**SHOP EZ – E – COMMERCE – APPLICATION**

**TEAM MEMBERS :-**

1. LAYSHMA M - Team Leader
2. KARTHIKA RS - Backend Developer
3. KARAN PAL P - Frontend Developer
4. KUMARASELVAN V - Documentation Specialist

**PROJECT OVERVIEW :-**

**PURPOSE :**

ShopEZ is your one-stop destination for effortless online shopping. With a user-friendly interface and a comprehensive product catalog, finding the perfect items has never been easier. Seamlessly navigate through detailed product descriptions, customer reviews, and available discounts to make informed decisions. For sellers, our robust dashboard provides efficient order management and insightful analytics to drive business growth. Experience the future of online shopping with ShopEZ today.

**FEATURES :**

ShopEZ aims to revolutionize online shopping by delivering a seamless and personalized experience for both consumers and sellers.

* **Intuitive User Interface:** A user-friendly design will simplify navigation, and  find desired products without feeling overwhelmed.
* **Personalized Recommendations:** Utilizing AI algorithms, ShopEZ will offer  product suggestions based on user preferences, previous purchases, ensuring a more engaging shopping experience.
* **Streamlined Checkout Process:** The checkout will be optimized for efficiency and saved payment information to minimize steps and reduce cart abandonment.
* **Comprehensive Seller Dashboard:** Sellers will benefit from a robust dashboard that provides real-time analytics and efficient order processing tools.
* **Secure** **Payment and Order Tracking:** A secure payment gateway, combined with real-time order tracking, will enhance trust and transparency throughout the purchasing journey.Customer Feedback Integration: Prominently displaying customer reviews will help buyers make informed decisions.

**ARCHITECTURE :-**

**1. Frontend (React):**

* **Structure:** Component-based design with pages like Home, Product Details, Cart, and Checkout.
* **State Management:** Uses Redux for handling global state (e.g., cart, authentication).
* **Routing & API:** React Router for navigation; Axios to make API requests.
* **Styling & Notifications:** Styled with Tailwind CSS or Bootstrap; react-toastify for alerts.

**2. Backend (Node.js + Express.js):**

* **API Structure:** Organized routes for authentication, products, cart, orders, and user profiles.
* **Middleware & Security:** JWT-based authentication, error handling, and security middleware.
* **Services:** Includes business logic for cart, orders, and integrations (e.g., Stripe for payments).

**3. Database (MongoDB):**

* **Schema:** Collections for Users, Products, Cart, and Orders.
* **User:** Storesuser info, credentials.
* **Product:** Product details, stock, pricing.
* **Cart:** User cart items, quantities.
* **Order:** Completed orders, payment, and order status.

**SETUP INSTRUCTIONS :-**

**ShopEZ E-commerce Web Application Setup Guide**

**Prerequisites**

To set up and run the ShopEZ application, ensure the following software dependencies are installed:

1. Node.js (v14.x or higher) - Required to run the backend server.

2. npm - Package manager for handling dependencies.

3. MongoDB - NoSQL database to store application data.

4. Git - Version control for cloning the project repository.

**Installation**

1. Clone the Repository **:** git clone https://github.com/username/shopEZ.git

cd shopEZ.

2. Install Dependencies

Backend:

cd backend

npm install

Frontend:

npm run start

**3. Set Up Environment Variables**

Create a .env file in the backend directory and define the following variables:

PORT=5000

MONGODB\_URI=<your-mongodb-connection-string>

JWT\_SECRET=<your-jwt-secret>

STRIPE\_SECRET\_KEY=<your-stripe-secret-key> # Optional, if using Stripe for payments.

**4. Run the Application**

Backend:

npm run dev

5. Access the Application

Open a browser and go to http://localhost:3000 to view the frontend.

**ShopEZ E-commerce Application Folder Structure**

1. **Client (React Frontend)**

The frontend is organized to keep components modular and reusable, with a focus on separating concerns like routing, state management, and UI.

* public/: Contains static assets like images and the root index.html file.
* src/: Main source folder for the React application.
* components/: Holds reusable UI components, such as ProductCard, Navbar, and Footer.
* pages/: Contains main pages of the application like Home, ProductDetail, Cart, and Checkout.
* redux/: State management folder that includes:
* slices/: Individual slices for different app features (e.g., cartSlice, productSlice).
* store.js: Sets up the Redux store.
* services/: Handles API calls using Axios to interact with backend endpoints (e.g., productService.js).
* App.js: Root component that includes routes for navigating between pages.
* index.js: Entry point that renders the app into the DOM.

**2. Server (Node.js Backend)**

The backend is structured for scalability and separation of business logic, routing, and database interactions.

* controllers/: Contains controllers to handle API logic for each route, like productController.js, userController.js.
* models/: Defines Mongoose schemas for collections, including Product.js, User.js, Order.js, and Cart.js.
* routes/: Holds route files for organizing API endpoints, such as productRoutes.js, userRoutes.js, and authRoutes.js.
* middlewares/: Stores middleware functions for authentication, error handling, and other tasks.
* config/: Holds configuration files, such as db.js for MongoDB connection and environment configuration.
* services/: Business logic and external service integration, such as payment processing in paymentService.js.
* server.js: Entry point of the backend application that sets up the Express server, loads middleware, and routes.

