

CERTIFICATE

Project Title: “**BookNest: Where Stories Nestle** “is a bona fide work carried out by the following students:

➤ **TEAM ID:** LTVIP2025TMID60701

- **Pallapothu Hemanth Satya Subramanyam (Team Leader)**
- **Pirla Abhinash (Team Member)**
- **Ramya Sai Tirumalasetty (Team Member)**
- **Rebbavarapu Anusha (Team Member)**

Date Of Submission: 28-06-2025

TABLE OF CONTENTS

1. Introduction
2. Project Overview
 - Purpose
 - Features
3. Architecture
 - Frontend
 - Backend
 - Database
4. Setup Instructions
 - Prerequisites
 - Installation
5. Folder Structure
 - Frontend
 - Backend
6. Running the Application
 - Frontend
 - Backend
7. API Documentation
8. Authentication
9. User Interface
10. Testing
11. Screenshots or Demo
12. Known Issues
13. Future Enhancements

1. INTRODUCTION

BookNest: Where Stories Nestle

BookNest: Where Stories Nestle is a full-stack web application developed using the MERN stack (MongoDB, Express.js, React, Node.js). It is designed to deliver a seamless and interactive online bookstore experience tailored for users, sellers, and administrators. The application provides a responsive interface and real-time communication between the frontend and backend. It integrates key functionalities that replicate real-world e-commerce platforms, specifically for book discovery and purchase.

Key Features:

- Role-based access for Users, Sellers, and Admins
- Secure registration and login with authentication
- Book search, filter, and categorized browsing
- Wishlist and cart management for users
- Order placement and tracking system
- Seller dashboard for managing books and orders
- Admin panel for system-wide monitoring and control
- Responsive design for mobile and desktop use
- RESTful API integration with MongoDB backend
- Book image uploads using Multer

Project Description:

The BookNest application delivers a complete digital bookstore experience with dynamic content and user interaction. Users can register, explore a wide variety of books, and manage their orders efficiently. Sellers are provided with tools to list new books, update inventory, and fulfill orders. Admins have full access to oversee users, books, and platform activity. The project emphasizes clean UI, real-time data updates, and modular code structure, making it suitable for academic submission and real-world deployment.

2.PROJECT OVERVIEW

Purpose

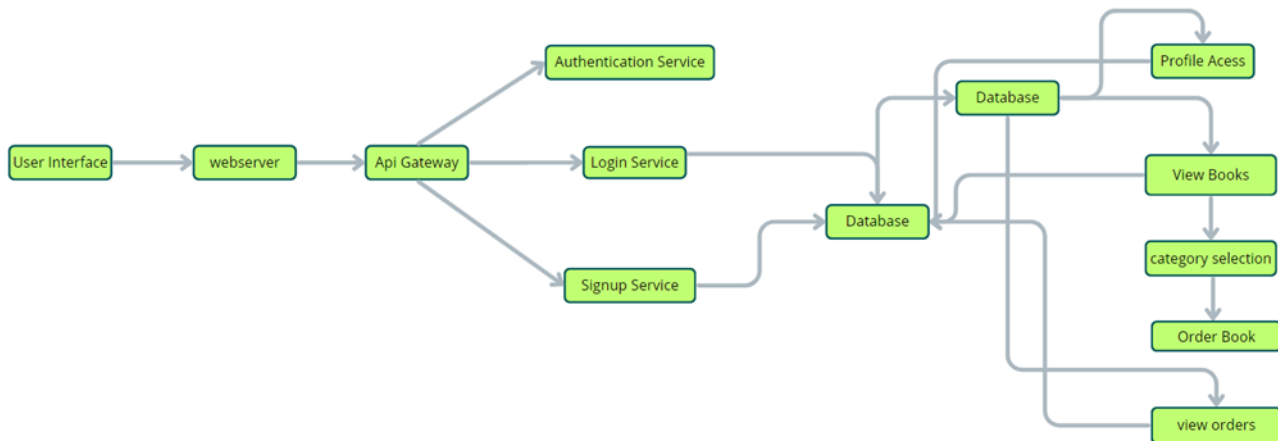
The primary objective of the **BookNest** project is to develop a modern, efficient, and responsive web application that replicates the core functions of an online bookstore. It aims to provide users with an accessible platform to browse and purchase books from various categories, while also enabling sellers to manage their listings and fulfill orders. Administrators are given tools to monitor and maintain the system, ensuring smooth operations. The application supports secure user authentication, dynamic content management, and a clean user interface to enhance the reading and buying experience.

Features

- **Role-based Access:** Supports three user roles—User, Seller, and Admin—each with specific functionalities.
- **User Authentication:** Secure registration and login system with protected routes.
- **Book Browsing:** Allows users to search and filter books by genre, author, and more.
- **Cart and Wishlist:** Users can add books to a cart or wishlist for later reference or purchase.
- **Order Placement and History:** Enables users to place orders and track current or past purchases.
- **Seller Dashboard:** Lets sellers add, update, and manage book listings and view order details.
- **Admin Panel:** Admins can monitor users, manage content, and oversee seller and order activity.
- **API Integration:** Backend APIs handle data operations, ensuring real-time communication.
- **File Upload:** Book images are uploaded through the backend using Multer.
- **Responsive Design:** A mobile-friendly interface developed using React and Bootstrap.

