**Full Stack Development with MERN**

**Project Documentation format**

**1. Introduction**

* **Project Title:** Book Store
* **Team Members:**
  + A Pardhiv-lead,frontend
  + A Manideep-backend
  + K Keerthi-frontend,documentation
  + M Shrinivasan-backend,documentation

**2. Project Overview**

* **Purpose:**

Welcome to the literary haven of the digital age—introducing our revolutionary Book-Store Application, a masterpiece crafted with precision using the powerful MERN (MongoDB, Express.js, React, Node.js) Stack. Immerse yourself in a world where the love for reading converges seamlessly with cutting-edge technology, redefining the way bibliophiles explore, discover, and indulge in their literary pursuits.

Tailored for the modern book enthusiast, our MERN-based Book-Store Application seamlessly blends robust functionality with an intuitive user interface. From the joy of discovering new releases to the nostalgia of revisiting timeless classics, our platform promises an immersive reading experience customized to cater to your literary preferences.

Fueling the backbone of our application is MongoDB, ensuring a scalable and efficient database infrastructure that facilitates swift access to an extensive collection of literary works. Express.js, with its streamlined web application framework, establishes a responsive and efficient server, while Node.js ensures high-performance, non-blocking I/O operations—resulting in a seamless and enjoyable user experience.

At the heart of our Book-Store Application lies React, a dynamic and feature-rich JavaScript library. Dive into a visually enchanting and interactive interface where every click, search, and book selection feels like a literary journey. Whether you're exploring on a desktop, tablet, or smartphone, our responsive design ensures a consistent and delightful experience across all devices.

Say farewell to the constraints of traditional bookstores and embrace a new era of possibilities with our MERN Stack Book-Store Application. Join us as we transform how you connect with literature, making the discovery of your next favorite read an effortless and enriching experience. Get ready to turn the digital pages of a new chapter in reading, where every book is just a click away, and the literary world is at your fingertips. It's time to open the door to a future where the love for books meets the convenience of modern technology.

* **Features:**

**User Registration and Authentication:** Allow users to register accounts securely, log in, and authenticate their identity to access the book store platform.

**Book Listings:** Display a comprehensive list of available books with details such as title, author, genre, description, price, and availability status.

**Book Selection:** Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

**Purchase Process:** Allow users to add books to their cart, specify quantities, and complete purchases securely. Upon successful completion, an order is generated, and the inventory is updated accordingly.

**Order Confirmation:** Provide users with a confirmation page or notification containing details of their order, including book information, total price, and order ID.

**Order History:** Allow users to view their past and current orders, providing options to track shipments, review purchased books, and rate their shopping experience.

**3. Architecture**

* **Frontend:**

React components for each major functionality.

State management using Redux.

Routing with React Router.

API calls to the backend

* **Backend:**

Express.js server handling API requests.

Separate routes, controllers, and models for clarity.

Middleware for authentication and error handling.

MongoDB database with Mongoose schemas

* **Database:**

MongoDB for storing user, book, and order data.

Mongoose for schema definitions and interactions.

**4. Setup Instructions**

* **Prerequisites:** Nodejs,MongoDB,React,Express
* **Installation:**

To develop a full-stack Book Store App using React js, Node.js,Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download:<https://nodejs.org/en/download/>

• Installation instructions:<https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download:<https://www.mongodb.com/try/download/community>

• Installation instructions:<https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

