# BOOK STORE USING MERN

# INTRODUCTION:

The Book Store App is an innovative online book purchasing platform designed to streamline the experience of discovering, reviewing, and acquiring books. This system allows users to easily browse, search, and purchase books across various genres, all through a user-friendly interface. By offering functionalities like genre filtering, book previews, review submissions, and secure order management, the app caters to the needs of readers, book enthusiasts, and administrators alike.

Users can explore books based on genre, author, and language preferences, ensuring they find the right literary material for their interests. Once a book is selected, users can view detailed information, add it to their cart, place orders, and receive updates on availability and delivery. Readers benefit from personalized recommendations and the ability to leave and read reviews, while administrators manage the book inventory, monitor orders, and ensure smooth operation of the platform.

Built with a robust technical architecture, the Book Store App leverages a client-server model, using front-end frameworks like Bootstrap and Material UI for an engaging user experience, and a back end powered by Express.js and MongoDB for secure data handling. This system delivers a seamless and efficient book browsing and purchasing experience, addressing the growing demand for accessible, organized, and user-friendly online literary platforms.

**KEY FEATURES**

**USER REGISTRATION & PROFILE CREATION:**

●  SECURE SIGN-UP using email and password authentication.  
●  PROFILE CREATION that stores user preferences, reading history, and saved wishlists for a personalized experience.

**BOOK BROWSING & FILTERING:**

●  SEARCH AND FILTER BOOKS based on genre, author, language, and availability.  
●  REAL-TIME STOCK UPDATES ensure users view only the books currently available for purchase or reservation.

**PURCHASE & ORDER MANAGEMENT:**

●  INTUITIVE PURCHASE INTERFACE allowing users to add books to cart, select delivery options, and complete payment securely.  
●  ORDER CONFIRMATION AND TRACKING notifications via email or SMS keep users updated on their purchases.

**USER DASHBOARD:**

●  ACCESS TO ORDER HISTORY, wishlist, and saved books for quick reordering or tracking.  
●  REVIEW AND RATING SYSTEM enabling users to provide feedback on books and share their reading experiences.

**ADMIN CONTROLS & INVENTORY MANAGEMENT:**

●  ADMIN APPROVAL FOR BOOK ADDITIONS, ensuring quality and relevance of listed items.  
●  CENTRALIZED INVENTORY MANAGEMENT including book details, stock levels, user management, and platform policies for efficient operations.

**DESCRIPTION :**

The **Book Store App** is a user-friendly platform designed to revolutionize the way readers explore and purchase books online. Built to cater to the needs of modern bibliophiles, the app offers a seamless digital interface where users can browse, search, and filter books based on genre, author, language, and availability.

For readers, the app provides secure registration, personalized profile creation, and wishlist functionality, along with real-time notifications and order tracking to ensure a smooth purchasing journey. The platform also features user-generated reviews and ratings to assist in informed decision-making. Admins benefit from a comprehensive control panel to manage book listings, monitor orders, maintain inventory, and resolve customer concerns efficiently.

Developed using **Bootstrap** and **Material UI** for a sleek and responsive front-end design, the app integrates **Axios** for real-time communication between front end and back end. The server-side logic is handled by **Express.js** and **Node.js**, while **MongoDB** ensures efficient, scalable data storage for books, users, and transactions. Additional utilities like **bcrypt** provide secure user authentication, and tools such as **Moment.js** (if included) can be used for accurate timestamping of purchases and delivery tracking.

With features that enhance discoverability, purchasing efficiency, and user engagement, the **Book Store App** addresses the growing demand for accessible and reliable digital book

shopping. Whether you're discovering new releases or building your personal library, the app offers a refined, hassle-free experience that bridges the gap between literature and technology.

**SCENARIO-BASED CASE STUDY :**

**1. USER REGISTRATION :**

Sarah, an avid reader with a love for fiction and contemporary literature, discovers the Book Store App while searching for a more convenient way to shop for books. She installs the app and begins the registration process by providing her email address and creating a password. Once successfully registered, Sarah receives a welcome message and logs into her newly created account.

**2. BROWSING BOOKS :**

After logging in, Sarah is directed to the home dashboard, where a wide range of books is displayed. She uses the app’s intuitive filtering options to browse books by genre, author, price, and rating. Sarah filters the list to show best-selling fiction and newly released novels, finding several that spark her interest.

**3. BOOK SELECTION :**

Intrigued by a novel titled *"The Forgotten Pages"*, Sarah clicks on the book to view more details. The app provides a detailed description, author bio, reader reviews, ratings, availability status, and pricing. Convinced by the positive reviews, Sarah adds the book to her shopping cart.

**4. PURCHASE PROCESS :**

Sarah proceeds to the checkout page. She confirms her shipping address, selects a delivery option, and applies a discount coupon. Using a secure payment gateway integrated within the app, she completes her purchase. Once the payment is successful, she receives a notification stating that her order has been placed successfully.

**5. ORDER CONFIRMATION :**

Immediately after checkout, Sarah is redirected to an order confirmation screen. It displays the order ID, book title, total amount, estimated delivery time, and shipping details. A confirmation email and SMS are also sent to her registered email address and phone number.

**6. ORDER HISTORY :**

A few days later, Sarah wants to revisit her previous orders. She accesses the “My Orders” section from her dashboard. Here, she can view her current and past orders, check the delivery status, and re-order books she previously enjoyed. The app allows her to download invoices and rate the books she has purchased.

**7. ADMIN APPROVAL (BACKGROUND PROCESS) :**

In the background, the platform's admin has reviewed and approved new book submissions to the catalog. Each listing, including *"The Forgotten Pages"*, has been vetted for accuracy, availability, and content quality. Admins ensure that only verified and relevant books are displayed on the platform.

**8. PLATFORM GOVERNANCE :**

The admin team continually monitors the platform’s performance and user activity. They handle customer support queries, resolve payment disputes, and enforce listing policies to ensure a smooth, compliant, and user-focused experience for all readers and sellers.

**9. INVENTORY MANAGEMENT :**

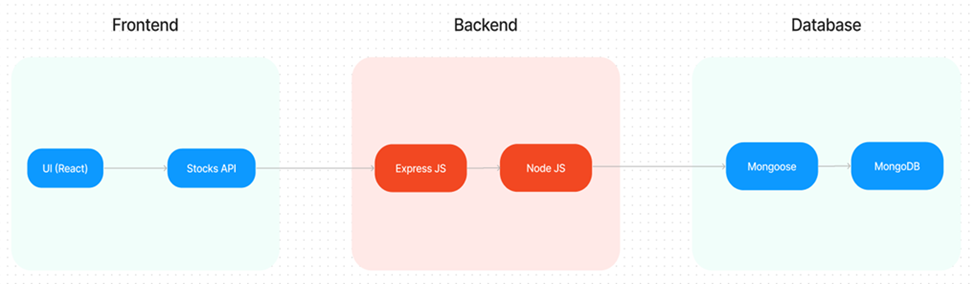
Simultaneously, the system updates the inventory. Once Sarah's order is confirmed, the stock count for *"The Forgotten Pages"* is reduced. The admin dashboard provides real-time insights into stock levels, helping with restocking decisions and avoiding order delays due to out-of-stock items.

**10. POST-PURCHASE EXPERIENCE :**

After receiving her book, Sarah returns to the Book Store App to leave a review and rating for *"The Forgotten Pages"*. She praises the book and the fast delivery service. The app suggests similar books she might enjoy, helping her continue her literary journey. With her wishlist and reading history saved, Sarah now enjoys a personalized, engaging book-shopping experience every time she logs in.

# Technical Architecture:

The Book Store App is developed using a modern MERN Stack architecture, comprising MongoDB, Express.js, React.js, and Node.js, structured on a client-server model.The frontend leverages React with Bootstrap and Material UI to create a responsive and intuitive user interface. Axios is used for smooth and secure API communication with the backend.The backend is built using Express.js and Node.js, ensuring efficient server-side operations and routing. MongoDB provides scalable and flexible data storage for user accounts, book details, orders, and reviews.JWT is used for authentication and bcrypt ensures secure password encryption. Role-Based Access Control (RBAC) manages permissions between users and administrators.The admin panel oversees book listings, inventory updates, and user management. The system supports real-time updates, efficient order tracking, and is designed for performance and scalability through modular code, caching, and API optimization.



**FRONTEND TECHNOLOGIES :**

* **Bootstrap and Material UI:** Provide a responsive and modern UI that adapts to various devices, ensuring a user-friendly experience.
* **Axios:** A promise-based HTTP client for making requests to the backend, ensuring smooth data communication between the frontend and server.

**BACKEND FRAMEWORK :**

* **Express.js:** A lightweight Node.js framework used to handle server-side logic, API routing, and HTTP request/response management, making the backend scalable and easy to maintain.

**DATABASE AND AUTHENTICATION :**

* **MongoDB**: A NoSQL database used for flexible and scalable storage of user data, doctor profiles, and appointment records. It supports fast querying and large data volumes.
* **JWT (JSON Web Tokens):** Used for secure, stateless authentication, allowing users to remain logged in without requiring session storage on the server.
* **Bcrypt**: A library for hashing passwords, ensuring that sensitive data is securely stored in the database.

