# Project Title

# BookNest: Where Stories Nestle (MERN)

## Team Members:

Here List team members and their roles:

**1. P.Sriasha** (Full Stack Developer): Combines both frontend and backend responsibilities, ensuring smooth communication between the two. This role also handles bug fixing, feature integration, and overall system performance.

**2. P.Geetha Varshini** (Frontend Developer): Responsible for designing the user interface using React.js. This role focuses on ensuring a responsive, user-friendly design, as well as integrating the frontend with backend APIs.

**3**. **N.Kamal** (Backend Developer): Develops the backend server using Node.js and Express.js, ensuring the creation of secure, scalable RESTful APIs, as well as handling authentication, data processing, and business logic.

**4.N.Deena**(Backend Developer): Develops the backend server using Node.js and Express.js, ensuring the creation of secure, scalable RESTful APIs, as well as handling authentication, data processing, and business logic.

## Introduction:

Tired of scattered book collections? Introducing BookNest: Where Stories Nestle—a revolutionary platform that redefines reading. Discover, organize, and share your favorite books effortlessly. With seamless book tracking, personalized recommendations, and a vibrant reading community, your next great story is just a click away!

## Project Overview:

Get ready to transform your reading experience with BookNest: Where Stories Nestle, an innovative platform designed for book lovers. Discover, organize, and share your favorite books effortlessly while engaging with a vibrant community of readers.

Effortlessly track your reading progress, create personalized book collections, and receive tailored recommendations that match your literary taste. Whether you're diving into a new novel, sharing insights with fellow readers, or saving must-read titles for later, BookNest makes it all seamless.

We prioritize your reading experience and data security. Our platform ensures a user-friendly interface with secure access, allowing you to explore, connect, and organize your books without worry.

## Goals of Project:

• Enhance reader engagement by encouraging discussions, book reviews, and shared reading experiences.
• Facilitate knowledge sharing by allowing users to save, recommend, and review books.
• Ensure data privacy and security with robust protection for user information and reading preferences.
• Promote book discovery through personalized recommendations and curated reading lists.

## Features of BookNest:

**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.

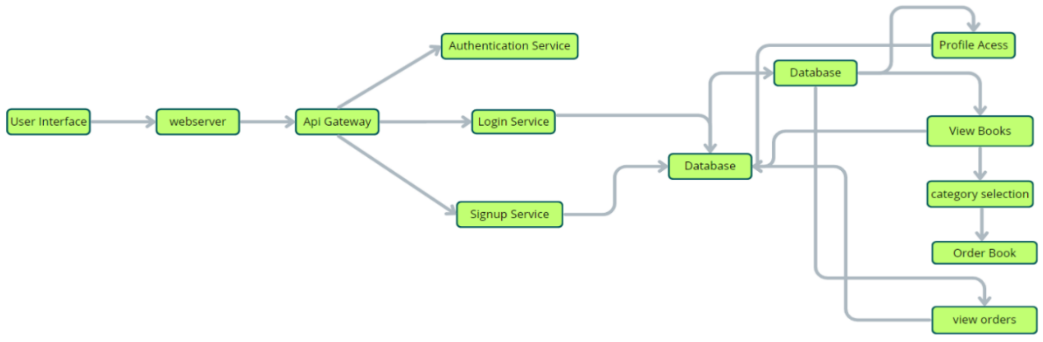
**Book Selection:** Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

**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.

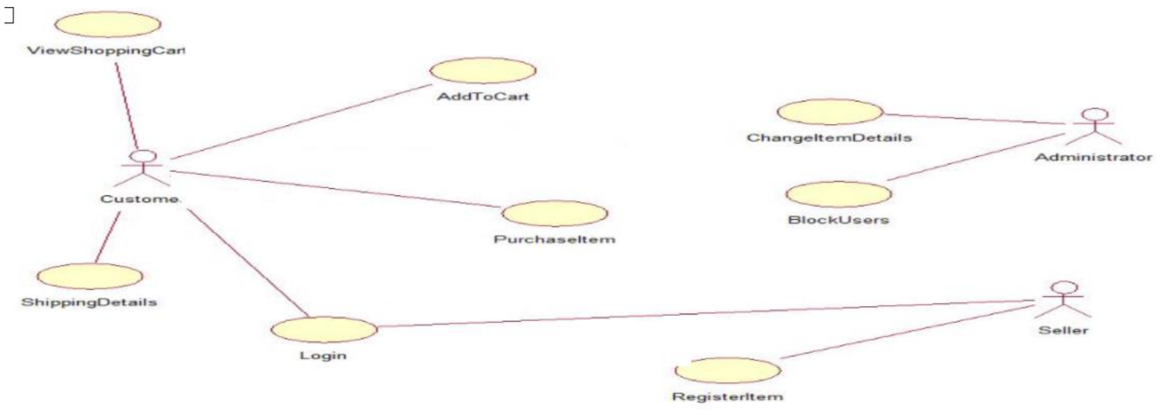
View Books: This feature allows users to browse through the available books. They can explore different categories and genres to discover books of interest.

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:



**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)

## **PRE-REQUISITES**:

Here are the key prerequisites for developing a full-stack application using Node.js, Express.js, MongoDB, React.js, Socket.io:

- ✓ **Node.js and npm**:

Node.js is a powerful JavaScript runtime environment that allows you to run JavaScript code on the server-side. It provides a scalable and efficient platform for building network applications.

Install Node.js and npm on your development machine, as they are required to run JavaScript on the server-side.

- Download: https://nodejs.org/en/download/
- Installation instructions: https://nodejs.org/en/download/package-manager/

✓ **Express.js**:

Express.js is a fast and minimalist web application framework for Node.js. It simplifies the process of creating robust APIs and web applications, offering features like routing, middleware support, and modular architecture. Install Express.js, a web application framework for Node.js, which handles server-side routing, middleware, and API development.

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

```
npm install express
```

✓ **MongoDB**:

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

- Download: https://www.mongodb.com/try/download/community
- Installation instructions: https://docs.mongodb.com/manual/installation/

✓ **React.js**:

React.js is a popular JavaScript library for building user interfaces. It enables developers to create interactive and reusable UI components, making it easier to build dynamic and responsive web applications.

Install React.js, a JavaScript library for building user interfaces.

Follow the installation guide: https://reactjs.org/docs/create-a-new-react-app.html

## 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 Bookstore

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. To Connect the Database with Node JS go through the below provided link:
• https://www.section.io/engineering-education/nodejs- mongoosejs-mongodb/

✓ **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.

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

To run the existing BookNest App project downloaded from google drive:

Firstly, download the code from the google drive link provided below

https://github.com/Sriasha123/Booknest-Where-Stories-Nestle

**Install Dependencies:**

• Navigate into the cloned repository directory:

```
cd Bookstore
```

• Install the required dependencies by running the following commands:

```
cd Frontend
npm install
cd ../Backend
npm install
```

✓ **Start the Development Server**:

• To start the development server, execute the following command:

```
npm run dev
```

• The video conference app will be accessible at http://localhost:3000

✓ **Access the App:**

• Open your web browser and navigate to http://localhost:3000.

• You should see the login page, indicating that the installation and setup were successful.

You have successfully installed and set up the social media app on your local machine. You can now proceed with further customization, development, and testing as needed.