3. ARCHITECTURE

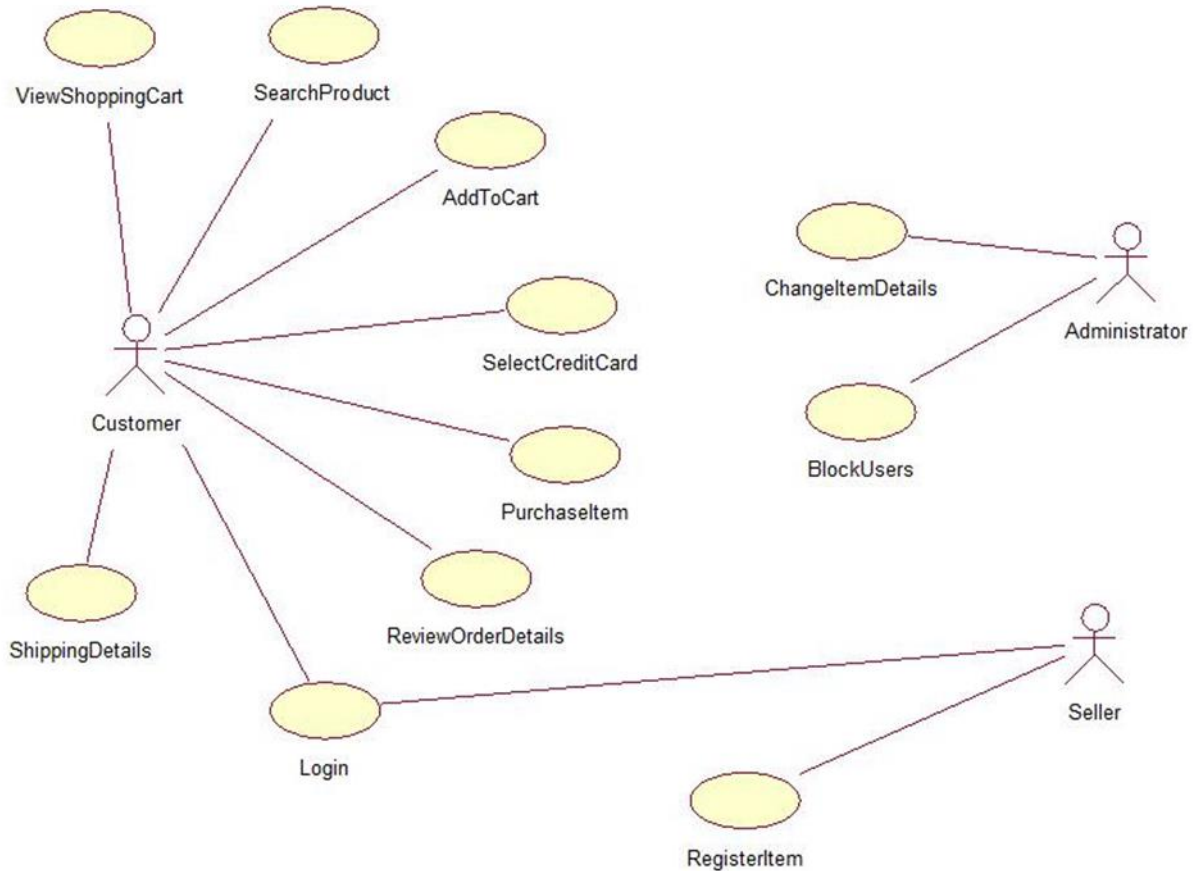
TECHNICAL ARCHITECTURE:



The technical architecture of the **BookNest** application is structured for scalability and modularity, ensuring smooth communication between components. It begins with the User Interface, which acts as the front-facing layer where users interact with the system. All requests from the UI are sent to the Web Server, which then routes them through an API Gateway. This API Gateway functions as a central hub that delegates requests to relevant backend services such as the Authentication Service, Login Service, and Signup Service, depending on the user's actions. These services interact directly with the Database to verify credentials, register new users, and manage session data securely.

Once authenticated, users gain access to application features such as Profile Access, Viewing Books, Selecting Categories, Placing Orders, and Viewing Past Orders. Each of these operations involves backend data retrieval or modification, all managed through consistent database interactions. The use of a centralized database and modular services ensures data consistency, performance, and ease of maintenance, providing a seamless experience for users across all devices.

ER DIAGRAM:



The ER diagram of the BookNest application illustrates the interaction between three primary entities:

Customer, **Seller**, and **Administrator**. Each entity is associated with specific operations reflecting their role in the system. The **Customer** entity interacts with functions such as Login, SearchProduct, AddToCart, ViewShoppingCart, SelectCreditCard, PurchaseItem, ReviewOrderDetails, and ShippingDetails, enabling a complete shopping and order management experience.