**ADMIN PANEL & GOVERNANCE :**

* **Admin Interface:** Provides functionality for platform admins to approve doctor registrations, manage platform settings, and oversee day-to-day operations.
* **Role-based Access Control (RBAC):** Ensures different users (patients, doctors, admins) have appropriate access levels to the system’s features and data, maintaining privacy and security.

**SCALABILITY AND PERFORMANCE :**

* **MongoDB:** Scales horizontally, supporting increased data storage and high user traffic as the platform grows.

Load Balancing: Ensures traffic is evenly distributed across servers to optimise performance, especially during high traffic periods.

* **Caching:** Reduces database load by storing frequently requested data temporarily, speeding up response times and improving user experience.

**TIME MANAGEMENT AND SCHEDULING**

* **Moment.js:** Utilised for handling date and time operations, ensuring precise appointment scheduling, time zone handling, and formatting.

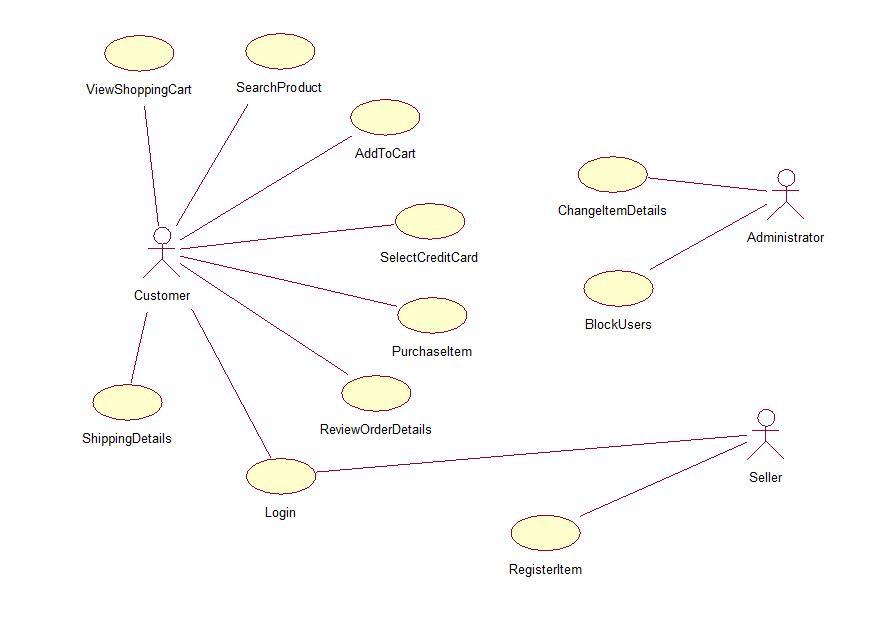
**SECURITY FEATURES :**

* **HTTPS**: The platform uses SSL/TLS encryption to secure data transmission between the client and server.
* **Data Encryption:** Sensitive user information, such as medical records, is encrypted both in transit and at rest, ensuring privacy and compliance with data protection regulations.

**NOTIFICATIONS AND REMINDERS :**

* **Email/SMS Integration:** Notifications for appointment confirmations, reminders, cancellations, and updates are sent to users via email or SMS, ensuring timely communication.

**ER-Diagram:**

****

**User-Book Relationship:**

Type: Many-to-Many (M:M). A single user can read or interact with many books, and a single book can be accessed by many users.

Implementation: Introduce an intermediate entity, "Interaction", with foreign keys to both User and Book tables. This table could store additional information like reading progress, reviews, or ratings.

**Book-Inventory Relationship:**

Type: One-to-Many (1:M). Each book can have multiple copies in inventory, but each copy belongs to one book.

Implementation: Maintain a separate Inventory table with fields like BookID (foreign key), quantity, location, and condition.

**User-Order Relationship:**Type: One-to-Many (1:M). A single user can place multiple orders, but each order belongs to one user.Implementation: Keep the UserID foreign key in the Order table to track user purchase history.

**Additional Relationships:**

Book-Author Relationship: Many-to-Many (M:M). A book can have multiple authors, and an author can write multiple books. (Similar to User-Book, use an intermediate "WrittenBy" table)

Book-Genre Relationship: Many-to-Many (M:M). A book can belong to multiple genres, and a genre can have many books. (Similar to User-Book, use an intermediate "CategorizedAs" table)

Review-User Relationship: Many-to-One (M:1). A review is written by one user, but a user can write many reviews. (Keep UserID as a foreign key in the Review table)

**Key Features:**

**User Registration and Authentication:** Allow users to register accounts securely, log in, and authenticate their identity to access the book store platform.

**Book Listings:** Display a comprehensive list of available books with details such as title, author, genre, description, price, and availability status.

**Book Selection:** Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

**Purchase Process:** Allow users to add books to their cart, specify quantities, and complete purchases securely. Upon successful completion, an order is generated, and the inventory is updated accordingly.

**Order Confirmation:** Provide users with a confirmation page or notification containing details of their order, including book information, total price, and order ID.

**Order History:** Allow users to view their past and current orders, providing options to track shipments, review purchased books, and rate their shopping experience.

**Organizer Dashboard:** Offer administrators an interface to manage book listings, inventory levels, user accounts, orders, and other platform-related activities.

**Create Item:** Organizer can create items and add new items and he can get the items and he can update items.

**Admin Dashboard:** Offer administrators an interface to manage book listings, inventory levels, user accounts, orders, and other platform-related activities. Manage the users and organizers.

**Reporting and Analytics:** Generate reports and analytics on book sales, popular genres, user demographics, and other relevant metrics to gain insights into platform usage and performance.

**Integration with External APIs:** Integrate with third-party APIs for services like payment processing, shipping logistics, and book recommendations to enhance the functionality and user experience of the book store platform.

## PRE REQUISITES:

To develop a full-stack Book Store App using React js, Node.js,Express js and MongoDB, there are several prerequisites you should consider. Here are the key prerequisites for developing such an application:

**Node.js and npm:** Install Node.js, which includes npm (Node Package Manager), on your development machine. Node.js is required to run JavaScript on the server side.

• Download: <https://nodejs.org/en/download/>

• Installation instructions:<https://nodejs.org/en/download/package-manager/>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

• Download:<https://www.mongodb.com/try/download/community>

• Installation instructions:<https://docs.mongodb.com/manual/installation/>

**Express.js:** Express.js is a web application framework for Node.js. Install Express.js to handle server-side routing, middleware, and API development.

• Installation: Open your command prompt or terminal and run the following

command: **npm install express**

**React js: React** is a JavaScript library for building client-side applications.

And Creating Single Page Web-Appliaction

**Getting Started**

Create React App is an officially supported way to create single-page React applications. It offers a modern build setup with no configuration.

**Quik Start**

npm create vite@latest

cd my-app

npm install

npm run dev

If you've previously installed create-react-app globally via npm install -g create-react-app, we recommend you uninstall the package using npm uninstall -g create-react-app or yarn global remove create-react-app to ensure that npx always uses the latest version.

**Create a new React project:**

• Choose or create a directory where you want to set up your React project.

• Open your terminal or command prompt.

• Navigate to the selected directory using the cd command.

• Create a new React project by running the following command: npx create-react-app your-app-name.Wait for the project to be created:

• This command will generate the basic project structure and install the necessary dependencies

**Navigate into the project directory:**

• After the project creation is complete, navigate into the project directory by running the following command**: cd your-app-name**

**Start the development server:**

• To launch the development server and see your React app in the browser, run the following command:  **npm run dev**

• The npm start will compile your app and start the development server.

• Open your web browser and navigate to <https://localhost:5173> to see your React app.

You have successfully set up React on your machine and created a new React project. You can now start building your app by modifying the generated project files in the src directory.

Please note that these instructions provide a basic setup for React. You can explore more ad- vanced configurations and features by referring to the official React documentation: <https://react.dev/>

**HTML, CSS, and JavaScript:** Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

**Database Connectivity:** Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Node.js server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations.

**Front-end Library:** Utilize React to build the user-facing part of the application, including products listings, booking forms, and user interfaces for the admin dashboard.

**Version Control**: Use Git for version control, enabling collaboration and tracking changes throughout the development process. Platforms like GitHub or Bitbucket can host your repository.

* Git: Download and installation instructions can be found at:<https://git-scm.com/downloads>

**Development Environment:** Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from <https://code.visualstudio.com/download>

• Sublime Text: Download from

<https://www.sublimetext.com/download>

• WebStorm: Download from <https://www.jetbrains.com/webstorm/download>

**Roles and Responsibility**

**User:**

• Registration: Users are responsible for registering an account on the BookEase book store app by providing essential details such as name, email, and password.

• Profile Management: Users have the capability to manage their profiles, allowing them to update information like email, name, and password.

• Book Browsing: Users can browse through the available books, explore different genres, and search for specific titles or authors.

• Purchase: Users can add books to their cart, specify quantities, and complete purchases securely.

• Feedback: Provide feedback and ratings for purchased books and sellers on the BookEase platform.

• Logout: Lastly, they can logout from the BookEase book store app.

