**Full Stack Development with MERN**

**1. Introduction**

* **Project Title:** Online Bookstore
* **Team Members:**
  + Kavya Swetha J: Project Manager, Backend Developer
  + Ashok kumar M: Frontend Developer
  + Arul Kumar T: Database Administrator, Frontend Developer
  + Nataraj M: Quality Assurance, Documentation
  + Mahaveer Priyadarshan A: Backend Developer, Documentation

**2. Project Overview**

* **Purpose**:  
  Book-Store is an online bookstore platform aimed at providing a seamless and user-friendly environment for both customers and sellers. The platform allows users to browse, purchase, and review books, while enabling sellers to manage their inventory and fulfill orders. Admins have full control over the platform, from managing users and books to overseeing transactions.
* **Features**:
  + **User Registration and Login**: Secure authentication for users to create accounts and log in.
  + **Book Listings**: Display a catalogue of books with detailed information such as title, author, genre, and price.
  + **Search and Filters**: Users can filter and search for books based on genres, authors, ratings, and more.
  + **Cart and Checkout**: Users can add books to their cart, specify quantities, and proceed to secure checkout.
  + **Order History**: Users can view their past orders, track current orders, and request returns.
  + **Seller Dashboard**: Sellers can list books, manage inventory, and fulfill customer orders.
  + **Admin Dashboard**: Admins have full control over the system, including managing users, books, and reports.
  + **Reviews and Ratings**: Users can leave feedback and ratings for books they've purchased.

**3. Architecture**

* **Frontend (React):**The frontend is built using React. The application is divided into several components (e.g., Home, Book Details, Cart, Profile, Admin Dashboard) to manage different parts of the application. It interacts with the backend using RESTful API calls, consuming data in JSON format.
* **Backend (Node.js & Express.js):**  
  The backend is built with Node.js and Express.js, providing a RESTful API to handle requests related to book listings, user authentication, orders, and more. It is responsible for interacting with the database and sending responses to the frontend.
* **Database (MongoDB):**  
  The database is MongoDB, a NoSQL database that stores data such as books, users, orders, and reviews. It allows for flexible and scalable data storage. Key collections include users, books, orders, reviews, sellers, and inventory.

**4. Setup Instructions**

* **Prerequisites**:  
  To set up the application locally, you will need:
  + Node.js (version 14 or higher)
  + MongoDB (local or cloud instance like MongoDB Atlas)
  + npm (Node package manager)
* **Installation**:
  + - 1. Clone the repository:

git clone https://github.com/yourusername/bookease.git

* + - 1. Install Backend Dependencies: Navigate to the server directory and run:

npm install

* + - 1. Install Frontend Dependencies: Navigate to the client directory and run

npm install

* + - 1. Set Up Environment Variables:
    - Create a .env file in the server directory.
    - Add your MongoDB URI and JWT secret:

MONGO\_URI=mongodb://localhost:27017/bookease

JWT\_SECRET=your\_jwt\_secret

**5. Folder Structure**

* **Client**: The client folder contains the React application. Key subfolders:
  + src/components: React components (e.g., Home, BookDetails, Cart, Login).
  + src/pages: Different pages in the app (e.g., HomePage, LoginPage).
  + src/utils: Helper functions (e.g., for form validation).
  + src/styles: CSS files and styling (or SCSS if using).
* **Server**: The server folder contains the backend code. Key subfolders:
  + models: Mongoose models (e.g., User.js, Book.js).
  + controllers: Functions that handle requests (e.g., authController.js, orderController.js).
  + routes: API route definitions (e.g., userRoutes.js, bookRoutes.js).
  + middleware: Custom middleware (e.g., for JWT authentication).
  + config: Configuration files (e.g., db.js for MongoDB connection).