The **Seller** manages product-related operations like RegisterItem, while the **Administrator** handles platform-level controls including ChangeItemDetails and BlockUsers. These relationships define the structure for access control and data management in the application. The ER model promotes a clear, role-based workflow, ensuring that each user type has appropriate access and functionality within the BookNest system.

4. SET INSTRUCTIONS

PRE-REQUISITES/INSTALLATION:

1. Node.js and npm

Purpose:

Node.js is used to run JavaScript on the server-side, while npm (Node Package Manager) is essential for installing project dependencies.

Installation Steps:

- Download the LTS version of Node.js which includes npm.
- After installation, confirm the setup by running `node -v` and `npm -v` in the terminal.

Official Link:

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

2. MongoDB

Purpose:

MongoDB is a NoSQL database used to store application data such as user profiles, book information, orders, and more.

Installation Options:

- **Option 1:** Install MongoDB Community Edition locally.
- **Option 2:** Use **MongoDB Atlas** (a free cloud-hosted version).

Official Links:

- Local Download: <https://www.mongodb.com/try/download/community>
- Atlas Cloud: <https://www.mongodb.com/cloud/atlas/register>

Installation Guide:

<https://www.mongodb.com/docs/manual/installation/>

3. Express.js

Purpose:

Express is a lightweight framework used to build backend APIs with Node.js.

Installation:

Inside your project's backend folder, run:

```
bash
```

```
CopyEdit
```

```
npm install express
```

Documentation:

<https://expressjs.com/>

4. React**Purpose:**

React is used to build the frontend of the application, enabling dynamic and responsive user interfaces.

Installation using Vite (recommended for faster setup):

bash

CopyEdit

```
npm create vite@latest
```

```
cd project-name
```

```
npm install
```

```
npm run dev
```

Documentation:

<https://react.dev/>

5. Mongoose**Purpose:**

Mongoose is an ODM (Object Data Modeling) library that simplifies interactions between Node.js and MongoDB.

Installation:

Bash

CopyEdit

```
npm install mongoose
```

Documentation:

<https://mongoosejs.com/>

6. Postman or Thunder Client**Purpose:**

Used for testing REST APIs during backend development.

Download:

- **Postman:** <https://www.postman.com/downloads/>
 - **Thunder Client (VS Code Extension):** Install directly from the VS Code Extensions Marketplace.
-

7. Git and GitHub**Purpose:**

Git is used for version control and GitHub is used to host your repository and collaborate with teammates.

Download Git:

<https://git-scm.com/downloads>

Create GitHub Account:

<https://github.com/join>

8. Code Editor

Recommended: Visual Studio Code (VS Code)

Download:

<https://code.visualstudio.com/download>

After installation, install helpful extensions like ESLint, Prettier, and Thunder Client.

9. Additional Frontend Libraries

Install these from your React app directory:

bash

CopyEdit

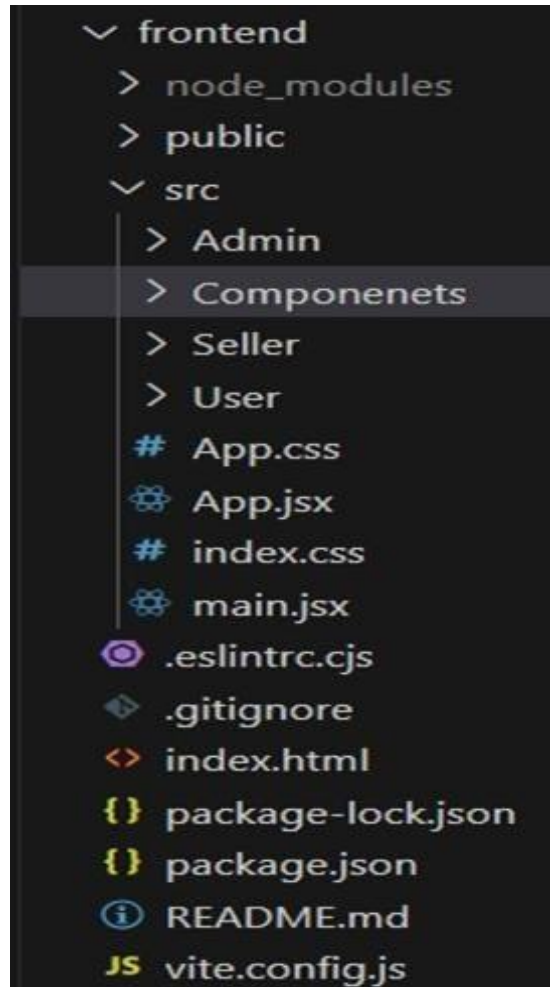
npm install axios react-router-dom bootstrap react-bootstrap

These tools form the core environment needed to build and run your MERN stack bookstore application efficiently. Proper installation and configuration will ensure a smooth development workflow from start to finish.