**Seller:**

• Registration: Sellers register an account on the BookEase book store app by providing necessary details such as business name, email, and password.

• Profile Management: Sellers have the capability to manage their profiles, allowing them to update information like email, business name, and password.

• Book Listing: Sellers can add new books to the platform, including details such as title, author, genre, description, price, and quantity available.

• Inventory Management: Sellers can manage their book inventory, updating stock levels, removing inactive listings, and handling book ratings.

• Order Fulfillment: Sellers are responsible for fulfilling orders placed by users, including packaging and shipping books in a timely manner.

• Logout: Finally, they can logout from the BookEase book store app.

**Admin:**

• System Management: Admins have full control over all aspects of the book store system, overseeing functionalities, configurations, and security.

• User Management: Admins can manage user information, including creating, updating, and deleting accounts. They also have authority over user ratings.

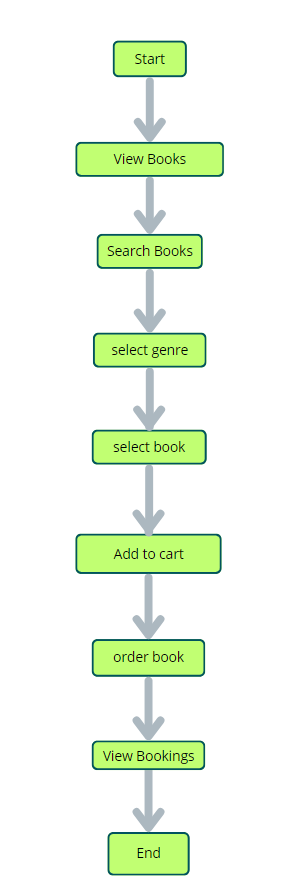
• Book Management: Admins can manage book listings, including adding new books, updating details, and removing inactive listings from the platform.

• Seller Management: Admins have the authority to manage seller information, including approving new seller accounts, updating profiles, and handling seller ratings.

• Logout: Finally, they can logout from the BookEase book store app.

This adaptation aligns user, seller, and admin functionalities with those of a book store app, emphasizing actions and terminology relevant to book browsing, purchasing, and selling.

**User Flow:**

****

Start: Users open the BookEase app to explore a vast collection of books.

Home Page: Users land on the home page, which provides an overview of the book store's offerings. From here, they can navigate to various sections of the app.

Access Profile: Users have the option to access their profiles, allowing them to view or update personal information, preferences, and order history.

Book Selection: After accessing their profiles, users proceed to browse and select books to purchase. The app presents a list of available books, along with details such as title, author, genre, and price.

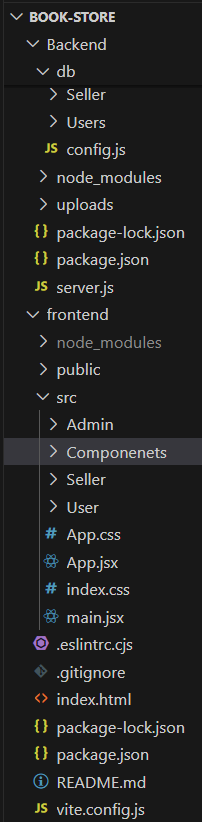
Book Purchase: Users navigate through the available book options and specify the quantity of each book they wish to purchase. They can also choose additional options such as e-book format or special editions.

View Orders: Users have the option to view their current and past orders. This section provides details about ordered books, order status, and payment history.

Order Confirmation: For new purchases, users can initiate the ordering process. This involves selecting books, specifying quantities, confirming the order, and receiving an order confirmation.

End: The flow concludes as users have completed their desired actions within the BookEase app.

**Project Structure:-**

****

**PROJECT FLOW:-**

Before starting to work on this project, let’s see the demo.

Demo link:-

<https://drive.google.com/file/d/14zsjfT65GYyZdnw2v5StoNeRn5bApsxa/view?usp=drive_link>

Use the code in:-

[Book store Using Mern Source Code](https://drive.google.com/drive/folders/1kC_gVVwZlHaZ6VuGvs71-4ynQXmiC2Kh)

or follow the videos below for better understanding.

**Milestone 1: Project Setup and Configuration:**

**1. Install required tools and software:**

* Node.js.
* MongoDB.
* Create-react-app.

**2. Create project folders and files:**

* Client folders.
* Server folders.

**3. Install Packages:**

**Frontend npm Packages**

* Axios.
* React-Router –dom.
* Bootstrap.
* React-Bootstrap.

**Backend npm Packages**

* Express.
* Mongoose.
* Cors.

Link:-

<https://drive.google.com/file/d/1Acv3Lx3PtJcOYkUjREWAzIoC-i6w96Tl/view?usp=drive_link>

**Milestone 2: Backend Development:**

* **Setup express server**

1. Create index.js file in the server (backend folder).
2. Create a .env file and define port number to access it globally.
3. Configure the server by adding cors, body-parser.

* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Define API Routes:**
  + Create separate route files for different API functionalities such as users orders, and authentication.
  + Define the necessary routes for listing products, handling user registration and  login,managing orders, etc.
  + Implement route handlers using Express.js to handle requests and interact with the database.
* **Implement Data Models:**
  + Define Mongoose schemas for the different data entities like products, users,  and orders.
  + Create corresponding Mongoose models to interact with the MongoDB database.
  + Implement CRUD operations (Create, Read, Update, Delete) for each model to perform database operations.
* **User Authentication:**
  + Create routes and middleware for user registration, login, and logout.
  + Set up authentication middleware to protect routes that require user authentication.
* **Error Handling:**

• Implement error handling middleware to catch and handle any errors that occur during the API requests.

• Return appropriate error responses with relevant error messages and HTTP status codes.

Link:- <https://drive.google.com/file/d/14Vut4GVqofFnPO-z1DjWKgQsVq3R4-r1/view?usp=drive_link>

**Milestone 3: Database:**

**1. Configure MongoDB:**

* Install Mongoose.
* Create database connection.
* Create Schemas & Models.

**2. Connect database to backend:**

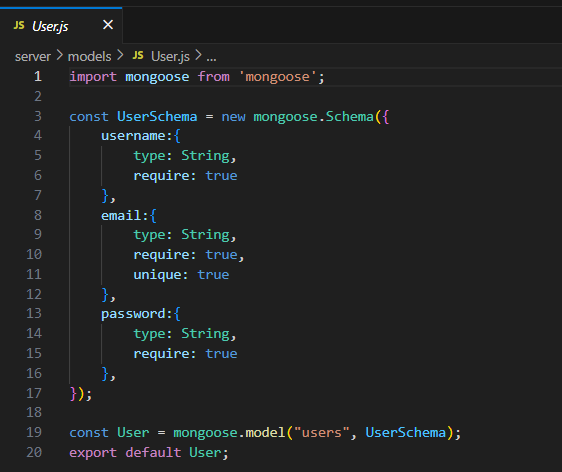
Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



**3. Configure Schema:**

Firstly, configure the Schemas for MongoDB database, to store the data in such pattern. Use the data from the ER diagrams to create the schemas.

The Schemas for this application look alike to the one provided below.



**Milestone 4: Frontend Development:**

**1. Setup React Application:**

• Create React application.

• Configure Routing.

• Install required libraries.

**2. Design UI components:**

• Create Components.

• Implement layout and styling.

• Add navigation.