**6. Running the Application**

* Frontend: To start the React development server, navigate to the client directory and run:

npm start

* Backend: To start the backend server, navigate to the server directory and run:

npm start

**7. API Documentation**

* **POST /api/register**  
  Register a new user.
  + Request Body: { "name": "John", "email": "john@example.com", "password": "password123" }
  + Response:

json

Copy code

{ "message": "User registered successfully" }

* **POST /api/login**  
  Login a user and receive a JWT token.
  + Request Body: { "email": "john@example.com", "password": "password123" }
  + Response:

{ "token": "JWT\_TOKEN" }

* **GET /api/books**  
  Get a list of all available books.
  + Response:

[

{ "id": 1, "title": "Book Title", "author": "Author Name", "price": 20.99, "genre": "Fiction" }

]

* **POST /api/orders**  
  Create a new order for the user.
  + Request Body:

{

"userId": "1",

"books": [{ "bookId": 1, "quantity": 2 }],

"totalAmount": 41.98

}

* + Response:

{ "orderId": "12345", "message": "Order placed successfully" }

**8. Authentication**

* **JWT Authentication:**  
  The app uses JSON Web Tokens (JWT) for authentication. After logging in, the user receives a token that is stored in local storage or a cookie. This token is then sent with each request to protected routes (e.g., viewing order history).
* **Authorization:**  
  Middleware checks for valid JWT tokens on protected routes to ensure the user is authenticated before accessing resources like order details or profile information.

**9. User Interface**

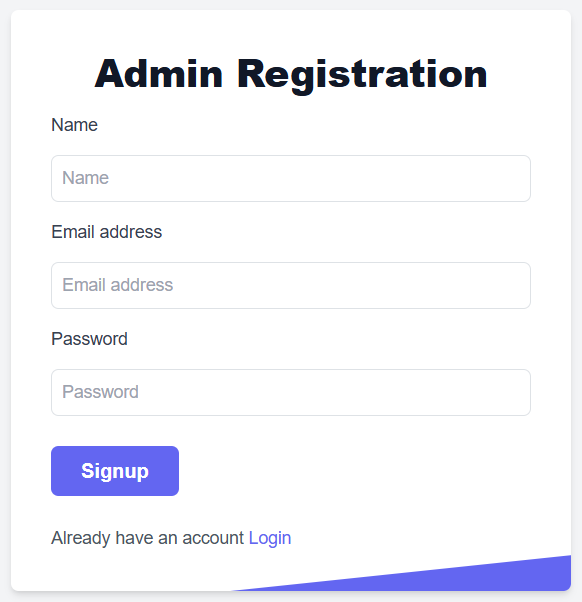
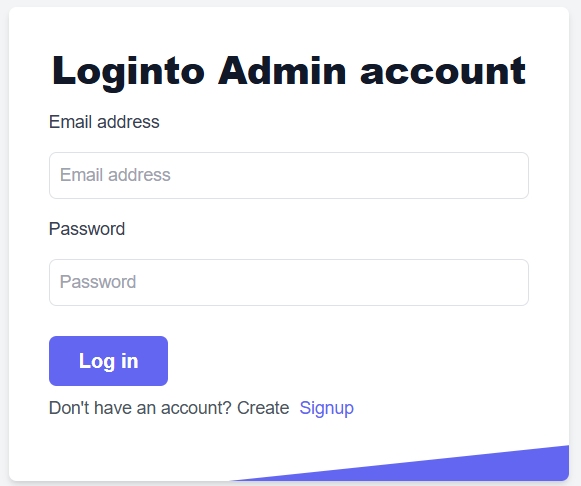
* **Home Page**:  
  The home page displays a list of featured books and offers a search bar for finding specific titles, authors, or genres.
* **Book Details Page**:  
  Displays detailed information about each book, including description, author, and price. Users can add books to their cart from this page.
* **Cart**:  
  Users can view their selected books, modify quantities, and proceed to checkout.
* **Profile**:  
  Users can view and update their profile details (name, email, password) and view order history.

