**Full Stack Development with MERN**

**Project Documentation**

**1. Introduction**

Project Title: Shopez

**Team Id:** LTVIP2025TMID55646

**Team Members:**

* **Yarramasetty Anusha**:(Team Leader) Backend Development
* **Sanagala Yashwanth**: Backend Development, Rotes &API Integration,.
* **Yarribona Sai Sunanda**: (Team member)
* **Yaddanapudi Kavya**: Frontend Development,Authentication, Shopping , Cart & Order Logic.

**2. Project Overview**

**Purpose**

Shopez is a shopping platform built with the MERN stack. The goal is to connect users with local products through an easy-to-use, secure platform that allows browsing, ordering, and managing orders. Seller owners can manage their product catalog, and admins can promote products and oversee the platform's operation.

**Features**

* User authentication (sign up, login, JWT token)
* Browse products, products, and categories
* Shopping Cart management (add/remove items)
* Order placement and tracking
* Seller management (add, update, delete products)
* Admin dashboard for seller promotions

**3. Architecture**

**Frontend**

The frontend is built using **React.js**. It includes:

* **React Router** for dynamic page routing.
* **Redux** (optional) for state management.
* Custom components for user interface design, including reusable components like the header, footer, shopping cart, and product cards.

**Backend**

The backend uses **Node.js** and **Express.js** to handle:

* API routes for user authentication, shopping cart management, and order processing.
* Role-based access control (user, seller, admin).
* JWT token authentication for secure communication.

**Database**

The database uses **MongoDB** for storing:

* **Users** (authentication data)
* Sellers (details, products, orders)
* **Orders** (order status, payment tracking)

**4. Setup Instructions**

**Prerequisites**

* **Node.js**: Version 14.x or higher
* **MongoDB**: Installed locally or using MongoDB Atlas
* **npm**: Node package manager

**Installation**

1. Clone the repository:  
   git clone https://github.com/your-repo/Shopez.git
2. **Install Dependencies**:
   * For **Frontend** (React):
   * cd client
   * npm install
   * For **Backend** (Node.js, Express):
   * cd server
   * npm install
3. **Set up environment variables**:
   * Create .env files in both client and server directories:
     + **Frontend**: Set the API base URL (REACT\_APP\_API\_URL)
     + **Backend**: Set MongoDB URI (MONGO\_URI), JWT secret (JWT\_SECRET)
4. **Start the Application**:
   * For **Frontend**:
   * cd client
   * npm start
   * For **Backend**:
   * cd server
   * npm start

**5. Folder Structure**

**Client**

* client/: React.js frontend folder.
  + src/
    - components/: Reusable UI components (e.g., Header, Footer)
    - pages/: React components for different routes (e.g., Home, Shopping Cart, Orders)
    - redux/: Optional state management files (if Redux is used)

**Server**

* server/: Node.js backend folder.
  + controllers/: Logic to handle API requests.
  + models/: MongoDB schema models.
  + routes/: API routes (e.g., user, seller, order).
  + middleware/: Custom middleware (e.g., authentication, error handling).
  + config/: Environment variables and database connection.

**6. Running the Application**

To start the application locally, follow these commands:

* **Frontend**:
* cd client
* npm start
* **Backend**:
* cd server
* npm start

**7. API Documentation**

**User Endpoints**

* **POST /api /register**: Register a new user
* **POST /api /login**: Log in and receive JWT token
* **GET /api /profile**: Get user details (Protected route)

Seller Endpoints

* POST /api/products/login: Seller login
* GET /api/products/:id/products: Get products for a specific seller
* POST /api/products/:id/products: Add a product

**Order Endpoints**

* POST /api/orders: Create a new order
* GET /api/orders/:id: Get order details

**Admin Endpoints**

* POST /api/admin/promote/:id: Promote seller to homepage

**8. Authentication**

Authentication in Shopez is handled using bcrypt for secure password hashing and React Context API or Redux for client-side authentication state management.