## Project structure:

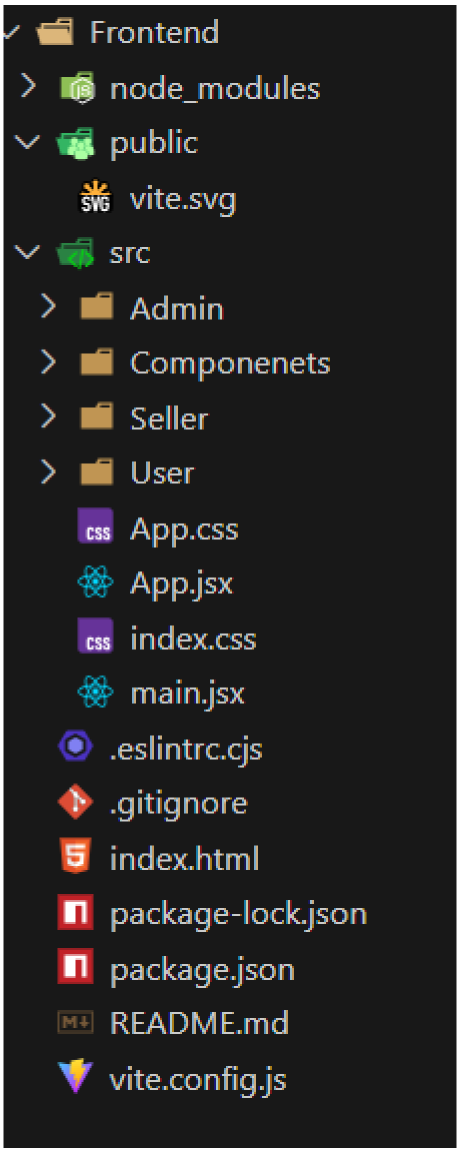
- Inside the Bookstore directory, we have the following folders

  Backend

  Frontend

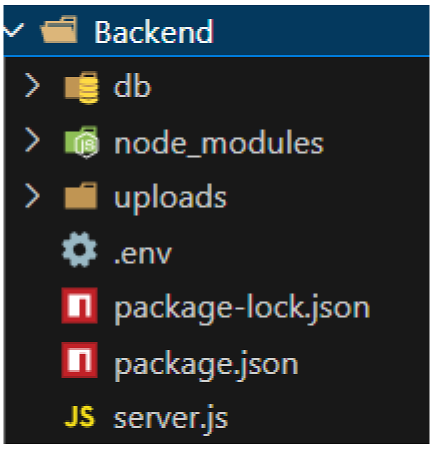
- **Frontend directory:**

  The below directory structure represents the directories and files in the client folder (front end) where, react js is used.



```
Frontend
  node_modules
  public
    vite.svg
  src
    Admin
    Componenets
    Seller
    User
    App.css
    App.jsx
    index.css
    main.jsx
  .eslintrc.cjs
  .gitignore
  index.html
  package-lock.json
  package.json
  README.md
  vite.config.js
```

- **Server directory:**
  The below directory structure represents the directories and files in the server folder (back end) where, node js, express js and mongodb.



```
Backend
  db
  node_modules
  uploads
  .env
  package-lock.json
  package.json
  server.js
```

## Project Flow:

### Project demo:

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

Demo link:

https://drive.google.com/file/d/1AlGrV2bCelM_6iKwJ7XK9d4zY1yzIBna/view?usp=drive_link

Use the code in:

https://github.com/Sriasha123/Booknest-Where-Stories-Nestle

## Milestone 1: Project setup and configuration.

- **Folder setup**:

  Now, firstly create the folders for frontend and backend to write the respective code and install the essential libraries.

  - frontend folders.
  - backend folders

- **Installation of required tools**:

Open the frontend folder to install necessary tools. For frontend (client), we use:

| Tools/libraries | Installation command |
|---|---|
| React Js | `npx create-react-app .` |
| Vite | `npm create vite@latest ./` |
| Tailwind css | `npm install bootstrap` |
| Axios | `npm install axios` |
| Firebase | `npm install firebase` |
| Frame-Motion | `npm install frame-motion` |

Open the backend folder to install necessary tools. For backend (server), we use:

| Tools/libraries | Installation command |
|---|---|
| Express Js | `npm install express` |
| Mongoose | `npm install mongoose` |
| Bcrypt | `npm install bcrypt` |
| Body-parser | `npm install body-parser` |
| Cors | `npm install cors` |
| Dotenv | `npm install dotenv` |

## Milestone 2: Backend development

- **Set Up Project Structure:**
  - Create a new directory for your project and set up a package.json file using npm initcommand.
  - Install necessary dependencies such as Express.js, Mongoose, and other requiredpackages.

- **Set Up Project Structure:**
  - Create a new directory for your project and set up a package.json file using npm initcommand.
  - Install necessary dependencies such as Express.js, Mongoose, and other requiredpackages.
- **Create Express.js Server:**
  - Set up an Express.js server to handle HTTP requests and serve API endpoints.
  - Configure middleware such as body-parser for parsing request bodies and cors forhandling cross-origin requests.

- **Define API Routes:**
  - Create separate route files for different API functionalities such as authentication, create Books, upload Books, etc.,
  - Implement route handlers using Express.js to handle requests and interact with thedatabase.

- **Implement Data Models:**
  - Define Mongoose schemas for the different data entities.
  - Create corresponding Mongoose models to interact with the MongoDB database.
  - Implement CRUD operations (Create, Read, Update, Delete) for each model toperform database operations.

- **User Authentication:**
  - Implement user authentication using strategies like JSON Web Tokens (JWT) orsession-based authentication.
  - Create routes and middleware for user registration, login, and logout.
  - Set up authentication middleware to protect routes that require user authentication.

- **Handle new chats and posts:**
  - Allow users to chat with other users using the userId.
  - Also the users will make the posts on social timeline.
- **Error Handling:**
  - Implement error handling middleware to catch and handle any errors that occurduring the API requests.
  - Return appropriate error responses with relevant error messages and HTTP statuscodes.

Reference video for backend code:

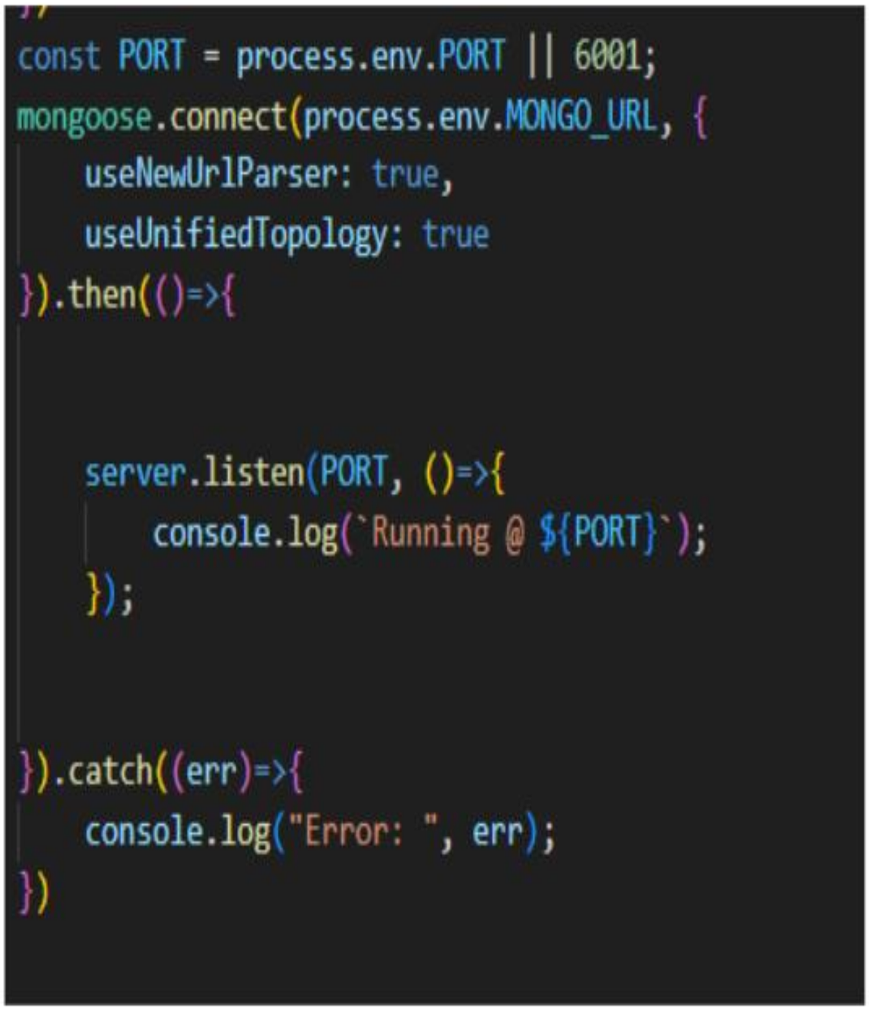
https://drive.google.com/drive/folders/1eiTGiVpYe3nzfsOiWX-hRC0pKniRf24M?usp=drive_link