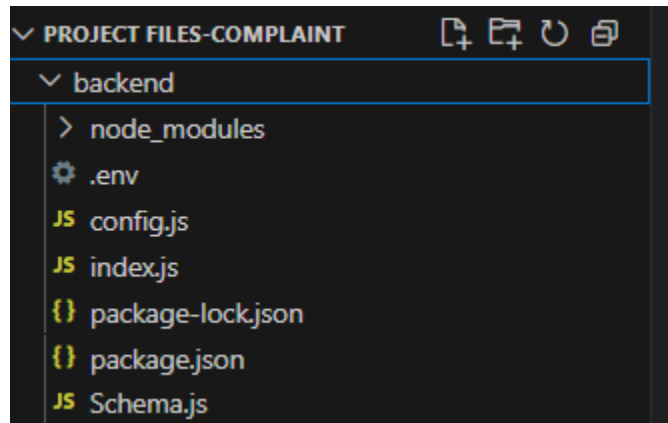
5.PROJECT STRUCTURE

Create a Project folder that contains files as shown below:

1. Frontend:



2. Backend:



- The first image is of frontend part which is showing all the files and folders that have been used in UI development
- The second image is of Backend part which is showing all the files and folders that have been used in backend development

6. Running the Application:

A. FRONTEND:

➤ To run the **React frontend**-

- ✓ Open terminal and navigate to the frontend folder:

```
cd frontend
```

- ✓ Install dependencies:

```
npm install
```

- ✓ Start the frontend:

```
npm start
```

- ✓ Open browser and visit:

<http://localhost:5173>

B. BACKEND:

➤ To run the **Node.js + Express backend**:

- ✓ Open another terminal and navigate to the backend folder:

```
cd backend
```

- ✓ Install dependencies:

```
npm install
```

- ✓ Start the backend server:

```
npm start
```

- ✓ Server runs at:

<http://localhost:5173>

7.API DOCUMENTATION

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

.

