**Full Stack Development with MERN**

**Frontend Development Report**

|  |  |
| --- | --- |
| Date | 15/07/2024 |
| Team ID | SWTID1720014974 |
| Project Name | BookStore |
| Maximum Marks |  |

**Project Title: BookStore**

Date: 15/07/2024

Prepared by: Manideep , Pardhiv , Shrinivasan , Keerthi

**Objective**

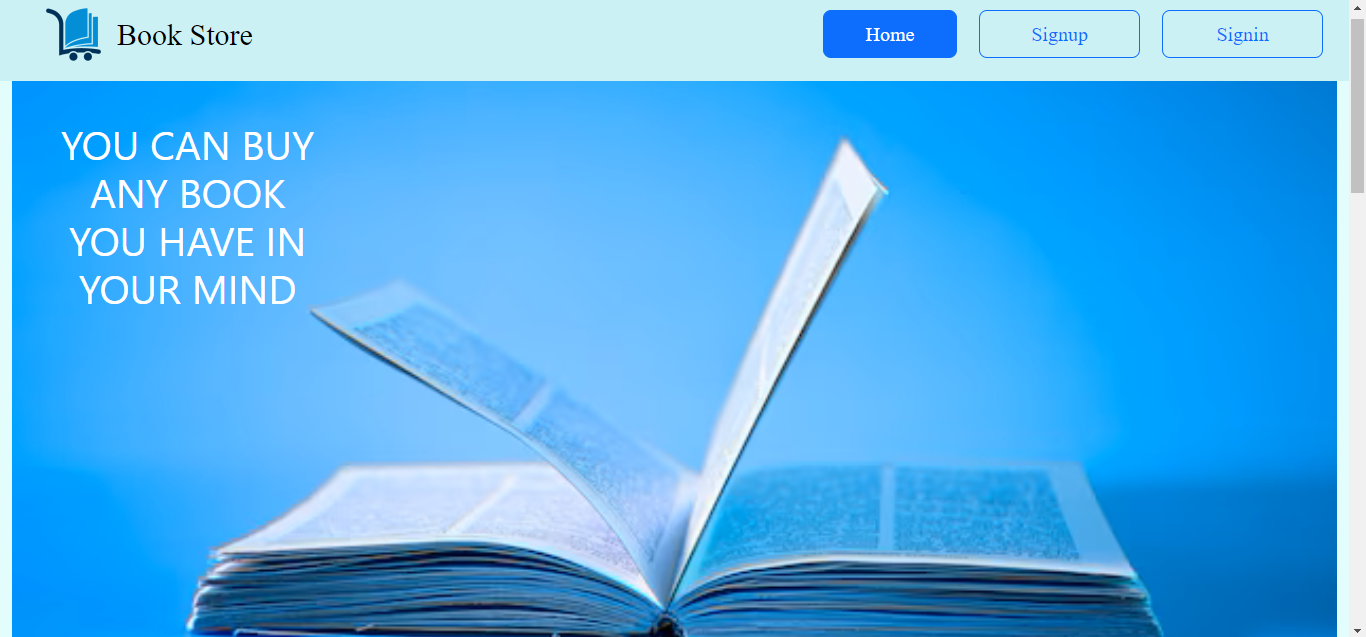
The objective of this report is to document the frontend development progress and key aspects of the user interface implementation for the BookStore project.

