**BOOK STORE PROJECT USING MERN STACK**

**INTRODUCTION :**

Introducing our **MERN Stack Book-Store Application**, where technology meets literature. Built with **MongoDB, Express.js, React, and Node.js**, this app offers a seamless, interactive experience for book lovers. Explore a vast collection, from the latest releases to timeless classics, all at your fingertips.

With **MongoDB** for fast access to books, **Express.js** for a reliable server, and **Node.js** for high performance, your reading experience is smooth and efficient. **React** powers a dynamic, responsive interface that works across all devices.

Embrace the future of reading—discover, browse, and enjoy books effortlessly, anywhere, anytime.

**DESCRIPTION:**

Welcome to the **MERN Stack Book-Store Application**—a revolutionary digital platform designed for book lovers, blending modern technology with the timeless joy of reading. Built using the powerful **MERN** stack—**MongoDB, Express.js, React, and Node.js**—this application provides an immersive, interactive experience for users to discover, explore, and enjoy a vast collection of literary works.

**Key Features:**

1. **MongoDB (Database):**
   * At the core of the application lies **MongoDB**, a NoSQL database that ensures efficient and scalable storage for an ever-growing collection of books. MongoDB’s flexible document-based structure allows us to store detailed book information, including titles, authors, genres, and user reviews, making it easy to access and manage a large catalog.
2. **Express.js (Server Framework):**
   * **Express.js** is a fast and minimalist web application framework for Node.js that forms the backbone of the server-side of the application. It simplifies the routing and handling of HTTP requests, ensuring the application remains responsive and quick, even with a high volume of users interacting with the platform.
3. **Node.js (Runtime Environment):**
   * **Node.js** provides the server-side runtime that powers the application, leveraging its non-blocking, event-driven architecture to handle multiple user requests simultaneously. This ensures smooth, high-performance operations, making the browsing and book discovery process fast and responsive.
4. **React (Frontend Library):**
   * The user interface is powered by **React**, a JavaScript library known for its flexibility and efficiency. React ensures a dynamic, interactive, and visually appealing experience, where every action (such as searching, browsing, and book selection) feels instant and engaging. With **React Router**, users can smoothly navigate through different pages and features without page reloads, offering a seamless browsing experience.
   * The application is fully responsive, adjusting beautifully to any screen size, ensuring a consistent experience across desktops, tablets, and smartphones.

**SCENARIO :**

Sarah is an avid reader with a passion for exploring new genres and authors. However, her busy schedule often leaves her with limited time to visit physical bookstores. Sarah is looking for a solution that allows her to discover and purchase books conveniently, without compromising her reading preferences or the joy of browsing through a bookstore.

**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.

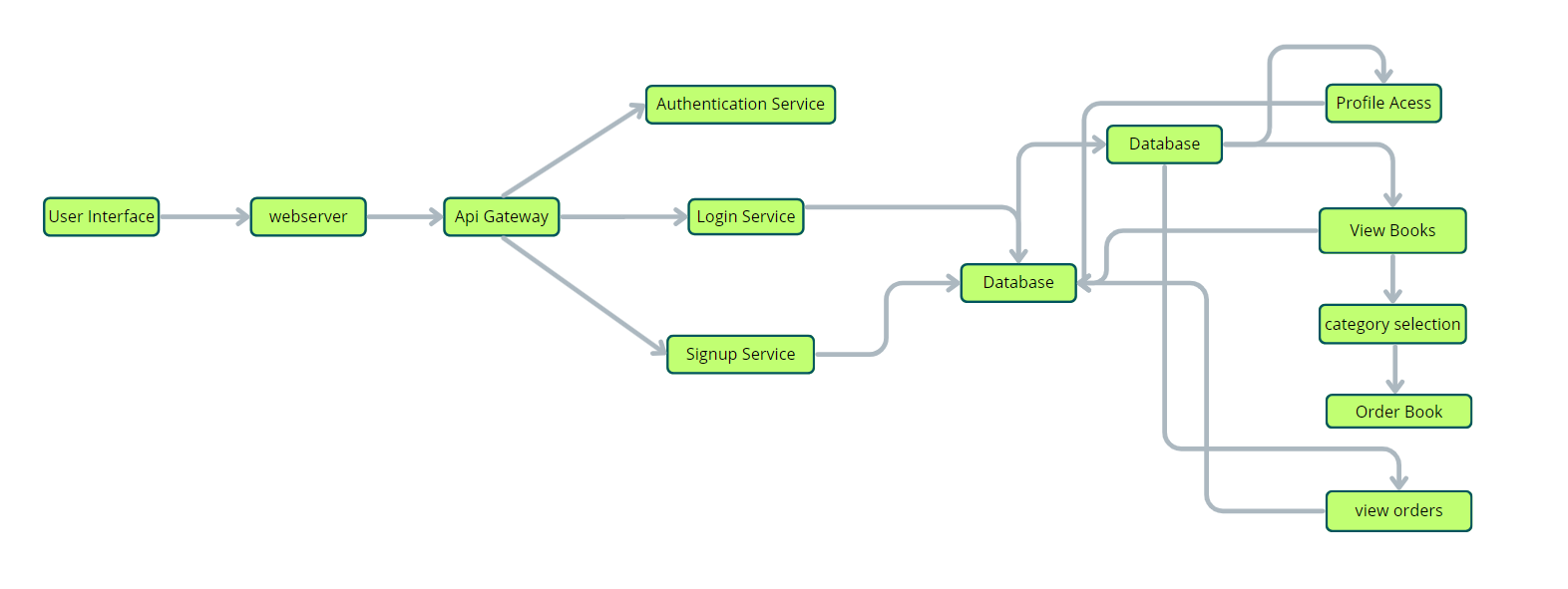
**Book Selection:** Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

**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.

**Technical Architecture:**



 User Interface: The user interface will serve as the platform where customers can browse books, search for specific titles or authors, read book descriptions, and make purchases. It should be intuitive and user-friendly, enabling easy navigation and exploration of available books.