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>

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;
```

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](#)

6. AUTHENTICATION

User Authentication

User authentication is a critical component of the BookNest application. It ensures secure access control, protecting sensitive user data and allowing only authorized users to perform specific actions. The authentication system includes user registration, login, and access to protected routes based on roles such as User, Seller, and Admin.

Purpose

- Ensures that only verified users can access the system.
 - Maintains data privacy and security across user roles (User, Seller, Admin).
 - Provides role-based access control to restrict unauthorized actions.
-

Technologies Used

- **MongoDB & Mongoose** – To store and manage user credentials securely.
 - **bcryptjs** – For hashing passwords before storing them in the database.
 - **jsonwebtoken (JWT)** – For generating and verifying secure user tokens.
 - **Express.js** – To handle HTTP requests and define authentication routes.
-

Authentication Workflow

1. Registration Process

- Users provide username, email, and password.
- The system checks if the email is already registered.
- Password is hashed before storing in the database.
- New user is saved with a default role (e.g., User or Seller).
- A success response is returned upon completion.

2. Login Process

- User submits email and password.
 - System validates if the email exists.
 - Password is compared with the hashed one stored in the database.
 - If valid, a JWT token is generated and sent to the frontend.
 - The token is stored locally (e.g., in localStorage) for future requests.
-

Token-Based Authentication

- JWT token contains user ID and role as payload.
 - The token is sent with each request to access protected routes.
 - Backend verifies the token before allowing access.
-

Protected Routes

- Some routes (e.g., order history, seller dashboard, admin panel) are protected.
 - Middleware checks for a valid token before granting access.
 - Unauthorized users are blocked from accessing these routes.
-

Logout Mechanism

- On logout, the token is cleared from storage on the frontend.
 - No backend token invalidation is needed unless implementing refresh tokens.
-