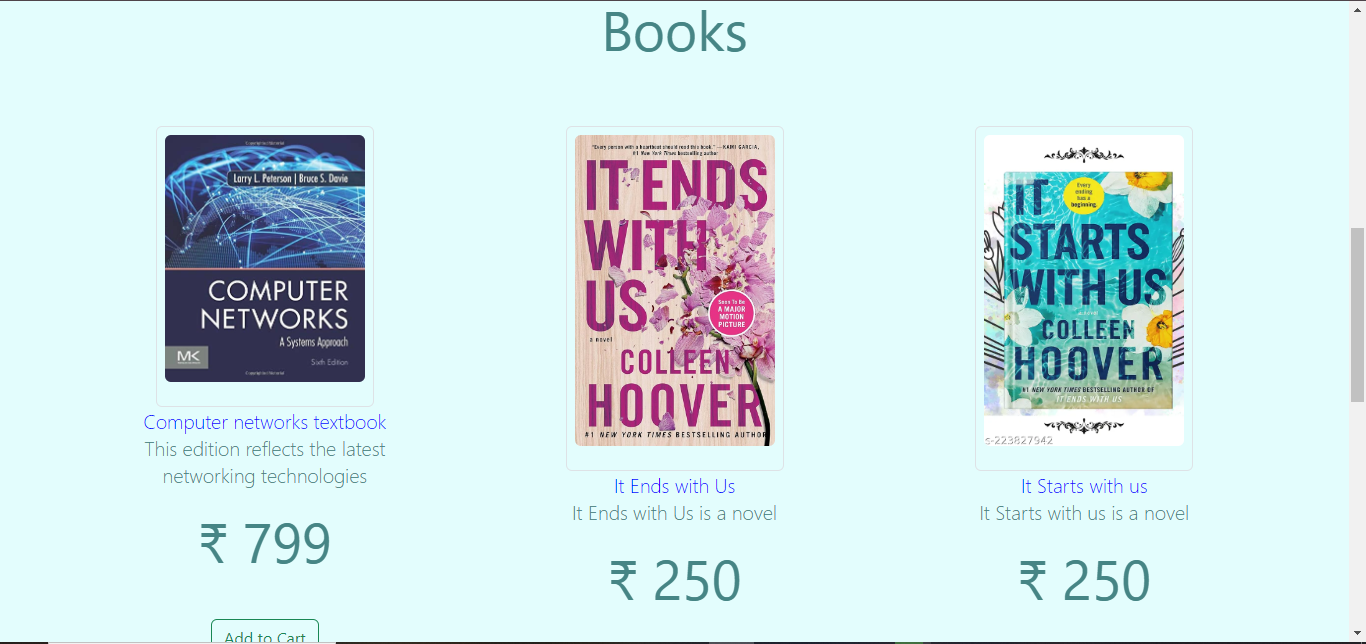
**Technologies Used**

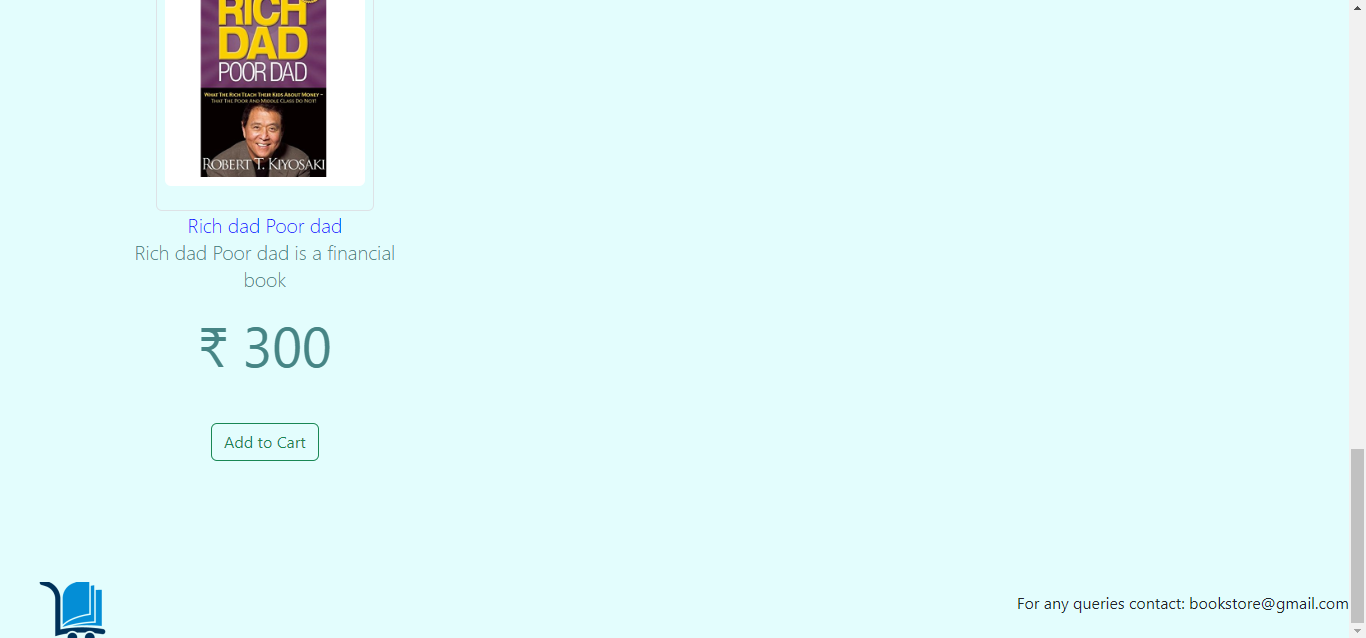
* **Frontend Framework:** React.js
* **State Management:** React hooks (useState, useEffect)
* **UI Framework/Libraries:** Bootstrap
* **API Libraries:** Custom implementation