**RUNNING THE APPLICATIONS ;-**

**Running the ShopEZ Application Locally**

To start the application locally, use the following commands:

Frontend: Run the following command in the client directory to start the React frontend :

cd client

npm i

npm run start

Backend: Run the following command in the server directory to start the Node.js backend.

cd server

npm i

npm run dev

The frontend will be available at http://localhost:3000, and the backend server will run on <http://localhost:5000>.

**API DOCUMENTATION :-**

**1. Authentication**

Register User

* POST /api/auth/register
* Body: { "name": "John Doe", "email": "john@example.com", "password": "password123" }
* Response: { "message": "User registered successfully", "user": { "id": "123", "name": "John Doe" } }

Login User

* POST /api/auth/login
* Body: { "email": "john@example.com", "password": "password123" }
* Response: { "message": "Login successful", "token": "jwt-token" }

**2. Products**

Get All Products

* GET /api/products
* Response: List of products with details.

Get Product by ID

* GET /api/products/:id
* Response: Details of a specific product.

**3.** **Cart**

Add Item to Cart

* POST /api/cart/add
* Body: { "productId": "101", "quantity": 2 }
* Response: { "message": "Item added to cart", "cart": { ... } }

Get User Cart

* GET /api/cart
* Response: { "items": [...], "totalPrice": 59.98 }

**4.** **Orders**

Place Order

* POST /api/orders
* Body: { "cartId": "xyz123", "paymentMethod": "credit\_card" }
* Response: { "message": "Order placed successfully", "order": { ... } }

Get Order by ID

* GET /api/orders/:id
* Response: Details of a specific order.

**5. User Profile**

* Get User Profile
* GET /api/users/:id
* Response: { "id": "123", "name": "John Doe", "email": "john@example.com", .}

**AUTHENTICATION :-**

**Authentication and Authorization in ShopEZ**

In the ShopEZ application, authentication and authorization are managed using JSON Web Tokens (JWT) for secure user access to protected routes and resources.

1. **Authentication Process**

* User Login: When a user logs in, the backend verifies their credentials. Upon successful authentication, the server generates a JWT.
* Token Generation: The JWT is signed with a secret key and includes user information (e.g., user ID, role) in the payload. This token is sent back to the frontend.

**2. Authorization**

* Token-Based Access: The frontend stores the JWT (e.g., in local storage) and includes it in the headers of API requests to access protected routes (such as viewing the cart or placing an order).
* Role-Based Access: The token contains user roles (e.g., "admin" or "customer"), allowing the backend to authorize users for specific actions, like managing products (admin-only).

**3. Token Validation**

* Middleware: The backend uses middleware to check for a valid JWT on protected routes. If a valid token is present, the request proceeds; otherwise, it returns an authentication error.
* Session Management: The JWT-based approach is stateless, meaning no server-side sessions are maintained. Token expiration is set to limit the duration of user sessions, and users can re-authenticate to receive a new token.

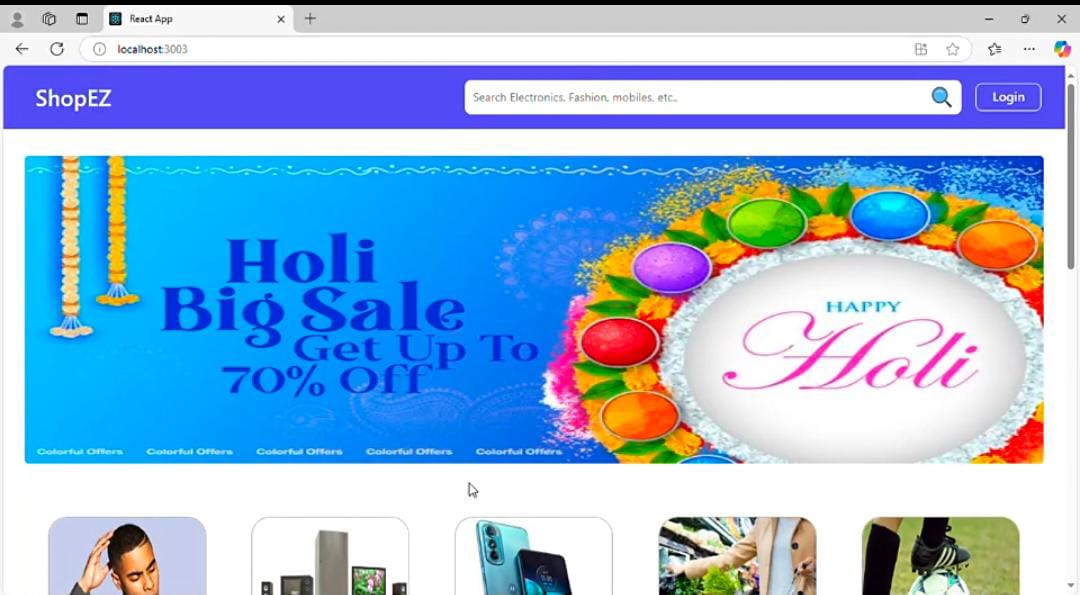
**USER INTERFACE :**

The ShopEZ application features a user-friendly interface designed for a smooth shopping experience. Below are some key UI components:

**1.** **Home Page**

Showcases featured products, categories, and promotional banners.