## Milestone 3: Database development:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.

- Create a database and define the necessary collections for users, posts, stories, chats, etc.,

  The code for the database connection to the server is

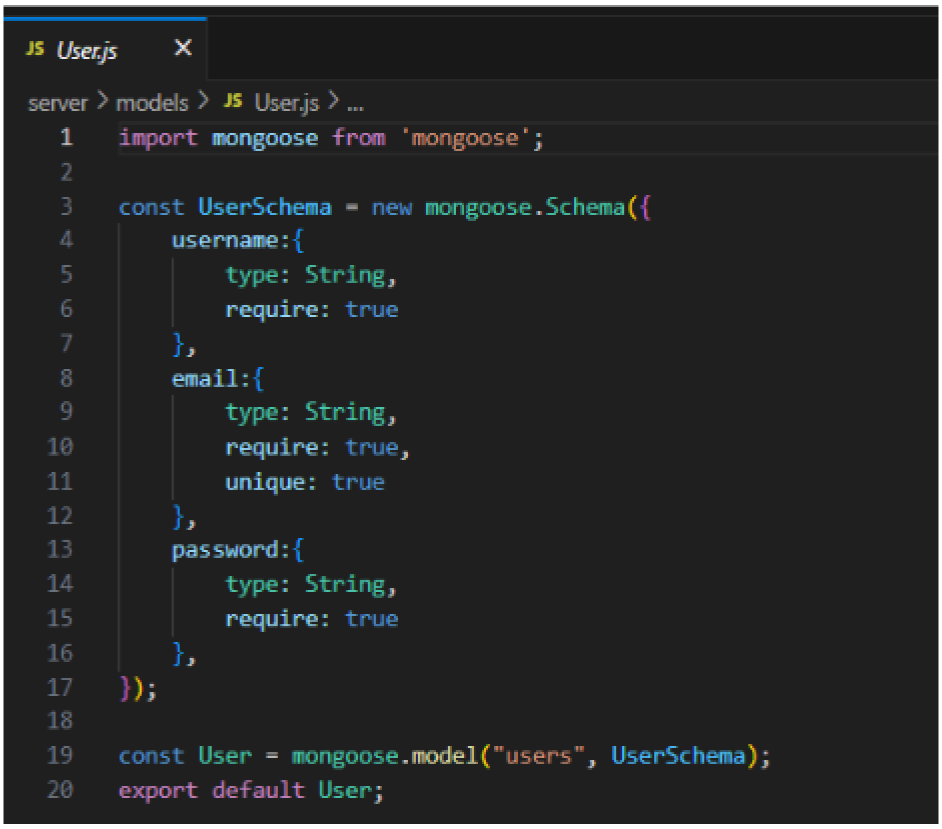


```
const PORT = process.env.PORT || 6001;
mongoose.connect(process.env.MONGO_URL, {
    useNewUrlParser: true,
    useUnifiedTopology: true
}).then(()=>{


    server.listen(PORT, ()=>{
        console.log(`Running @ ${PORT}`);
    });


}).catch((err)=>{
    console.log("Error: ", err);
})
```

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



```
import mongoose from 'mongoose';

const UserSchema = new mongoose.Schema({
    username:{
        type: String,
        require: true
    },
    email:{
        type: String,
        require: true,
        unique: true
    },
    password:{
        type: String,
        require: true
    },
});

const User = mongoose.model("users", UserSchema);
export default User;
```

## Milestone 4: Frontend development & Integration

**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.**

Reference for frontend code:

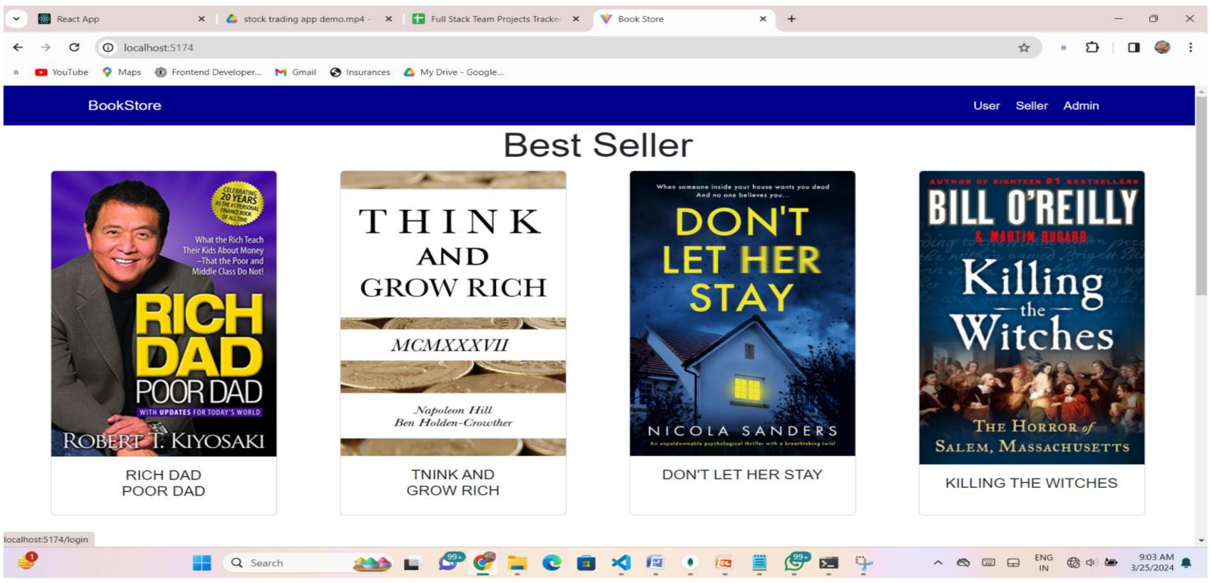
https://drive.google.com/drive/folders/16JkTzPlhBmk3-F-tAYpjKMw_RHpPsZEE?usp=drive_link

