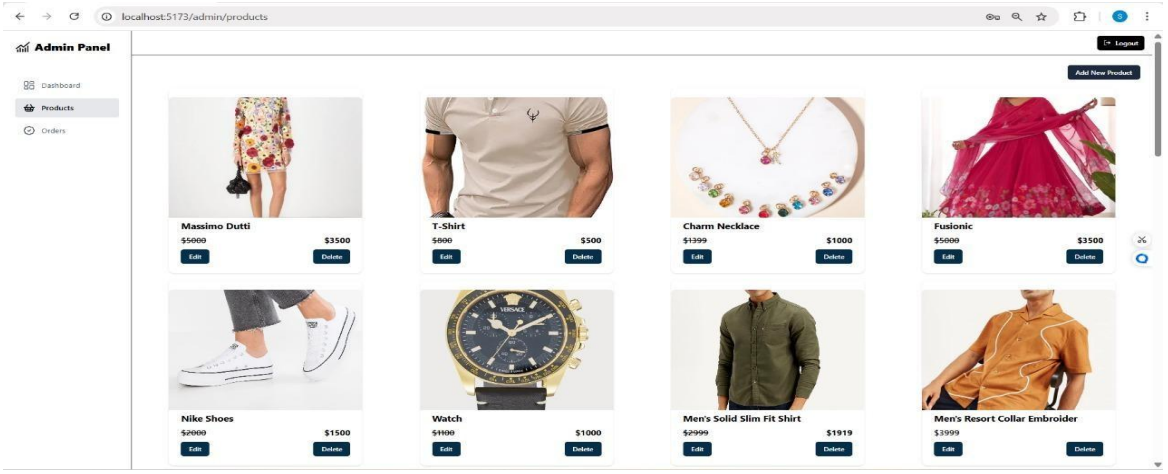
10. Testing

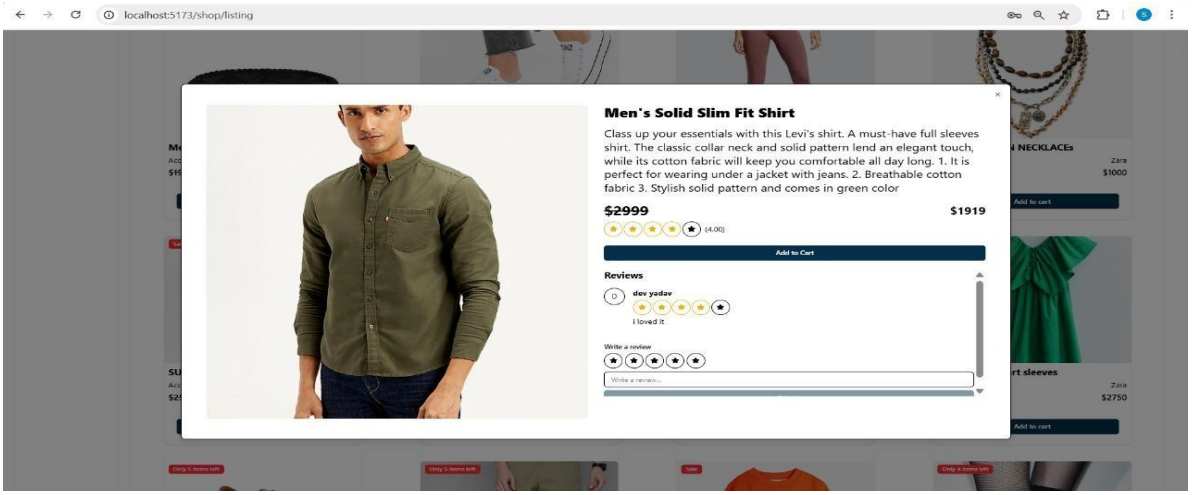
Testing Approaches:

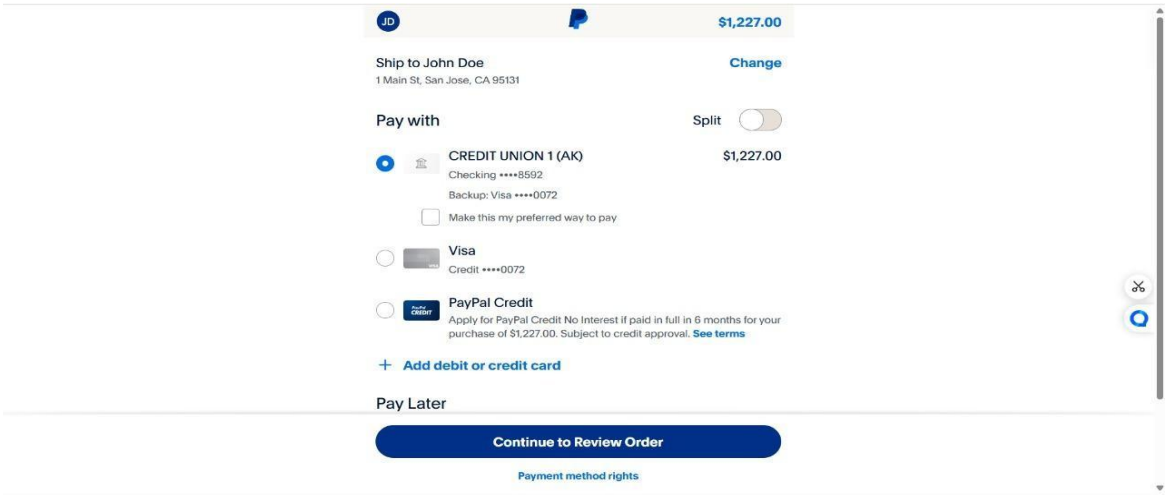
- **Functional Testing** – Manual verification of flows
- **API Testing** – Used Postman collections to verify endpoint integrity

- **Performance Testing** – Load testing with Apache JMeter
- **Frontend Audit** – Google Lighthouse for speed and accessibility scores.

11. Screenshots or Demo







https://drive.google.com/file/d/1sf0OhwgIi3d6qpX8PHjiiuWHaaonJbmT/view?usp=sharing

12. Known Issues

- Voice search depends on browser capabilities.

13. Future Enhancements

To ensure ShopEZ continues evolving into a highly scalable, intelligent, and competitive ShopEZ platform, the following future enhancements are proposed. These developments aim to improve user engagement, streamline backend operations, and enhance overall performance and accessibility:

- **Mobile App:** Develop native mobile app using React Native
- **Order Tracking:** Real-time delivery tracking and status updates
- **Notification System:** Email/SMS confirmation post checkout
- **Promo Engine:** Add coupon codes and discounts
- **Advanced Roles**: Fine-grained access control for admins and sellers
- **Analytics:** Dashboard for sales insights and visual reports
- **Chatbot:** Integrate AI-based support assistant for customers