Web Server: The web server hosts the user interface of the book store app, serving dynamic web pages to users and ensuring a seamless browsing and shopping experience.

API Gateway: Similar to the original architecture, the API gateway will serve as the central entry point for client requests, directing them to the relevant services within the system. It will handle requests such as fetching book information, processing orders, and managing user accounts.

Authentication Service: The authentication service manages user authentication and authorization, ensuring secure access to the book store app and protecting sensitive user information during the browsing and purchasing process.

Database: The database stores persistent data related to books, including information such as titles, authors, genres, descriptions, prices, and availability. It also stores user profiles, purchase history, and other essential entities crucial to the book store app.

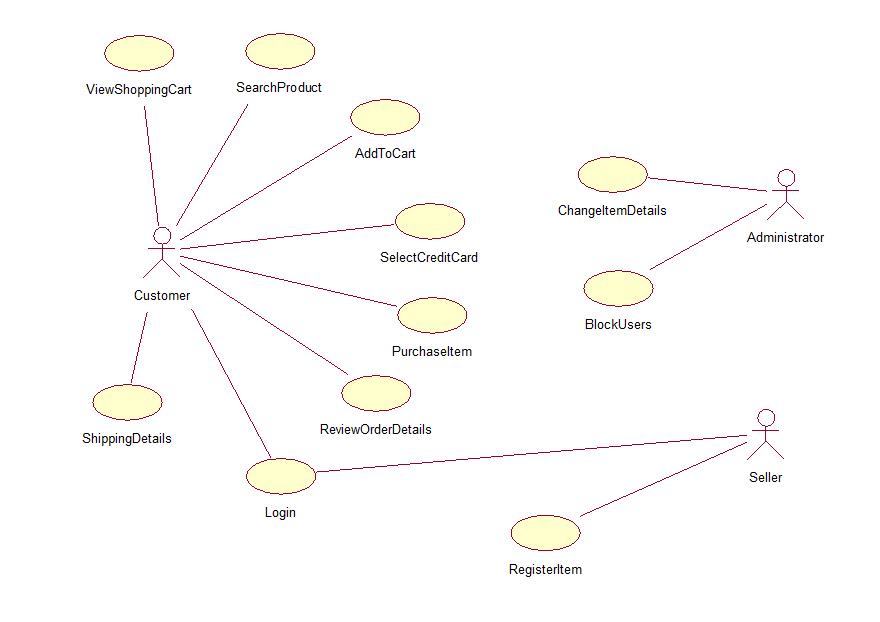
View Books: This feature allows users to browse through the available books. They can explore different categories and genres to discover books of interest.

Category Selection: Users can select specific categories or genres to filter and refine their book browsing experience, making it easier to find books tailored to their preferences.

Inventory Management Service: This service manages information about available books, including their availability, stock levels, and ratings. It ensures efficient management of the book inventory and seamless integration with the browsing and purchasing process.

Order Management Service: This service facilitates the ordering process, allowing users to add books to their cart, specify quantities, and complete purchases securely. It also handles order tracking and status updates in real-time.

**ER-Diagram**

****

The Book Store platform features a structured relational model with key relationships like **Many-to-Many** between users and books (via an "Interaction" table), **One-to-Many** between books and inventory, and **One-to-Many** between users and orders. Additional relationships include **Many-to-Many** for books and authors/genres, and **Many-to-One** for reviews and users. Key features include secure **user registration**, comprehensive **book listings**, personalized **book selection** (by genre, author, ratings), and a streamlined **purchase process** with order confirmation and **order history** tracking. Admins can manage books, inventory, and users via the **organizer** and **admin dashboards**, while **reporting and analytics** offer insights into sales and user trends. Integration with **third-party APIs** enhances the platform with payment processing, shipping, and book recommendations.

**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](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.

V**ersion 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.

**Application 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.

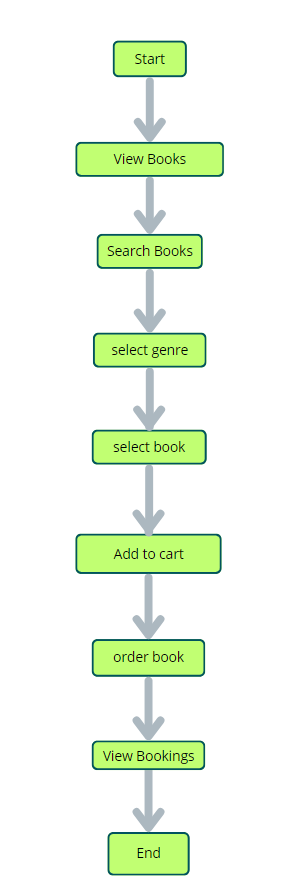
Access Profile: Users have the option to access their profiles, allowing them to view or update personal information, preferences, and order history.