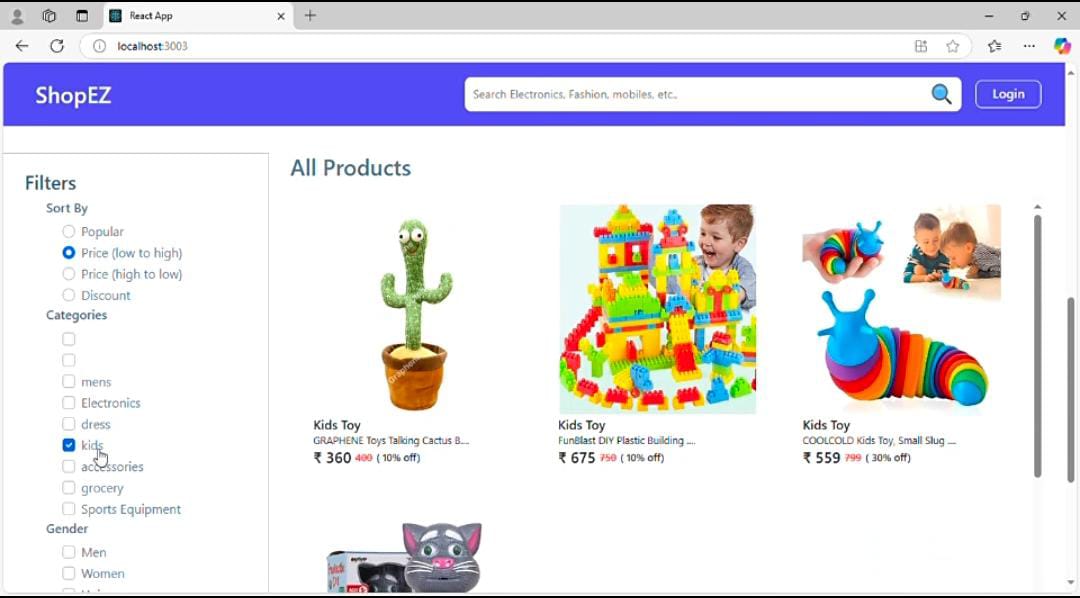
Includes a search bar for product discovery.



**2. Product Details Page**

Displays product images, description, price, and available stock.

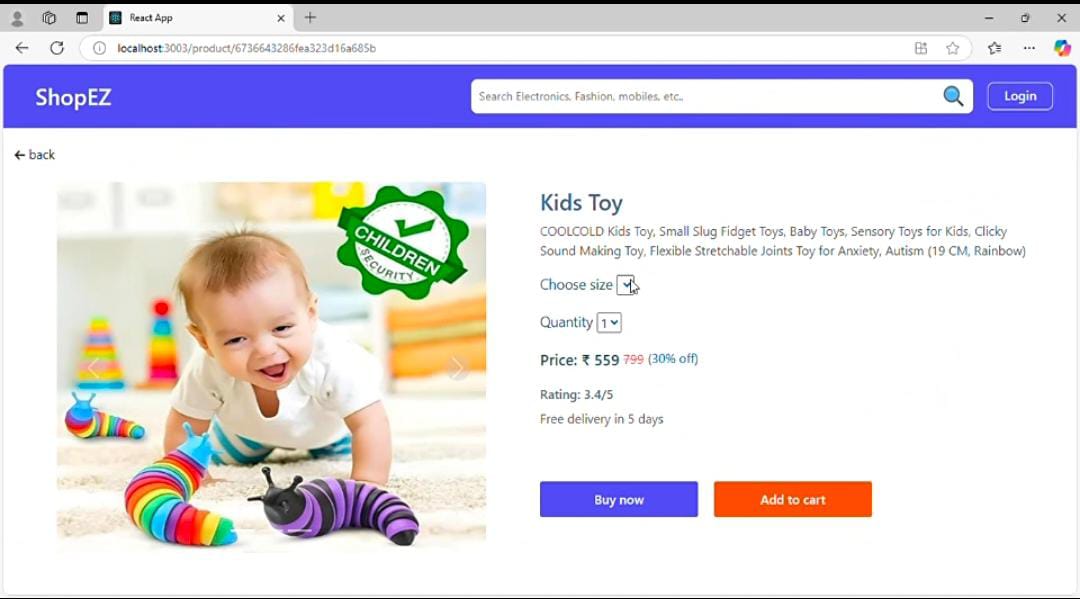
Offers an "Add to Cart" button for easy purchasing.



**3. Shopping Cart**

Shows items added to the cart with quantity, price, and total cost.