Role-Based Access

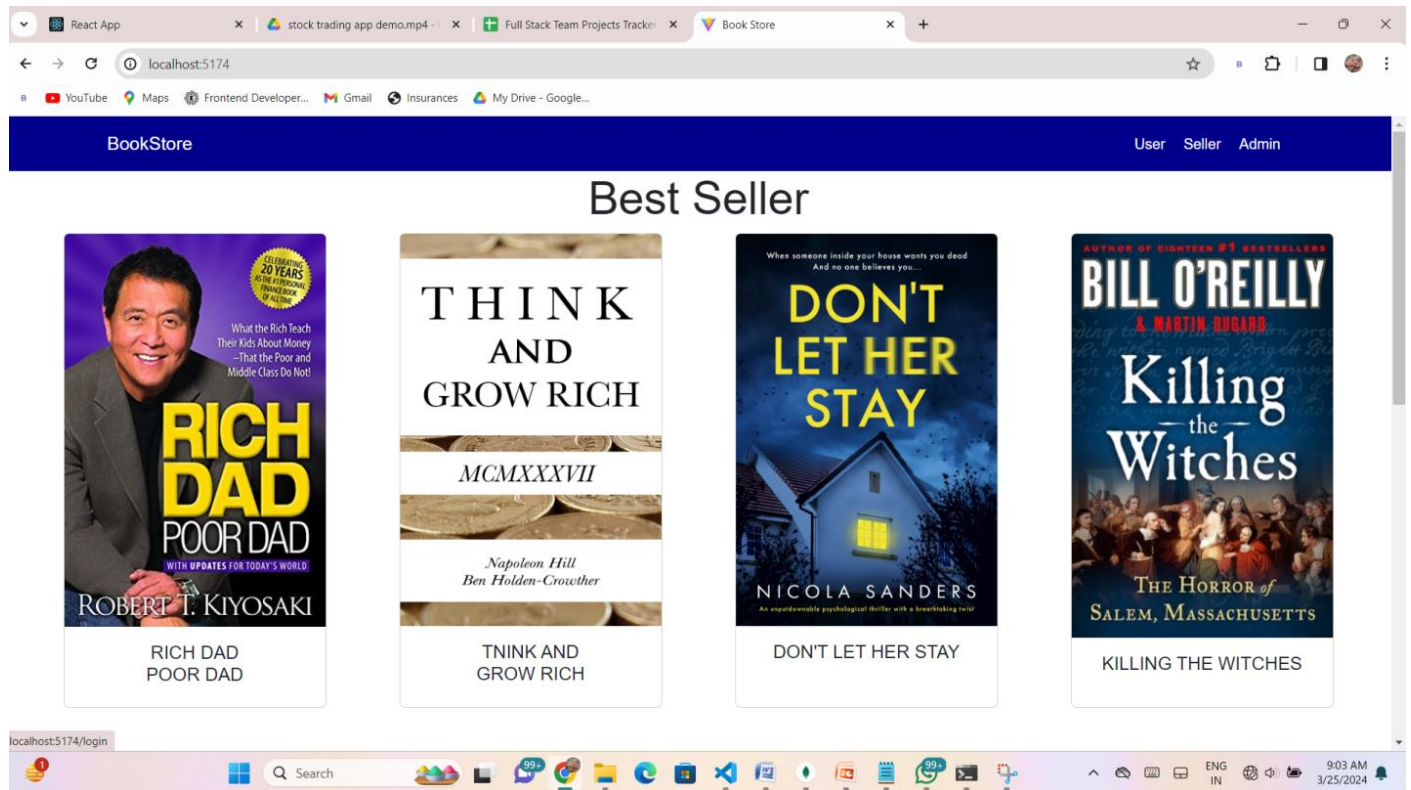
- The token stores the user's role (User, Seller, Admin).
- Role-based middleware ensures only authorized roles can access specific routes.
- Example: Only Admins can view and manage all users or sellers.

7. USER INTERFACE

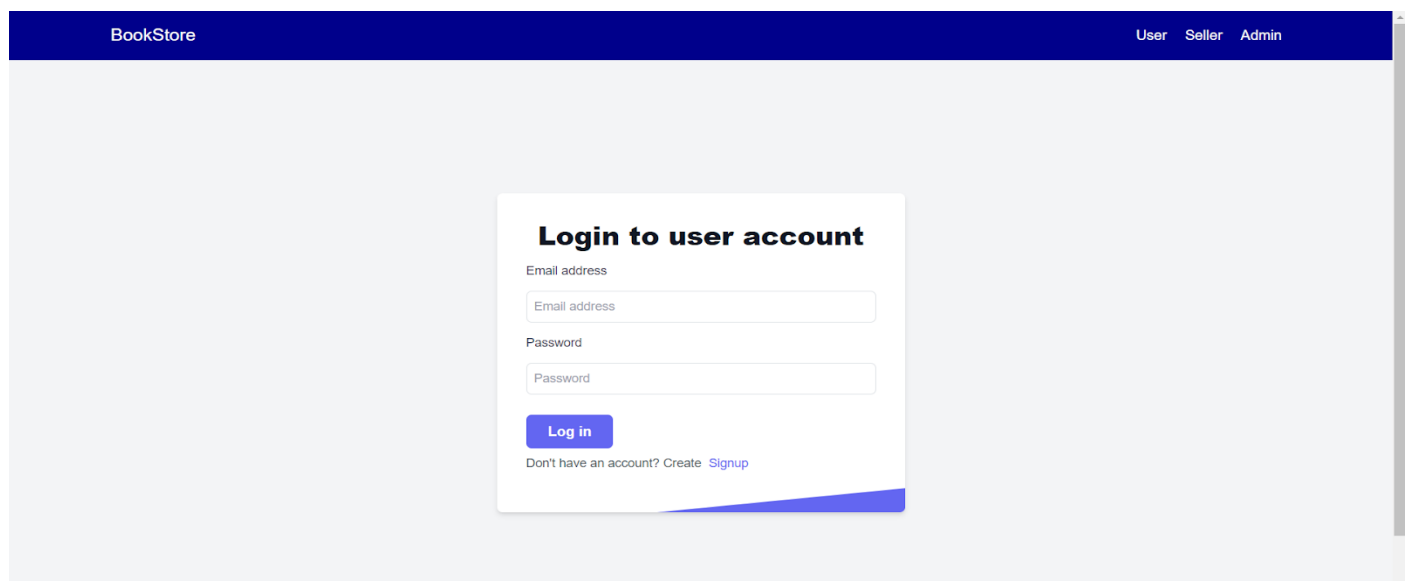
Project Implementation:

Finally, after finishing coding the projects we run the whole project to test its working process and look for bugs. Now, let's have a final look at the working of our Cab Booking application.

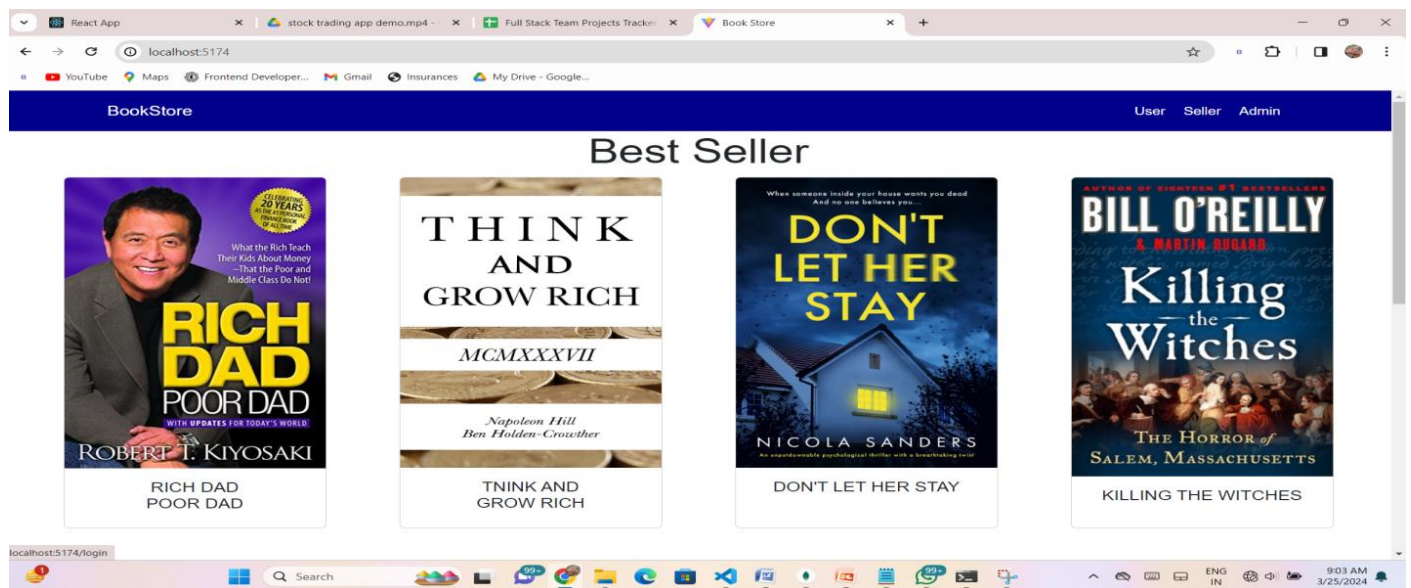
Landing page: -



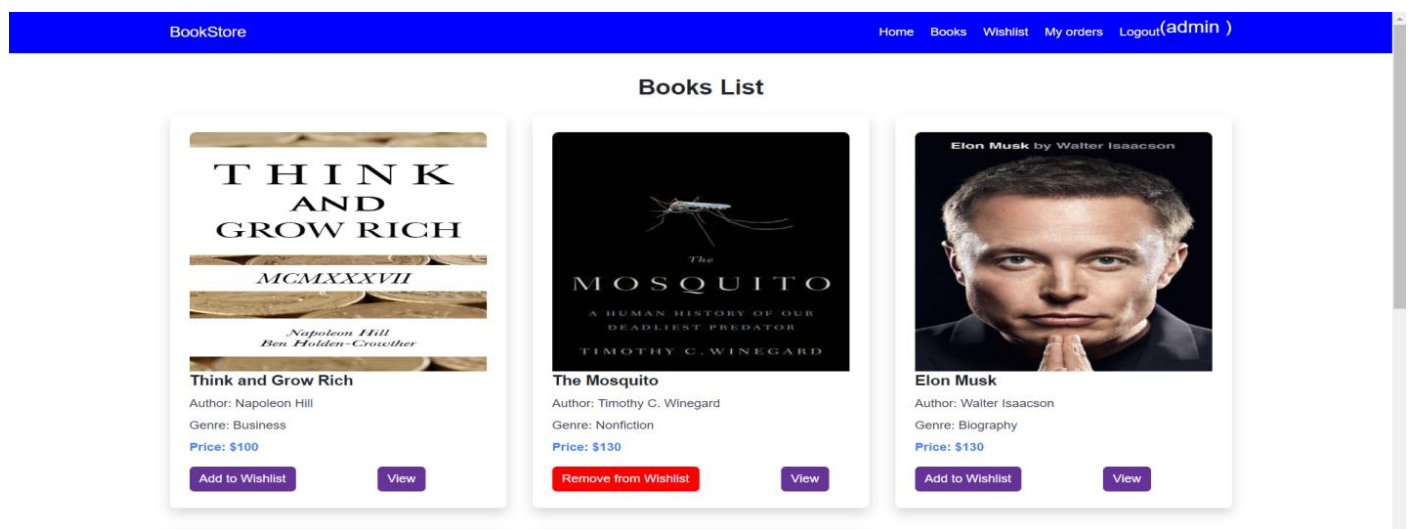
Login Page: -



Home Page: -



Books Page: -



Wishlist Page: -







My Bookings Page: -

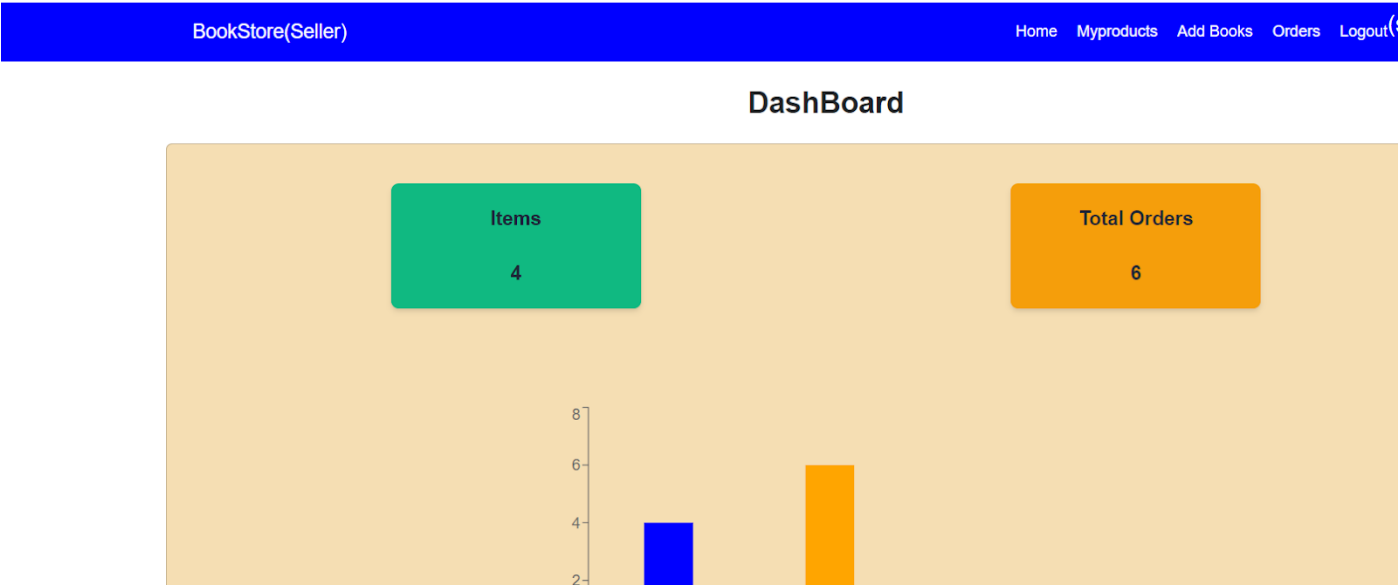
BookStore

HomeBooksWishlistMy ordersLogout(syed)

My Orders

	ProductName: -0449	Orderid: 6580449015	Address: dsfksf, 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: , ,0, .	Seller: syed	BookingDate: 25/3/2024	Delivery By: 4/1/2024	Price: \$229	Status: ontheway
	ProductName: -0f25	Orderid: 6600f25f67	Address: , ,0, .	Seller: syed	BookingDate: 25/3/2024	Delivery By: 4/1/2024	Price: \$229	Status: ontheway

Seller Dashboard: -










Seller Items: -

BookStore(Admin)

HomeUsersSellersLogout (syed)

Vendor Products

	Product Name: -d98a	Orderid: 655d98a04f	Warranty: 1 year	Price: 100	
	Product Name: -d9a1	Orderid: 655d9a184f	Warranty: 1 year	Price: 130	
	Product Name: -d9d0	Orderid: 655d9d0d4f	Warranty: 1 year	Price: 130	
	Product Name:	Orderid:	Warranty:	Price:	

Admin Dashboard: -

BookStore(Admin)

HomeUsersSellersLogout (syed)

DashBoard

USERS

3

Vendors

2

Items

5

Total Orders

6

Users

vendors

Items

Orders

8

6

4

2

0

value

localhost:5174/ahome

Users Page: -

BookStore(Admin)

Home

Users

Sellers

Logout (syed)

Users

sl/no	UserId	User name	Email	Operation
1	655d9f4c4f4ace5f198b9abf	arshad	arshad@gmail.com	<div><div></div><div></div><div>view</div></div>
2	655e571f62e8144c8a9cb27f	shivani	shivani@gmail.com	<div><div></div><div></div><div>view</div></div>
3	6580442915b2ba8ff503a659	syed	syed@gmail.com	<div><div></div><div></div><div>view</div></div>

User Orders:

BookStore(Admin)

HomeUsersSellersLogout (syed)

Users


slno

Userid

User name

Email

Operation



Product Name

Orderid

Address

Buyer