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.

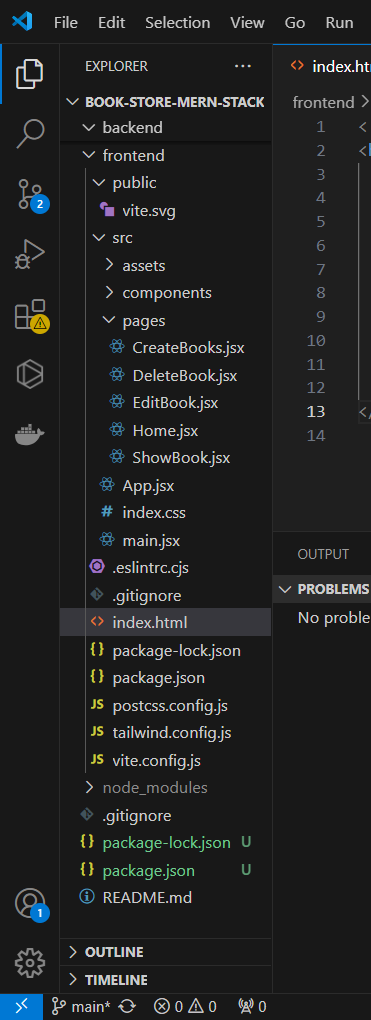
Book Purchase: Users navigate through the available book options and specify the quantity of each book they wish to purchase. They can also choose additional options such as e-book format or special editions.

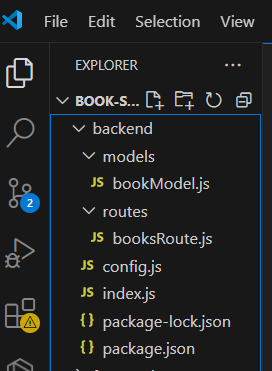
View Orders: Users have the option to view their current and past orders. This section provides details about ordered books, order status, and payment history.

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:**

**Project Setup and Configuration**

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

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

**STEP-2 Create project folders and files:**

* Client folders.
* Server folders.

**STEP-3 Install Packages:**

**Frontend npm Packages**

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

**Backend npm Packages**

* Express.
* Mongoose.
* Cors.

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

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

**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.

Referance Link:- <https://drive.google.com/file/d/1QwDswQXZ8e22PxYTb4HEmvJFQ9AzQR3F/view?usp=drive_link>

REFERENCE VIDEO: <https://drive.google.com/file/d/1QwDswQXZ8e22PxYTb4HEmvJFQ9AzQR3F/view?usp=drive_link>

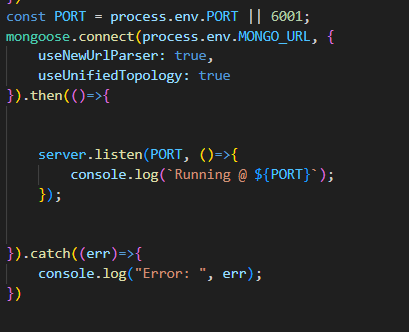
**Database**

**1. Configure MongoDB:**

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

**2. Connect database to backend:**

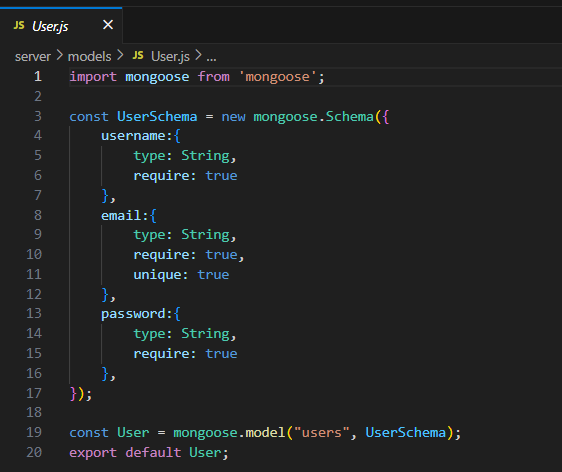
                Now, make sure the database is connected before performing any of the actions through the backend. The connection code looks similar to the one provided below.