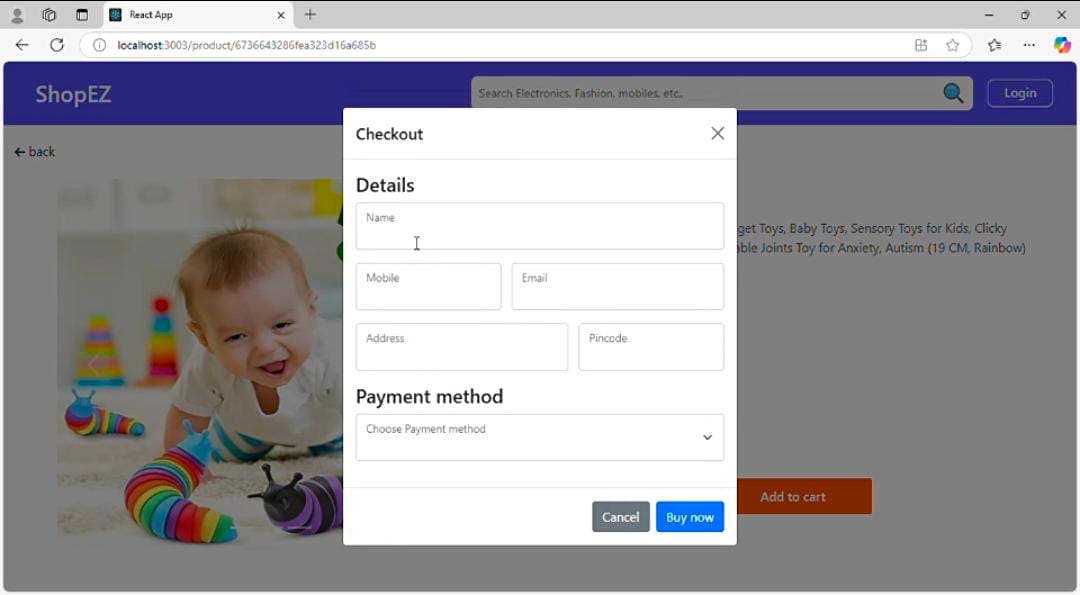
Allows users to update quantities or remove items.



**4. Checkout Page**

Guides users through order details, shipping information, and payment options.

Provides a summary of the total order amount before payment.



**5. User Profile**

Displays account details, order history, and options to update personal information.Screenshots or GIFs of each of these sections should be included to illustrate the user interface and features in action, providing a visual overview of the ShopEZ experience.

**TESTING :-**

To ensure a reliable and user-friendly experience, we employed the following testing approach for ShopeEz:

**1. Unit Testing**

Purpose: Verify each component works individually.

Tool: Jest

**2. Integration Testing**

Purpose: Test that the frontend and backend communicate correctly.

Tool: Postman

**3. Functional Testing**

Purpose: Confirm key features like login, cart, and checkout work as expected.

Tool: Selenium

**4. Usability Testing**

Purpose: Ensure intuitive navigation and ease of use.