## Milestone 5: Project Implementation:

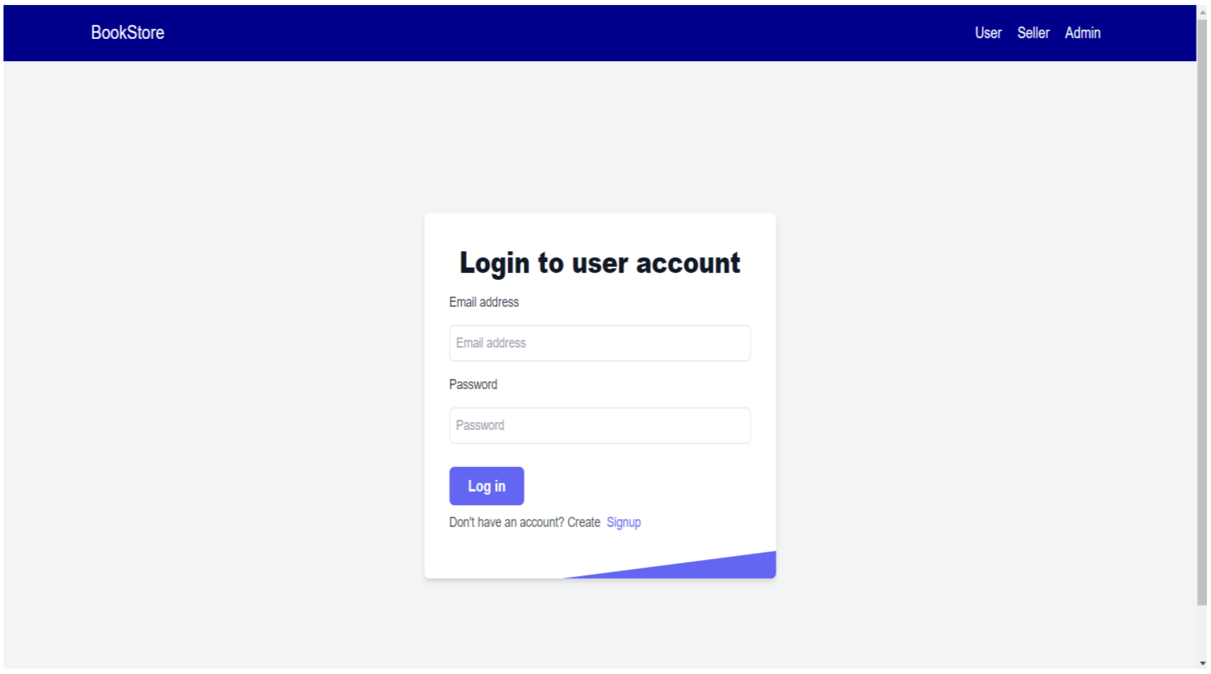
On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the one's provided below.

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:-


BookStore

User Seller Admin

Login to user account

Email address

Email address

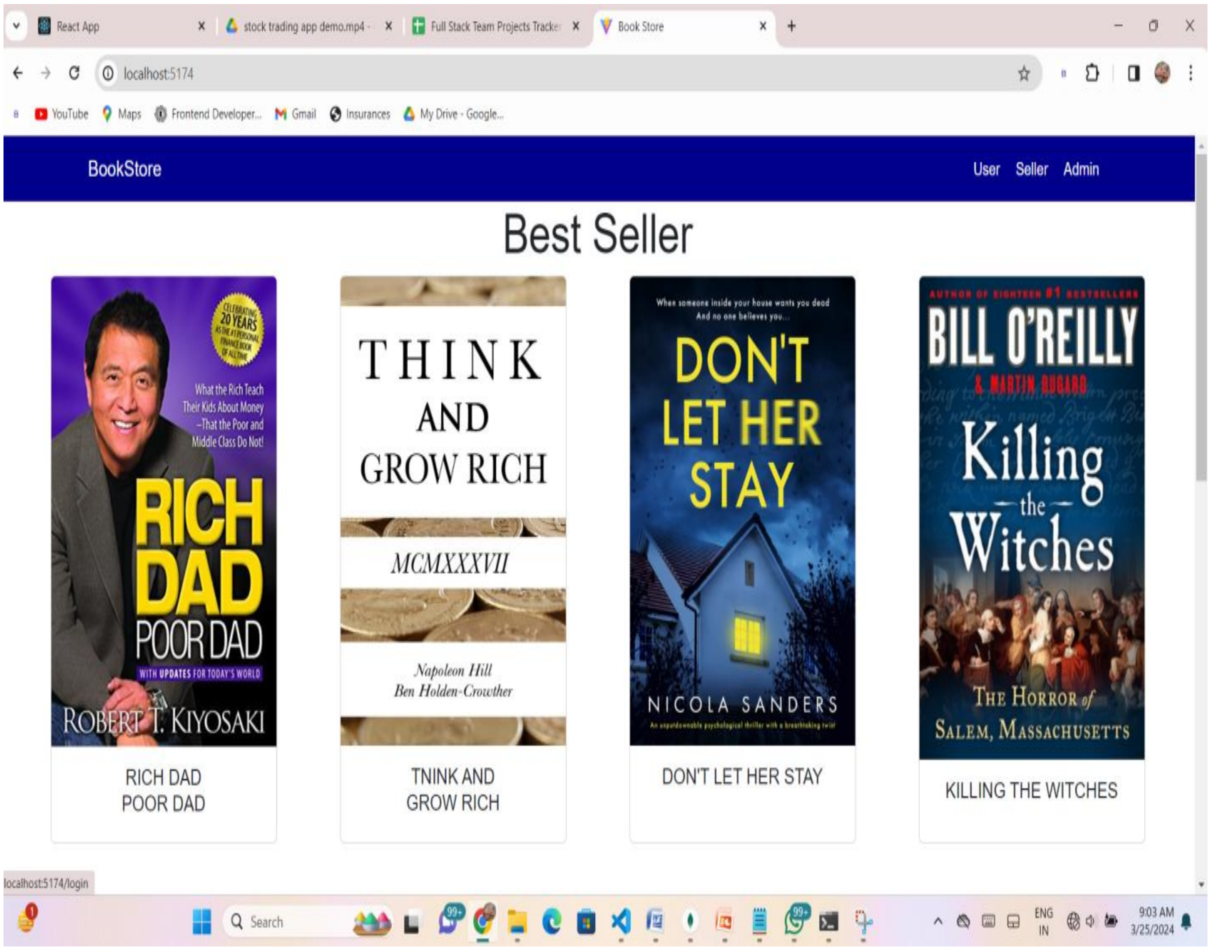
Password

Password

Log in

Don't have an account? Create Signup

Home Page:-


React App | stock trading app demo.mp4 | Full Stack Team Projects Tracker | Book Store

localhost:5174

YouTube Maps Frontend Developer... Gmail Insurances My Drive - Google...