**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.



**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.

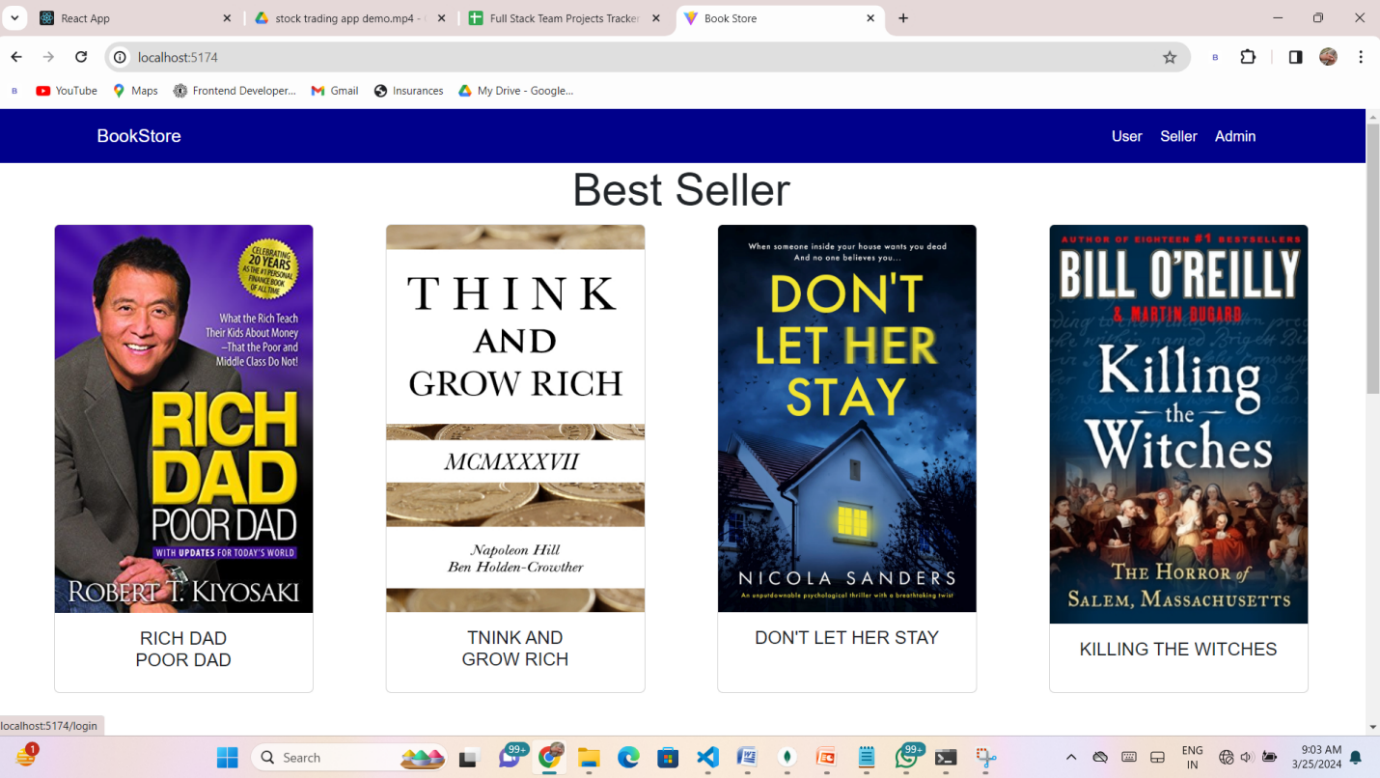
• Implement data binding.

Referance:- <https://drive.google.com/file/d/1t7oZj1Au_Itgv5iyykvb_c6FU4S_b0Q/view?usp=drive_link>

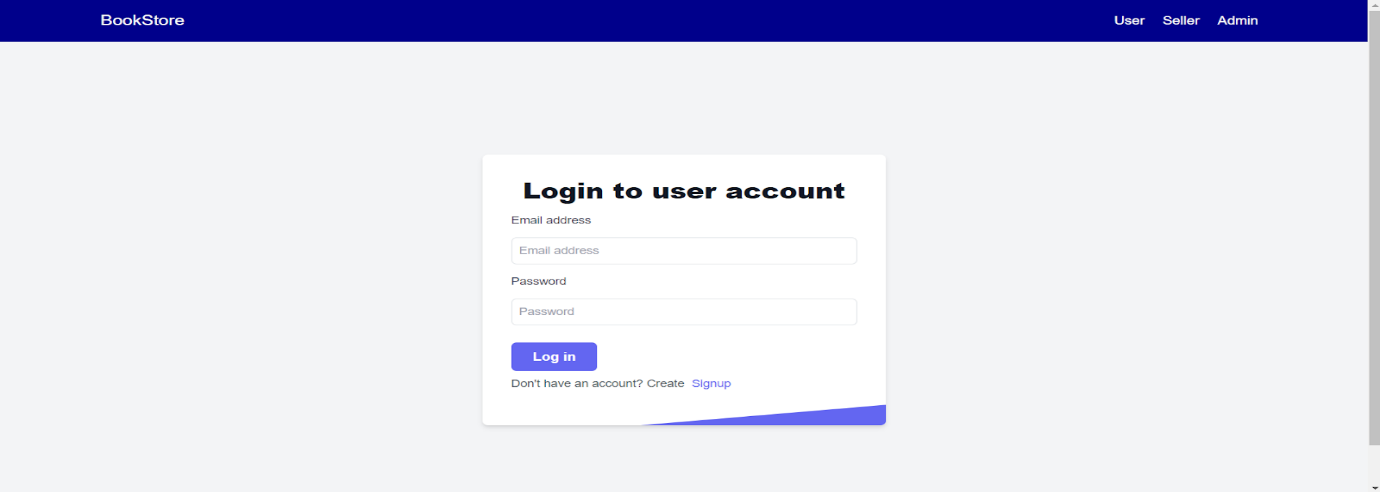
REFERENCE VIDE: <https://drive.google.com/file/d/1t7oZj1Au_Itgv5iyykvb_c6FU4S_b0Q/view?usp=drive_link>