Seller

BookingDate

Delivery By

Warranty

Price

Status

-da8a

655da8aa40

22_sri apartments,
bangalore,(516209),
karnataka

arshad


syed

22/11/2023

11/29/2023

1 year

180

delivered 

Close

Seller's page:

BookStore(Admin)

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

Vendors

sl/no	Userid	User name	Email	Operation
1	655da3154992a4960bff6489	elf	elf@gmail.com	<input checked="" type="checkbox"/> <input type="checkbox"/> view
2	655c4b32b451e85b2daade96	syed	syed@gmail.com	<input checked="" type="checkbox"/> <input type="checkbox"/> view

UI Overview and Key Modules

The user interface (UI) of *BookNest: Where Stories Nestle* is designed to deliver an intuitive and responsive bookstore experience for three distinct roles: **User**, **Seller**, and **Admin**. Each user type is presented with a role-specific dashboard that streamlines access to the features they need most.

Objectives

- To provide a user-friendly, responsive interface across devices.
 - To ensure clear navigation and smooth access to role-based features.
 - To separate functionalities based on roles for security and clarity.
-

Role-Based Dashboards

- **User Dashboard:** Allows users to view books, manage wishlists, place orders, and track purchase history.
 - **Seller Dashboard:** Enables sellers to add, update, or delete books, manage inventory, and process customer orders.
 - **Admin Dashboard:** Grants admins full control to manage users, sellers, books, and monitor platform.
-

Key Components

- Navigation Bar for seamless page transitions.
 - Book listings with filters by category, author, or price.
 - Forms for registration, login, and order placement.
 - Interactive dashboards with analytics and controls.
-

Technologies Used

- **React.js** (Frontend)
 - **Bootstrap** (UI Styling)
 - **Node.js & Express.js** (Backend)
 - **MongoDB & Mongoose** (Database)
 - **JWT** for authentication
 - **Axios** for API requests
-

User Experience Goals

- Mobile-friendly design with responsive layouts.
- Clear CTAs (Call to Actions) and feedback messages.
- Fast load times and minimal navigation steps.

8. TESTING

1. Manual Testing

- All user roles (User, Seller, Admin) were tested by manually performing tasks such as registration, login, adding books, placing orders, and viewing dashboards.