• Installation: Open your command prompt or terminal and run the following

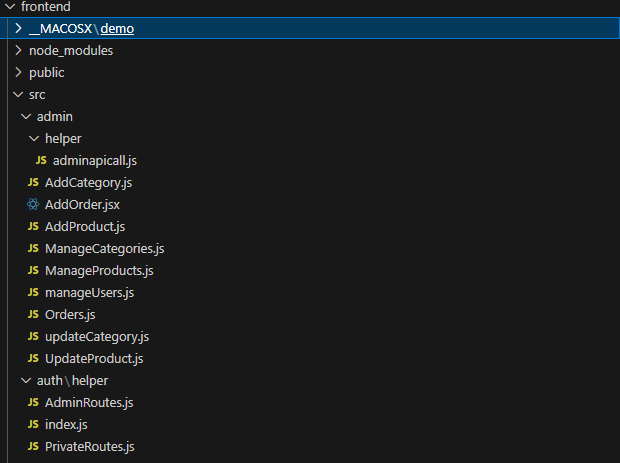
command: **npm install express**

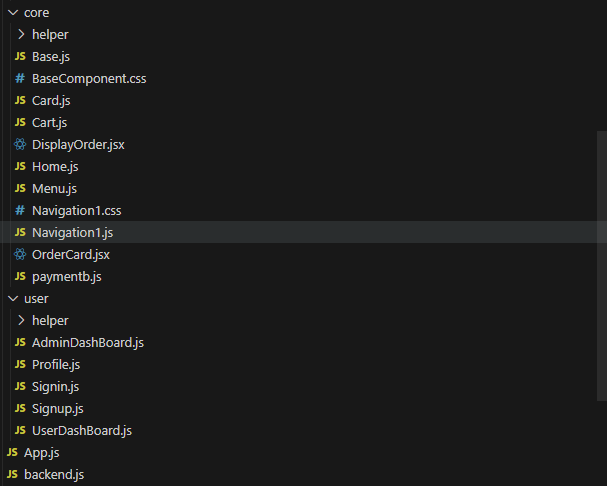
**React js: React** is a JavaScript library for building client-side applications.

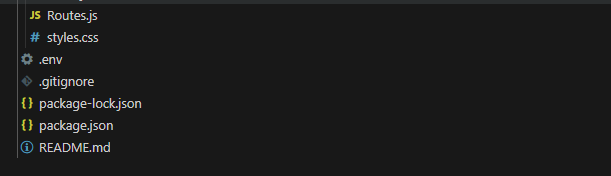
And Creating Single Page Web-Appliaction

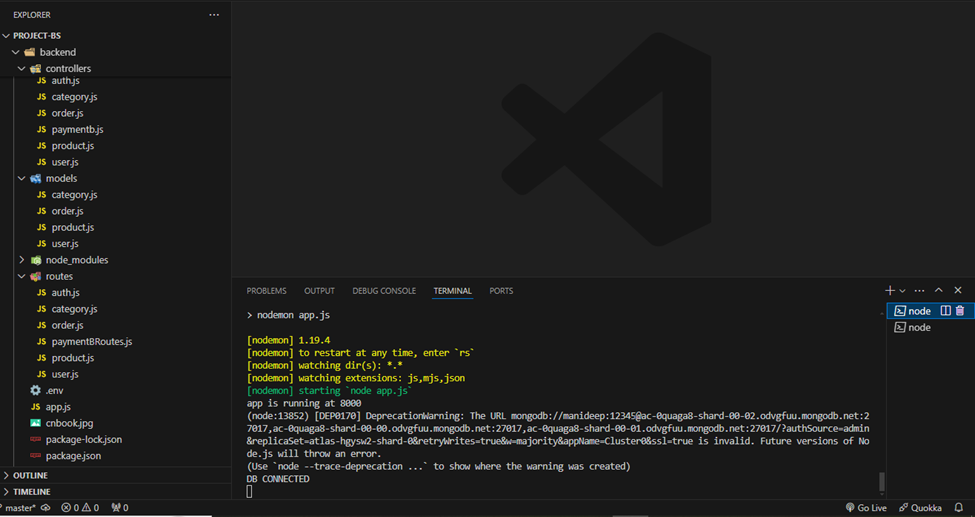
**5. Folder Structure**

* **Client:**

****

****

****

* **Server:**
* ****

**6. Running the Application**

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

**7. API Documentation**

* Document all endpoints exposed by the backend.
* The backend of the bookstore web application exposes several endpoints for various functionalities. The base URL for all endpoints is prefixed with `/api`.
* Authentication endpoints include:
* - Register User (`/signup`): A `POST` request that registers a new user by accepting a username, email, and password. The response includes a success message and user details.
* - Login User (`/login`) : A `POST` request for logging in an existing user, which also requires an email and password. Successful login returns user details and possibly a token for session management.
* Book-related endpoints include:
* - Get All Books (`/products`) : A `GET` request fetching a list of all books available in the store.
* - Get Book by ID (`/products/:productId`) : A `GET` request retrieving details of a specific book by its ID.
* This documentation helps in understanding the functionalities and interactions available within the backend of the bookstore application, crucial for both developers and maintainers.

**8. Authentication**

1. User Registration

- User submits registration details (username, email, password).

- Backend hashes the password and stores user details in the database.

2. User Login :

- User submits login credentials (email, password).

- Backend verifies credentials, generates a JWT if valid, and sends it to the frontend.