BookStore

User Seller Admin

Best Seller

RICH DAD POOR DAD

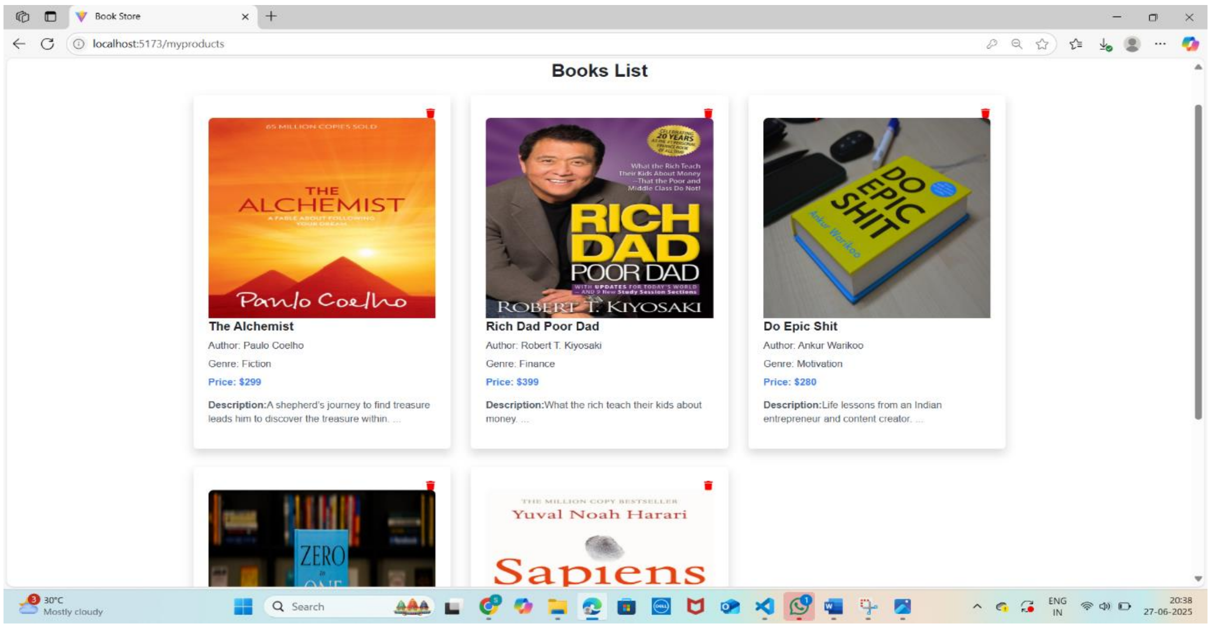
TNINK AND GROW RICH

DON'T LET HER STAY

KILLING THE WITCHES

localhost:5174/login

Books Page:-



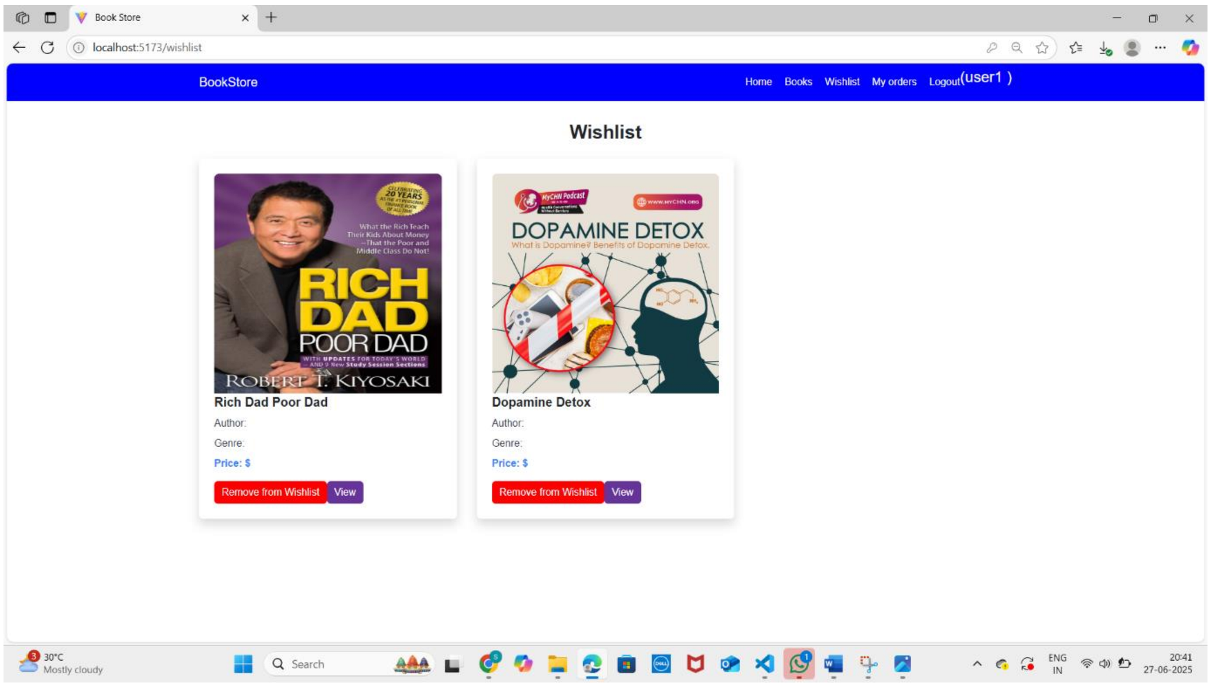
Books List

**The Alchemist**
Author: Paulo Coelho
Genre: Fiction
Price: $299
**Description:**A shepherd's journey to find treasure leads him to discover the treasure within. ...

**Rich Dad Poor Dad**
Author: Robert T. Kiyosaki
Genre: Finance
Price: $399
**Description:**What the rich teach their kids about money ...

**Do Epic Shit**
Author: Ankur Warikoo
Genre: Motivation
Price: $280
**Description:**Life lessons from an Indian entrepreneur and content creator. ...

Wishlist Page:

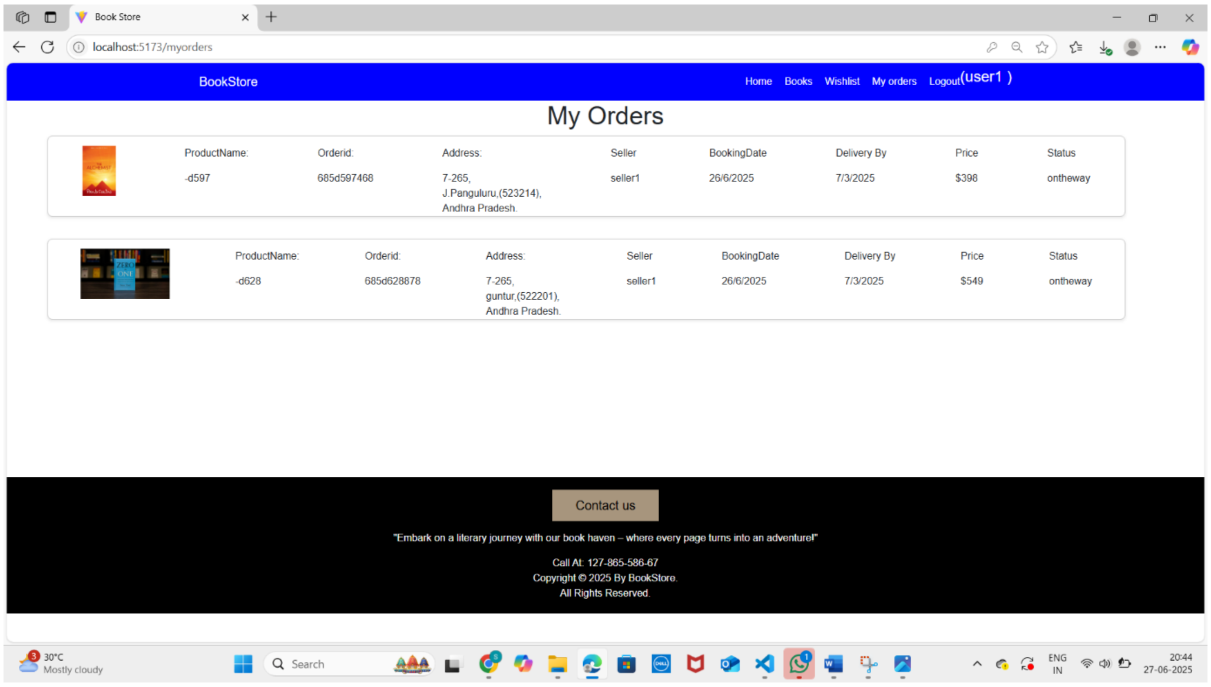


Wishlist

**Rich Dad Poor Dad**
Author:
Genre:
Price: $

**Dopamine Detox**
Author:
Genre:
Price: $

My Bookings Page:



Book Store

localhost:5173/myorders

BookStore — Home Books Wishlist My orders Logout(user1 )

My Orders

| ProductName: | Orderid: | Address: | Seller | BookingDate | Delivery By | Price | Status |
|---|---|---|---|---|---|---|---|
| -d597 | 685d597468 | 7-265, J.Panguluru,(523214), Andhra Pradesh. | seller1 | 26/6/2025 | 7/3/2025 | $398 | ontheway |
| -d628 | 685d628878 | 7-265, guntur,(522201), Andhra Pradesh. | seller1 | 26/6/2025 | 7/3/2025 | $549 | ontheway |

Contact us

"Embark on a literary journey with our book haven – where every page turns into an adventure!"

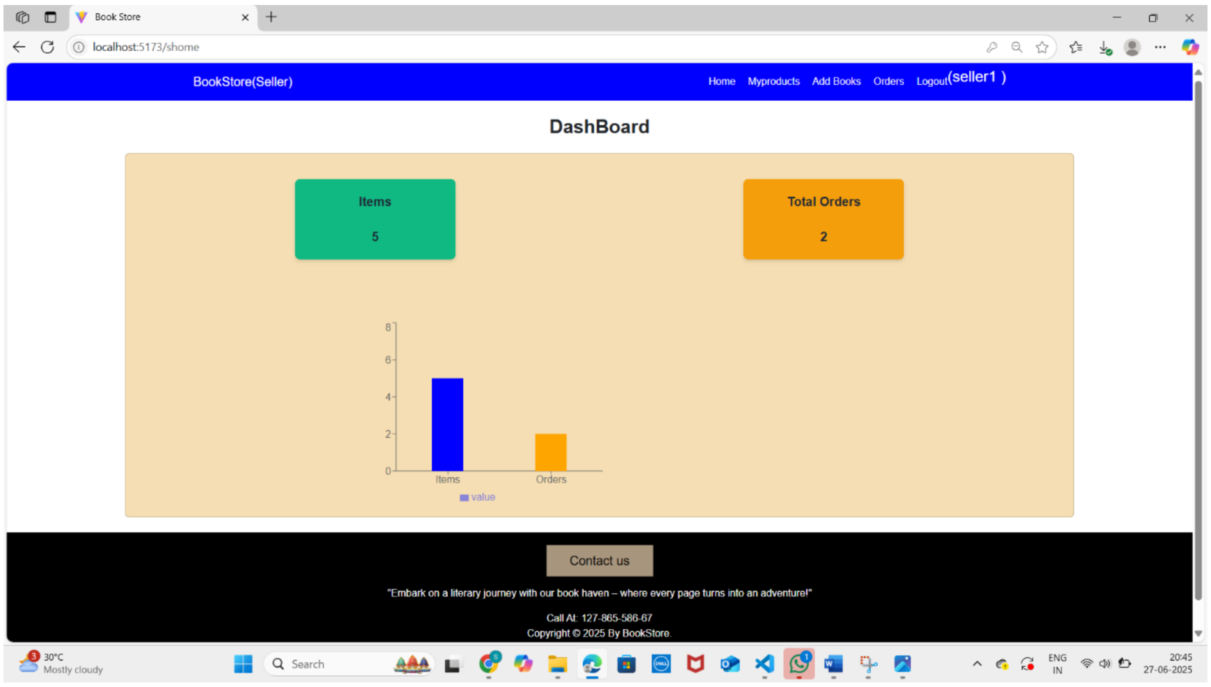
Call At: 127-865-586-67

Copyright © 2025 By BookStore.

All Rights Reserved.

30°C Mostly cloudy — Search — ENG IN — 20:44 27-06-2025

Seller Dashboard:



Book Store

localhost:5173/shome

BookStore(Seller) — Home Myproducts Add Books Orders Logout(seller1 )

DashBoard

Items

5

Total Orders

2

Contact us

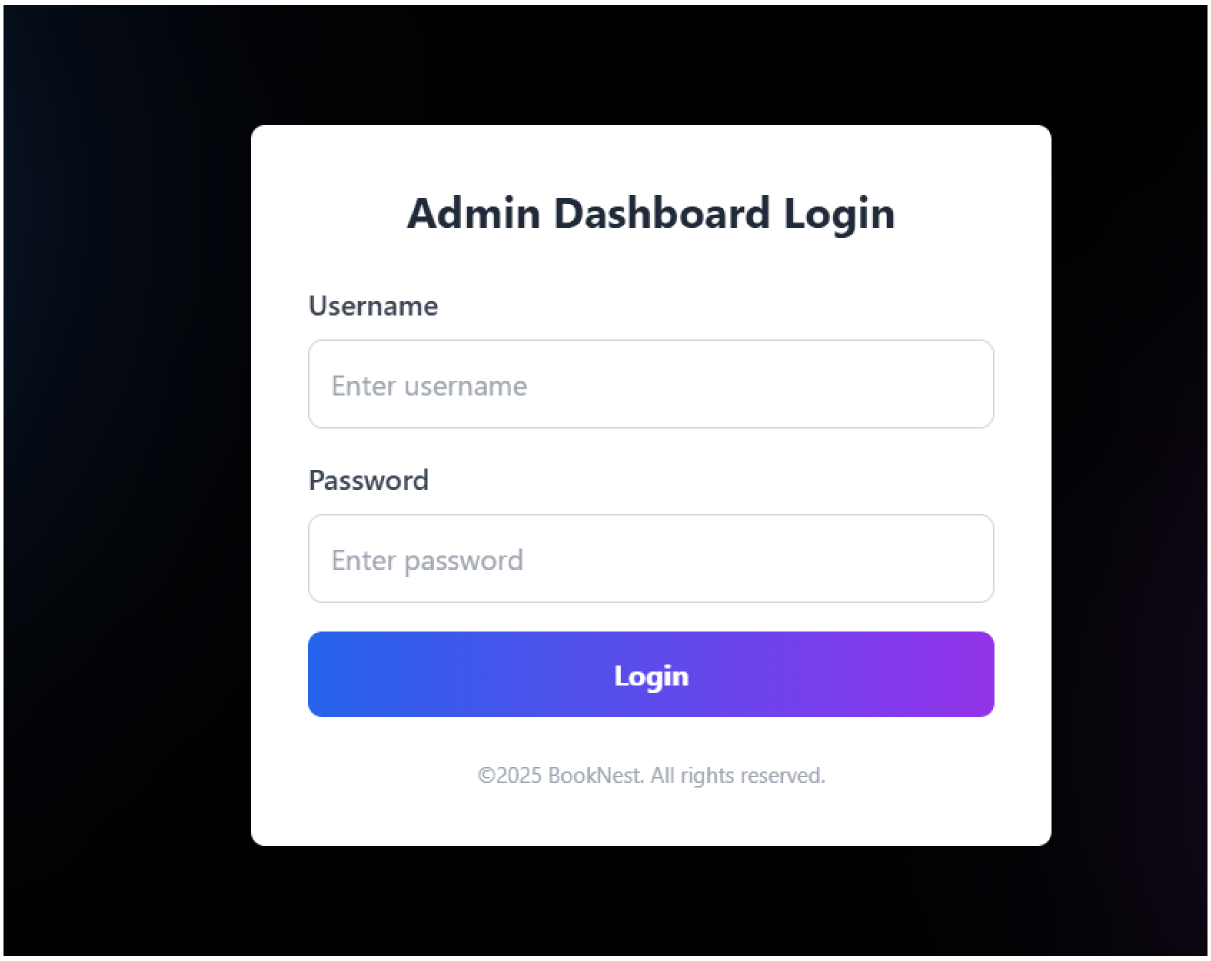
"Embark on a literary journey with our book haven – where every page turns into an adventure!"

