BOOK STORE APPLICATION

USING MERN STACK BY MONGO DB

Submitted by

JEEVA.J	410821104009
MATHAVAN.M	410821104015
JEEVA KUMAR.S	410821104010
MADHESHWARAN.K	410821104013

In partial fulfilment for the award of degree

Of

BACHELOR OF ENGINEERING

In

COMPUTER SCIENCE ENGINEERING

GKM COLLEGE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY, CHENNAI 63
NOVEMBER-2024

GKM COLLEGE OF ENGINEERING AND TECHNOLOGY

CHENNAI-600063

NAAN MUDHALVAN PROJECT

TEAM ID-NM2024TMID06335 x

TEAM MEMBERS	NM ID
JEEVA.J	82B38C8A53DB79B86C043BFDDCCE0856
MATHAVAN.M	FFCA82D517DDDA182FD454AA462F613F
JEEVA KUMAR.S	B576228482C8A1C889C4DE13249ACD06
MADHESHWARAN.K	9A8549709C47A1F97974AB28DEA09D8C

BOOK STORE

INTRODUCTION

Welcome to Bookstore, a comprehensive and user-friendly online platform designed to transform the book-buying experience. Whether you're a dedicated bibliophile, a casual reader, or a student, Bookstore offers a seamless way to discover, explore, and purchase books across a wide range of genres, authors, and formats.

Bookstore brings together an extensive catalog of titles, including bestsellers, classic literature, academic resources, and much more. With detailed descriptions, customer reviews, and personalized recommendations, Bookstore empowers users to make informed purchasing decisions and dive into new literary adventures with confidence.

The platform is designed with simplicity and ease in mind. Users can browse by genre, author, or featured categories, making it effortless to find exactly what they're looking for. Once a book is chosen, purchasing is quick and secure with options for preferred payment methods, personalized delivery, and a streamlined checkout process.

Bookstore caters not only to individual readers but also supports publishers and independent authors by providing a specialized dashboard to manage inventory, pricing, and sales analytics. This robust, multi-faceted platform redefines online book shopping, delivering a well-rounded and engaging experience for everyone involved in the world of books.

SCENARIO:

Solution with Bookstore App:

Meet Lisa, a college student burning the midnight oil to finish her assignment. As the clock strikes midnight, her stomach grumbles, reminding her that she skipped dinner. Lisa doesn't want to interrupt her workflow by cooking, nor does she have the energy to venture outside in search of food.

Solution with Food Ordering App:

1. **Book Search and Availability Check:** Alex opens the Bookstore app and quickly navigates to the "Academic Books" section. He types in the title of the textbook he needs and is relieved to see it listed with the option for instant digital access and next-day delivery for the physical copy.

2. **Reviews and Preview:** Before committing to the purchase, Alex checks the ratings and reads customer reviews to ensure the book covers the material he needs. He also takes advantage of the preview feature, skimming through a few pages to confirm it aligns with his syllabus.

3. **Easy Checkout Process:** Satisfied with his choice, Alex adds the book to his cart and proceeds to checkout. He selects digital access for immediate study and next-day delivery for the physical copy, choosing a payment option that suits him best.

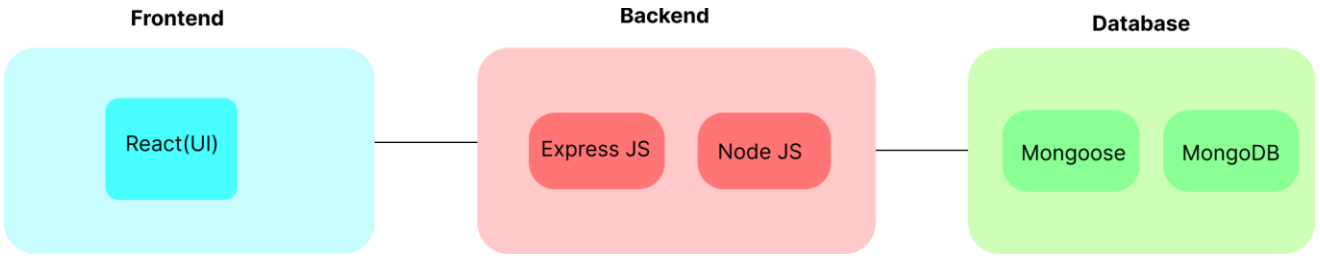
4. **Easy Checkout Process:** Satisfied with his choice, Alex adds the book to his cart and proceeds to checkout. He selects digital access for immediate study and next-day delivery for the physical copy, choosing a payment option that suits him best.

