

BookNest: Where Stories Nestle

TEAMID: LTVIP2025TMID55634

Member1: Katragadda Naga Kavya

Member3: Katta Madhuri

Member2: Koduri Pallavi

Member4: Katta Sirisha

Introduction:

Welcome to the literary haven of the digital era—**BookNest**, an innovative online Book-Store Application crafted with precision using the powerful **MERN Stack** (MongoDB, Express.js, React, Node.js). This application merges the timeless joy of reading with modern web technology, offering a seamless and engaging platform for book lovers to explore, discover, and enjoy literature like never before.

Designed with the modern reader in mind, BookNest combines **robust functionality** with an **intuitive user interface** to deliver a smooth and enjoyable experience. Whether you're uncovering the latest bestsellers or diving back into classic favorites, BookNest provides a personalized and immersive journey tailored to your reading preferences.

Description:

BookNest is a full-stack web application designed to serve as a modern online bookstore, offering users a seamless and engaging platform to explore, search, and enjoy a wide variety of books. Built using the **MERN Stack** (MongoDB, Express.js, React, and Node.js), the application exemplifies the integration of scalable backend services with an intuitive and responsive frontend interface.

This project showcases how powerful web technologies can be leveraged to create a real-world e-commerce experience tailored specifically for book enthusiasts. It supports a range of core features such as browsing books by category, detailed book views, real-time searching, and a user-friendly interface that adapts to all devices.

MongoDB provides a flexible and scalable NoSQL database to manage book data and user information efficiently. **Express.js** and **Node.js** work together to handle server-side operations and API routing, ensuring optimal performance and quick response

times. On the client side, **React** delivers a dynamic and interactive user experience through a component-based architecture and state management.

With a focus on performance, usability, and visual appeal, BookNest not only simplifies the process of discovering books but also enhances the overall digital reading experience. It serves as a complete solution for modern book browsing, demonstrating best practices in full-stack development and responsive web design.

Scenario Based Case Study:

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.

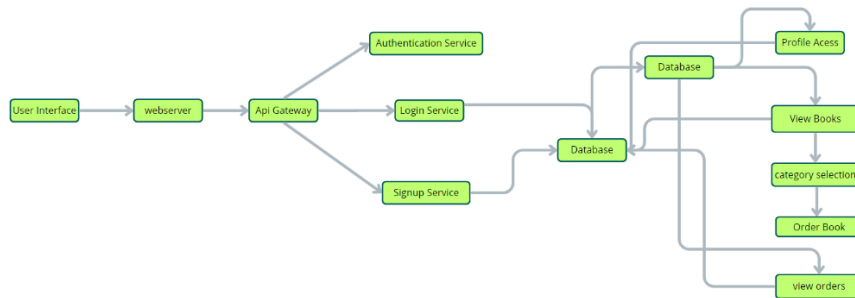
Book Selection: Provide users with options to select their preferred books based on factors like genre, author, ratings, and popularity.

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:-



1. User Interface

Provides a clean, user-friendly platform where users can browse books, search by title or author, view descriptions, and complete purchases. Designed for intuitive navigation and an engaging shopping experience.

2. Web Server

Hosts and delivers dynamic web pages, enabling smooth interaction between users and the application.

3. API Gateway

Acts as a central entry point for client requests, routing them to appropriate backend services. Handles operations like fetching book data, placing orders, and managing user accounts.

4. Authentication Service

Manages secure login, registration, and authorization, ensuring safe access and protection of user data during all interactions.

5. Database

Stores essential information including:

Book details (title, author, genre, price, availability)

User profiles

Purchase history

Inventory data

6. View Books

Allows users to browse all available books with sorting and filtering capabilities.

7. Category Selection

Enables users to filter books by genre or category, improving the ease and accuracy of book discovery.