**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.

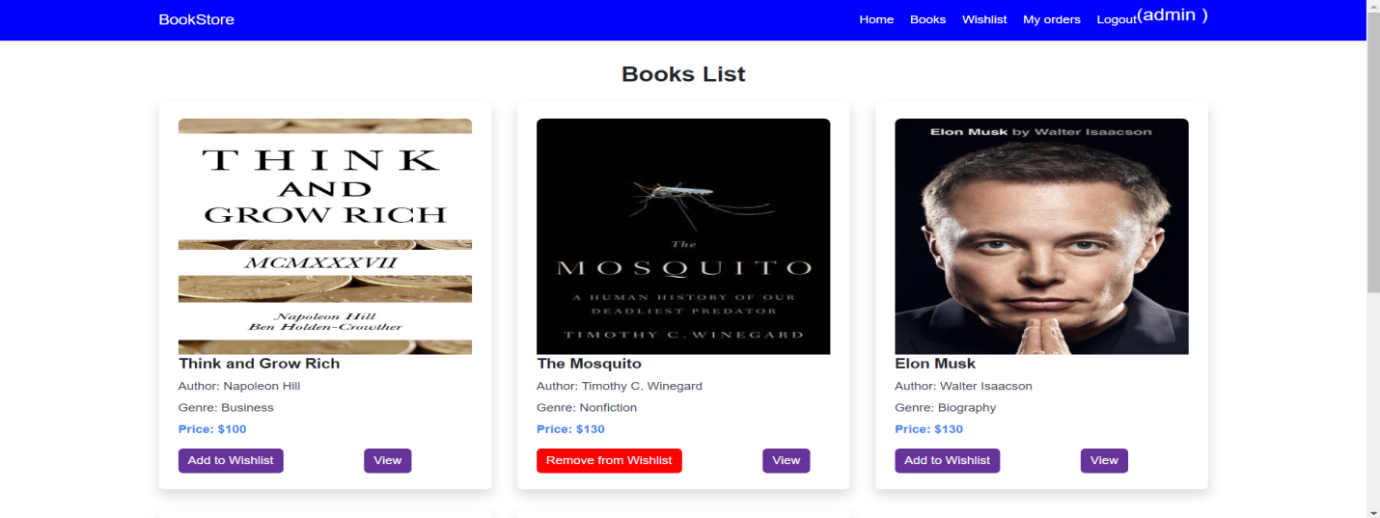
Landing page:- 

Login Page:-

