**Book Store - Detailed Project Report**

**Introduction**

* **Project Title**: LiteraHub: A central hub for literature and books.
* **Team Members**:
* **Harini T.E - Team Lead, Backend Development.**
* **Pravena.N - Frontend Development.**
* **Mrudhula M - Database Design and Integration.**
* **Nandhini T.S - Testing**
* **Sri harini - Quality Assurance.**

**Project Overview**

The Book-Store Application aims to provide a digital platform for book lovers to discover, browse, and purchase books conveniently. It offers an extensive collection of books with features like user registration, authentication, book browsing, cart management, and order tracking.

* User Registration and Authentication.
* Book Listings and Search.
* Order Processing and History.
* Cart Management.
* Responsive Design.

**Architecture**

**Frontend:**

* **HTML**: Structures pages with forms, buttons, and sections for books, registration, and cart.
* **CSS**: Styles pages and ensures responsive design across all devices.
* **JavaScript**: Adds interactivity by handling actions and dynamically updating webpage content.
* **JSON**: Used for handling structured data between the frontend and backend, such as book lists and user profiles.
* **Git Ignore**: Prevents unnecessary files (e.g., system files or sensitive credentials) from being pushed to version control repositories.

**Backend:**

* **Node.js:** A JavaScript runtime environment for building server-side applications.
* **Express.js:** A minimalist web framework for Node.js.
* **MongoDB:** A NoSQL database for storing user data, doctor profiles, and appointment information.
* **Moment.js:** A JavaScript library for manipulating dates and times.

**Database:**

* **Users**: Stores user details like name, email, password, and contact information.
* **Books**: Stores book details like title, author, genre, price, and reviews.
* **Orders**: Tracks orders, including book IDs, user ID, quantities, date, and status.

**Setup Instructions**

**Prerequisites:**

* Node.js and npm (or yarn) installed
* MongoDB database

1. A modern web browser (Chrome, Firefox, etc.) https://github.com/HariniElamaran/Book-Store.git

**Installation:**

1. Clone the repository: git clone Install dependencies:
   * cd client
   * npm install
   * cd server
   * npm install
2. Set up environment variables:
   * Create a .env file in both the client and server directories.
   * Add the following variables:
     + REACT\_APP\_API\_URL: PSD:\naanmudhalvan\PROJECT\Harini\server> node index.js
     + MONGODB\_URI: mongosh:Harini
3. Start the development server:
   * **Frontend:** **npm start** in the client directory
   * **Backend:** **node index.js** in the server directory

**Folder Structure**

* **Client (Frontend)**:
  + **/src**: Contains React components, styles, and assets.
  + **/public**: Public assets like images and the index.html file.
  + **/components**: Reusable UI components such as Header, Footer, Book Card, and Cart.
* **Server (Backend)**:
  + **/models**: Defines Mongoose models for Users, Books, and Orders.
  + **/routes**: Contains Express route handlers for API endpoints.
  + **/controllers**: Implements logic for user authentication, book management, and order processing.

**Running the Application**

* **Frontend:  
   Navigate to the client directory and run: npm start**
* **Backend:  
   Navigate to the server directory and run: npm start**