**3. Implement frontend logic:**

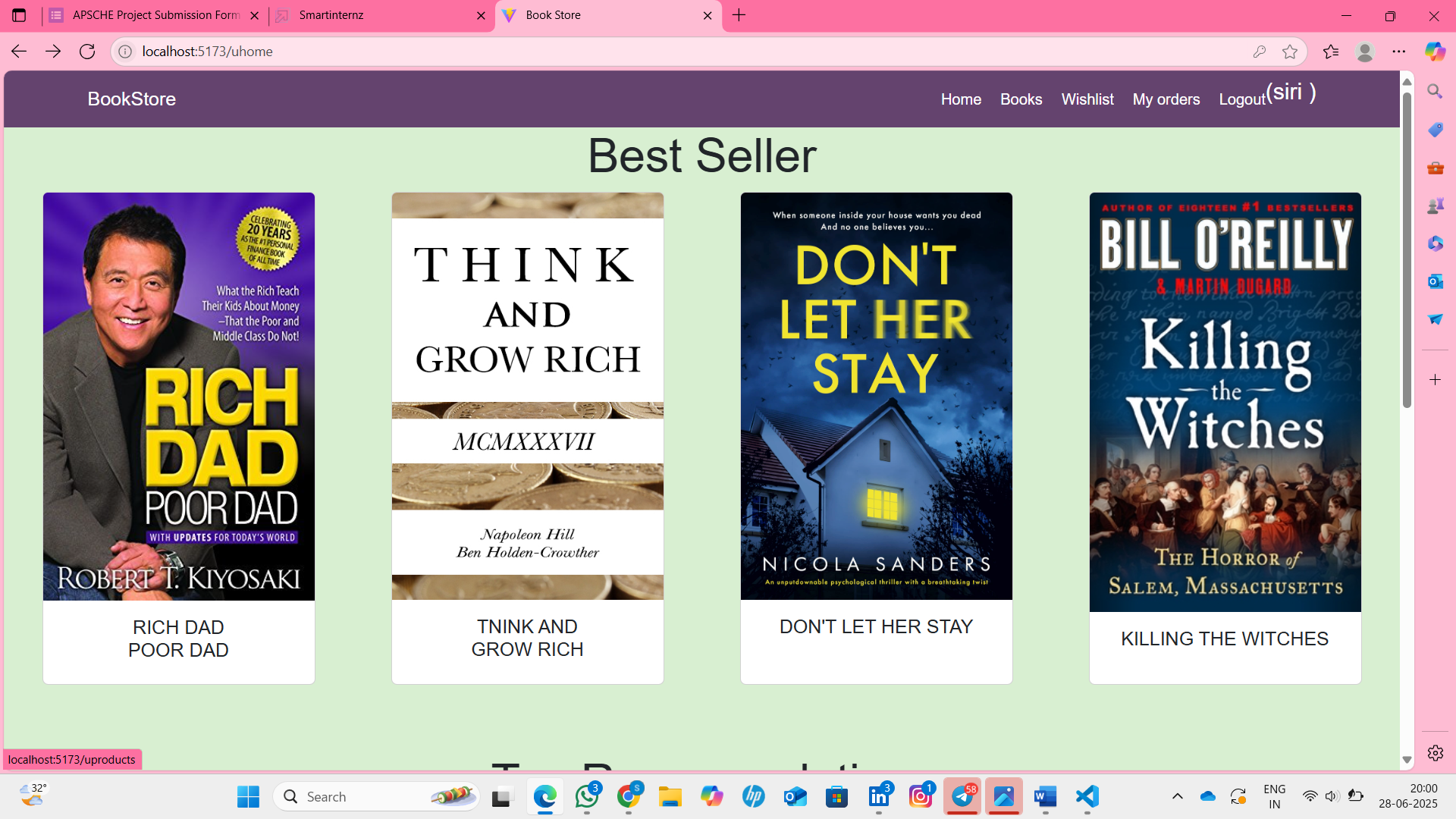
• Integration with API endpoints.

• Implement data binding.

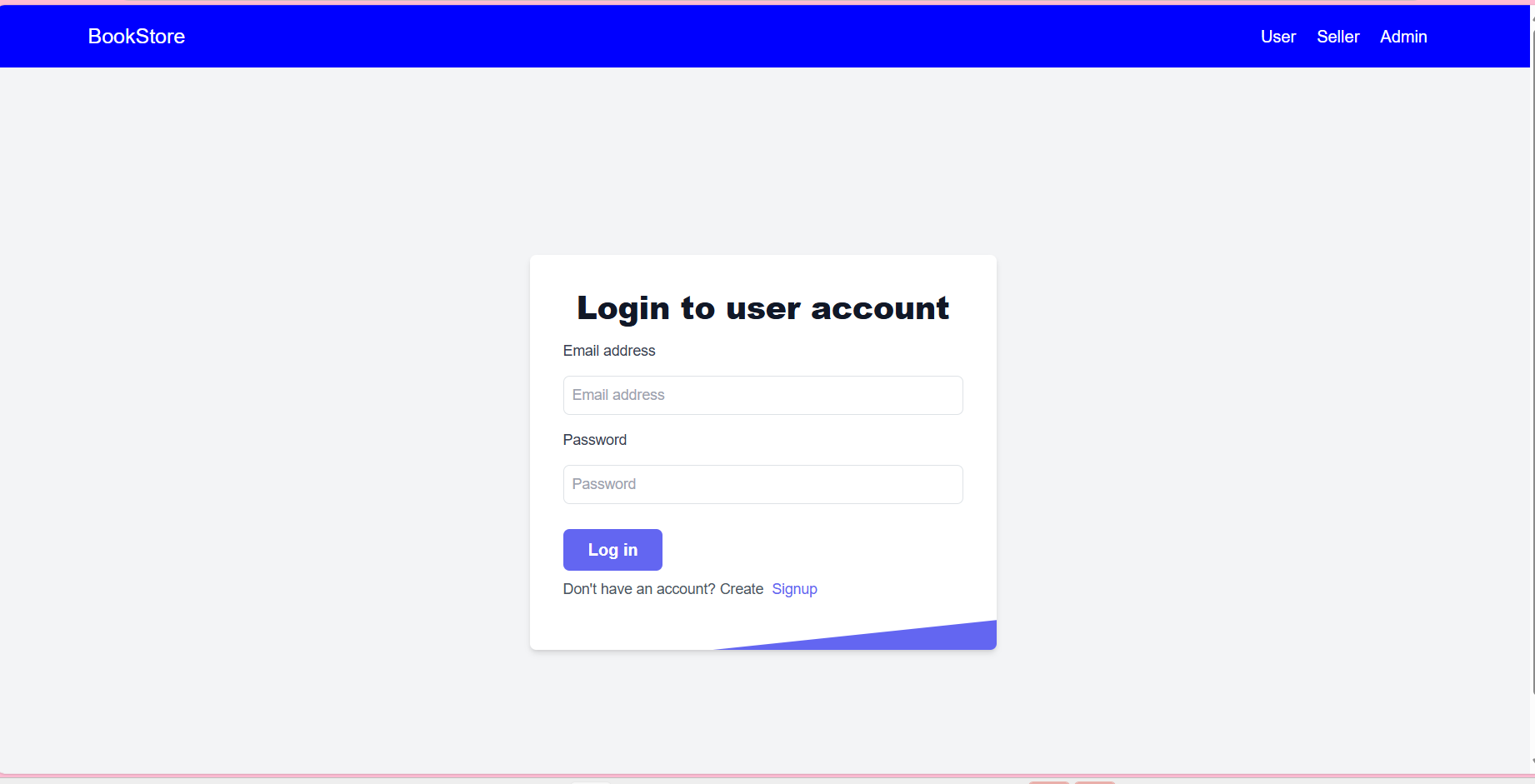
Link<https://drive.google.com/file/d/1L4aHUedhBJSAmnY2iGsoVhxwXpo5Fz4t/view?usp=drive_link>

**Milestone 5: Project Implementation:**Finally, after finishing coding the projects we run the whole project to test it’s working process and look for bugs. Now, let’s have a final look at the working of our Cab Booking application.

