**Full Stack Development with MERN: ShopEZ E-Commerce Application Documentation**

**1. Introduction**

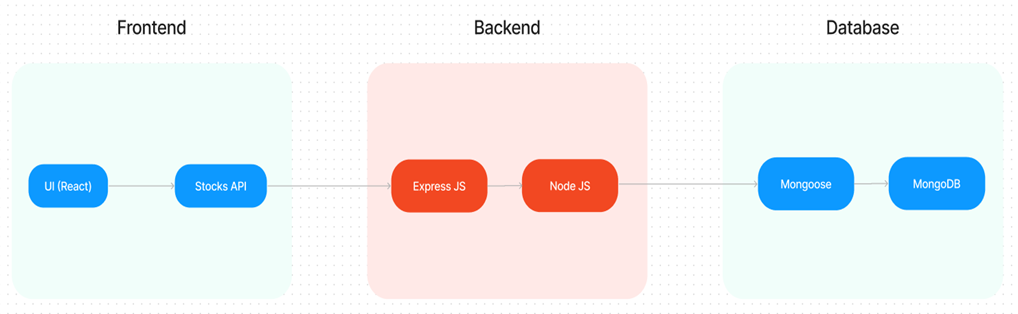
* **Project Title**: ShopEZ E-Commerce Application
* **Team Members**:
  + **Abinesh** - Frontend Developer
  + **Jeevarathinam** - Backend Developer
  + **Jagan Vignesh**- Database Engineer
  + **Gopi** - UI/UX Designer
  + **Gulbuthin Ansari** - Project Manager

**2. Project Overview**

* **Purpose**:  
   ShopEZ is a full-stack e-commerce platform designed to simplify the online shopping experience for customers while also providing robust support and complaint management features. The application facilitates browsing, purchasing, order tracking, and real-time complaint resolution.
* **Features**:
  + **User Registration**: Allows users to create accounts and track their shopping history
  + **Complaint Management:** Users can submit complaints regarding product quality, delivery issues, or customer service and track their progress in real-time.
  + **Product Discovery:** Advanced search and recommendation system to help users find products based on preferences.
  + **Seamless Checkout:** Streamlined checkout process with support for various payment methods.
  + **Order Tracking:** Users can monitor the status of their orders and receive notifications.
  + **Admin Dashboard:** Allows administrators to manage users, products, orders, payments, and promotions.

**3. Architecture**

* **Frontend:**  
  The frontend of ShopEZ is built using **React.js** for dynamic, responsive user interfaces. It communicates with the backend through restful APIs to fetch product details, handle user authentication, and manage orders. The frontend includes key features like product browsing, cart management, user authentication, and complaint submission.
* **Backend:**  
  The backend is developed using **Node.js** with **Express.js** to handle API requests and manage business logic. It includes several modules such as user authentication, order processing, payment handling, and complaint routing. The backend communicates with the MongoDB database for persistent data storage.
* **Database:**  
  The application uses **MongoDB** as the NoSQL database to store user profiles, products, orders, and payments. MongoDB's flexible schema allows for quick modifications and easy scalability. The database is structured into collections like Users, Products, Orders, and Complaints. It is connected to the backend via **Mongoose**, an ODM for MongoDB.



**4. Setup Instructions**

**Prerequisites:**

* **Node.js and npm**  
  Install **Node.js** (which includes **npm**).
  + [Node.js download link](https://nodejs.org/en/download/).
* **MongoDB**  
  Set up **MongoDB** either locally or use a cloud service like [MongoDB Atlas](https://www.mongodb.com/cloud/atlas).
  + [MongoDB download link](https://www.mongodb.com/try/download/community).
* **React**  
  Install **React** via the official React setup guide.
  + [React setup guide](https://reactjs.org/docs/create-a-new-react-app.html).
* **Git**  
  Install **Git** for version control.
  + [Git download link](https://git-scm.com/downloads).