**API Documentation**

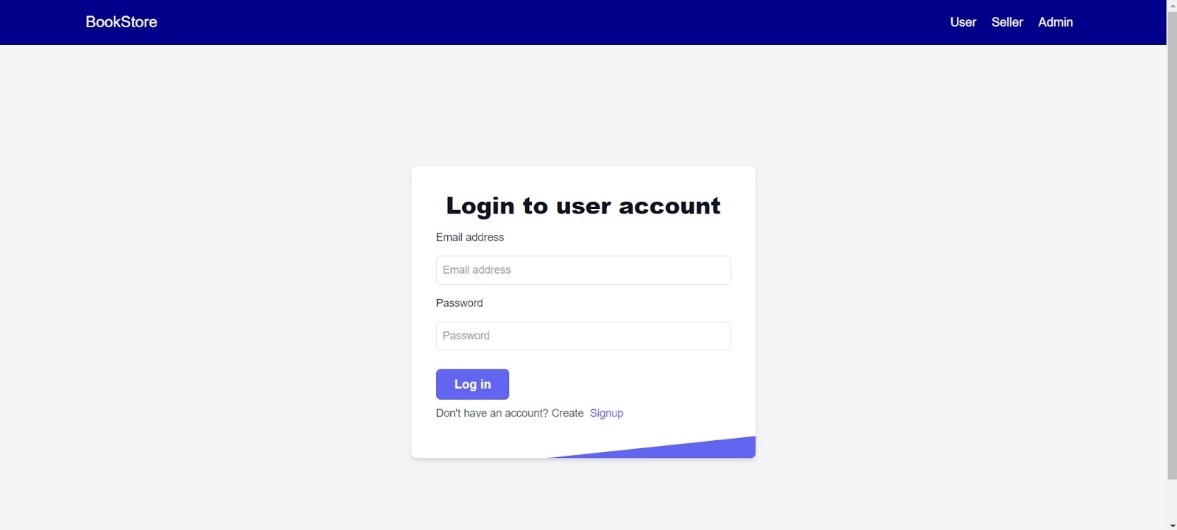
* **POST /Api/auth/register**
* Registers a new user.
* **Body**: { name, email, password, contact }
* **Response**: 201 Created, User data.
* **POST /Api/auth/login**
* Authenticates user and generates a JWT.
* **Body**: { email, password }
* **Response**: 200 OK, JWT token.
* **GET /Api/books**
* Fetches all available books with optional filters.
* **Query Params**: { genre, author, price, availability }
* **Response**: List of books.
* **POST /Api/cart**
* Adds books to the user's cart.
* **Body**: { book Id, quantity }
* **Response**: Updated cart.
* **POST /Api/orders**
* Processes a user's order.
* **Body**: { user Id, books, total Price, status }
* **Response**: Order ID and confirmation details.

**Authentication**

JWT-based authentication is the application uses **JWT (JSON Web Tokens)** to securely authenticate users. Tokens are issued upon successful login and stored in the frontend (e.g., in local Storage) to maintain session. Token validation occurs in the backend for each protected route.

**User Interface**

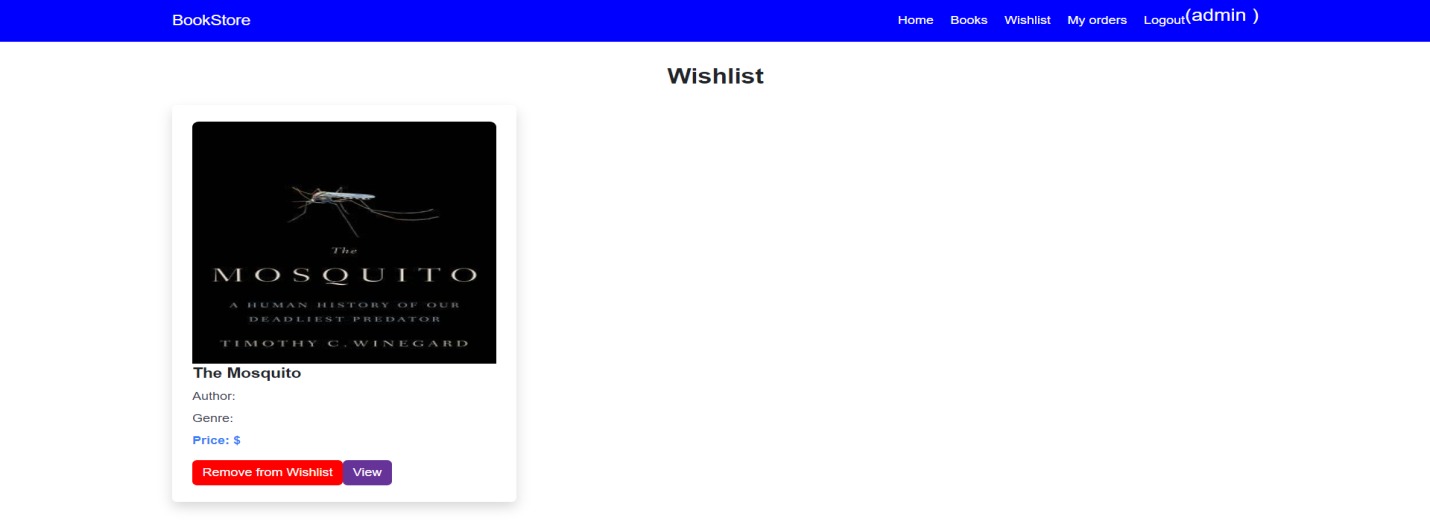
* Login Page:

****

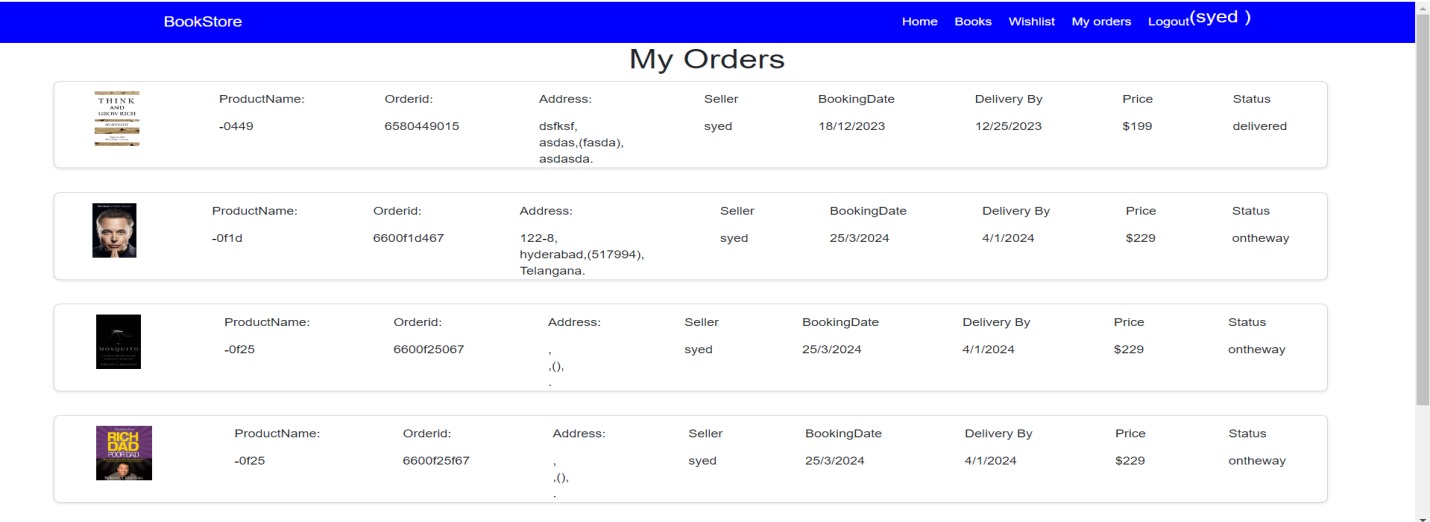
* Landing Page:



* Wishlist Page:



* My Booking Page:



**Testing**

* **Frontend**: Unit tests for React components using **Jest** and **React Testing Library**.
* **Backend**: API endpoint testing using **Mocha** and **Chai**.
* **User Testing**: Manual testing of the entire user flow from registration to order confirmation.

**Known Issues**

* Sometimes the inventory updates with a slight delay after purchase.
* Users on slow connections might experience delays in page rendering.

**Future Enhancements**

* **Mobile App**: Develop a companion mobile app for iOS and Android.
* **User Reviews and Ratings**: Allow users to rate and review books after purchase.
* **Recommendation System**: Implement an algorithm that recommends books based on user preferences and browsing history.