* **Password Handling**:
  + User passwords are hashed using bcrypt before being stored in the database.
  + During login, the hashed password is compared with the entered password securely.
* **Session Management**:
  + Once a user logs in, their information (like user ID, role, etc.) is stored in React’s**Auth Context**.
  + This context is used across the app to control access to protected routes and display appropriate UI components (e.g., user vs. seller header).
* **Role-Based Access**:
  + React Context tracks user roles (user, seller, admin) to ensure each role sees the correct dashboard and has access to relevant features.

**9. User Interface**

Screenshots showcasing key UI elements:

* Homepage: Displays seller categories and popular products.
* Shopping Cart Page: Lists added items with the option to update and checkout.
* Seller Dashboard: Allows seller owners to manage their products.

**10. Testing**

**Testing Strategy**

* **Unit Tests**: Tests for backend API routes, models, and controllers.
* **Integration Tests**: Full-stack testing (frontend to backend communication).
* Manual Testing: Functional testing of user flows such as login, shopping cart updates, and order placement.

**Tools Used**

* **Jest** for unit testing
* **Supertest** for API testing

**11. Screenshots**

* Screenshots:

Homepage (Product Listing with Sale Banner):

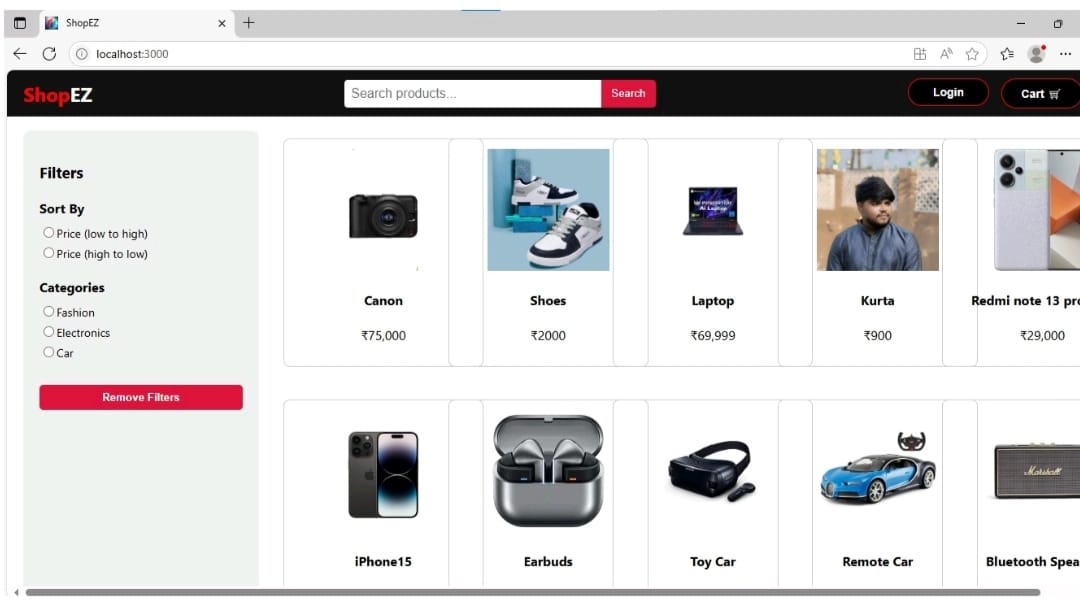
Product Details Page:

User Login Page:

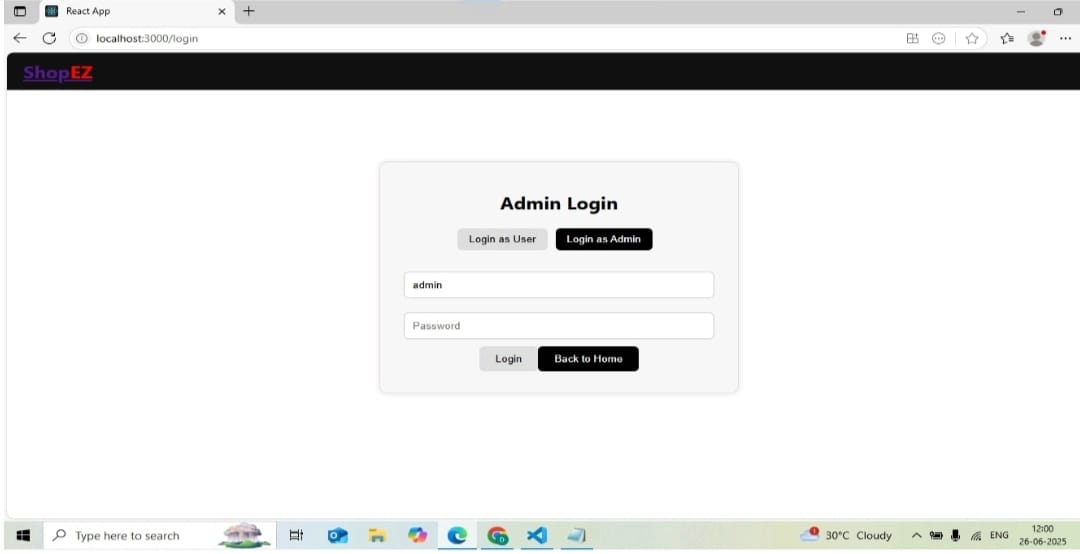
Admin Login Page:

Product Grid View with Filters:

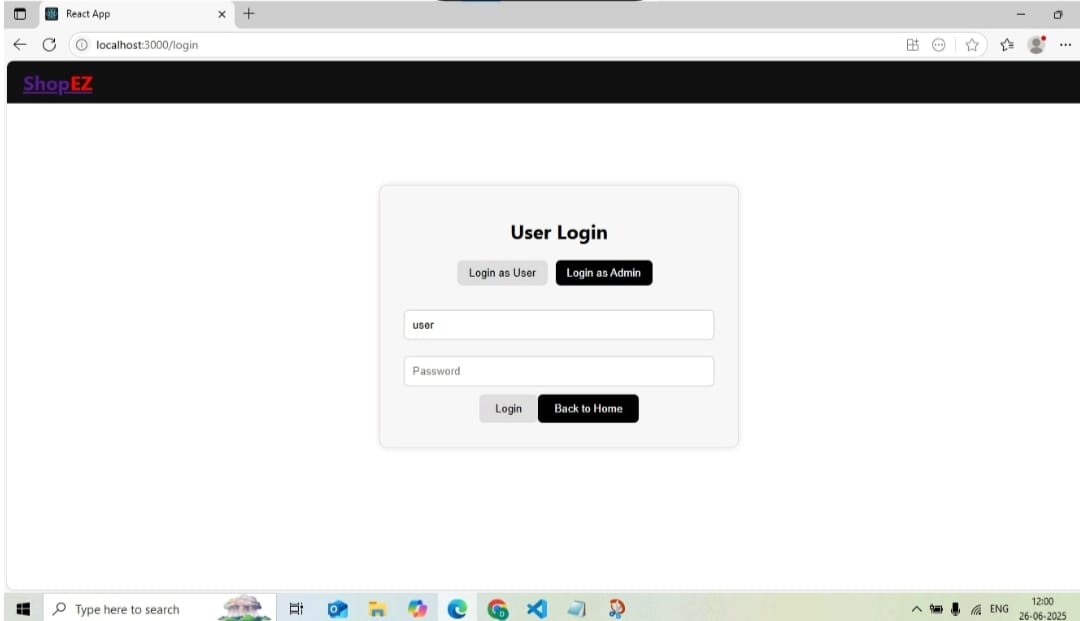
**Product details page:**



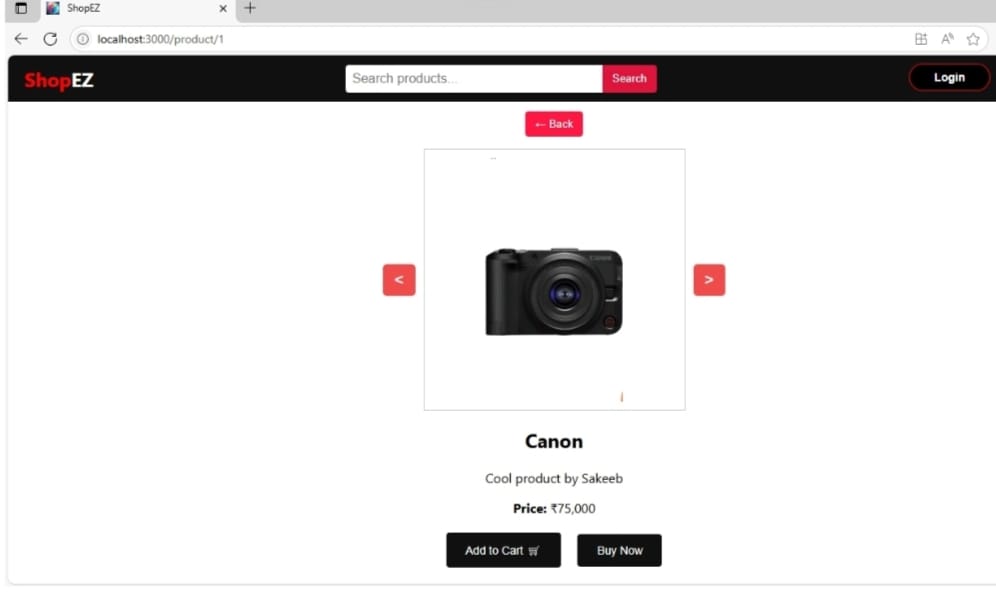
**Admin login:**



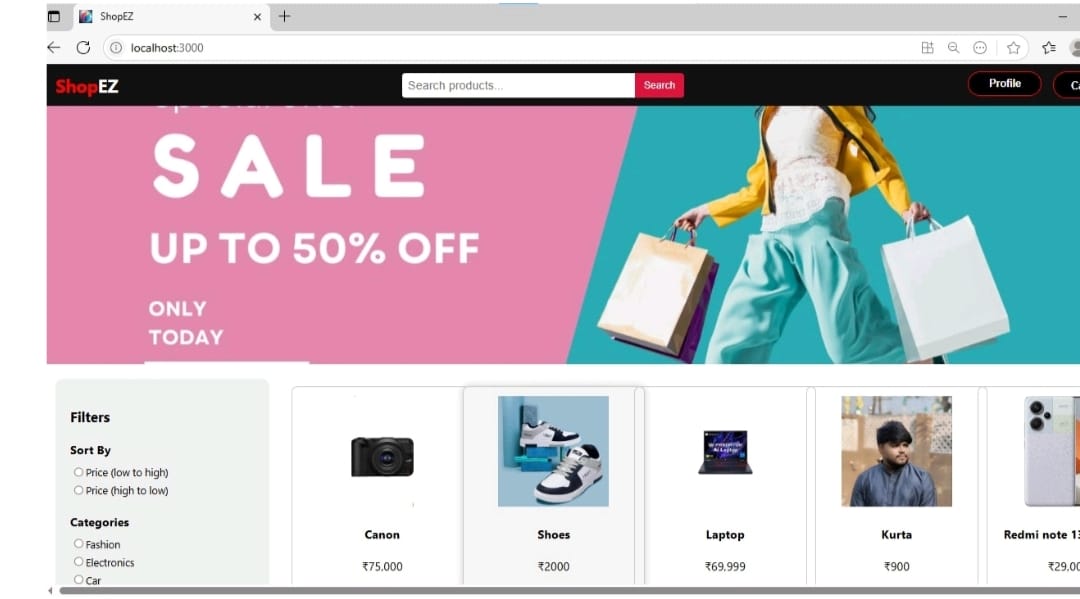
**User login:**



**Product details:**



**Home page:**



**12. Known Issues**

* **Payment Integration**: Not implemented yet, placeholder functionality.
* Mobile App (optional): Currently no mobile app version, only web-based.

**13. Future Enhancements**

* **Payment Gateway**: Integrate with a real payment provider (e.g., Stripe, Razorpay).
* Mobile App (optional): Develop a mobile version using React Native.
* Notifications: Implement order status notifications for users and products.
* Analytics Dashboard: Add real-time analytics for products and admins.