Call At: 127-865-586-67

Copyright © 2025 By BookStore.

30°C Mostly cloudy — Search — ENG IN — 20:45 27-06-2025

Admin Login Page:



## Admin Dashboard Login

Username

Enter username

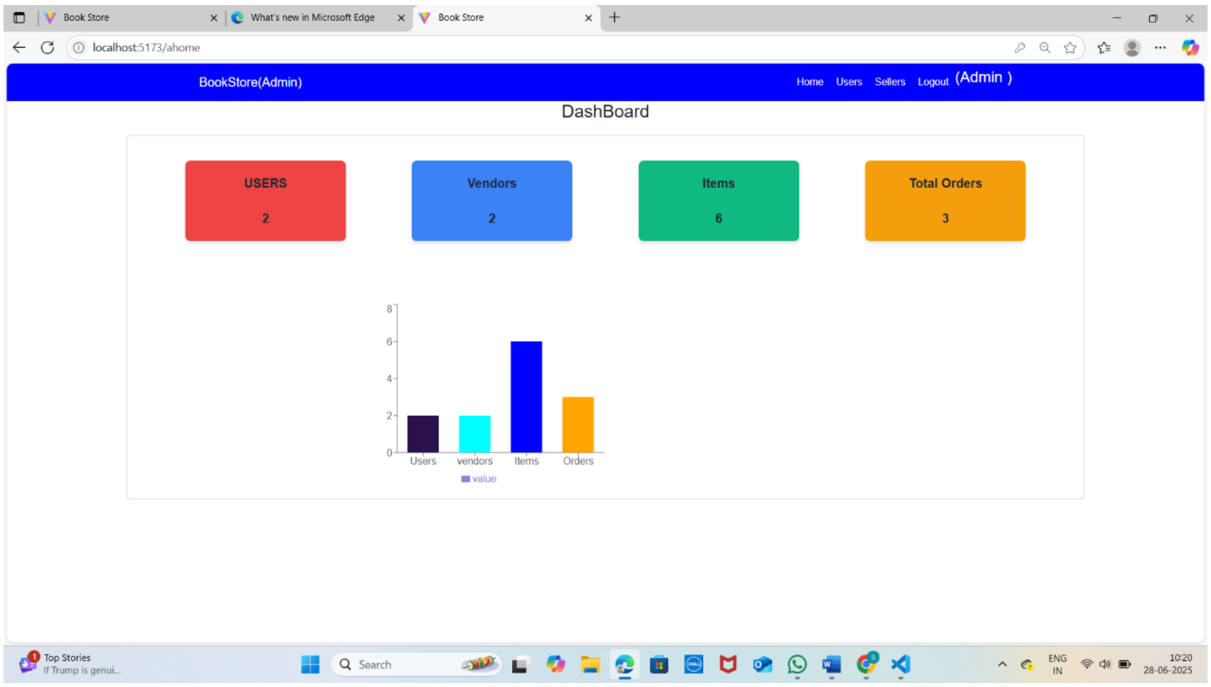
Password

Enter password

Login

©2025 BookNest. All rights reserved.

Admin Dashboard Page:



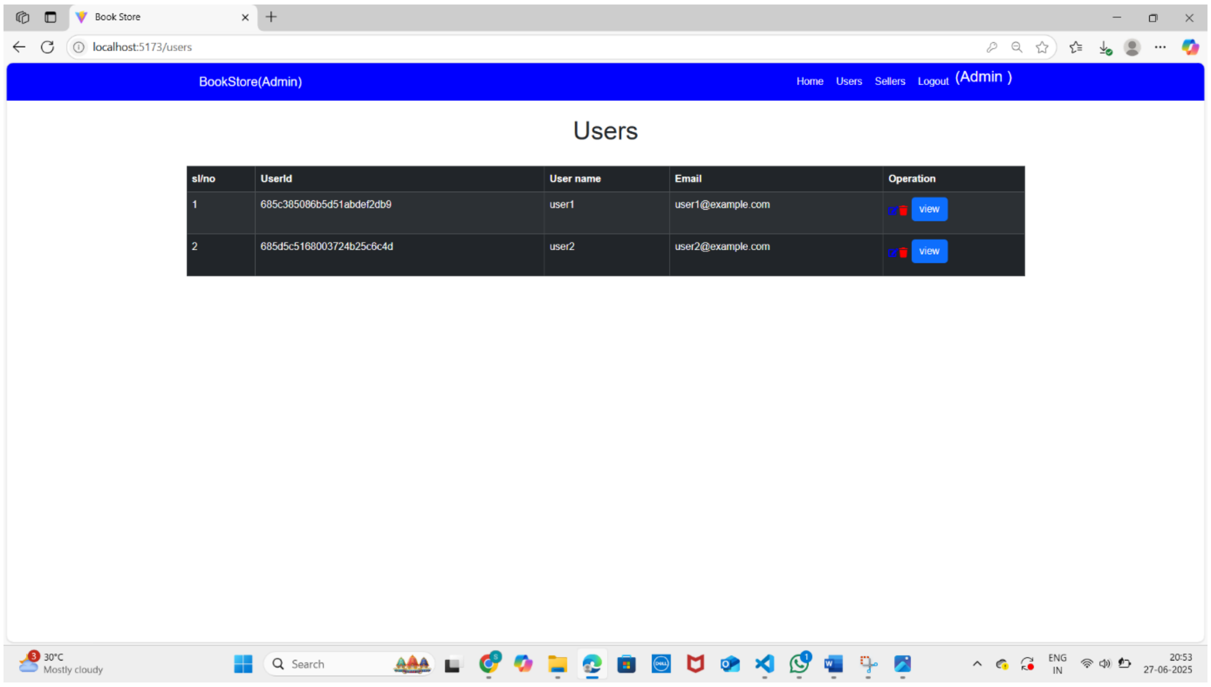
BookStore(Admin)

Home Users Sellers Logout (Admin )