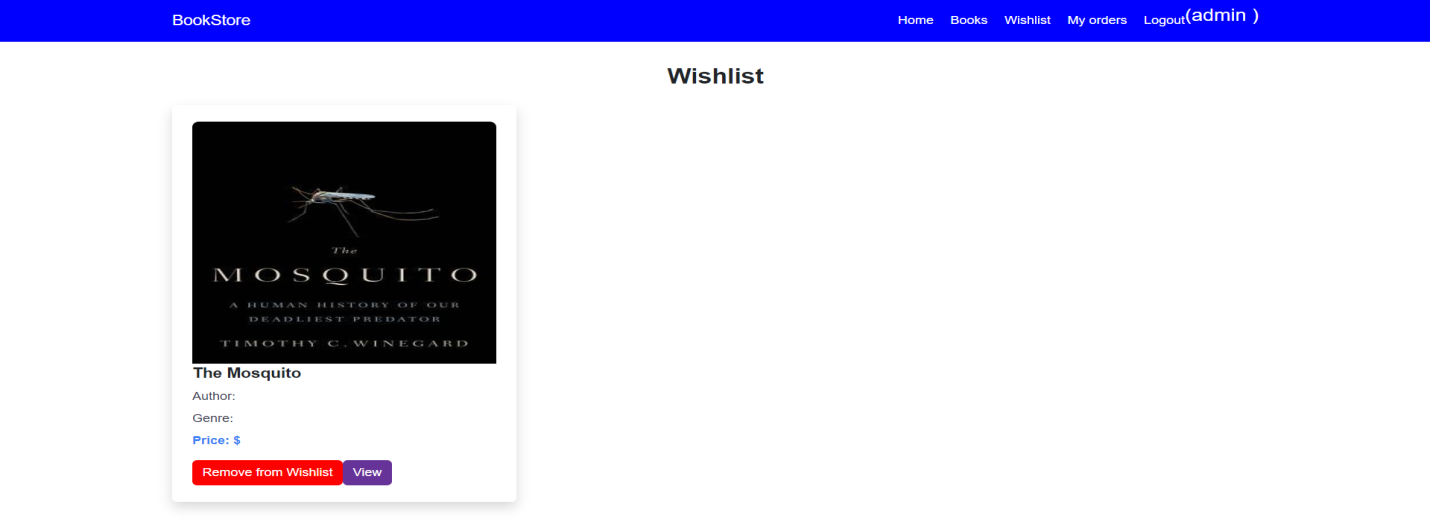


Home Page:- 

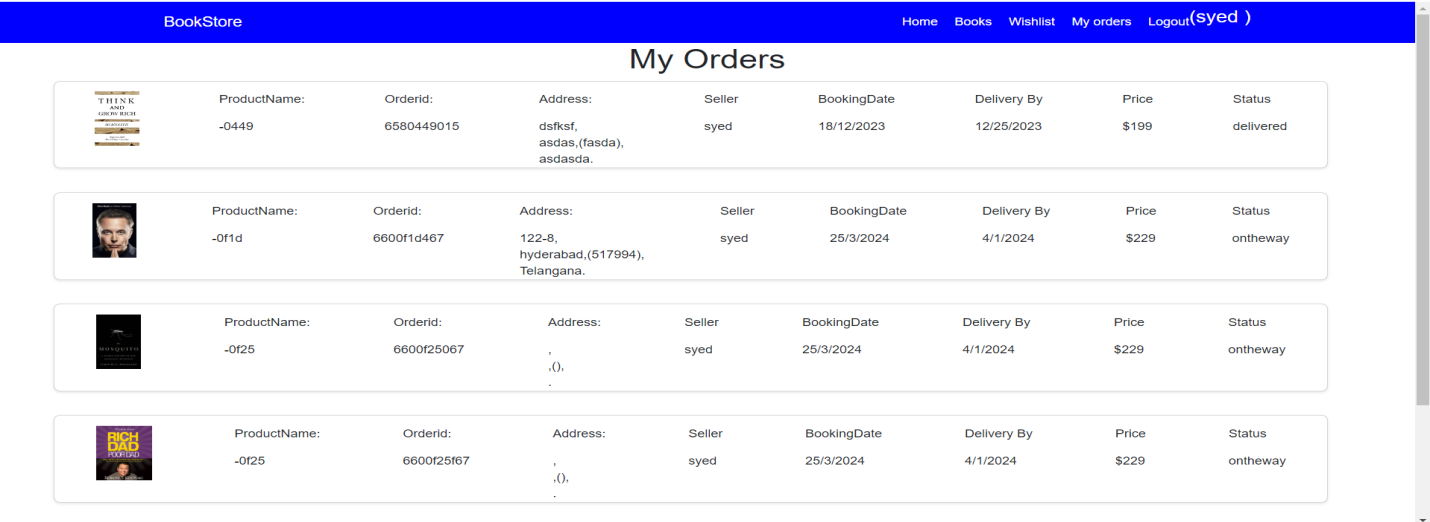
Books Page:-



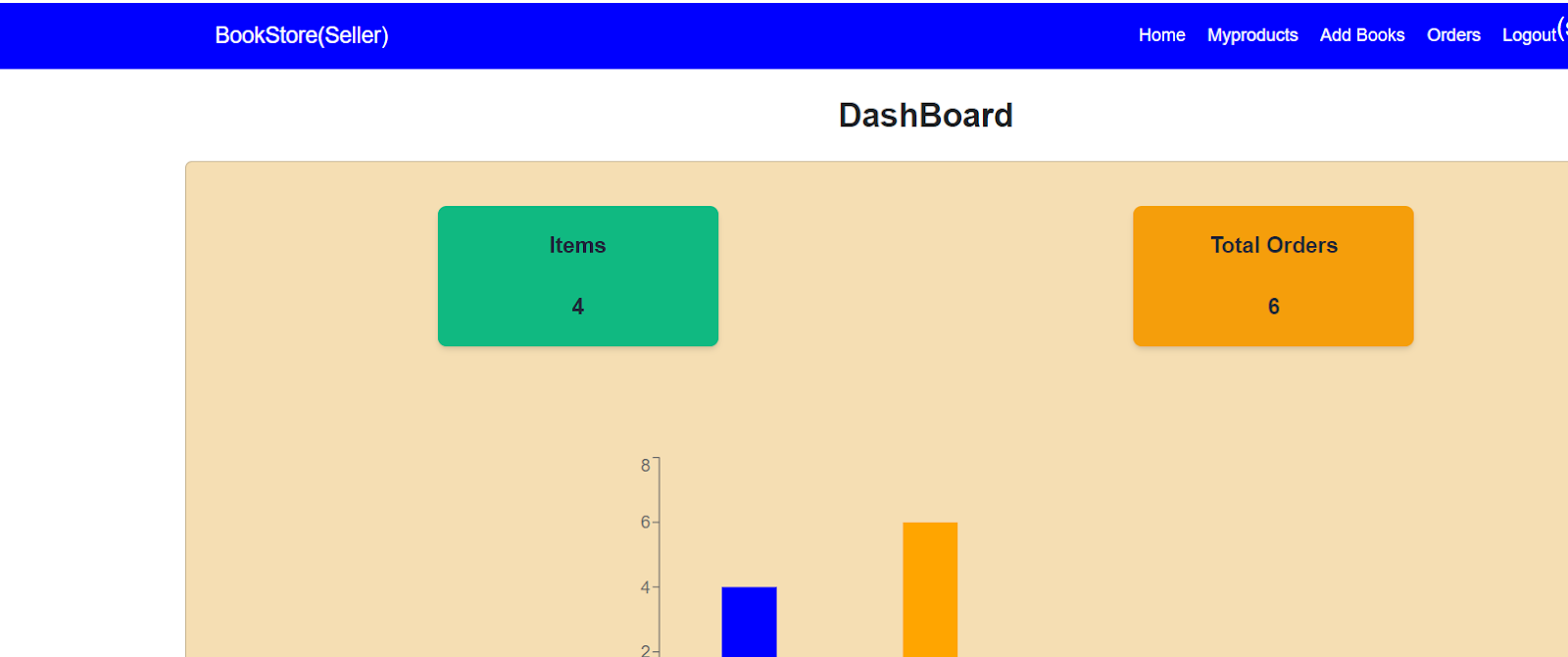
Wishlist Page:-