3. Storing Token :

- Frontend stores the JWT in local storage upon successful login.

Authorization

1. Protecting Routes :

- Middleware on the backend checks for a valid token in request headers.

- Routes requiring authentication are protected using this middleware.

2. Token Verification :

- Middleware verifies the JWT and allows access to protected routes if valid.

3. Frontend Token Check :

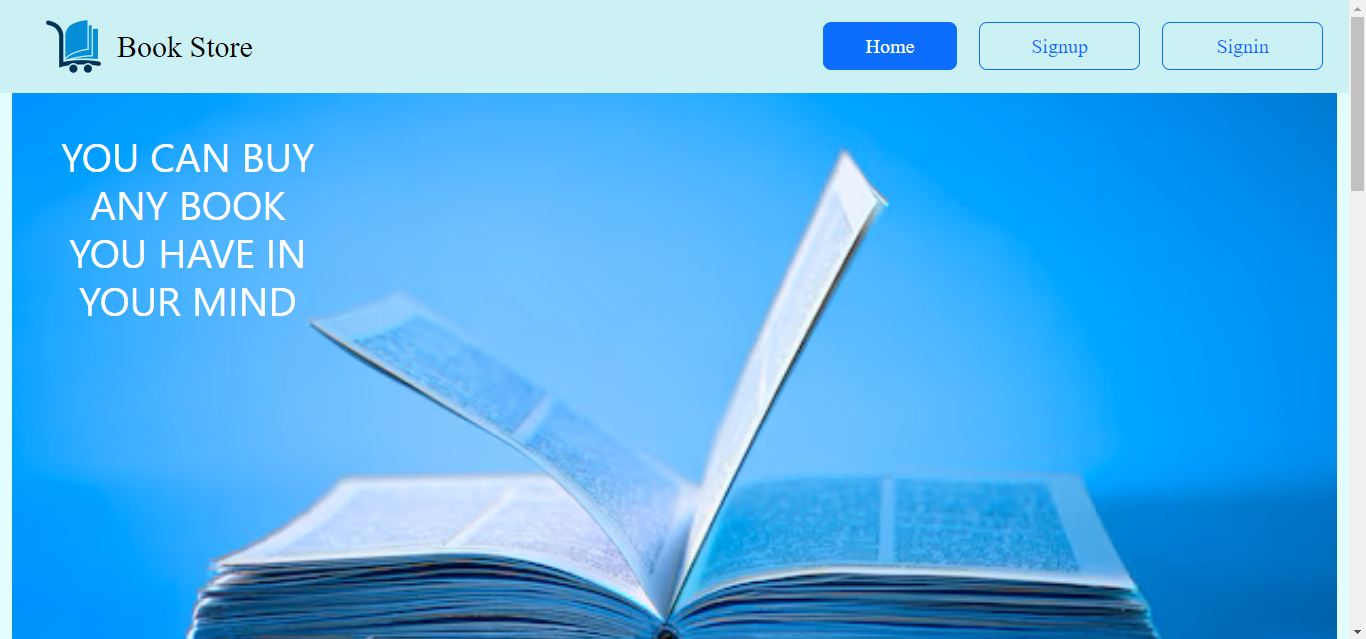
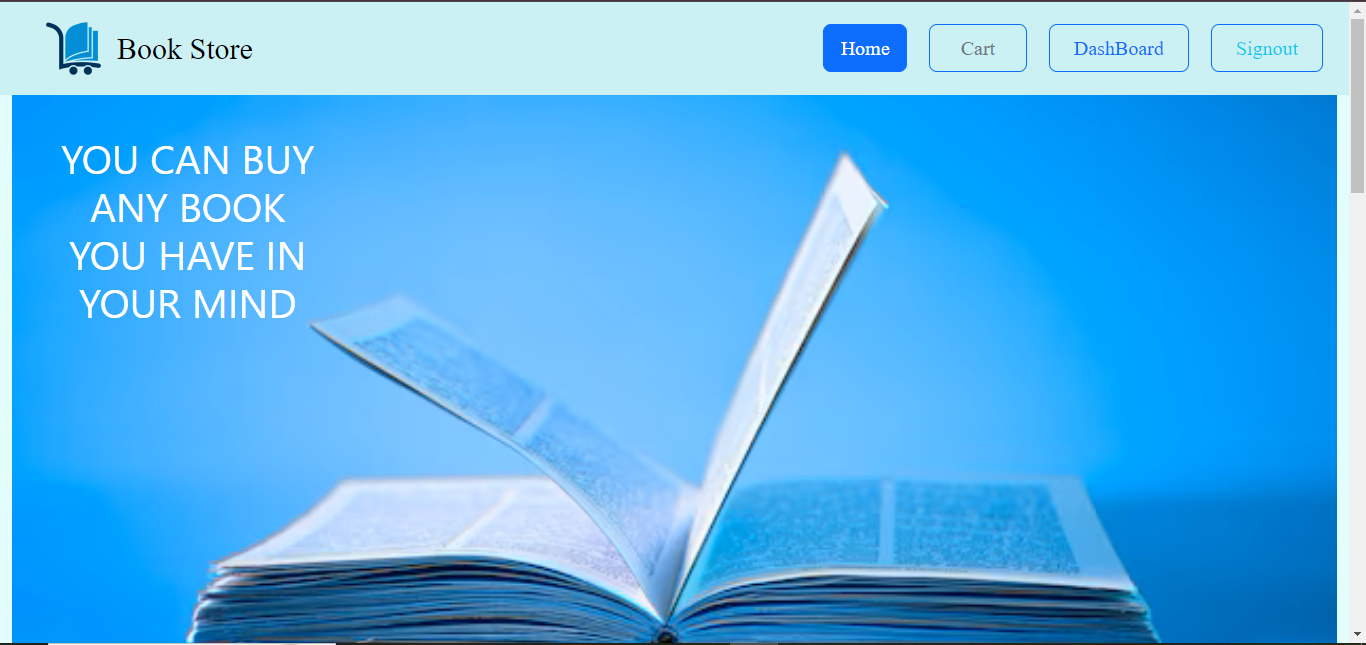
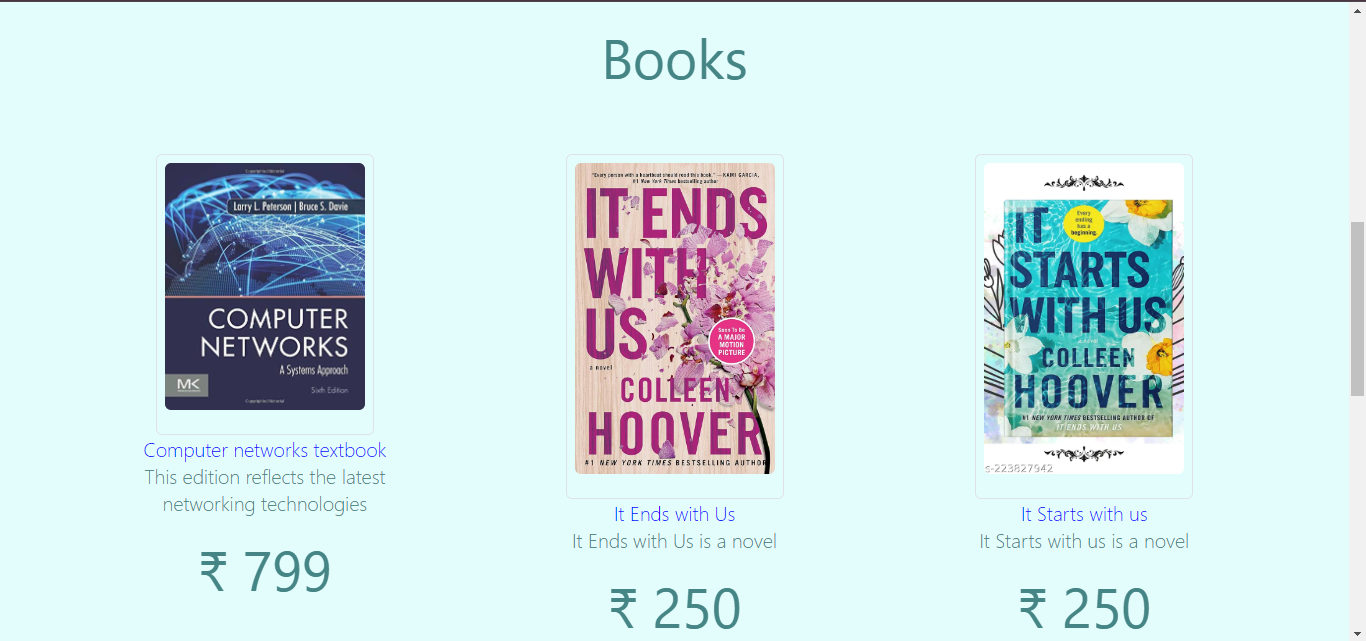
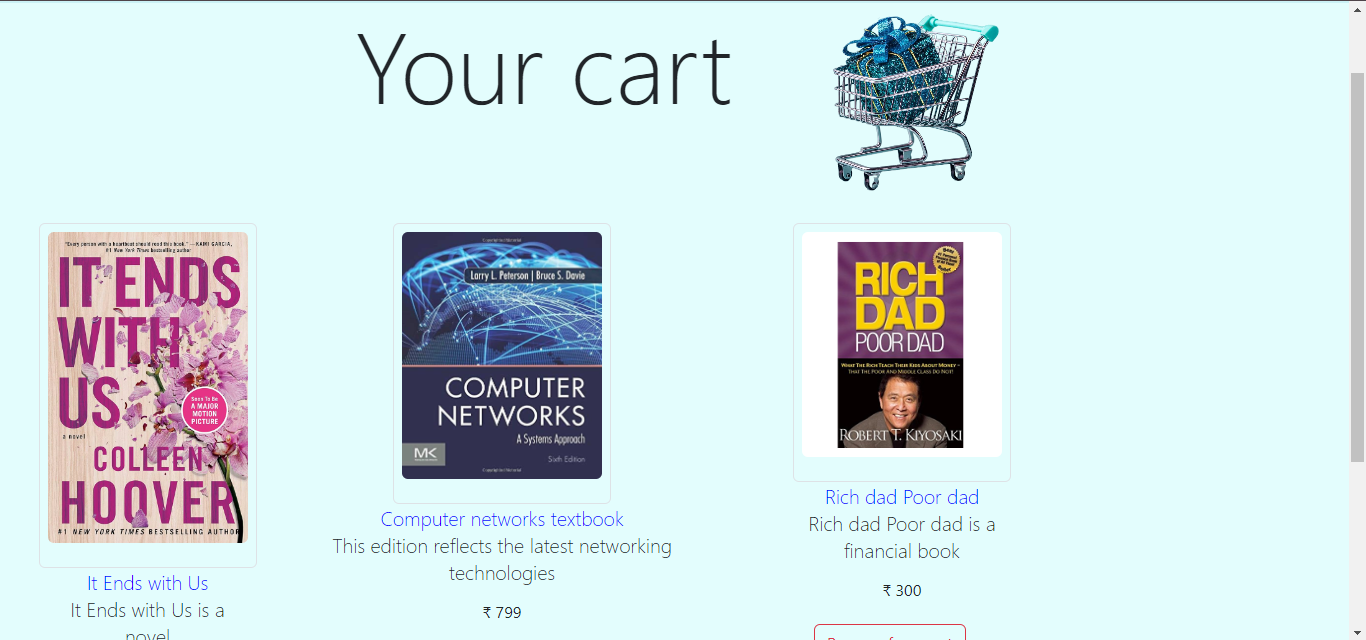
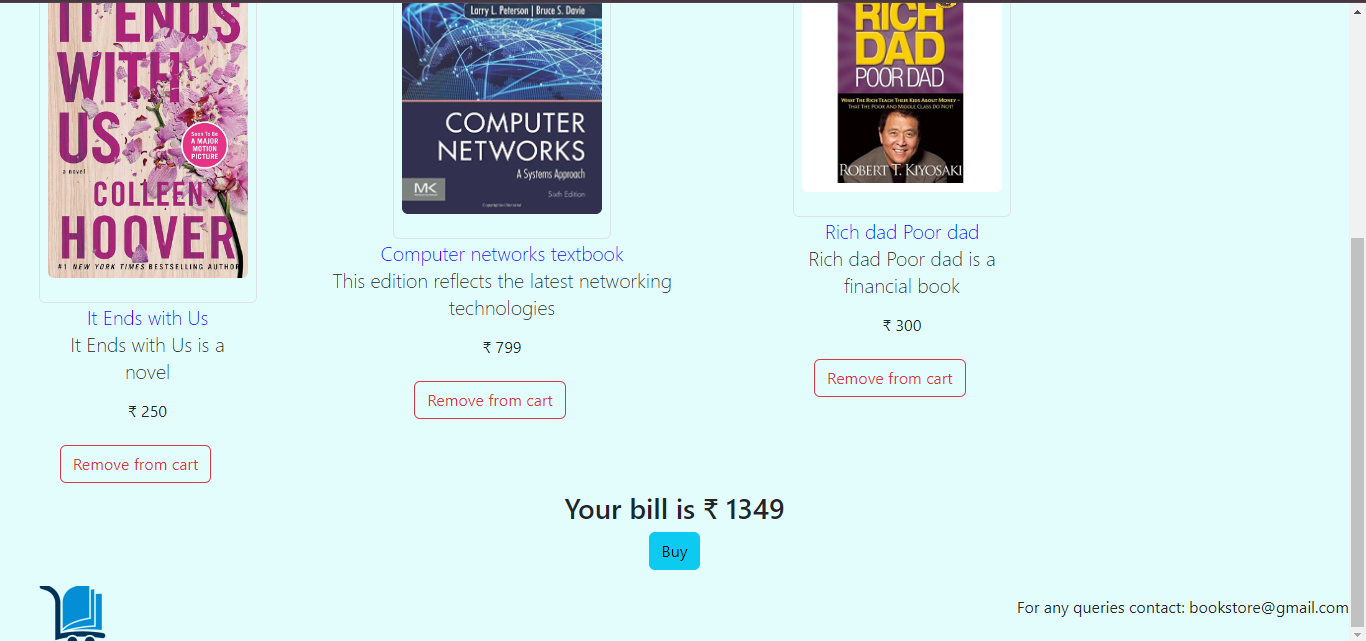