Tool: UserTesting.com

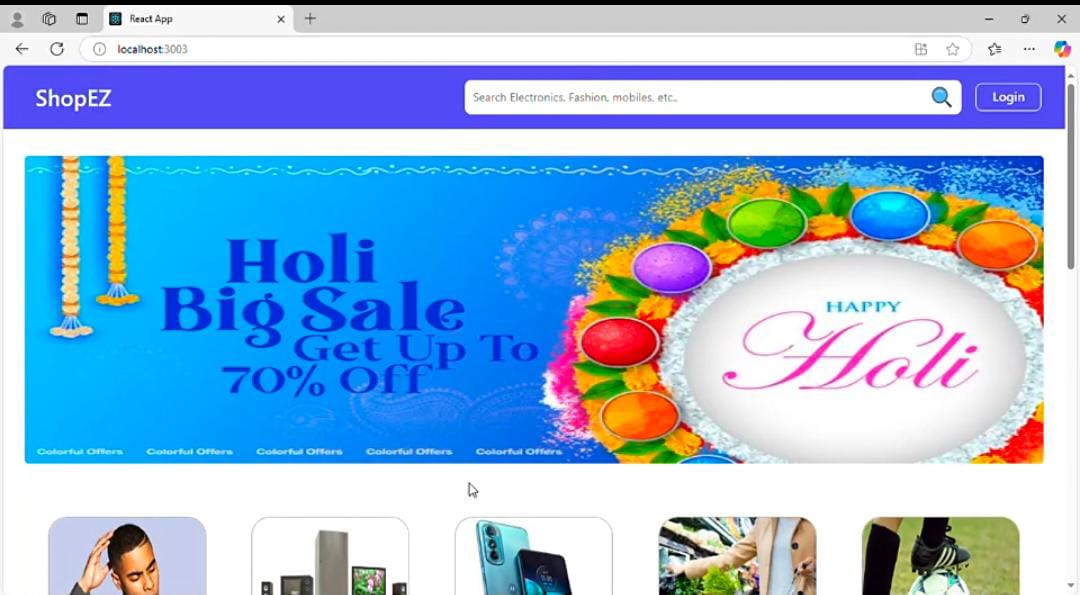
**5.App test.js**

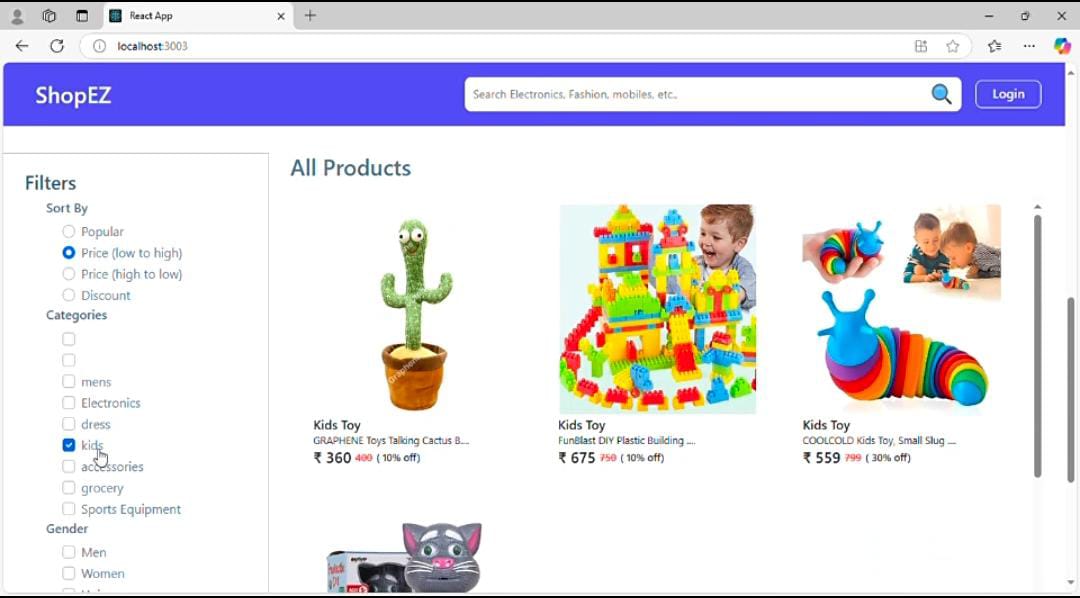
The primary purpose of App.test.js is to ensure that the App component loads and displays correctly with its initial elements, like a header or welcome text. If any fundamental issue prevents the app from rendering, this test will fail, alerting developers to fix it before proceeding.

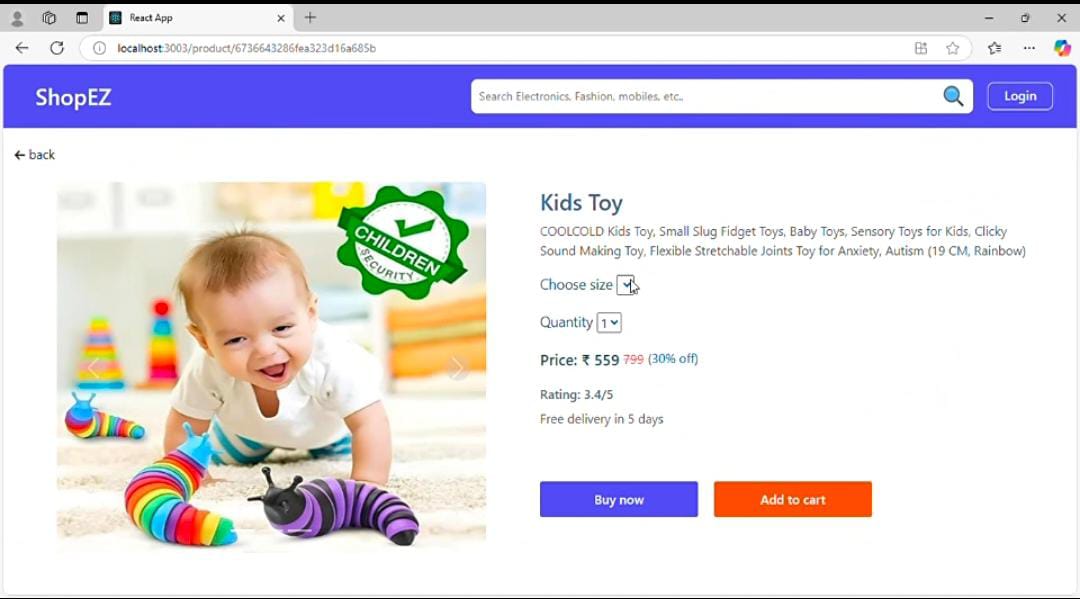
**6.Setup test.js**

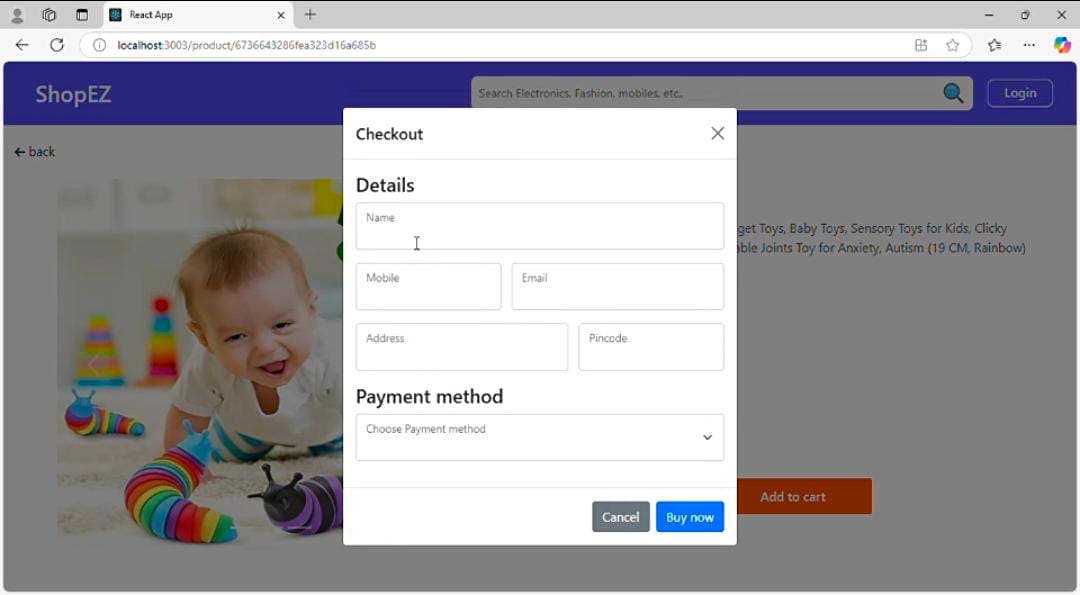
setupTests.js is a configuration file in React projects used to set up the testing environment before tests run.setupTests.js provides a stable, shared setup across all test files, making tests more efficient and consistent.