**10. Testing**

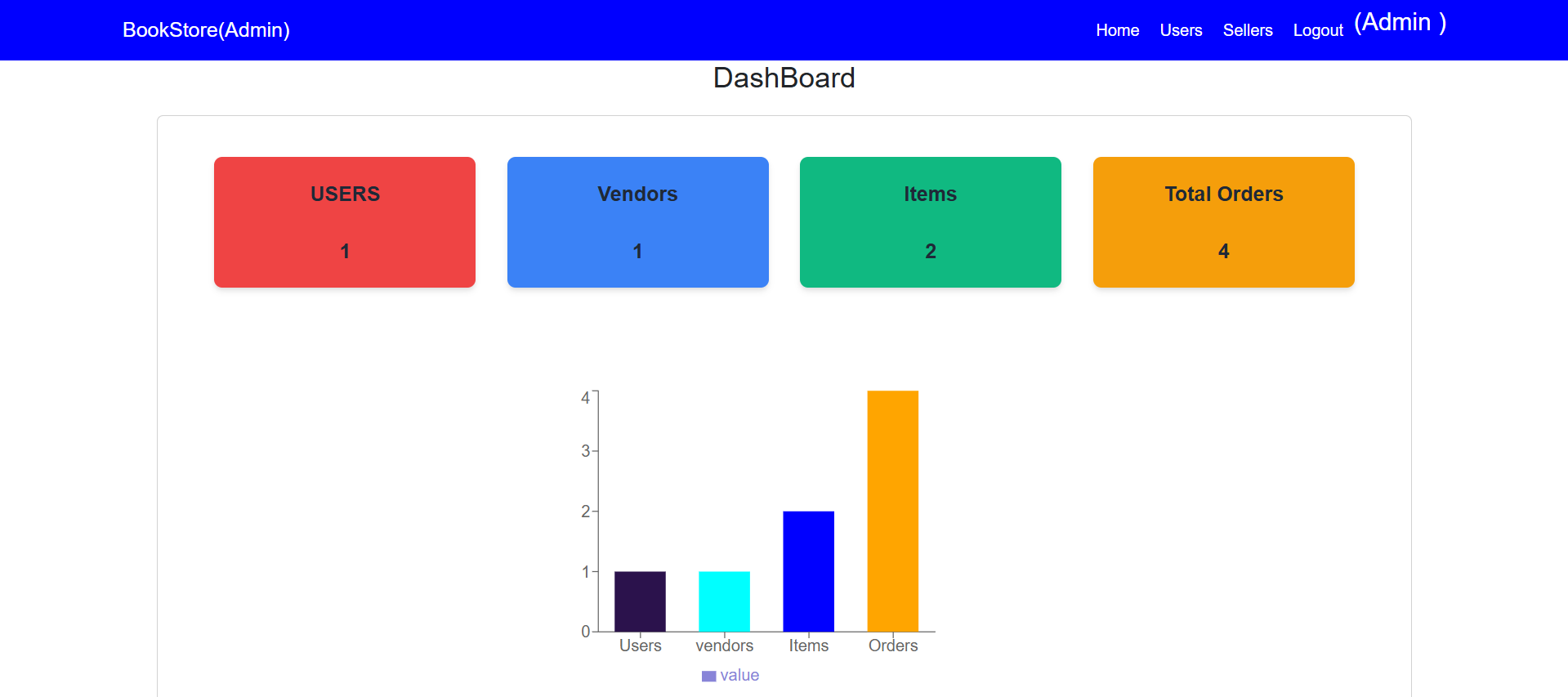
* **Unit Testing**:  
  Unit tests are written for backend controllers and frontend components using Jest and React Testing Library. Tests cover critical functionalities like user registration, login, and order placement.
* **Integration Testing**:  
  Integration tests ensure that the frontend and backend work together correctly. We test workflows such as user login, placing orders, and retrieving book details.
* **End-to-End Testing**:  
  End-to-end tests are performed using tools like Cypress to simulate real user interactions, such as logging in, searching for books, adding to the cart, and completing a purchase.