- Edge cases like invalid logins, missing form fields, and incorrect data types were tested to validate input handling.

2. API Testing (Postman/Thunder Client)

- Backend API endpoints were tested for expected response codes (200, 401, 404) and data structures.
- Authenticated routes (e.g., /orders, /books/add) were tested using JWT tokens to ensure access control.

3. UI Testing

- Frontend was tested across different browsers to ensure responsiveness and layout consistency.
- Navigation flow and interactive elements (buttons, links, dropdowns) were tested for click accuracy and state behavior.

4. Integration Testing

- Tested full flow: User login → Add book to cart → Place order → View order history, to verify interaction between frontend and backend modules.
- Verified that book stock is updated after an order and that sellers receive the correct order data.

9. **KNOWN ISSUES**

- Book images sometimes fail to upload if the file size is large.
- Validation messages on some forms are not consistently displayed.
- Admin panel lacks pagination for large user lists.
- UI layout may break slightly on very small mobile screens.
- Wishlist feature does not show real-time updates after removal.
- No confirmation prompt before deleting a book or order.
- Limited feedback messages on server errors (e.g., network failure).

11.FUTURE ENHANCEMENTS

Mobile App Version

- Build a React Native app for Android and iOS.
- Make the platform easily usable on smartphones.

Search Autocomplete and Filters

- Add smart suggestions while typing.
- Use filters for genre, price, and rating.

Review and Rating System

- Let users rate books and sellers.
- Display reviews on book detail pages.

Real-Time Chat Feature

- Enable live chat between buyers and sellers.
- Add message notifications for quick replies.

Email and SMS Notifications

- Send updates for orders and registrations.
- Notify users about deals and new arrivals.

Multi-language Support

- Allow users to switch between languages.
- Add language settings to the user profile.

DEMO LINKS

➤ **Video Demo Link:**

<https://drive.google.com/drive/folders/1d7HtQys9IQHuo-VsrFOGOSELicP8SqAT>

➤ **Project Links:**

https://drive.google.com/drive/folders/1kC_gVVwZlHaZ6VuGvs71-4ynQXmiC2Kh

➤ **Drive Link:**

<https://drive.google.com/drive/folders/1npuJb5pTc-hsHAdZ7resPYVr7BXouaoy>