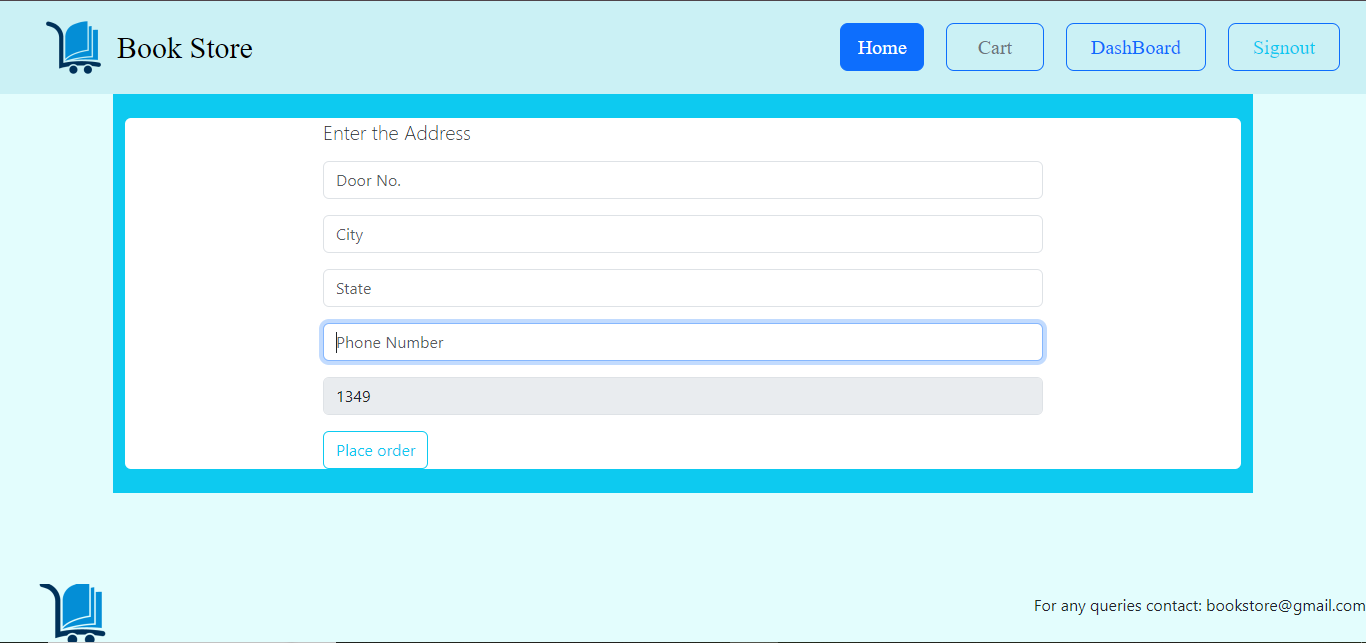
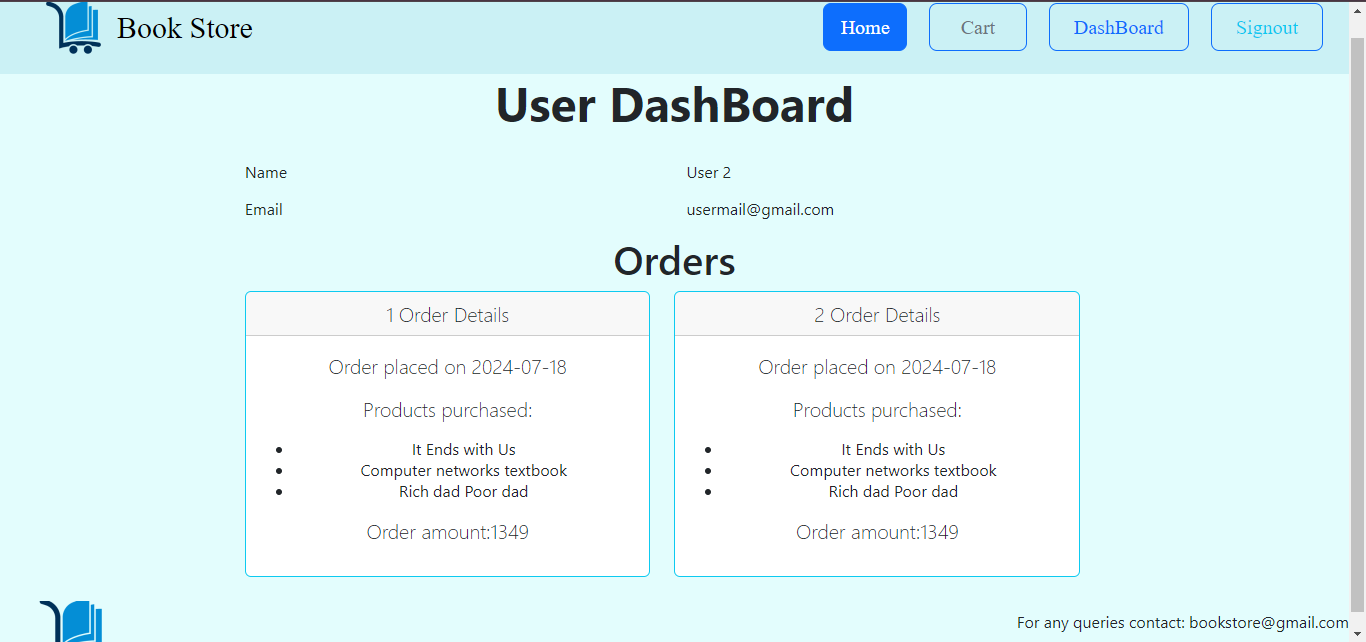
- Frontend checks for the presence of a token in local storage to determine if the user is logged in.