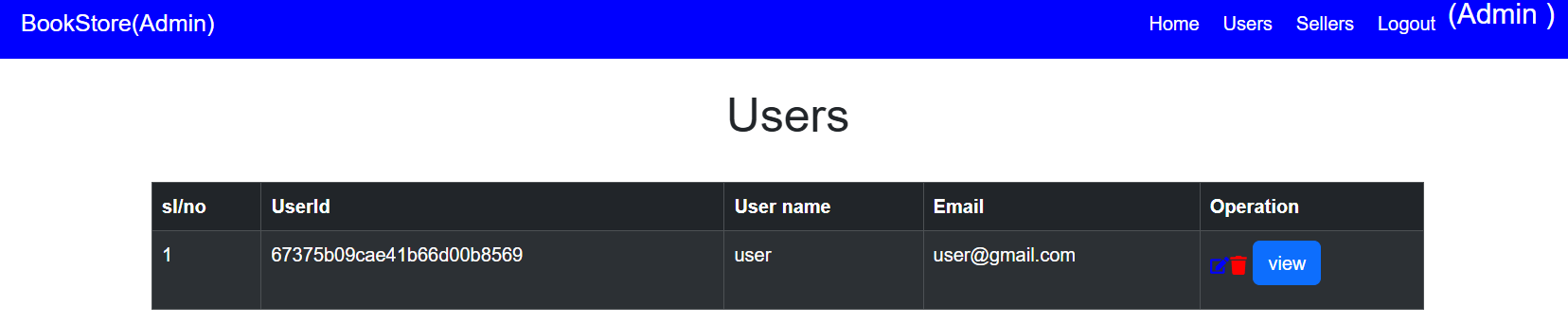
**11. Screenshots or Demo**

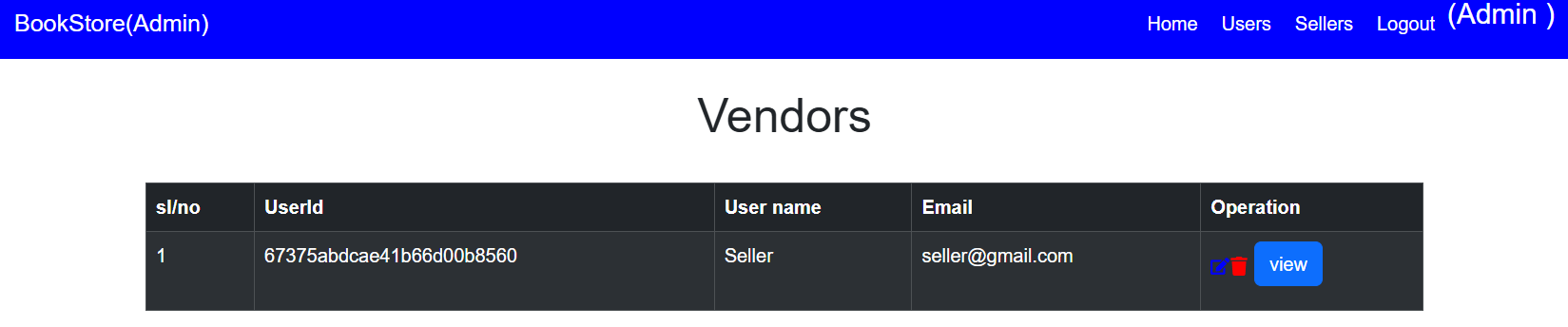
* [**Link to Demo**](https://drive.google.com/drive/folders/1sEz5e4t1VBEFA357Scpln3cwjn1UJVQO?usp=drive_link)
* **Screenshots**:

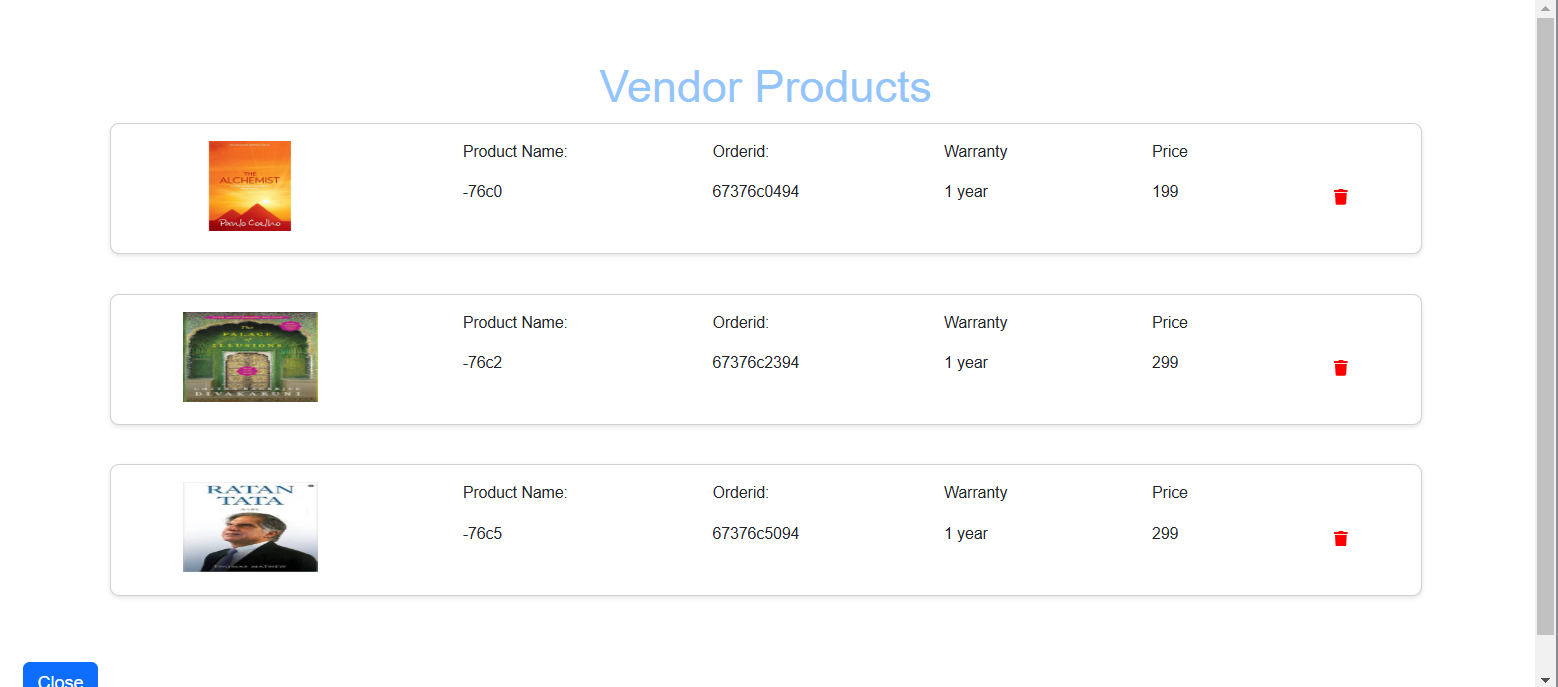
 

Admin Dashboard

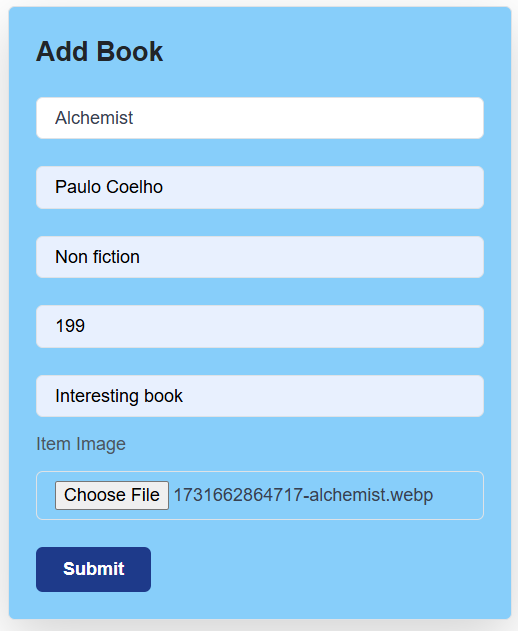


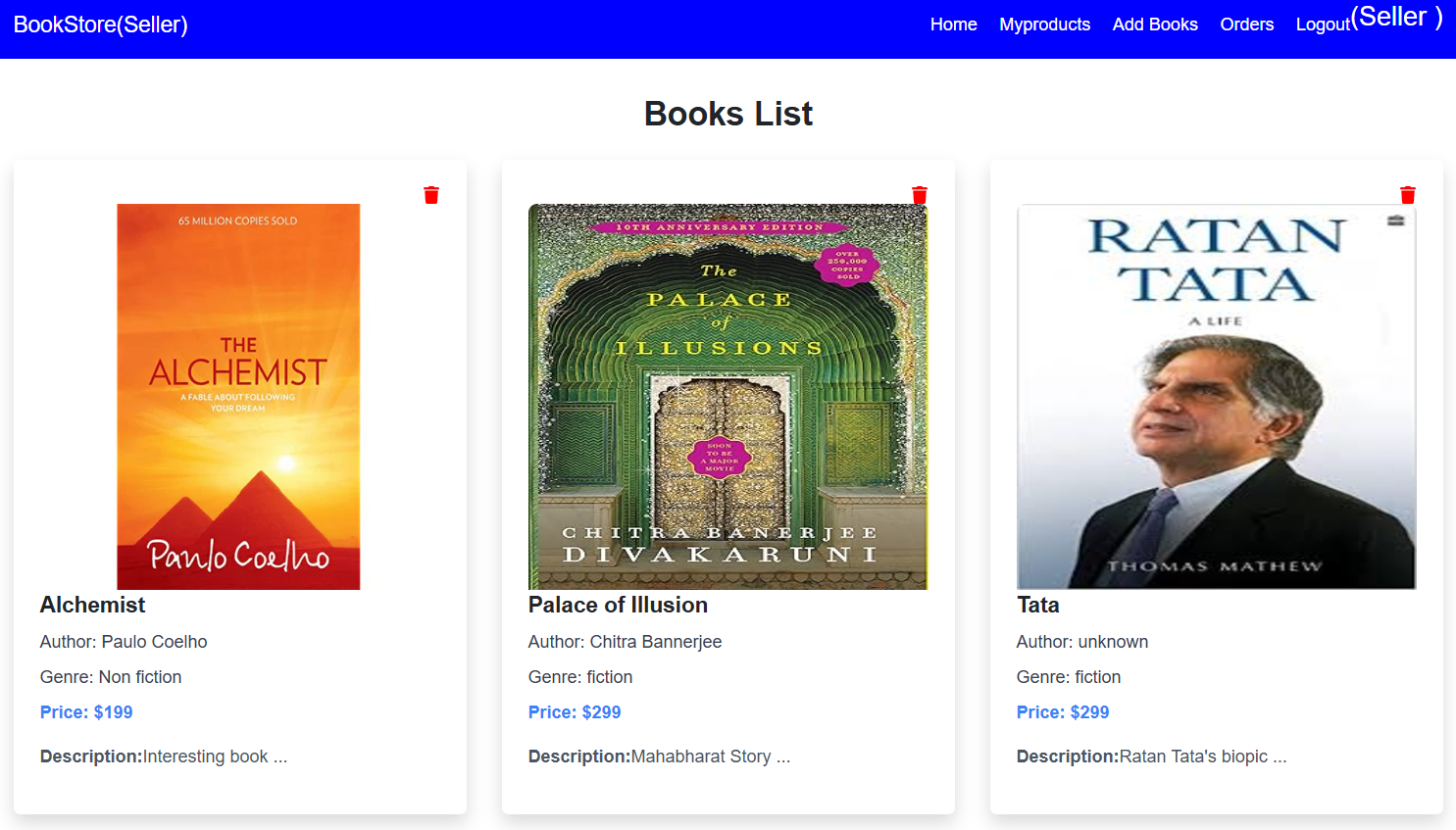




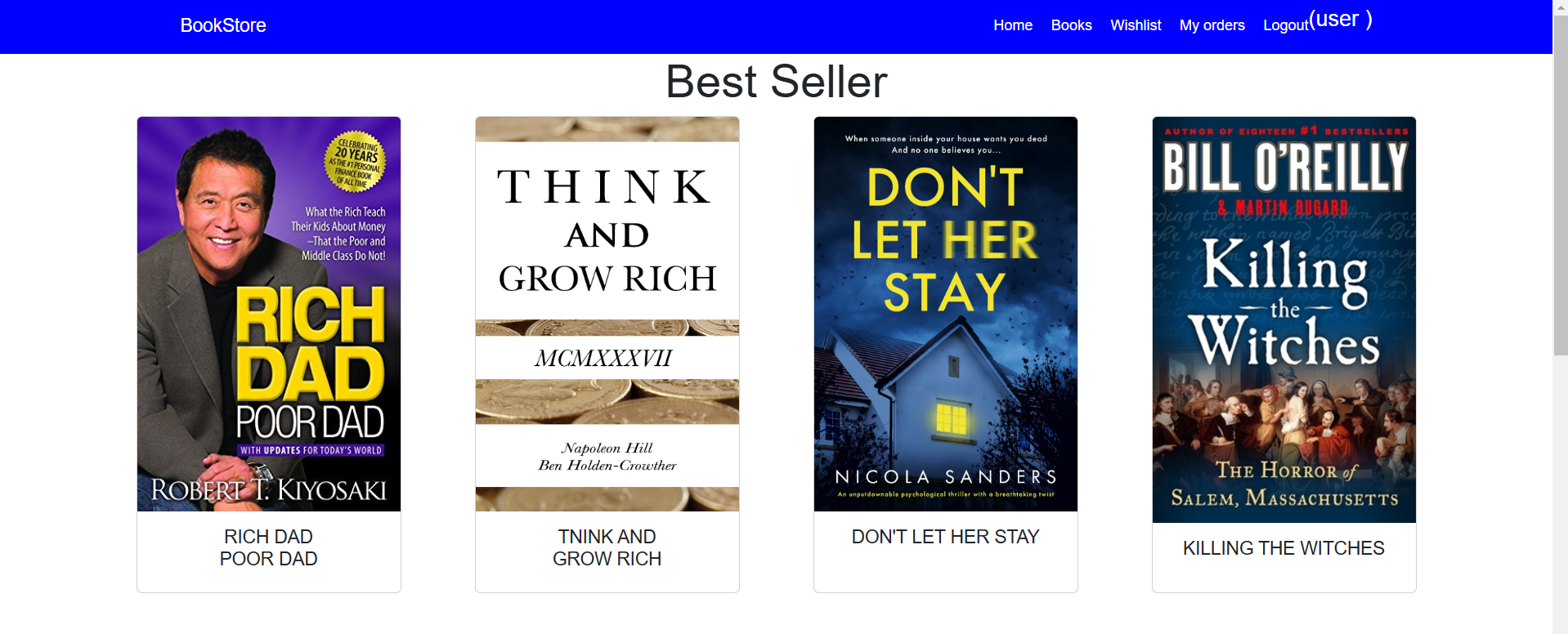


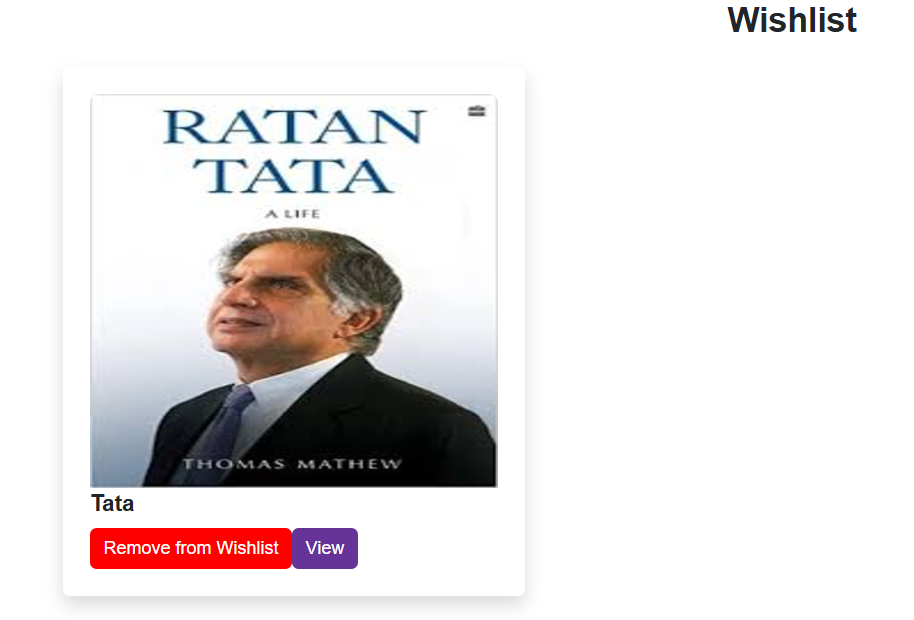