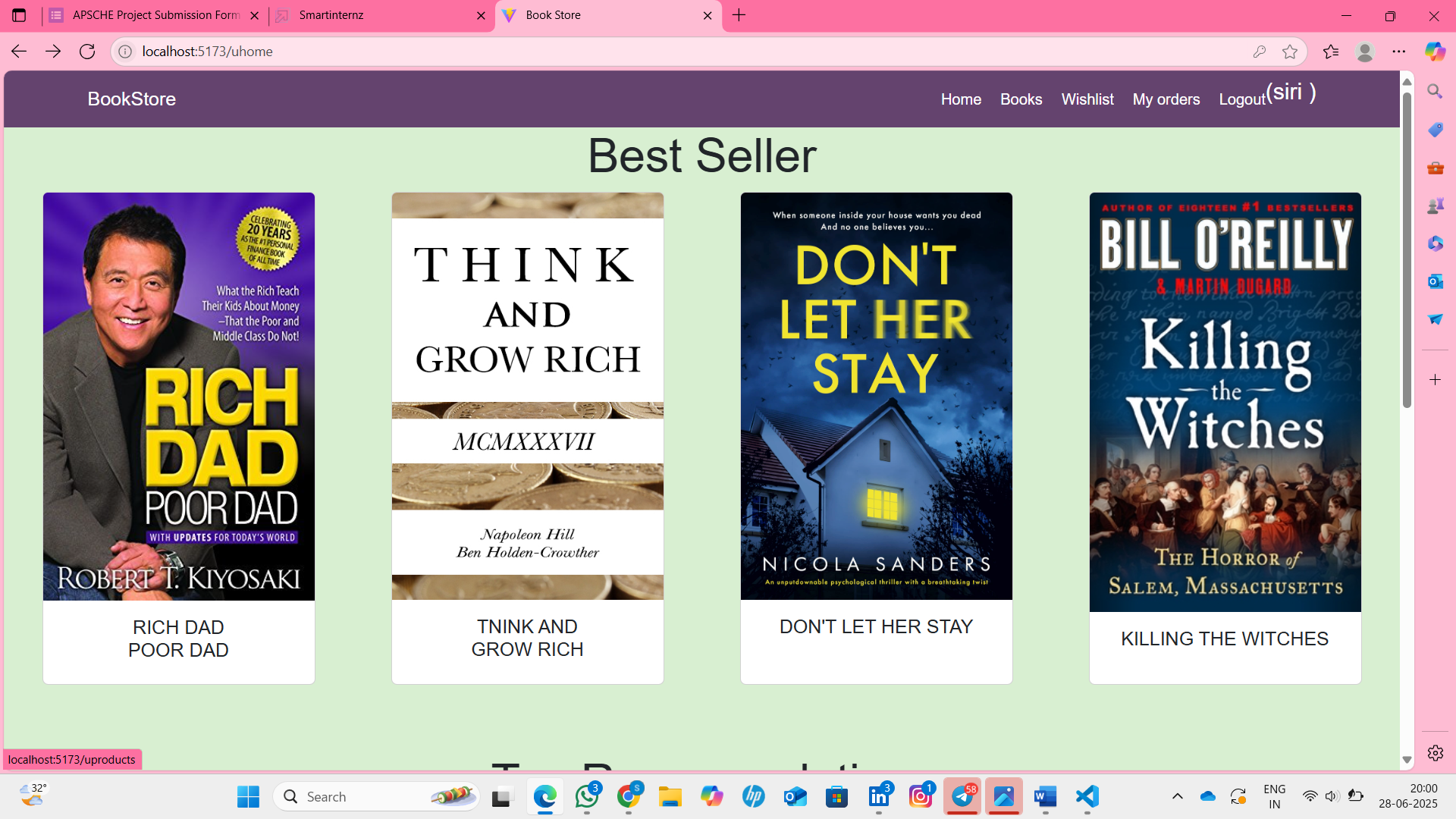
Landing page:-



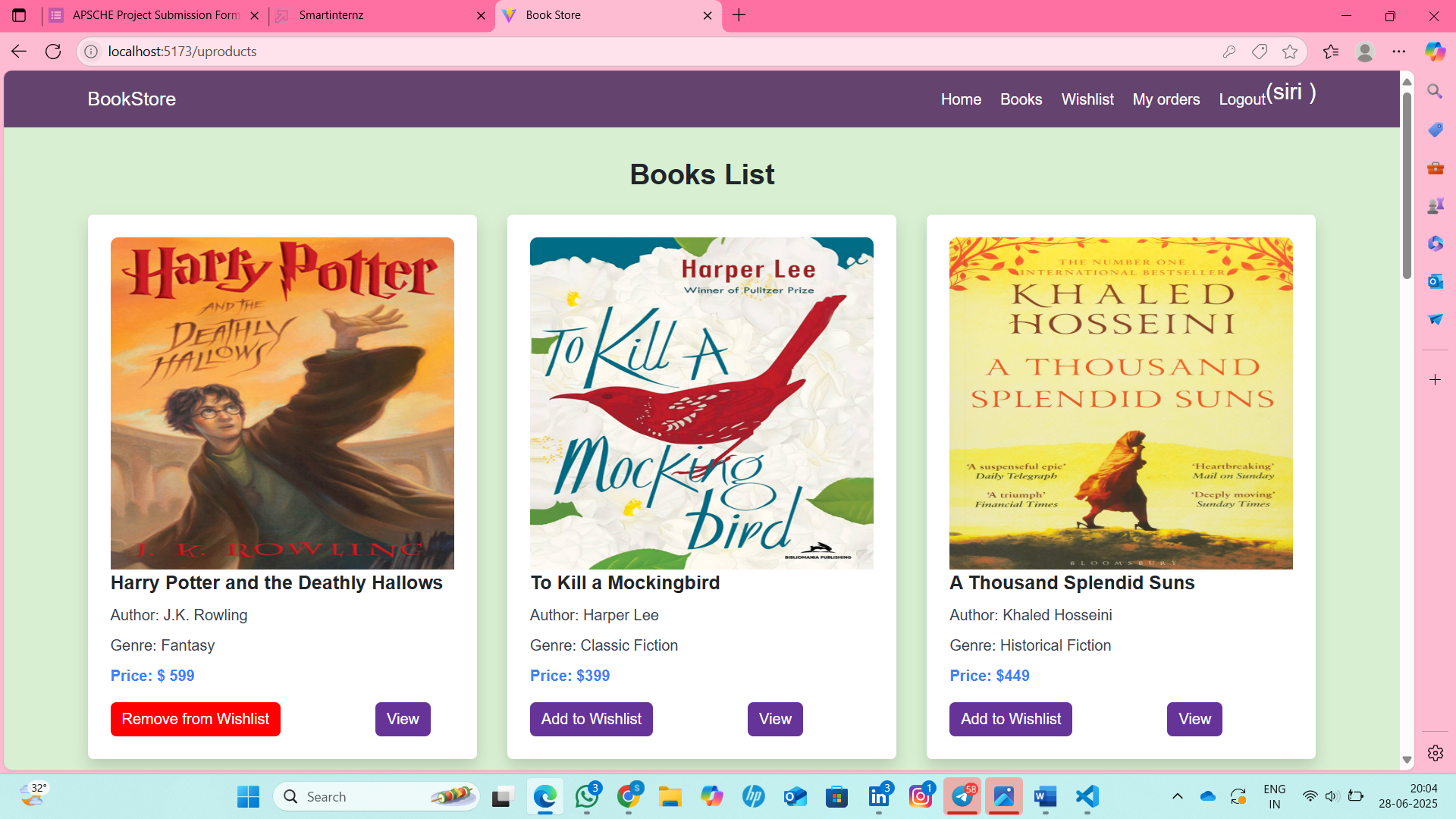
Login Page:-



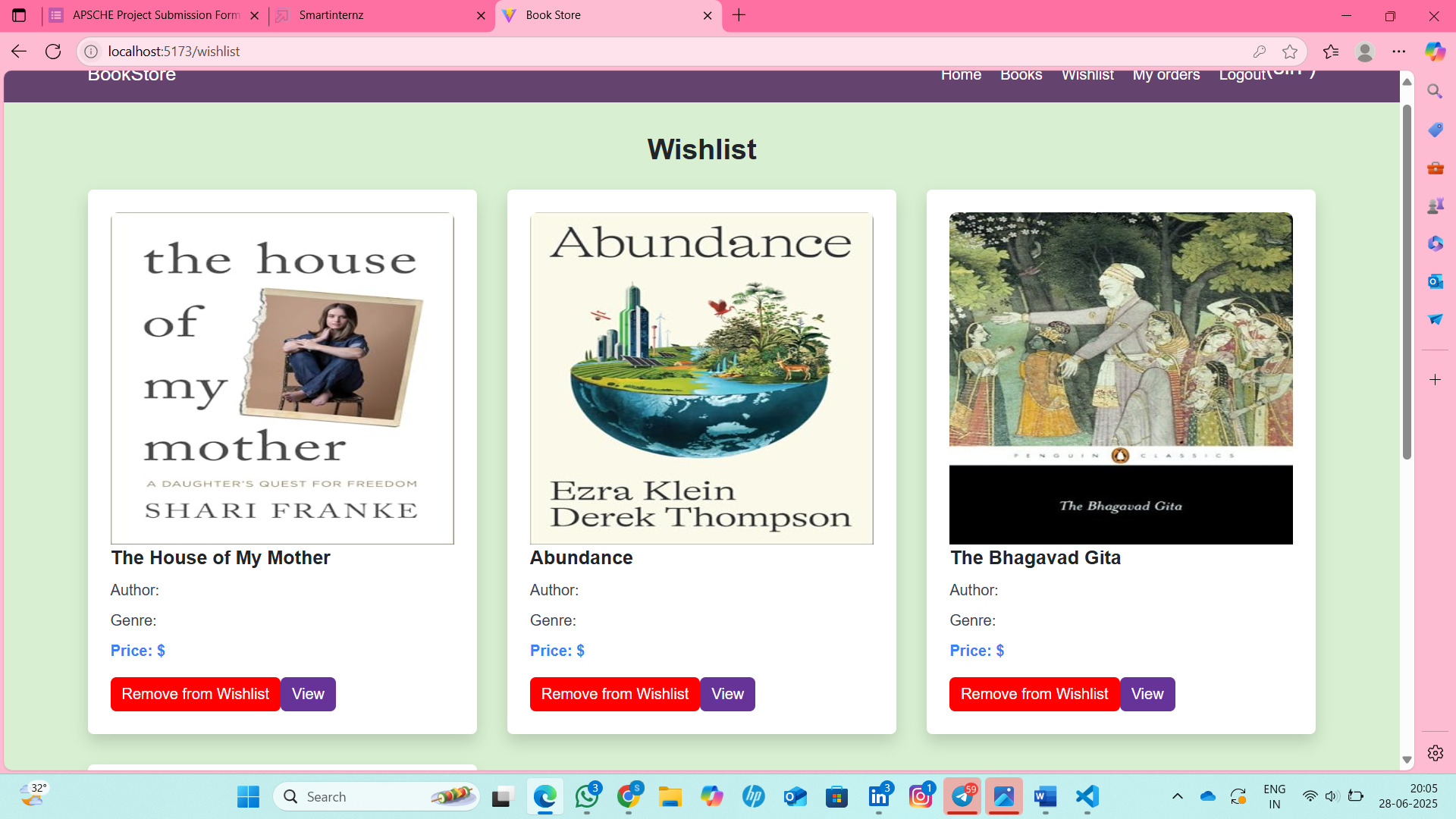