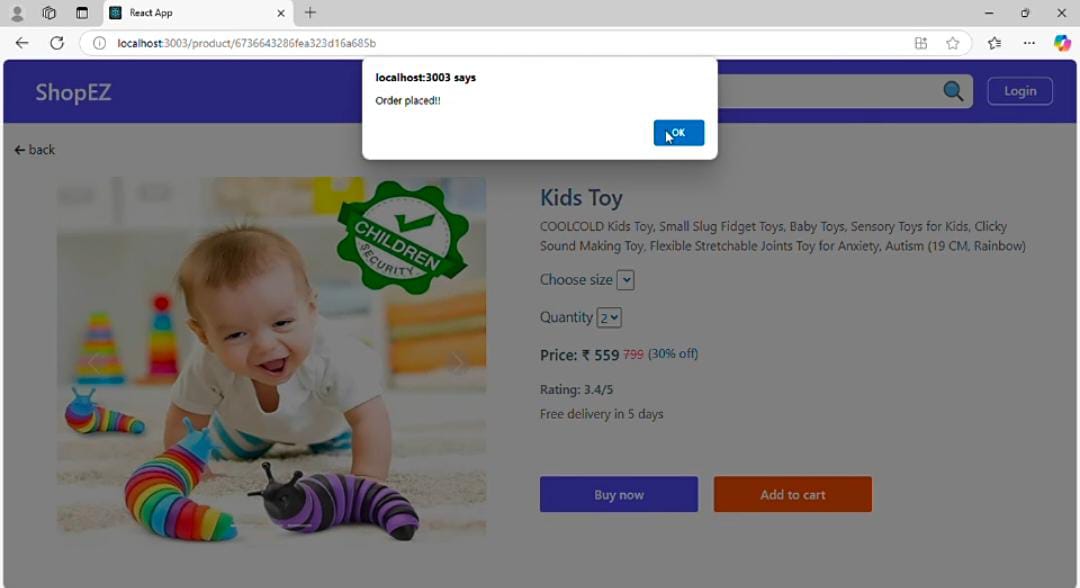
**SCREENSHOTS OR DEMO :-**

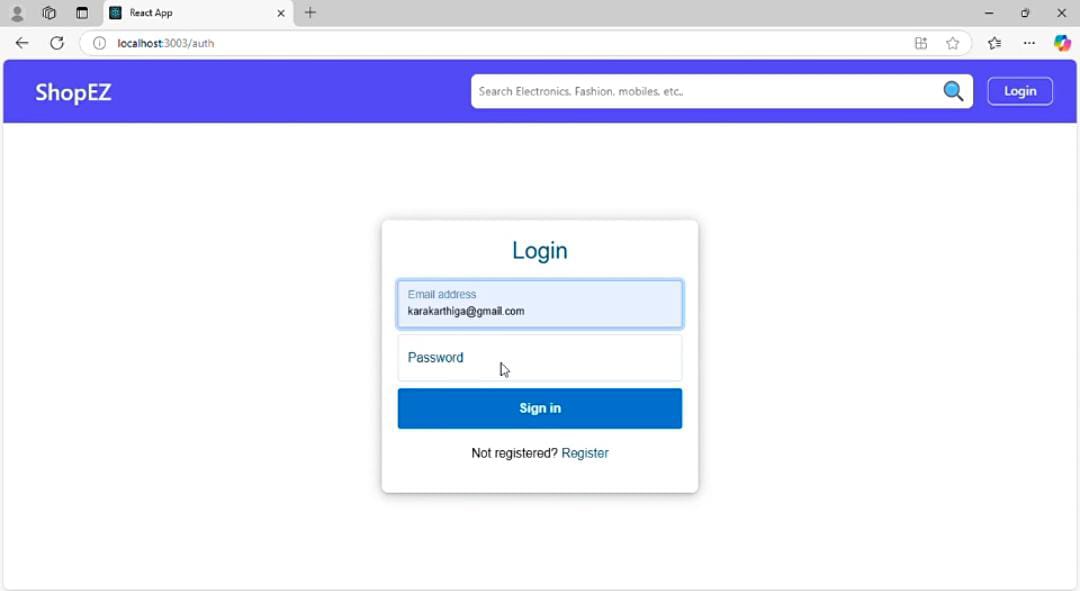
****

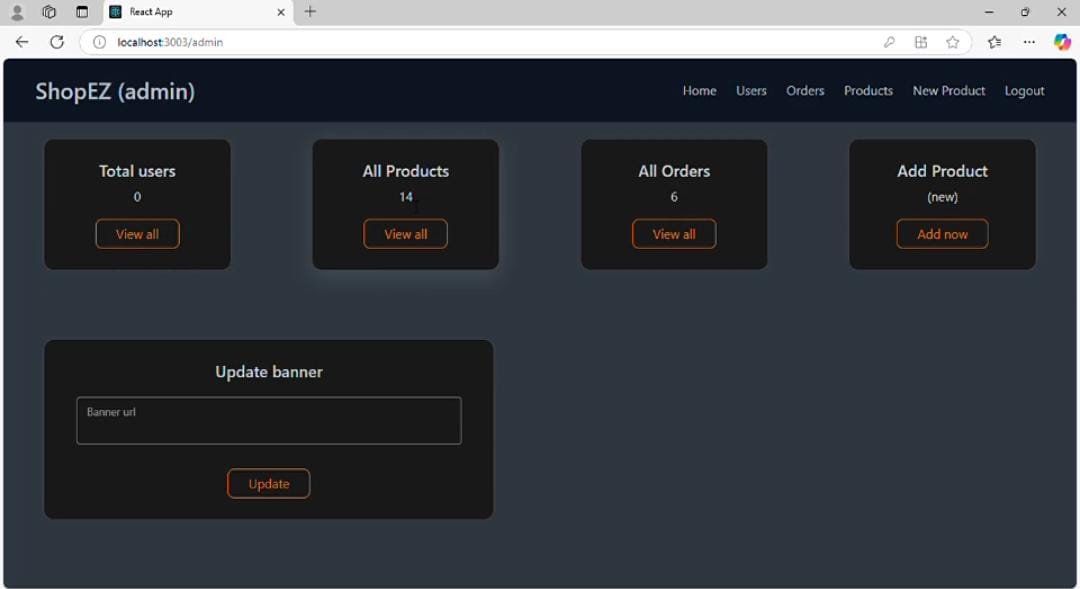
****

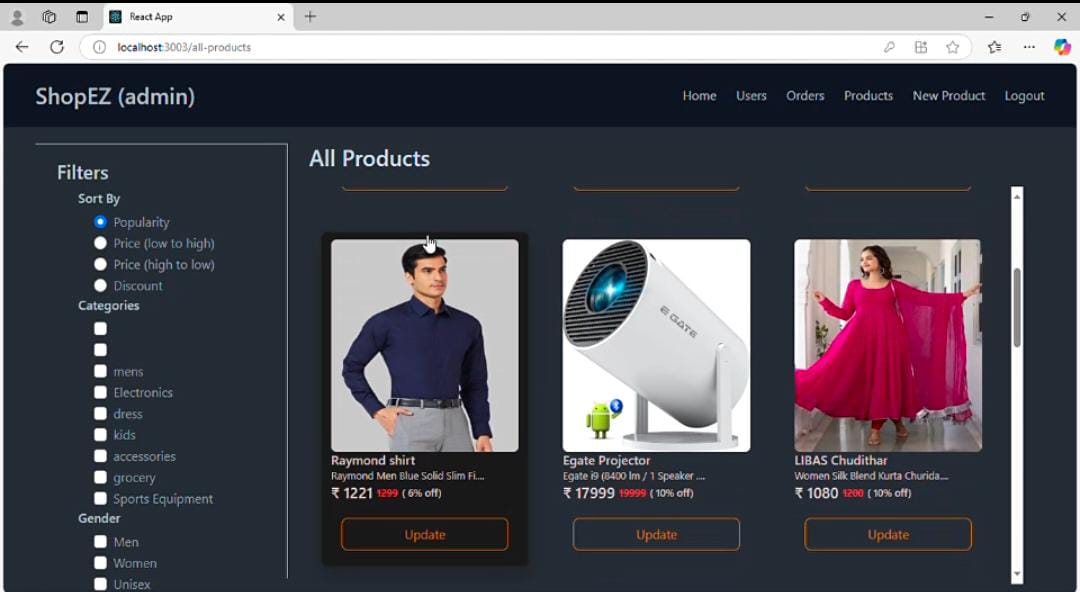
****

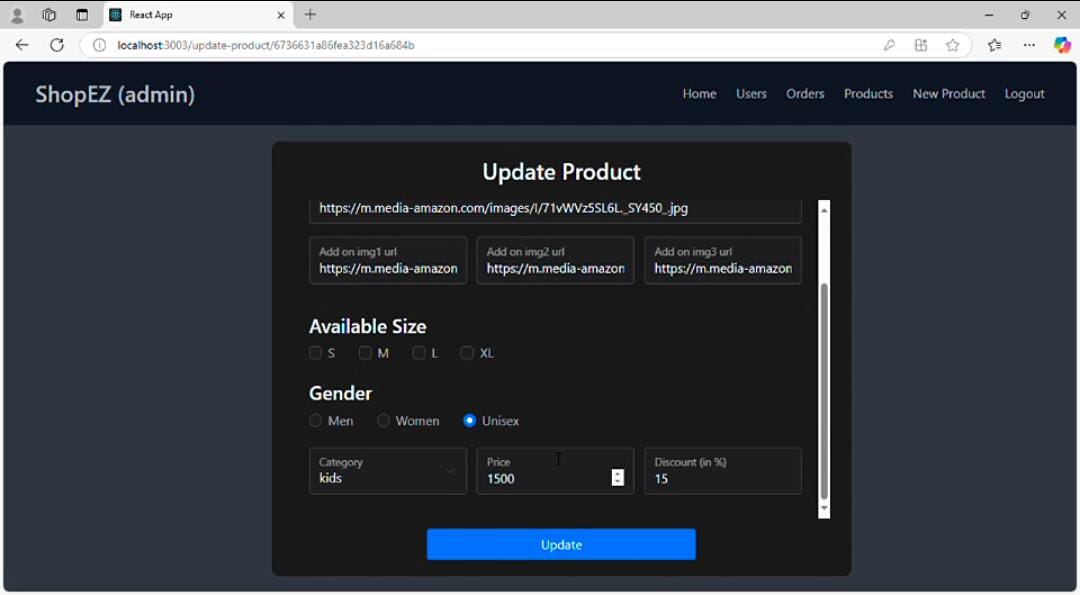
****

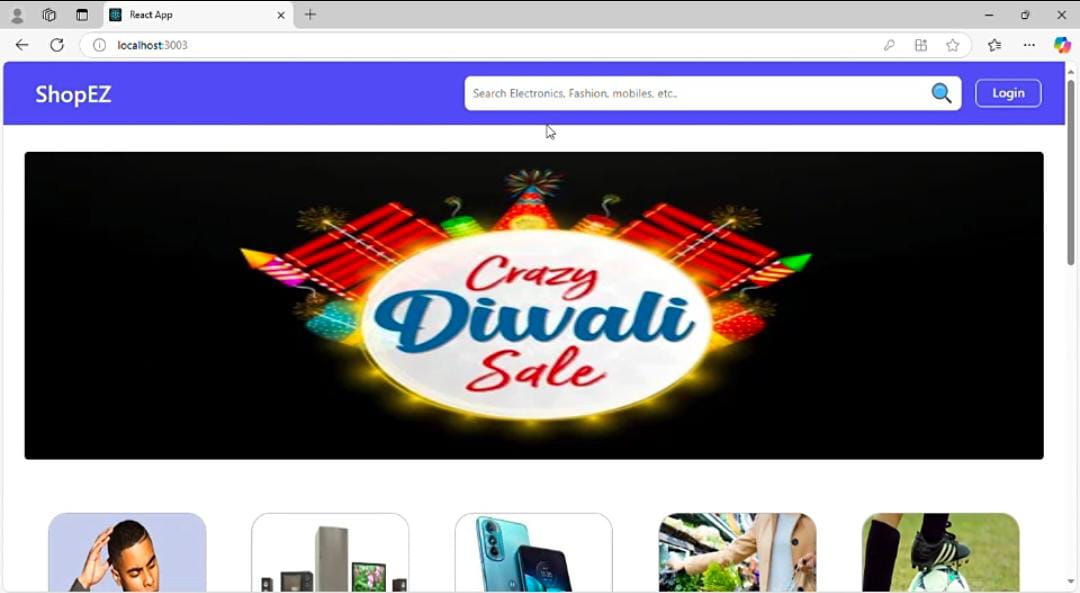
****

****

****

****

****

****

**KNOWN ISSUES :-**

**1.Slow Loading on Product Pages**

Description: Product pages take longer to load when images are high resolution.

Impact: Affects user experience on slower networks.

Planned Fix: Implement image optimization in future updates.

**2. Search Function Lag**

Description: The search feature experiences slight delays with complex queries.

Impact: Slows down search results, affecting the shopping experience.

Planned Fix: Optimize the search algorithm and database indexing.

**3. Payment Gateway Timeout Errors**

Description: Some users experience timeouts when confirming payments.