**Installation Steps:**

1. **Clone the Repository:**  
   Open your terminal and run:

bash

Copy code

git clone https://github.com/your-repo/ShopEZ-E-Commerce.git

1. **Install Backend Dependencies:**  
   Navigate to the server folder and install the required packages:

bash

Copy code

cd ShopEZ-E-Commerce/server

npm install

1. **Install Frontend Dependencies:**  
   Navigate to the client folder and install the required packages:

bash

Copy code

cd ../client

npm install

1. **Set Up Environment Variables:**
   * Create a .env file in the server directory and add MongoDB URI and other necessary environment variables:

makefile

Copy code

MONGODB\_URI=your-mongodb-uri

JWT\_SECRET=your-jwt-secret

PAYPAL\_CLIENT\_ID=your-paypal-client-id

1. **Start the Application:**
   * **Frontend:**  
     From the client folder, run:

bash

Copy code

npm start

* + **Backend:**  
    From the server folder, run:

bash

Copy code

npm run dev

1. **Visit** http://localhost:3000 in your browser to access the application.

**5. Folder Structure**

**Client (Frontend):**

* **public/:** Static files such as index.html, favicon, and images.
* **src/:** Contains all React components and hooks.
  + **components/:** Includes reusable UI components like ProductCard, CartItem, Navbar.
  + **pages/:** Each page of the app like HomePage, ProductPage, CheckoutPage.
  + **services/**: API services to communicate with the backend (e.g., productService.js, userService.js).
  + **App.js:** The root component that renders the app’s layout.

**Server (Backend):**

* **controllers/**: Contains the logic for handling requests for different entities like users, orders, products, complaints.
  + userController.js, orderController.js
* **models/:** Mongoose models for data schema.
  + User.js, Product.js, Order.js
* **routes/:** Defines all the routes (API endpoints) of the app.
  + userRoutes.js, orderRoutes.js, complaintRoutes.js
* **middlewares/:** Contains middleware functions for authentication, error handling, and logging.
  + authMiddleware.js, errorMiddleware.js
* **config/:** Database and server configurations.
  + db.js (MongoDB connection setup)

**6. Running the Application**

* **Frontend:**  
  To run the frontend, navigate to the client directory and run:

bash

Copy code

npm start

* **Backend:**To run the backend, navigate to the server directory and run:

bash

Copy code

npm run dev

The app will be live at http://localhost:3000.

**7. API Documentation**

**User Endpoints:**

* **POST** /api/users/register: Registers a new user.
  + Request Body: { name, email, password }
  + Response: 200 OK, { message: 'User registered successfully' }
* **POST** /api/users/login: Logs in a user.
  + Request Body: { email, password }
  + Response: 200 OK, { token, userId }

**Order Endpoints:**

* **POST** /api/orders/create: Creates a new order.
  + Request Body: { userId, products[], totalAmount, shippingAddress }
  + Response: 200 OK, { orderId, status: 'Processing' }

**8. Authentication**

ShopEZ uses **JWT (JSON Web Token)** for authentication. When a user logs in, a JWT is generated and sent back to the frontend. This token is used for subsequent requests that require authentication. The token is stored in local storage or cookies.

**9. User Interface**

* **Home Page:** Displays product categories, popular items, and personalized recommendations.
* **Product Page:** Contains detailed product information including price, description, reviews.
* **Cart:** Allows users to view and manage their selected items.
* **Checkout**: Users input shipping details and payment information before finalizing the order.

**Screenshots**:  
[Include screenshots or GIFs of key UI features]

**10. Testing**

Testing for the ShopEZ app is done using **Jest** for unit testing and **Supertest** for API testing. The testing suite covers:

* **API Tests:** Ensuring the backend endpoints are working as expected.
* **UI Tests:** Ensuring the frontend components render correctly and interact as expected.

**11. Screenshots or Demo**

[Provide screenshots or link to a live demo showcasing the application’s functionality]

**12. Known Issues**

* **Issue 1:** Occasionally, the product recommendation engine does not return the most relevant products.
* **Issue 2:** The checkout process sometimes times out due to high traffic during peak hours.

**13. Future Enhancements**

* **Feature 1:** Adding real-time chat support between users and customer service.
* **Feature 2:** Integrating more payment gateways (e.g., Apple Pay, Google Pay).
* **Feature 3:** Enhanced machine learning-based product recommendations.

Top of Form

Bottom of Form