Seller Dashboard

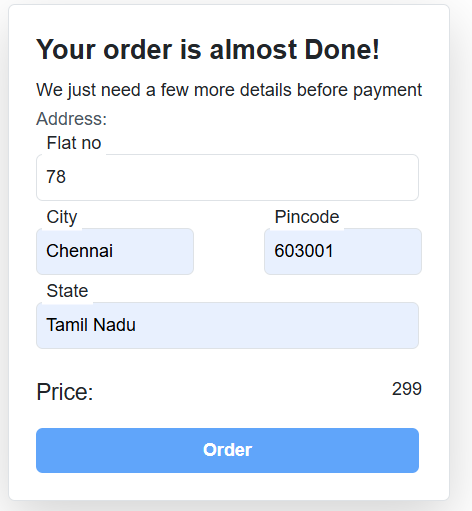


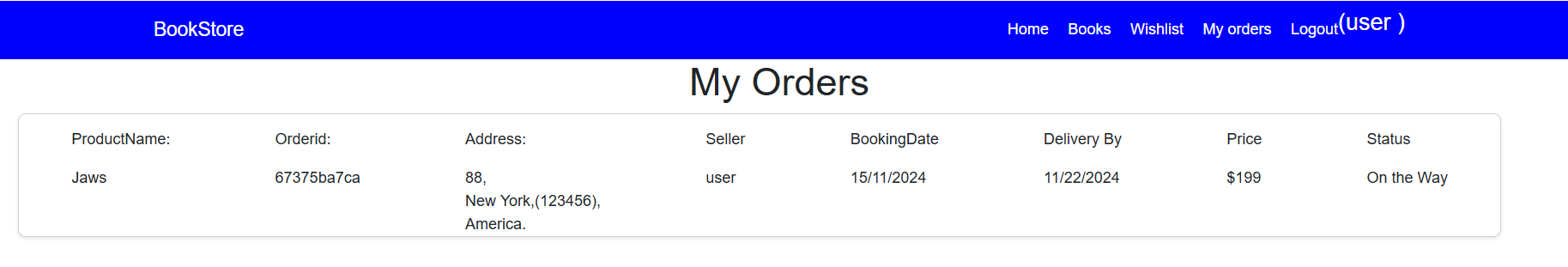


Customer Dashboard









**12. Known Issues**

* **Payment Gateway Delays**:  
  There may be occasional delays when processing payments, especially during high traffic periods.
* **Inventory Sync Issues**:  
  Inventory updates might not be immediate, leading to the possibility of overselling books in rare cases.

**13. Future Enhancements**

* **Book Recommendation Engine**:  
  Implement machine learning to recommend books based on user preferences and past purchase history.
* **Mobile App**:  
  Develop native mobile apps for iOS and Android to enhance accessibility