**Project Documentation: E-commerce**

**1. Overview**

This document provides comprehensive documentation for the Small E-commerce platform designed to facilitate seamless interactions between sellers and buyers. The platform is built using Next.js for the frontend, Node.js with Express.js for backend REST APIs, and PostgreSQL as the database. The application allows sellers to manage products and buyers to search for, view, add to cart and purchase products.

**2. Tech Stack**

Frontend: Next.js  
Backend: Node.js & Express.js  
Database: PostgreSQL  
Libraries: Tailwind CSS for styling

**3. Key Features**

User Authentication:  
- Users can sign up and log in as either a seller or a buyer.  
- JWT is used for authentication and maintaining sessions.

Seller Panel:  
- Sellers can add, edit, and delete products.  
- Product details include name, category, description, price, and discount.  
- Sellers can view a list of their products.

Buyer Panel:  
- Buyers can search for products by name.  
- Buyers can add products to their cart and remove them as needed.  
- A checkout process is available for buyers to complete their purchases.

Responsive Design:  
- The application is fully responsive, ensuring usability across various devices and screen sizes.

Error Handling:  
- Proper error messages are returned for invalid inputs, unauthorized access, and other edge cases.

**4. Application Structure**

**Frontend (Next.js):**

Pages: Organized using Next.js's file-based routing system.

-/register: Contains register page.  
- /login: Contains login page.  
- /seller: Dashboard for sellers, including product management.  
- /: Pages for all product.

-/cart: for access cart.

-/productdetail/:id : for accessing product detail.

Components: Reusable components such as headers, footers, and product cards.

Styles: Tailwind CSS is used for styling.

**Backend (Node.js & Express.js):**

API Routes:

-/api/register: Handler user register.  
- /api/login: Handles user login.  
- /api/products: CRUD operations for products.

-/api/allproducts: Access all products.

- /api/cart: Manage buyer’s cart.

Middleware: JWT-based authentication middleware to protect routes.

Database Models: Sequelize ORM is used to define models for users, products, and orders.

**5. Database Design**

Tables:  
- Users: Stores user information (id, username,useremail, password, role).  
- Products: Stores product details (id, name, category, description, price, discount, seller\_id,image).  
- Cart: Stores cart items for each buyer (user\_id, product\_id, quantity,product\_name,product\_id,price,discount,productimage).

**6. Installation & Setup**

Local Setup:  
1. Clone the repository: `git clone <repository-url>`  
2. Install dependencies:  
 - Frontend: `npm install`  
 - Backend: `npm install`  
3. Configure environment variables:  
 - Create `.env` files for both frontend and backend with appropriate configurations.

-for backend

* PORT=5000
* DB\_HOST=localhost
* DB\_USER=Your database user name
* DB\_PASSWORD=Your database password
* DB\_NAME=Your database name
* secret\_key=your\_secretkey\_here
* JWT\_SECRET=your\_secretkey\_here

-for frontend

* NEXT\_PUBLIC\_API\_URL=http://localhost:your backend\_port

4. Run the application:  
 - Frontend: `npm run dev`  
 - Backend: `npm start`  
5. Access the application:  
 - Open the browser and navigate to `http://localhost:3000` for the frontend.  
 - Use tools like Postman to interact with the backend APIs.

**7. API Documentation**

**Authentication Endpoints:**

POST /api/register: Register a new user.  
POST /api/login: Log in a user and return a JWT.

**Product Endpoints:**

GET /api/allproducts: Get all products.

GET /api/products/:id : Get product by id.  
POST /api/products: Add a new product (Seller only).  
PUT /api/products/:id: Update an existing product (Seller only).  
DELETE /api/products/:id: Delete a product (Seller only).

**Cart Endpoints:**

GET /api/cart: Get all items in the cart (Buyer only).  
POST /api/cart: Add a product to the cart (Buyer only).  
DELETE /api/cart/:id: Remove a product from the cart (Buyer only).

DELETE /api/cart: Clear the cart (Buyer only).

**8. Best Practices & Coding Standards**

Code Cleanliness:  
- Followed best practices for code readability and maintainability.  
- Used meaningful variable names, modularized code, and provided comments where necessary.

Error Handling:  
- Implemented comprehensive error handling to manage various scenarios like invalid input, unauthorized access, and non-existent resources.

Security:  
- All user inputs are validated and sanitized before processing.  
- Sensitive information, such as passwords, is hashed before storing in the database.

Responsiveness:  
- Ensured that the UI is responsive across different devices using Tailwind CSS.

**9. Future Enhancements**

Order tracking: We can add this feature to check our oreder status.

Payment Integration: Adding payment gateways for processing transactionsPayment Integration: Adding payment gateways for processing transactions.  
Advanced Search: Implementing filters and sorting for the product search.  
Reviews & Ratings: Allowing buyers to leave reviews and ratings for products.  
Notifications: Implementing email or SMS notifications for order updates.