**Project Structure**

Give Screenshot of Frontend Project with explanation

****

****

****

**Key Components**

1. **App.js**
   * Responsible for routing and main application layout.
2. **/components & /pages**:
   * /admin - Contains admin-specific components and helpers.
     + Helper:
       - adminapicall.js: Handles API calls related to admin operations.
       - AddCategory.js: Component for adding a new category.
       - AddOrder.js: Component for adding a new order.
       - AddProduct.js: Component for adding a new product.
       - ManageCategories.js: Component for managing categories.
       - ManageProducts.js: Component for managing products.
       - Orders.js: Component for managing orders.
       - updateCategory.js: Component for updating a category.
       - UpdateProduct.js: Component for updating a product.
   * /auth – Contains components related to authentication and route protection
     + Helper:
       - AdminRoutes.js: Handles protected routes for admin.
       - index.js: Authentication helpers.
       - PrivateRoutes.js: Handles protected routes for users.
   * /core - Contains core components and helper functions that are essential for the overall functionality of the application
     + Helper:
       - cartHelper.js: Helper functions for cart operations.
       - coreapicalls.js: General API call handlers.
       - ImageHelper.js: Handles image-related operations.
       - orderHelper.js: Helper functions for order operations.
       - paymenthelper.js: Handles payment-related operations.
     + Base.js: Base layout component.
     + Card.js: Component for displaying product cards.
     + Cart.js: Component for the shopping cart.
     + Home.js: Home page component.
   * /user - Contains user-specific components and helpers.
     + Helper:
       - AdminDashBoard.js: Component for the admin dashboard.
       - Profile.js: Component for user profile management.
       - Signin.js: Component for user sign-in.
       - Signup.js: Component for user sign-up.
       - UserDashBoard.js: Component for the user dashboard.
     + App.js: Main application component.
     + backend.js: Configuration for backend API calls.
     + index.js: Entry point for the React application.
     + Routes.js: Defines application routing.
     + styles.css: Global CSS styles.

**Routing**

 **Home Route**

* **Path**: /
* **Component**: Home

 **Signup Route**

* **Path**: /signup
* **Component**: Signup

 **Signin Route**

* **Path**: /signin
* **Component**: Signin

 **User Dashboard Route**

* **Path**: /user/dashboard
* **Component**: UserDashBoard

 **Admin Dashboard Route**

* **Path**: /admin/dashboard
* **Component**: AdminDashBoard

 **Add Category Route**

* **Path**: /admin/create/category
* **Component**: AddCategory

 **Manage Categories Route**

* **Path**: /admin/categories
* **Component**: ManageCategories

 **Add Product Route**

* **Path**: /admin/create/product
* **Component**: AddProduct

 **Manage Products Route**

* **Path**: /admin/products
* **Component**: ManageProducts

 **Update Product Route**

* **Path**: /admin/product/update/:productId
* **Component**: UpdateProduct

 **Update Category Route**

* **Path**: /admin/category/update/:categoryId
* **Component**: UpdateCategory

 **Cart Route**

* **Path**: /cart
* **Component**: Cart

 **Add Order Route**

* **Path**: /admin/add/order
* **Component**: AddOrder

**State Management (If Applicable)**

In this project, state management is handled using React's built-in hooks such as useState and useEffect.

State Management Approach:

useState:

The useState hook is used to create and manage state variables in functional components. It initializes state variables and provides a function to update these variables.

useEffect:

The useEffect hook is used to handle side effects in functional components, such as fetching data from APIs, subscribing to events, or manually changing the DOM. It runs after the component renders and can be configured to run only when certain dependencies change.

**Integration with Backend**

The frontend communicates with the backend APIs hosted on [backend URL]. Key endpoints include:

* **GET /api/category/{categoryId}** - Retrieves data for a specific category.
* **PUT /api/category/{categoryId}** - Updates a specific category.
* **POST /api/category/create** - Adds a new category.
* **GET /api/categories** - Retrieves a list of all categories.
* **POST /api/product/create** - Adds a new product.
* **GET /api/product/{productId}** - Retrieves data for a specific product.
* **PUT /api/product/{productId}** - Updates a specific product.
* **GET /api/products** - Retrieves a list of all products.
* **POST /api/order/create** - Creates a new order.
* **GET /api/orders** - Retrieves a list of all orders.
* **POST /api/user/signup** - Handles user signup.
* **POST /api/user/signin** - Handles user authentication.

**User Interface (UI) Design**

* The UI design follows the following principles:
  + - Clarity: The interface is designed to be clear and understandable. Each element is placed with purpose, ensuring that users can easily understand and navigate the application without confusion. Icons, labels, and instructions are straightforward, contributing to an intuitive user experience.
    - Simplicity: The design avoids unnecessary complexity. By focusing on the essential features and functionality, the interface remains clean and uncluttered. This simplicity helps users focus on their tasks without being overwhelmed by too many options or visual noise.
    - Consistency: Consistent design elements are used throughout the application. Colours, fonts, and button styles remain uniform.
    - Responsiveness: The UI is designed to be fully responsive, ensuring that the application works seamlessly on various devices and screen sizes. Whether accessed on a desktop, tablet, or mobile phone, the interface adjusts appropriately to provide an optimal viewing experience.
* The application uses Bootstrap as the primary UI framework. Bootstrap's pre-styled components and responsive grid system help ensure a consistent and mobile-friendly layout across different screen sizes and devices

**Third-Party Integrations**

React Router:

Used for handling client-side routing.

Bootstrap:

Used for styling and layout.