**Full Stack Development with MERN**

**Frontend Development Report**

|  |  |
| --- | --- |
| Date | 10-07-2024 |
| Team ID | SWTID1719997659 |
| Project Name | BOOK-NEST |
| Maximum Marks | 10 |

**Project Title: BOOK-NEST**

Date: 10.07.2024

Prepared by: J.MARTHAND BHARGAV, G.HARSHITH, G.AAKASH AND V.NAGENDRA

**Objective**

The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the Book-Nest project.

**Technologies Used**

1. **Frontend Framework:**

**React.js**

* For building the user interface of the bookstore application.
* Manages the presentation layer efficiently.

1. **State Management:**

* **Redux (or Context API)**: For managing application state (e.g., user login state, shopping cart state).
* Redux can be used with React to maintain a predictable state container.

1. **UI Framework/Libraries:**

**Bootstrap**:

* **Description**: A popular front-end framework for developing responsive and mobile-first websites.
* **Usage**: Provides a wide range of pre-styled components and utilities for rapid UI development.

**Tailwind CSS**:

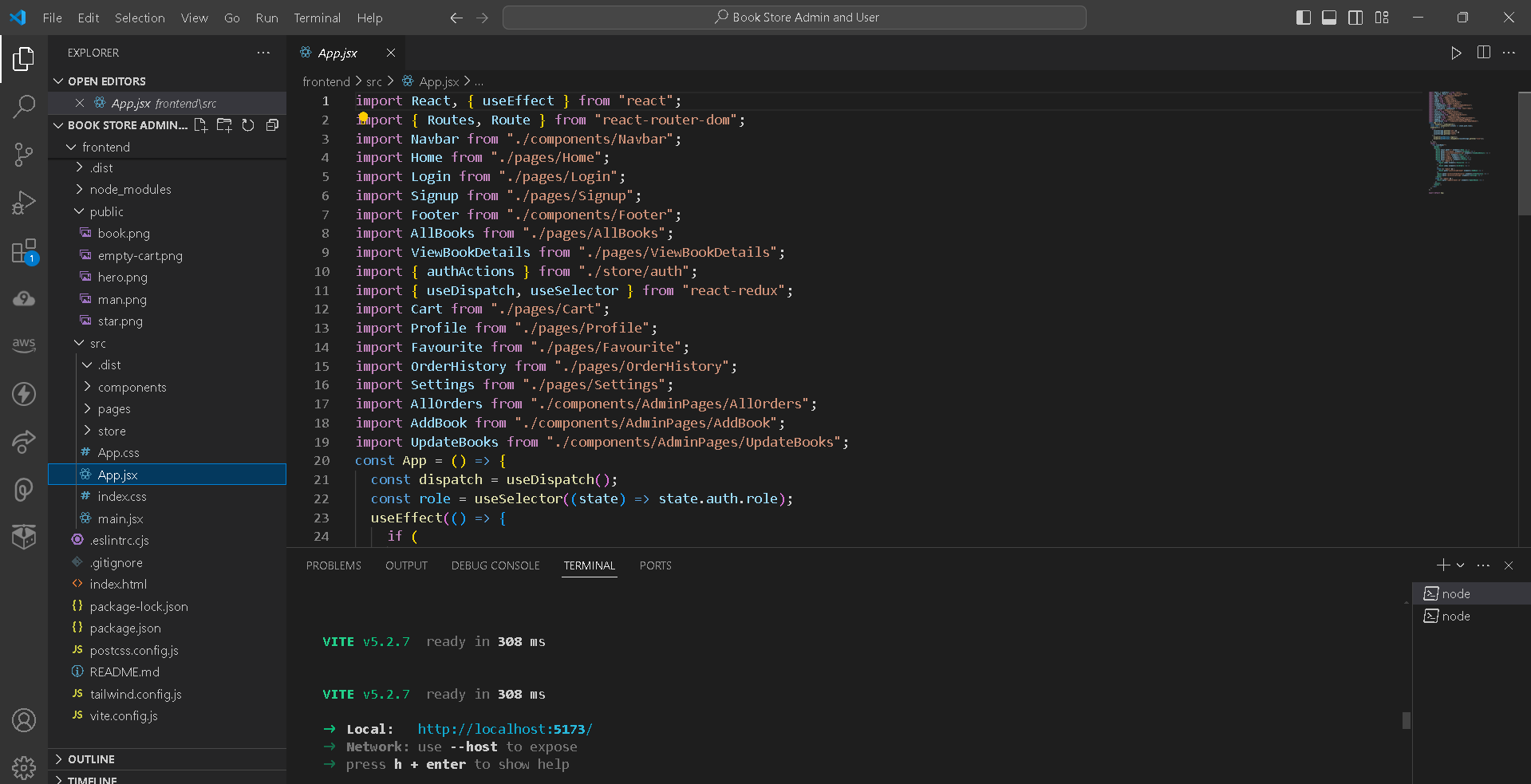
* **Description**: A utility-first CSS framework for rapidly building custom designs without leaving your HTML.
* **Usage**: Offers flexibility in styling and allows for highly customized designs without writing additional CSS.