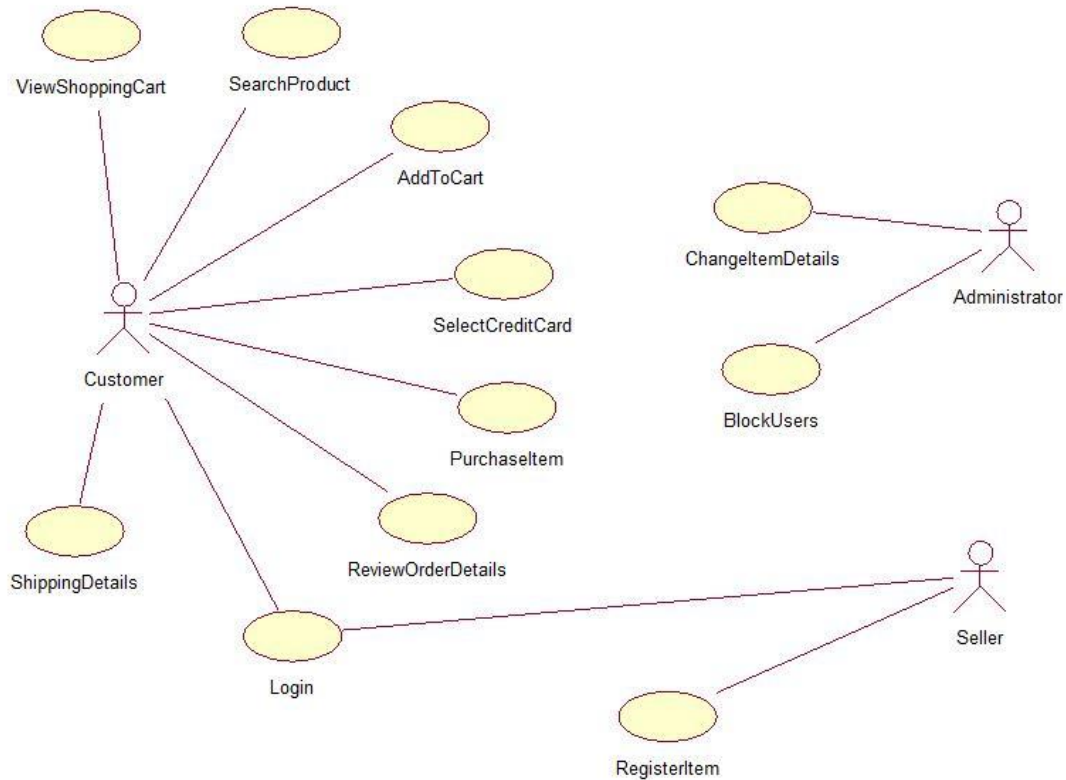
8. Inventory Management Service

Tracks stock levels, availability, and book ratings. Keeps inventory updated in real-time and integrates with the browsing and checkout experience.

9. Order Management Service

Handles cart operations, order placement, and secure payment. Also manages order tracking and status updates for a complete shopping lifecycle.

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-Application

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 advanced 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.

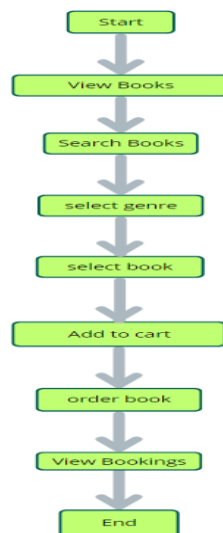
- Inventory Management: Sellers can manage their book inventory, updating stock levels, removing inactive listings, and handling book ratings.
- Order Fulfillment: Sellers are responsible for fulfilling orders placed by users, including packaging and shipping books in a timely manner.
- Logout: Finally, 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.

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 FLOW:-

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.

Reference 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.

Reference Link:-

<https://drive.google.com/file/d/1X84EhZJU-aHbiecO-FDVZ3Fb4gnNW2GT/view?usp=sharing>

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.

```

},
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);
})

```

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.

```

JS User.js  X
server > models > JS User.js > ...
1  import mongoose from 'mongoose';
2
3  const UserSchema = new mongoose.Schema({
4    username:{
5      type: String,
6      require: true
7    },
8    email:{
9      type: String,
10     require: true,
11     unique: true
12   },
13   password:{
14     type: String,
15     require: true
16   },
17 });
18
19 const User = mongoose.model("users", UserSchema);
20 export default User;

```

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:

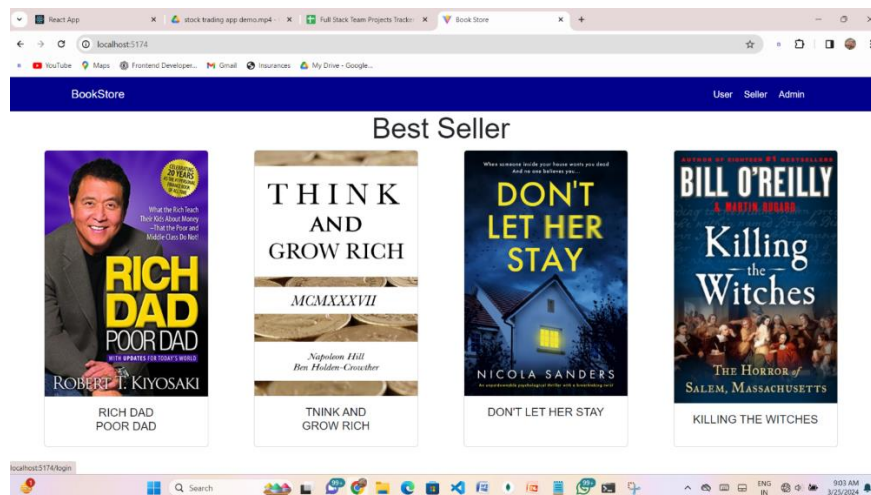
- Integration with API endpoints.
- Implement data binding.

Reference:- https://drive.google.com/file/d/10uZnPzmXBgH-NfS08y2YFvQqrjQd8bN7/view?usp=drive_link Video

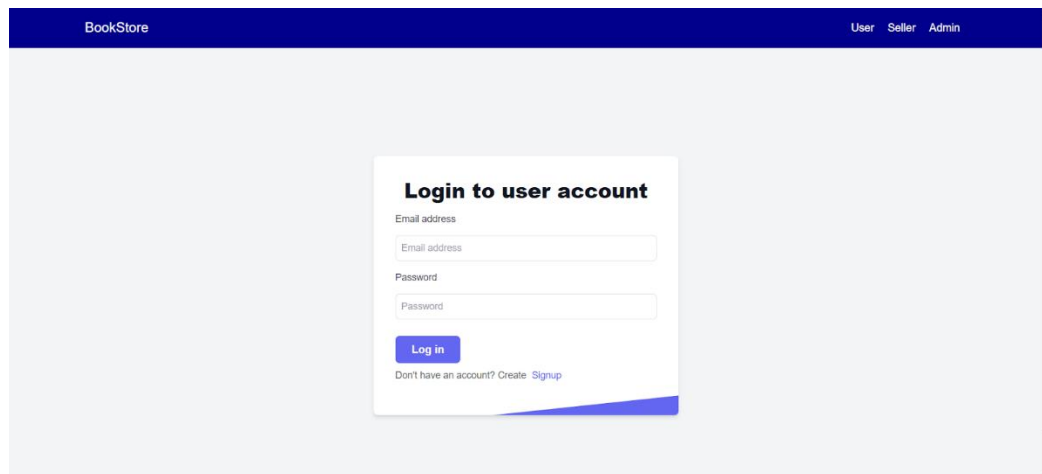
Milestone 5: Project Implementation:

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.