Home Page:-



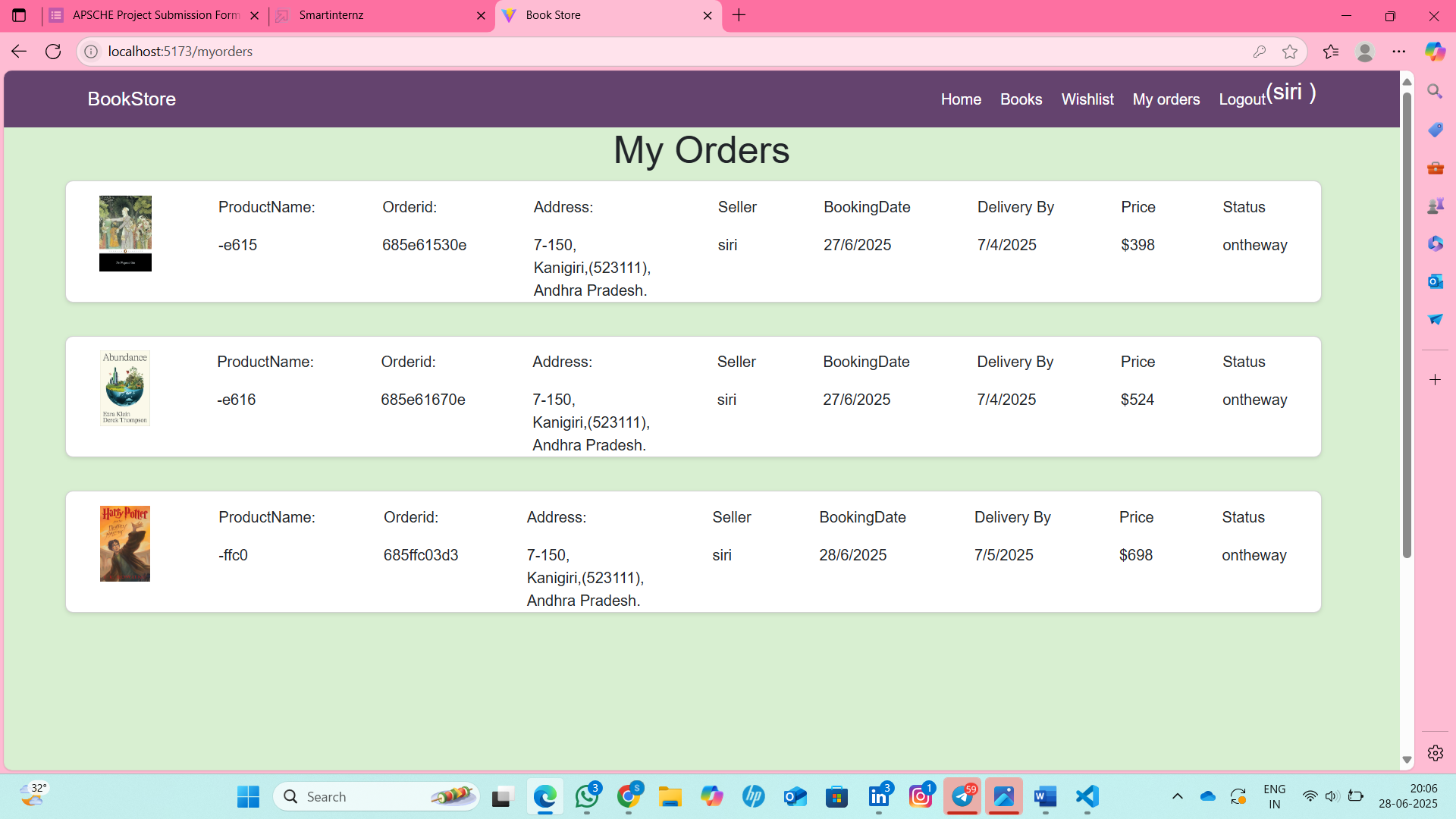
Books Page:-



Wishlist Page:-

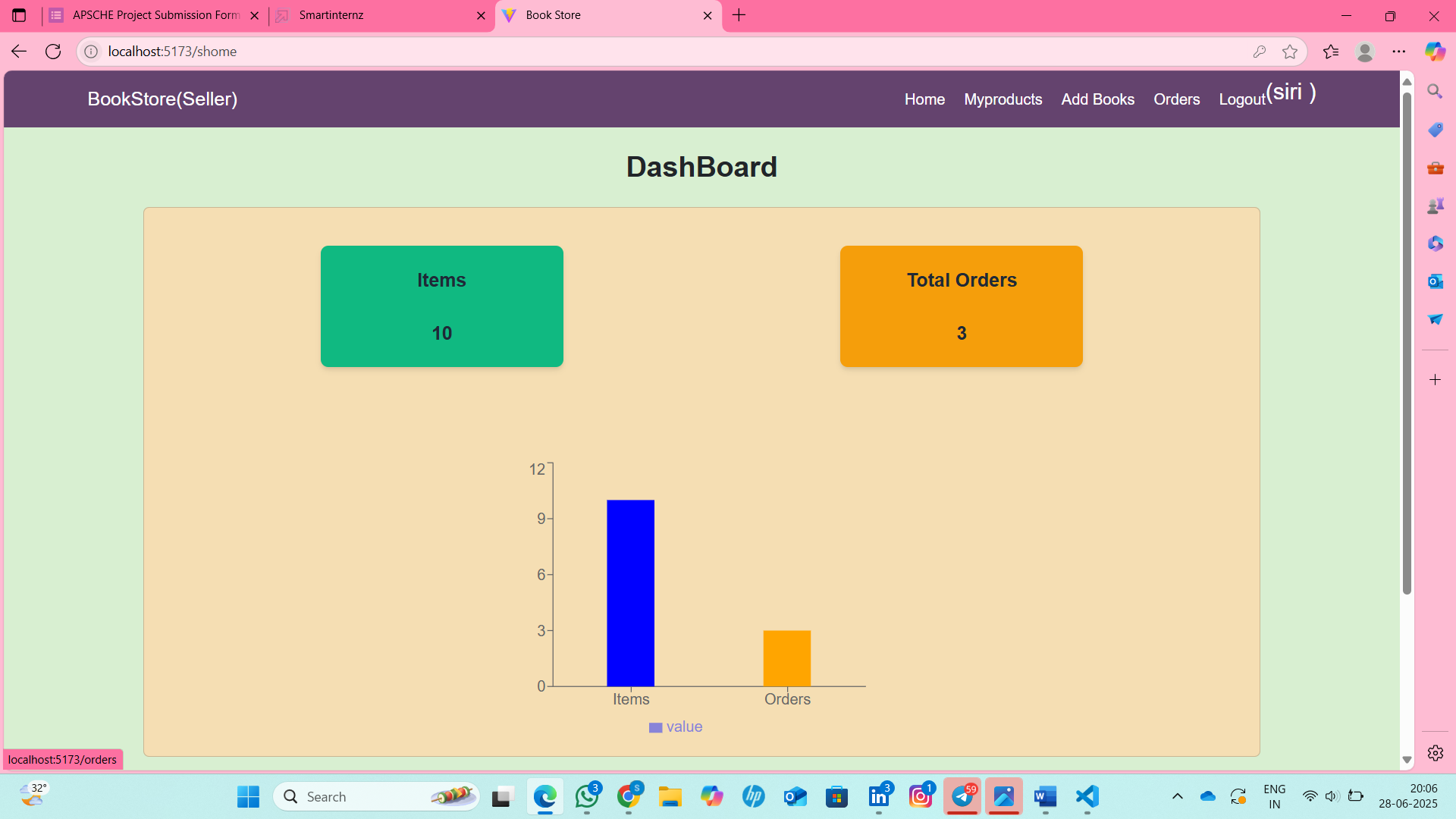