1. **API Libraries:**

 **Axios**: For making HTTP requests from the frontend to the backend API endpoints.

 Facilitates communication between the client-side and server-side.

**Project Structure:**

****

**File Structure:**

The left side of the screenshot shows the directory structure.The key folders and files:

**contexts**: contains reusable UI components (e.g., book cards, navigation).

**hooks**: include custom React hooks (e.g., for state management).

**pages**: holds different pages/routes (e.g., home, book details).

**reducers**: these are related to state management (e.g., Redux reducers).

Other folders handle styling, assets, and utility functions.

**Key Components**

1. **App.js**
   * Responsible for routing and main application layout.
2. **/components**
   * Contains reusable UI components used across the application.
3. **/pages**
   * Includes different pages for Web App.

**Routing**

Routing is managed using React Router. Here are the main routes:

**1. Home Route**

* **Path**: /
* **Component**: This route typically displays the homepage of your bookstore application. It include featured books, promotions, or general information about the bookstore.

**2. Books Route**

* **Path**: /books
* **Component**: Displays a list of all books available in the bookstore.
* **Functionality**:
  + Fetches and displays a paginated list of books from the database.
  + Allows users to browse through different categories or genres of books.

**3. Book Details Route**

* **Path**: /books/:id
* **Component**: Shows detailed information about a specific book based on its unique identifier (id).
* **Functionality**:
  + Fetches and displays detailed information about a specific book, including its title, author, description, price, etc.
  + Allows users to add the book to their cart or wishlist.

**4. User Authentication Routes**

* **Path**: /login, /register, /logout
* **Components**:
  + /login: Displays a login form for users to authenticate themselves.
  + /register: Displays a registration form for new users to create an account.
  + /logout: Logs out the authenticated user and redirects to the homepage.
* **Functionality**:
  + Handles user authentication using JWT or session-based authentication.
  + Validates user credentials and manages user sessions securely.

**5. Cart Route**

* **Path**: /cart
* **Component**: Displays the items currently added to the user's shopping cart.
* **Functionality**:
  + Shows a summary of items in the cart with options to update quantities or remove items.
  + Calculates and displays the total price of items in the cart.
  + Allows users to proceed to checkout or continue shopping.

**6. Checkout Route**

* **Path**: /checkout
* **Component**: Provides a checkout process for users to complete their purchases.
* **Functionality**:
  + Displays shipping and billing information forms.
  + Calculates taxes, shipping costs, and provides a total cost summary.
  + Integrates with payment gateways (e.g., Stripe, PayPal) to process payments securely.

**7. Admin Routes**

* **Path**: /admin/dashboard, /admin/books, /admin/orders, etc.
* **Components**: Various components for managing administrative tasks.
* **Functionality**:
  + Provides access to administrative features such as adding, editing, or deleting books.
  + Manages orders, user accounts, and other administrative tasks.
  + Requires authentication and authorization for access.