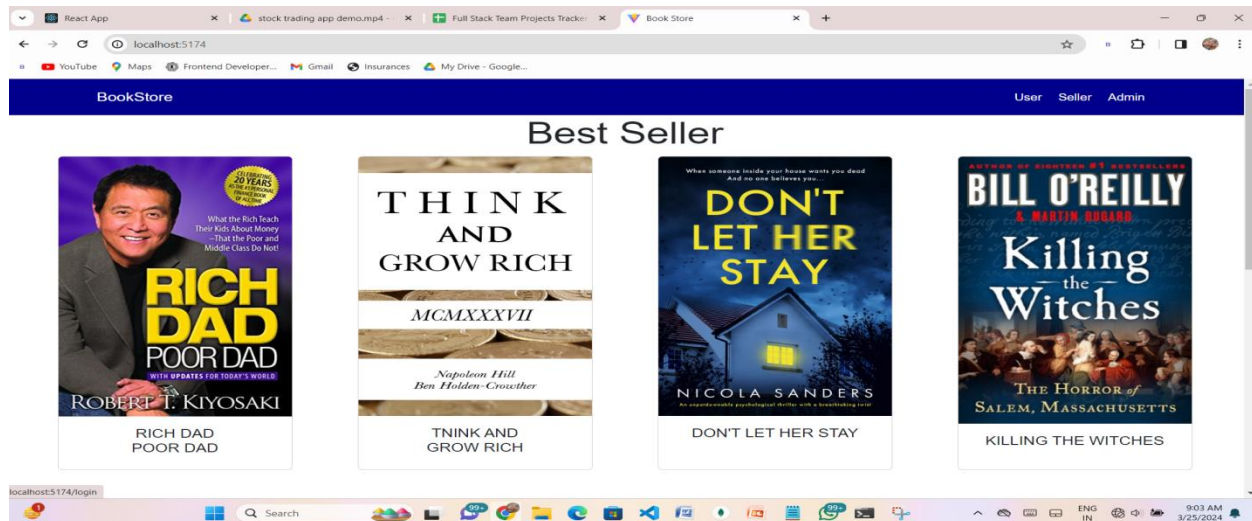
Landing page:-



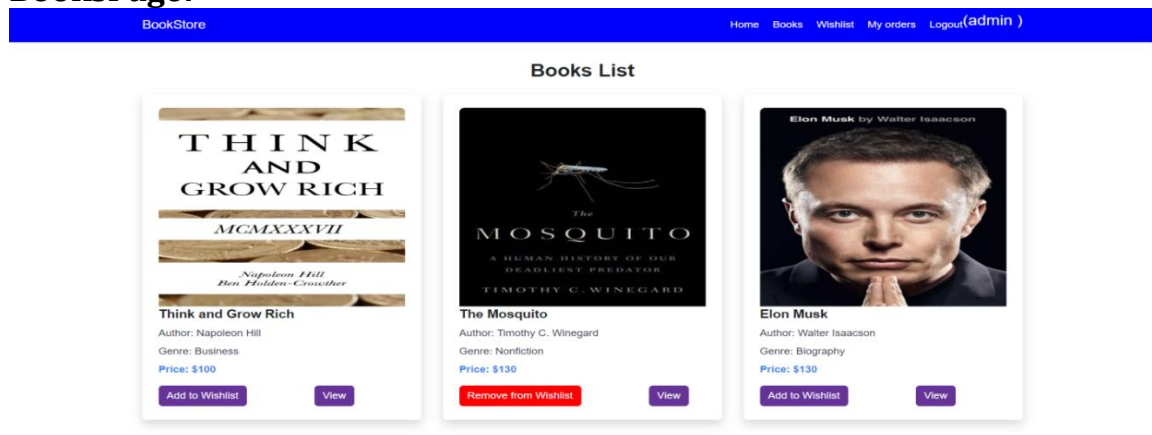
Login Page:-



Home Page:-



BooksPage:-

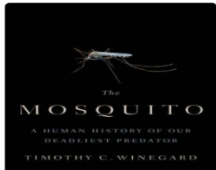


Wishlist Page:-

BookStore

HomeBooksWishlistMy ordersLogout(admin)

Wishlist



The Mosquito
A HUMAN HISTORY OF OUR DEADLIEST PREDATOR
TIMOTHY C. WINEGARD

Author:
Genre:
Price: \$





Remove from WishlistView

My Booking Page:-

BookStore

HomeBooksWishlistMy ordersLogout(syed)

My Orders

	ProductName: -0449	Orderid: 6580449015	Address: dsfkf, asdas,(fasda), asdasda.	Seller syed	BookingDate 18/12/2023	Delivery By 12/25/2023	Price \$199	Status delivered
	ProductName: -0f1d	Orderid: 6600f1d467	Address: 122-8, hyderabad,(517994), Telangana.	Seller syed	BookingDate 25/3/2024	Delivery By 4/1/2024	Price \$229	Status ontheway
	ProductName: -0f25	Orderid: 6600f25067	Address: , .(), .	Seller syed	BookingDate 25/3/2024	Delivery By 4/1/2024	Price \$229	Status ontheway
	ProductName: -0f25	Orderid: 6600f25867	Address: , .(), .	Seller syed	BookingDate 25/3/2024	Delivery By 4/1/2024	Price \$229	Status ontheway

Seller Dashboard:-

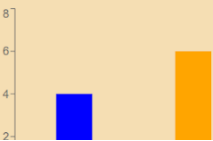
BookStore(Seller)

HomeMyproductsAdd BooksOrdersLogout(syed)

DashBoard

Items
4

Total Orders
6










Seller Items:-

BookStore(Admin)

