Project Report: Book Store App (MERN Stack)

1. Introduction

The **Book Store App** is an online platform that allows users to browse, search, and purchase books. Built using the **MERN stack** (MongoDB, Express.js, React.js, Node.js), the app provides a seamless and user-friendly interface for both customers and administrators. The application features secure user authentication, book management, and a shopping cart system, making it a complete solution for online book shopping.

The goal of this project was to build a scalable and easy-to-use platform that integrates with a robust backend to manage user data, book information, and transactions. This project demonstrates how the MERN stack can be effectively used to build a full-stack e-commerce application.

2. Project Overview

The **Book Store App** provides a platform where users can:

- **Browse** a wide range of books categorized by genres.
- **Search** for books by title, author, or ISBN.
- **Add** books to their shopping cart and proceed with checkout.
- **Create accounts**, log in, and manage their profiles.
- **Make purchases** via a secure payment system (could be integrated in future versions).

For administrators, the app offers the ability to:

- **Add, edit, and delete** books.
- **View orders** and manage stock levels.

Key Features:

- **User Authentication**: Login/Signup using JWT for secure authentication.
- **Book Catalog**: View books categorized by genre, search by author or title.
- **Shopping Cart**: Add/remove books and proceed to checkout.
- **Admin Panel**: Manage books, users, and orders.
- **Responsive UI**: The interface is fully responsive to provide a great experience on both desktop and mobile devices.

3. Architecture

The application follows a **client-server architecture**, which is divided into two primary sections:

Frontend:

- Built using **React.js** for the dynamic user interface.
- **Redux** is used for state management, especially to manage the shopping cart, user sessions, and book data.
- **React Router** handles navigation between pages (e.g., Home, Book Details, Cart, Profile).

Backend:

- The backend is built using **Node.js** and **Express.js** for routing and handling API requests.
- **MongoDB** is used as the database to store user data, book details, and orders.
- **JWT Authentication** is used to authenticate users securely.

Data Flow:

1. Users interact with the frontend, which sends requests to the backend API (e.g., to view books, add items to the cart).
2. The backend processes these requests, interacts with MongoDB to fetch or modify data, and returns the response to the frontend.
3. Real-time updates (like cart updates) are reflected immediately on the UI.

4. Setup Instructions

Follow the steps below to set up the Book Store App on your local machine:

Prerequisites:

- **Node.js** (v14 or later)
- **MongoDB** (local or MongoDB Atlas)
- **Git** (for version control)

Installation Steps:

1. **Clone the repository**:

   bash
   Copy code
   git clone https://github.com/yourusername/book-store-app.git
   cd book-store-app

2. **Install Backend Dependencies**: Navigate to the backend folder and install the necessary dependencies:

   bash
   Copy code
   cd backend
   npm install

3. **Set up MongoDB**:
    - Create a database in **MongoDB** (e.g., bookstoreDB).
    - Create a .env file in the backend folder and configure your environment variables:

      bash
      Copy code
      MONGO_URI=mongodb://localhost:27017/bookstoreDB
      JWT_SECRET=your_jwt_secret_key

4. **Install Frontend Dependencies**: Navigate to the frontend folder and install the required dependencies:

   bash
   Copy code
   cd frontend
   npm install

5. **Run the Application**:
    - To start the backend server, go to the backend folder:

      bash
      Copy code
      cd backend
      npm start

    - To start the frontend server, go to the frontend folder:

      bash
      Copy code
      cd frontend
      npm start

6. The app will be available at http://localhost:3000.

5. Folder Structure

The project is divided into two main directories: frontend (React app) and backend (Node.js API).

Frontend Folder Structure:

```
php
Copy code
frontend/
├── src/
│   ├── assets/       # Images, icons, stylesheets
│   ├── components/   # Reusable UI components (e.g., Navbar, Footer)
│   ├── pages/        # Pages (e.g., Home, BookDetails, Cart, Checkout)
│   ├── redux/        # Redux actions, reducers, and store
│   ├── App.js        # Main app component
│   └── index.js      # Entry point for React app
└── public/
    └── index.html    # HTML template
```

Backend Folder Structure:

```
bash
Copy code
backend/
├── controllers/      # Request handlers for routes (e.g., bookController.js)
├── models/           # Mongoose models for MongoDB collections (e.g., User, Book, Order)
├── routes/           # API routes (e.g., authRoutes, bookRoutes)
├── middleware/       # Middleware for JWT verification, error handling
├── config/           # Configuration files (e.g., database connection)
├── server.js         # Entry point for the Node.js server
```

└── .env # Environment variables

6. Running the Application

Once the dependencies are installed, you can run the application locally by following these steps:

1. **Start the Backend**: In the backend directory, run:

   bash
   Copy code
   npm start

2. **Start the Frontend**: In the frontend directory, run:

   bash
   Copy code
   npm start

Your app should now be running locally at http://localhost:3000.

7. API Documentation

The backend exposes several RESTful APIs for various functionalities.

Authentication API:

- **POST /api/auth/register**: Register a new user.
- **POST /api/auth/login**: Login an existing user and return a JWT token.
- **GET /api/auth/logout**: Logout the current user.

Book API:

- **GET /api/books**: Get a list of all books.
- **GET /api/books/**

  : Get details of a specific book.

- **POST /api/books**: Add a new book (Admin only).
- **PUT /api/books/**

  : Edit a book's details (Admin only).

- **DELETE /api/books/**

  : Delete a book (Admin only).

Cart API:

- **POST /api/cart**: Add a book to the user's cart.
- **GET /api/cart**: View items in the user's cart.
- **DELETE /api/cart/**

  : Remove a book from the user's cart.

Order API:

- **POST /api/orders**: Place an order (Checkout).
- **GET /api/orders**: Get all orders (Admin only).
- **GET /api/orders/**

  : Get details of a specific order.

8. Authentication

The app uses **JWT (JSON Web Token)** for user authentication:

- During login or registration, a JWT token is issued to the user.
- The token is stored in the local storage on the client-side.
- Subsequent requests to protected routes are made with the JWT token in the Authorization header.

JWT Verification:

- Middleware (authMiddleware.js) checks the validity of the JWT token on each protected route.

9. User Interface

The UI is built using **React.js** with a responsive design to ensure compatibility across devices. Main features of the UI include:

- **Homepage**: Displays featured books and categories.
- **Book Details Page**: Displays detailed information about each book.
- **Cart Page**: Displays items added to the cart, quantity adjustment, and total price.
- **Checkout Page**: Allows users to enter shipping details and proceed to payment.

10. Testing

The app has been manually tested for functionality and responsiveness across devices. Key features like user authentication, book search, adding books to the cart, and checkout have been thoroughly tested.

Testing Tools:

- **Jest**: For backend unit and integration tests.
- **React Testing Library**: For testing React components.
- **Postman**: For testing API endpoints.

11. Known Issues

- **Payment Integration**: The payment gateway integration is not yet implemented.
- **Search Performance**: The search function could be optimized for faster results as the book catalog grows.
- **Admin Panel**: Some admin functionalities (like user management) need additional work.