4. Sending Token with Requests :

- Token is included in the Authorization header for requests to protected routes.

This architecture ensures secure and efficient handling of user authentication and authorization in your MERN Stack Book-Store Application.

**9. User Interface**

* Provide screenshots or GIFs showcasing different UI features.
* 
* 
* 
* 
* 
* 
* 
* 

**10. Testing**

**Testing Report Summary**

**Unit Testing**

**- Frontend:** Tested React components for rendering and functionality using Jest and React Testing Library.

**- Backend:** Verified backend functions and endpoints.

**Integration Testing**

**- API Endpoints:** Tested data flow between frontend and backend using Postman.

**- Database Interactions:** Verified correct CRUD operations and data integrity.

**End-to-End (E2E) Testing**

**- User Scenarios:** Simulated user journeys (registration, book purchase) with Cypress.

**- Authentication/Authorization:** Tested protected routes and token usage.

**Usability Testing**

**- User Experience:** Gathered feedback and made improvements based on user testing sessions.

Overall, testing ensured the application is robust, secure, and user-friendly.

**11. Screenshots or Demo**

* https://drive.google.com/file/d/11tLqJJhbOW6\_UXJVnmLQHb7XNFulQmfE/view?usp=drivesdk.

**12. Known Issues**

* No Issues

**13. Future Enhancements**

1. **Recommendation System:**

* Implement a recommendation system to suggest books to users based on their browsing and purchase history, using machine learning algorithms.

2. **Advanced Search and Filtering:**

* - Enhance the search functionality with advanced filters (e.g., genre, author, price range) and implement fuzzy search to improve user experience in finding books.