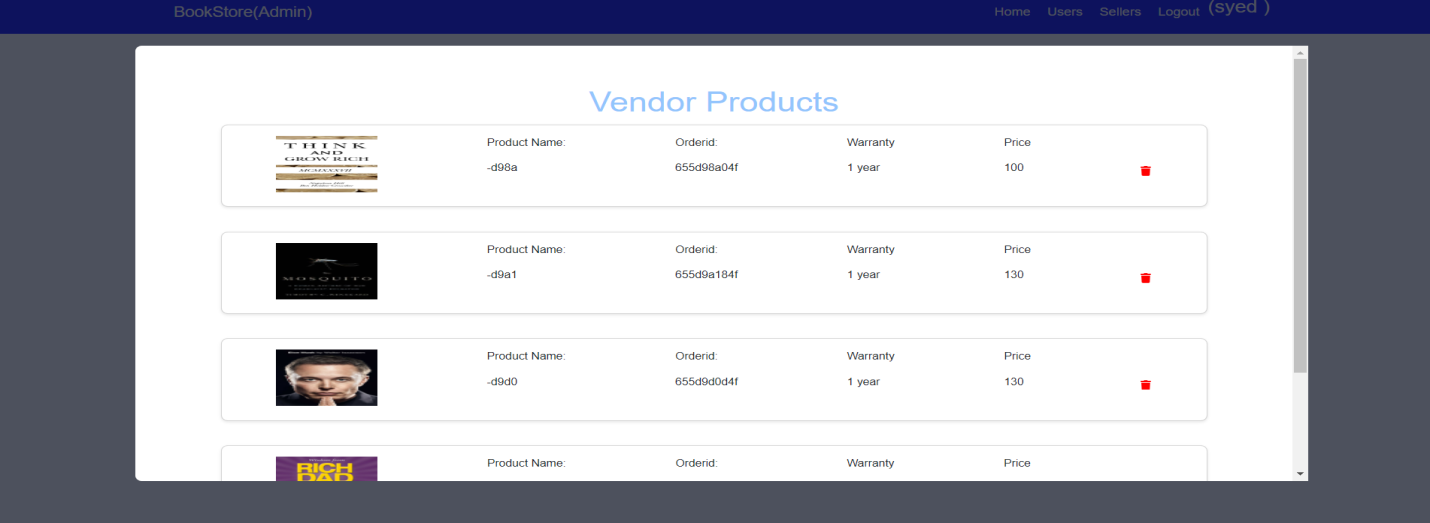
My Bookings Page :-



Seller Dashboard:-



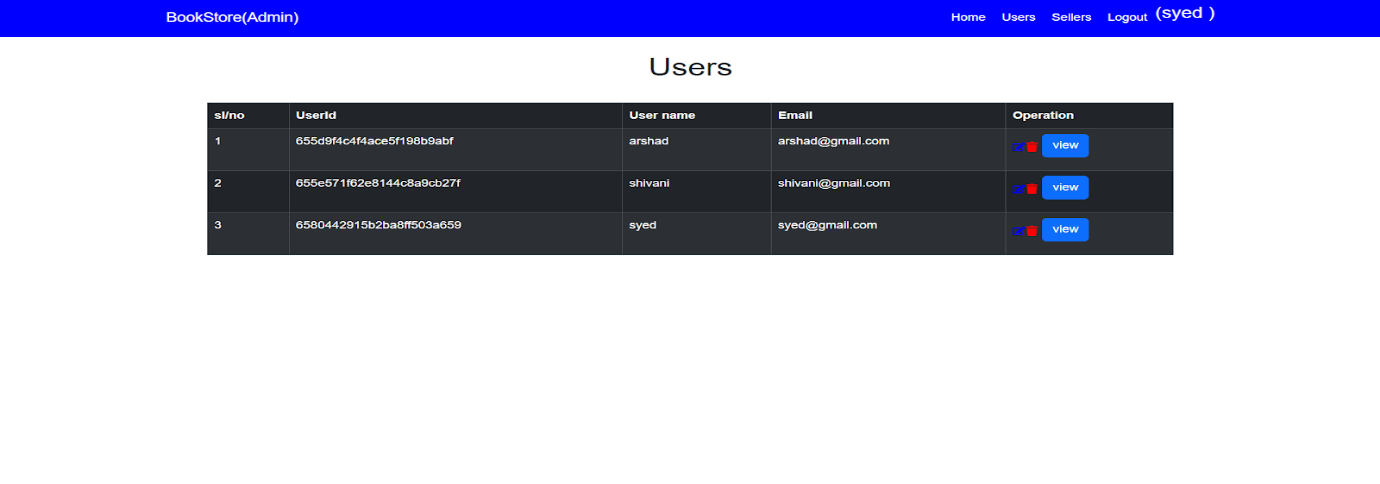
Seller Items:



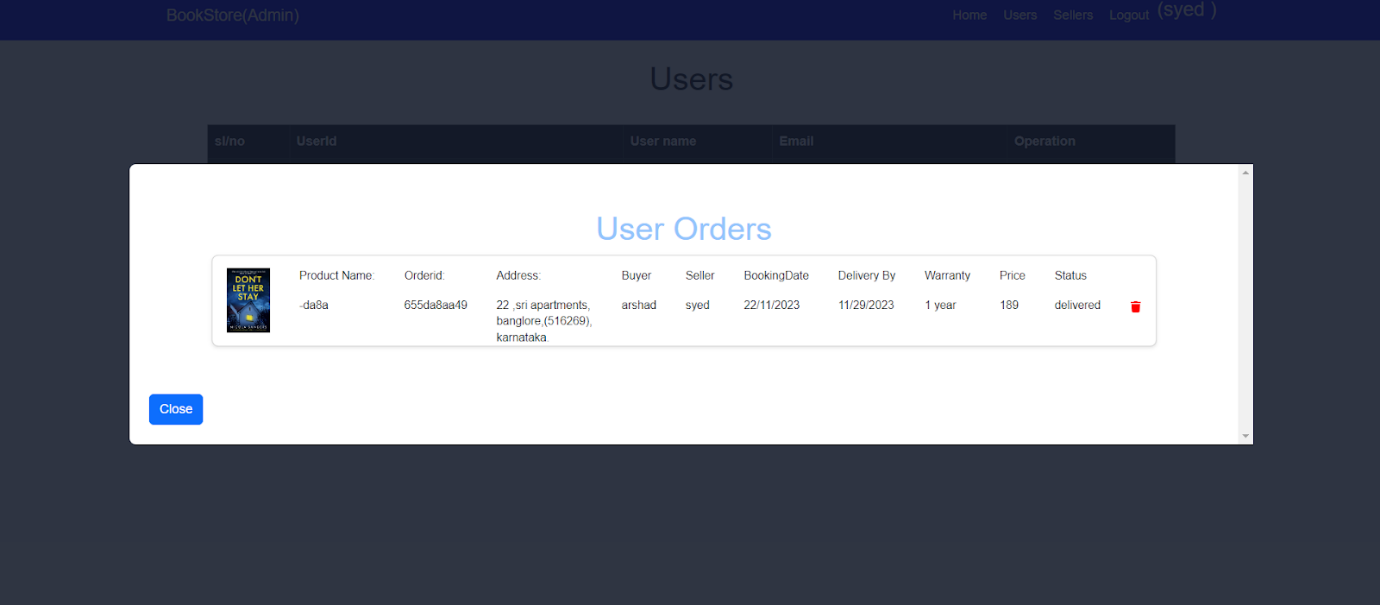
Admin Dashboard:-



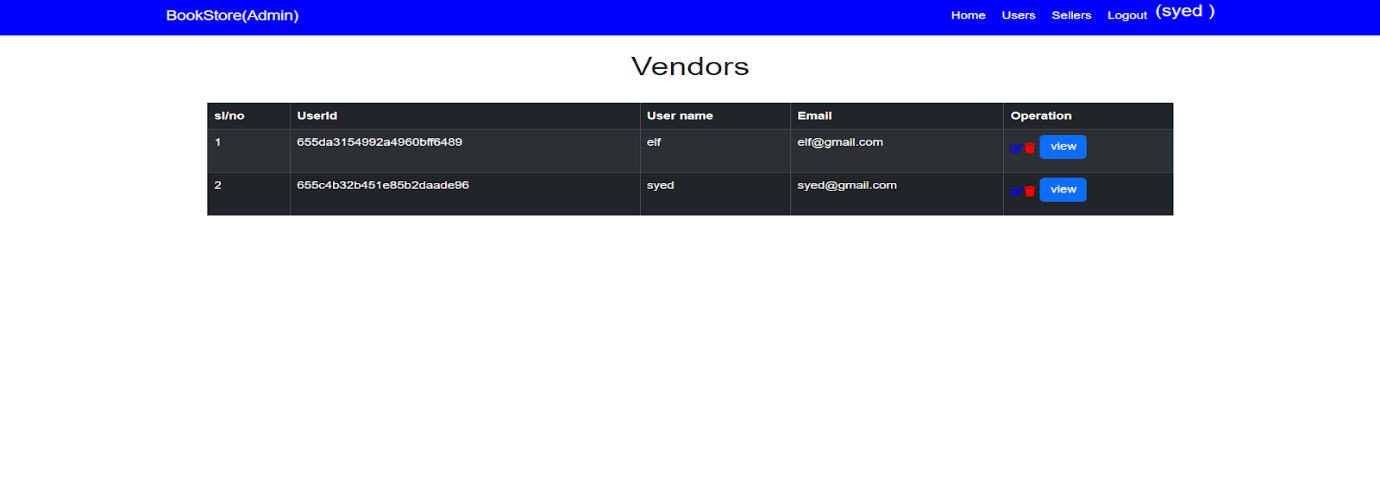
Users Page:



User Orders:



Sellers Page:



THE DEMO OF THE APP IS AVAILABLE AT :

<https://drive.google.com/file/d/1OPkPuGTeCiaTqljT-Lmt9hA2vgZt5T18/view>