Impact: Prevents transaction completion, affecting user satisfaction and potential sales.

Planned Fix: Working with payment gateway provider to improve stability.

**4. Wishlist Feature Not Syncing**

Description: Items added to the wishlist may not consistently appear across devices.

Impact: Inconvenient for users accessing their wishlist on multiple devices.

Planned Fix: Improve session management for cross-device synchronization.

**FUTURE ENHANCEMENTS :-**

**1. Personalized Product Recommendations**

Description: Implement a recommendation engine to suggest products based on user browsing and purchase history.

Benefit: Enhances user experience by making product discovery easier and increasing engagement.

**2. Advanced Search and Filtering**

Description: Add advanced search options and dynamic filters (e.g., price range, ratings, color, brand).

Benefit: Helps users find products faster, improving satisfaction and usability.

**3. Loyalty Program and Rewards**

Description: Implement a points-based loyalty program that rewards repeat customers with discounts or special offers.

Benefit: Encourages customer retention and brand loyalty.

**4. Chatbot and Live Support**

Description: Add AI-powered chat support and live chat options to assist users with questions in real-time.

Benefit: Improves customer support, making it easier for users to resolve issues quickly.

**5. Improved Order Tracking**

Description: Add detailed order tracking and push notifications for status updates.

Benefit: Keeps customers informed and enhances their post-purchase experience.

**6. Enhanced Security Features**

Description: Implement multi-factor authentication (MFA) and biometric login for added security.

Benefit: Provides stronger account protection, boosting user trust and safeguarding data.