[Home](#)
[Users](#)
[Settings](#)
[Logout \(syed\)](#)

Vendor Products

	Product Name: -d98a	Orderid: 655cd98a0d4f	Warranty: 1 year	Price: 100	
	Product Name: -d9a1	Orderid: 655cd9a1b4f	Warranty: 1 year	Price: 130	
	Product Name: -d9d0	Orderid: 655cd9d0d4f	Warranty: 1 year	Price: 130	
	Product Name:	Orderid:	Warranty:	Price:	

Admin Dashboard:






Users Page:-

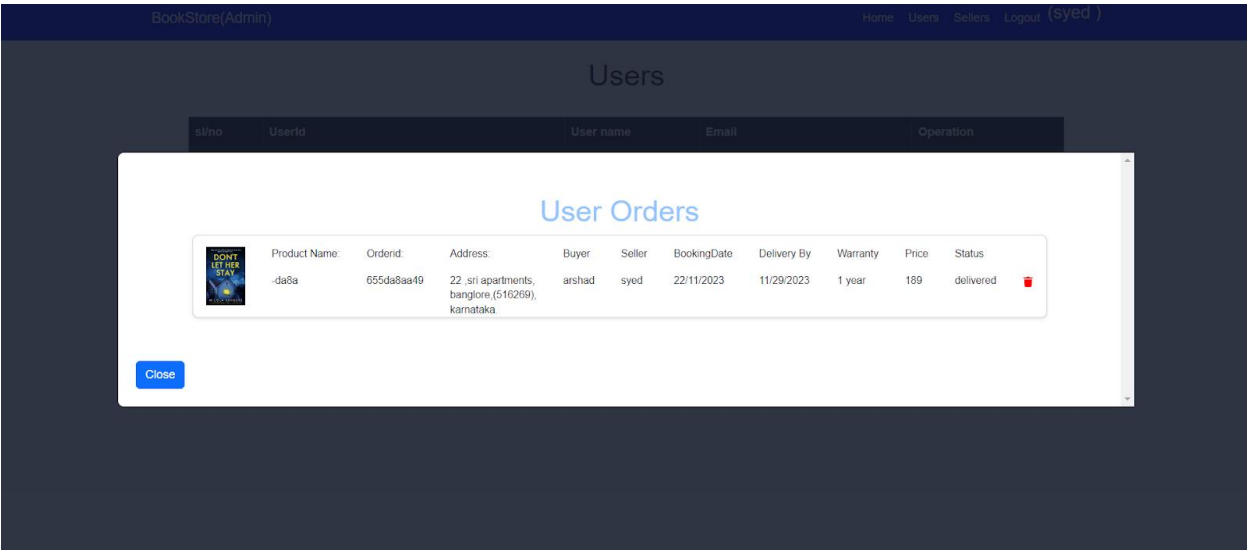
BookStore(Admin)

[Home](#)
[Users](#)
[Settings](#)
[Logout \(syed\)](#)



Users

idno	UserId	User name	Email	Operation
1	655cd9a1b4f0d4f	amrui	amrui@gmail.com	 View
2	655cd78c24f140d9b22f	amrui	amrui@gmail.com	 View
3	655cd9d0f1b2d4f9c4f6	syed	syed@gmail.com	 View

Users Orders:-



Sellers Page:

BookStore(Admin)				
Home Users Sellers Logout (syed)				
Vendors				
s/no	Userid	User name	Email	Operation
1	655da3154982a490b7f6409	arif	arif@gmail.com	 View
2	655c4b32b461e8502aade96	syed	syed@gmail.com	 View

CONCLUSION:

BookNest is a powerful demonstration of how modern web technologies can be used to build a seamless, efficient, and user-friendly online bookstore. By leveraging the MERN stack (MongoDB, Express.js, React, Node.js), the application offers a complete solution that meets the needs of readers, sellers, and administrators alike. From secure user registration and dynamic book listings to smooth purchase flows and detailed admin control, BookNest showcases a well-rounded, scalable, and responsive e-commerce experience tailored for literature lovers. This project not only enhances digital reading accessibility but also reflects best practices in full-stack web development.

The demo of the app is available at:-

<https://drive.google.com/file/d/1y7TPSSUC5krDNSxNI66VvYI4MTxPy8XV/view?usp=sharing>

*** Happy Hacking!! ***