**8. Error Handling Routes**

* **Path**: Catch-all route for handling 404 errors
* **Component**: Displays a "Not Found" page or redirects to the homepage.
* **Functionality**:
  + Catches undefined routes or invalid URLs and provides a user-friendly error message.

**State Management (If Applicable)**

In my MERN bookstore application, Redux facilitates centralized state management across components. It ensures that application state (such as user authentication status, shopping cart items, and book data) remains consistent and easily accessible throughout the application. By Redux, we maintain a predictable state container and efficiently manage state updates across my React components, enhancing both performance and maintainability of our application.

Top of Form

Bottom of Form

**Integration with Backend**

Integrating our MERN bookstore application involves setting up backend API routes with Express.js and MongoDB for data storage, while React.js handles the frontend user interface and interactions. By establishing HTTP communication between the two using Axios (or Fetch API) and ensuring consistent data flow, we create a seamless user experience with robust backend functionality. This approach ensures that our application can handle various operations such as fetching books, user authentication, and more, effectively integrating both frontend and backend components.

**User Interface (UI) Design**

Here’s the structured approach of designing the UI components based on typical functionalities in our bookstore application:

**1. Home Page**

* **Purpose**: Introduce users to your bookstore, highlight featured books, promotions, and navigation options.

**Components**:

* **Header**: Contains logo, navigation links (e.g., Home, Books, Login/Register), and possibly a search bar.
* **Featured Books Section**: Showcase visually appealing images or carousels of selected books with brief descriptions or ratings.
* **Categories or Genres**: Display links or buttons to browse books by categories (e.g., Fiction, Non-Fiction, Best Sellers).
* **Footer**: Includes contact information, social media links, and additional navigation links.

**2. Books Listing Page**

* **Purpose**: Display a list of all available books with options for filtering and sorting.

**Components**:

* **Sidebar Filters**: Allow users to filter books by categories, authors, genres, etc.
* **Book Cards/Grid**: Each book displayed as a card or grid item showing the book cover, title, author, and price.
* **Pagination**: If many books are available, implement pagination to manage large data sets.

**3. Book Details Page**

* **Purpose**: Provide detailed information about a selected book.

**Components**:

* **Book Cover Image**: High-resolution image of the book cover.
* **Title, Author, Description**: Display the book’s title, author name, and a brief synopsis or description.
* **Add to Cart/Wishlist**: Buttons or icons to add the book to the shopping cart or wishlist.
* **Reviews and Ratings**: If available, show user reviews and ratings for the book.
* **Similar Books**: Recommend other books similar to the selected one for cross-selling.

**4. User Authentication Pages**

* **Purpose**: Allow users to register, log in, and manage their profiles.

**Components**:

* **Login/Register Forms**: Input fields for username, password, and buttons to submit forms.
* **Profile Management**: Once logged in, users can view/edit their profile information (e.g., name, email, address).

**5. Shopping Cart Page**

* **Purpose**: Display items added to the shopping cart and allow users to proceed to checkout.

**Components**:

* **List of Cart Items**: Each item displayed with its image, title, quantity selector, price, and subtotal.
* **Total Summary**: Calculate and display the total price of items in the cart, including taxes and shipping costs.
* **Proceed to Checkout Button**: Button that redirects users to the checkout page.

**6. Checkout Process**

* **Purpose**: Guide users through the payment and order confirmation process.

**Components**:

* **Shipping Information**: Form fields for entering shipping address and preferred shipping method.
* **Payment Information**: Secure form for entering credit card details or integrating with payment gateways (e.g., Phonepe,Gpay etc..).
* **Order Summary**: Review of items in the cart, total cost, and confirmation button.

**7. Responsive Design**

**Components**:

* **Responsive Layout**: We Used responsive CSS frameworks (e.g., Bootstrap, Material-UI) or CSS Grid/Flexbox to adjust the layout based on screen size.
* **Mobile Navigation**: Consider a collapsible or drawer-style menu for mobile devices to conserve screen space.
* **UI Libraries used**: Bootstrap, Material-UI for pre-styled UI components and responsive design.