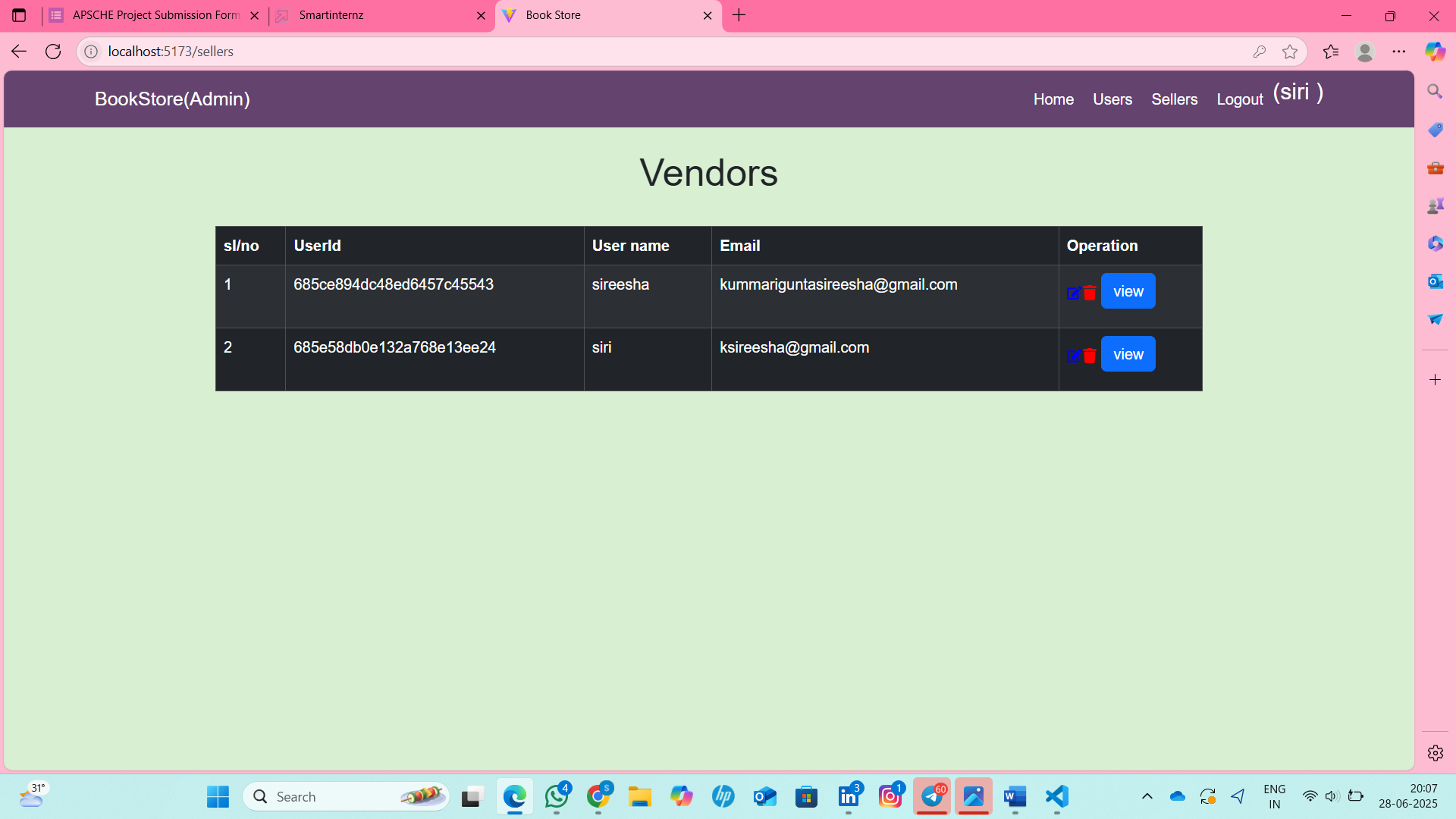
My Bookings Page :-



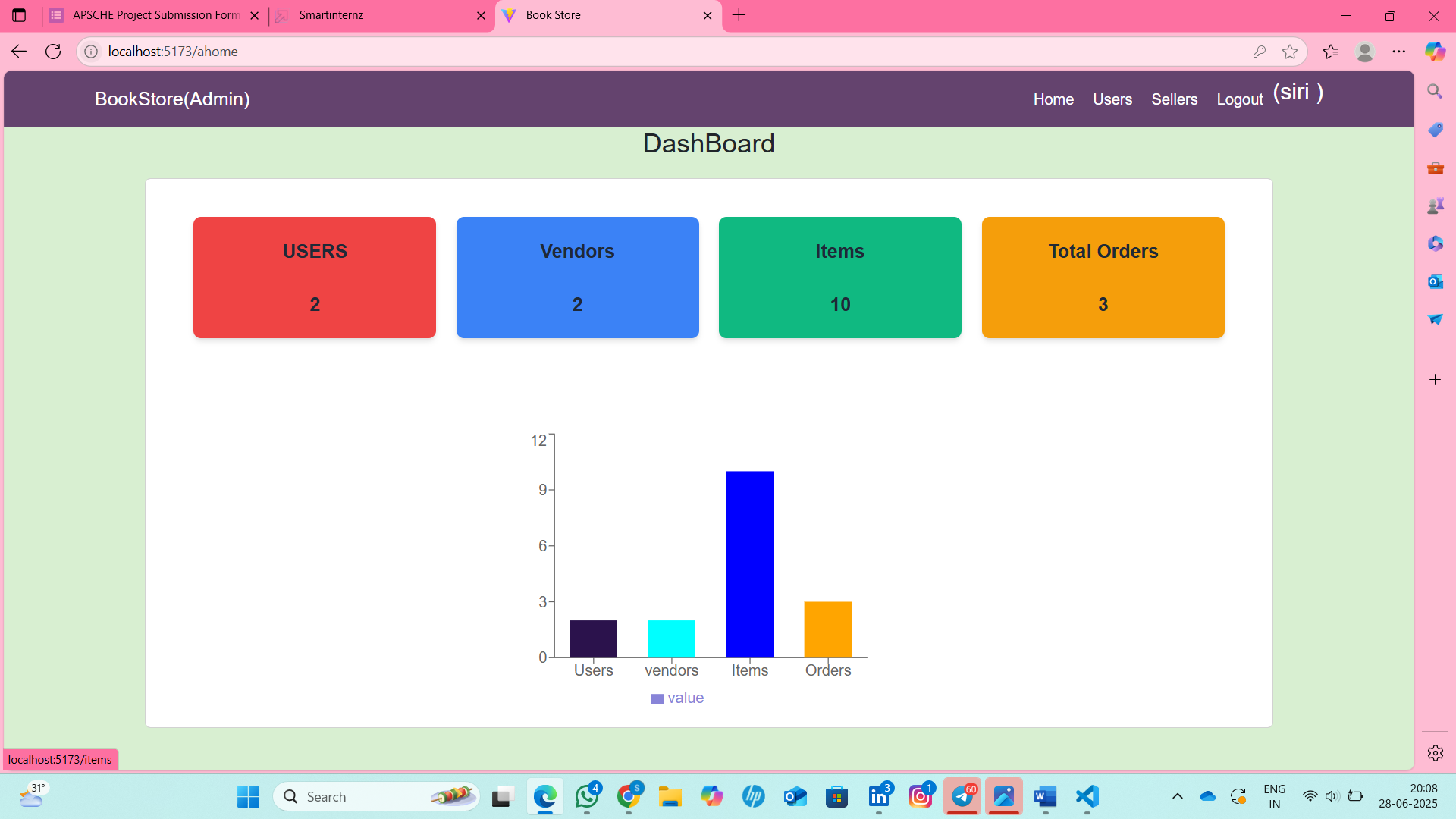
Seller Dashboard:-

Seller Items:

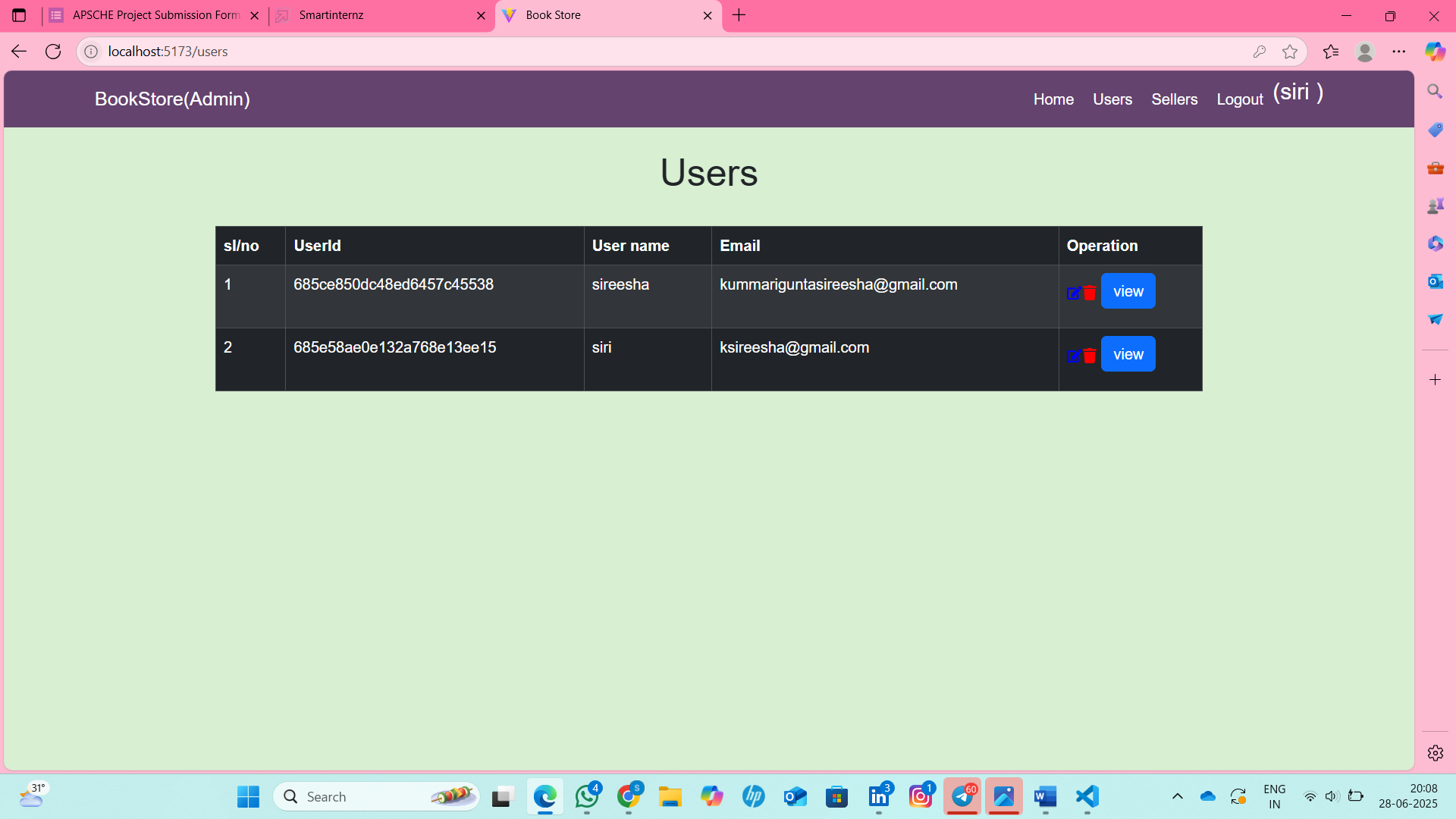




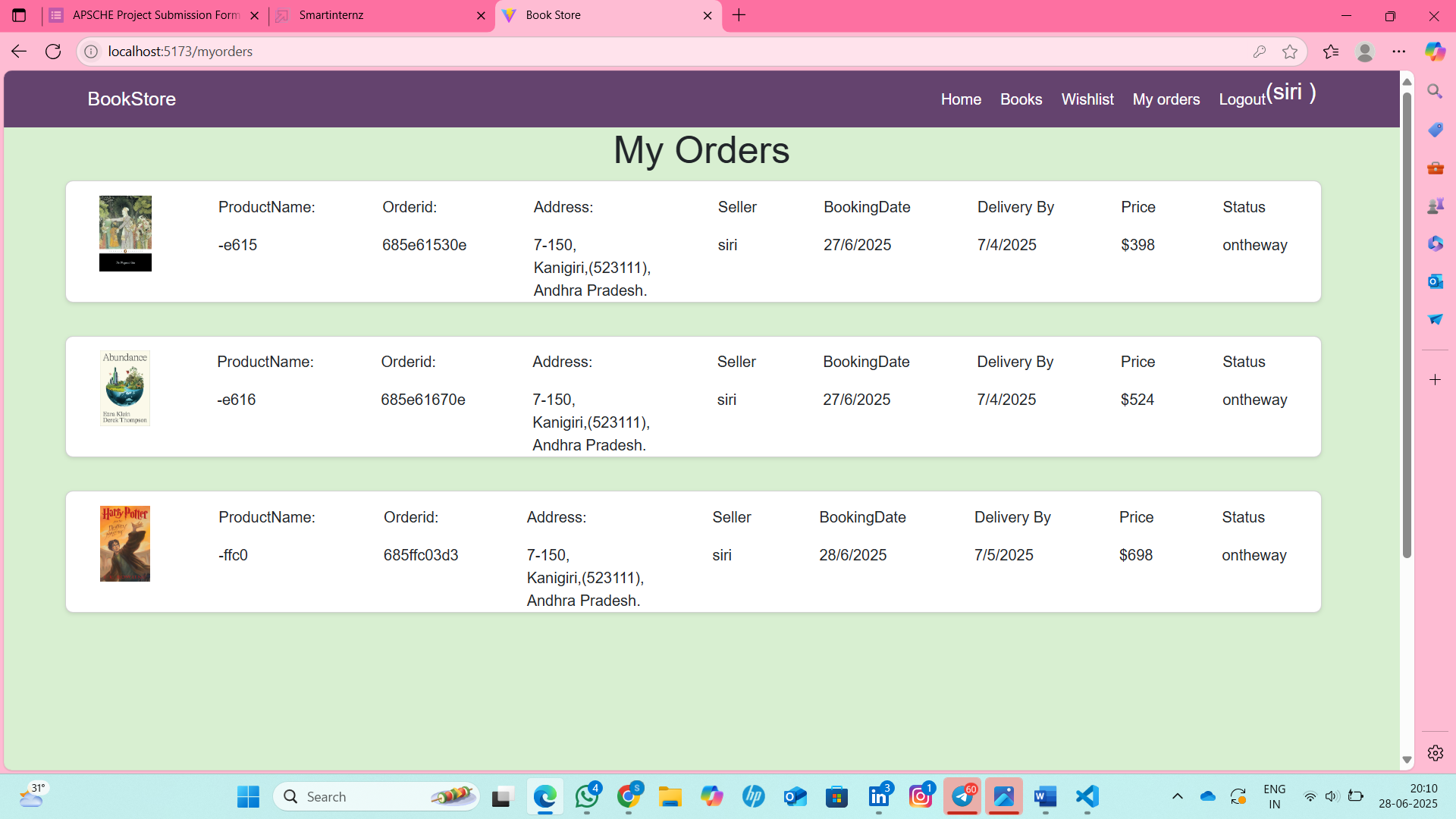
Admin Dashboard:-



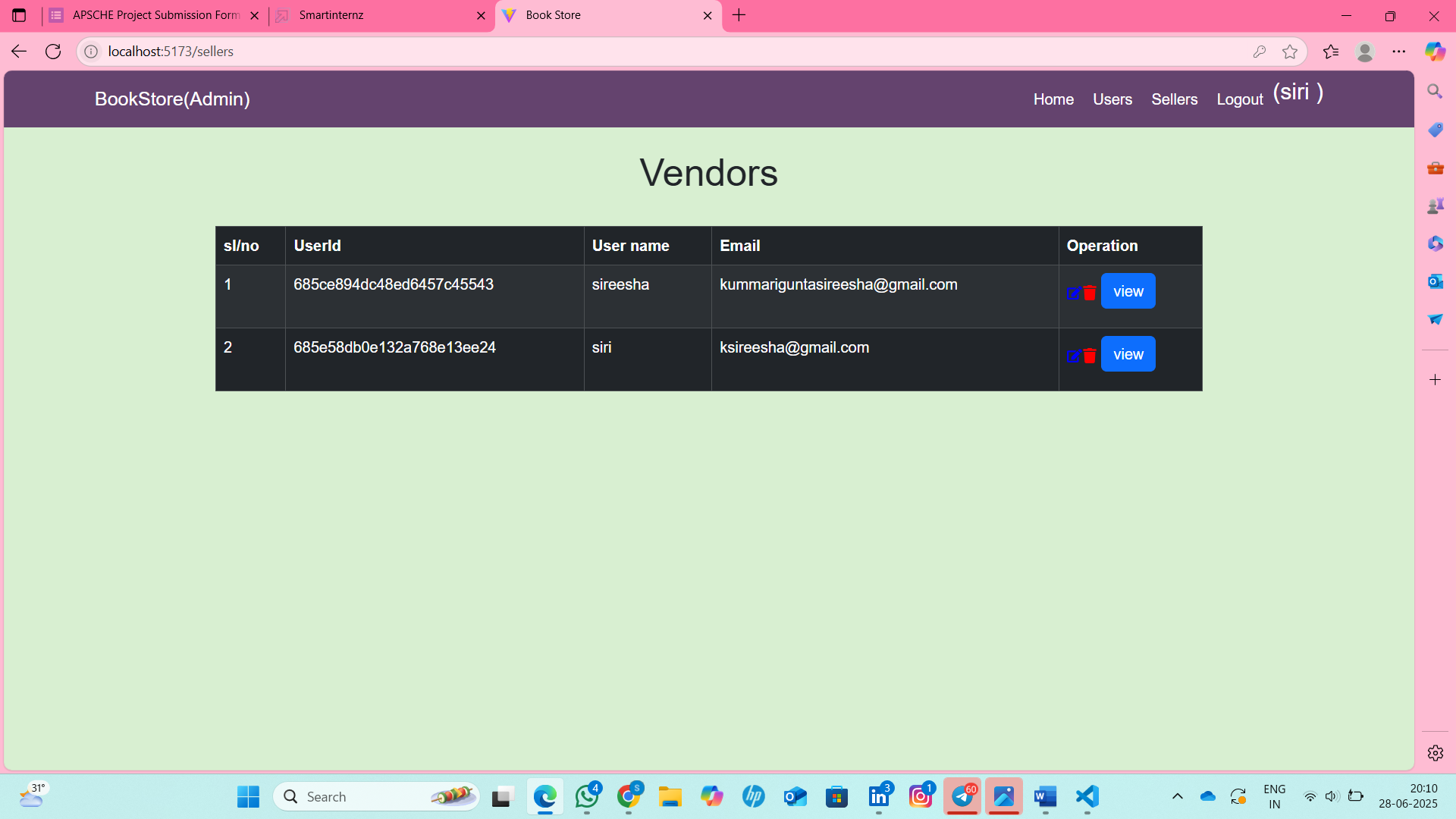
Users Page:



User Orders:



Sellers Page:



The demo of the app is available at:-

[**https://drive.google.com/file/d/19nVCk1Rw\_SS4aRDnEnnXaFPRRa527MLQ/view**](https://drive.google.com/file/d/19nVCk1Rw_SS4aRDnEnnXaFPRRa527MLQ/view)

**CONCLUSION:**The Book Store App is a simple and effective platform for buying books online. It helps users easily browse, search, and purchase books from different categories and authors.

Built using the MERN stack, the app provides a smooth user experience with secure login, fast performance, and real-time updates. Admins can manage books and users through a dedicated dashboard.With its clean design and useful features, the app makes reading more accessible and enjoyable for everyone.