DashBoard

| USERS | Vendors | Items | Total Orders |
|---|---|---|---|
| 2 | 2 | 6 | 3 |

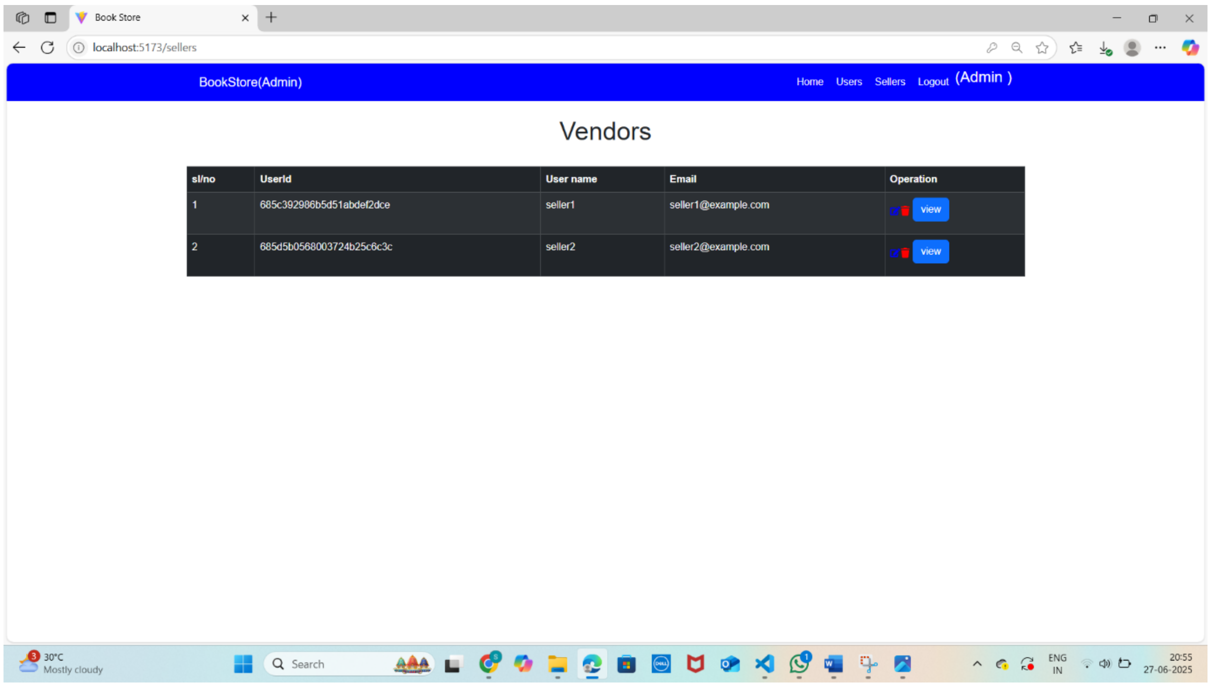
Users Page:



Users

| sl/no | UserId | User name | Email | Operation |
|---|---|---|---|---|
| 1 | 685c385086b5d51abdef2db9 | user1 | user1@example.com | view |
| 2 | 685d5c5168003724b25c6c4d | user2 | user2@example.com | view |

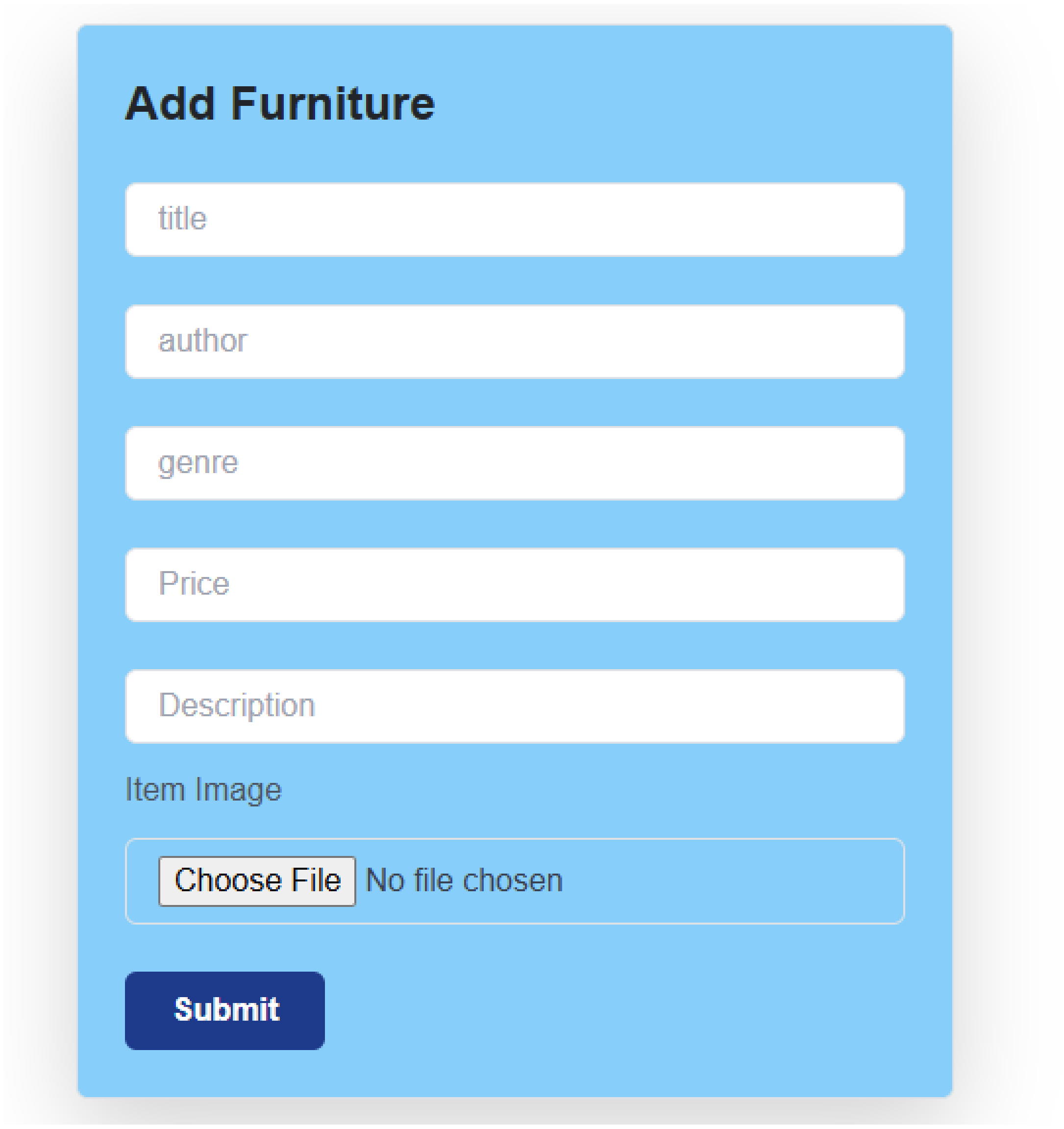
Sellers Page:



Vendors

| sl/no | UserId | User name | Email | Operation |
|---|---|---|---|---|
| 1 | 685c392986b5d51abdef2dce | seller1 | seller1@example.com | view |
| 2 | 685d5b0568003724b25c6c3c | seller2 | seller2@example.com | view |

Add a New Book Page:


Add Furniture

title

author

genre

Price

Description

Item Image

Finally, for any further assistance, use the links below:

Demo link:

https://drive.google.com/file/d/1AlGrV2bCelM_6iKwJ7XK9d4zY1yzIBna/view?usp=drive_link

Use the code in:

https://drive.google.com/drive/folders/1S2kvRDacQl3MT9RhqO2u0bY_tBRxvzzH?usp=drive_link