5. **Next-Day Delivery of Physical Copy:** The following day, as promised, the physical book arrives at Alex's doorstep. This allows him to make annotations and continue studying offline, feeling well-prepared and grateful for the convenience Bookstore provides.

Order Tracking and Notifications: Throughout the process, Alex receives real-time updates on his order status, including delivery tracking for the physical book.

This scenario illustrates how Bookstore meets users' urgent needs by offering a wide selection of books, flexible formats, and convenient delivery options—all while making the buying process fast, intuitive, and stress-free.

TECHNICAL ARCHITECTURE:



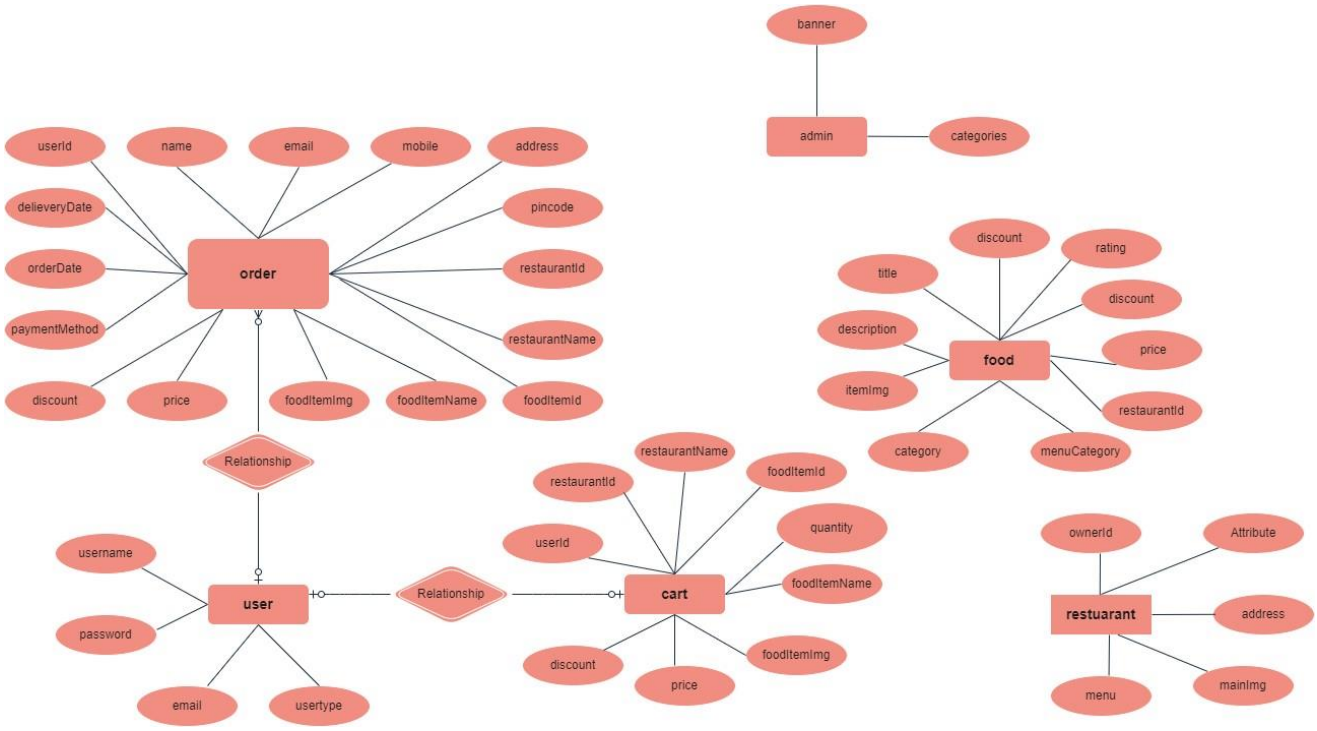
In this architecture diagram:

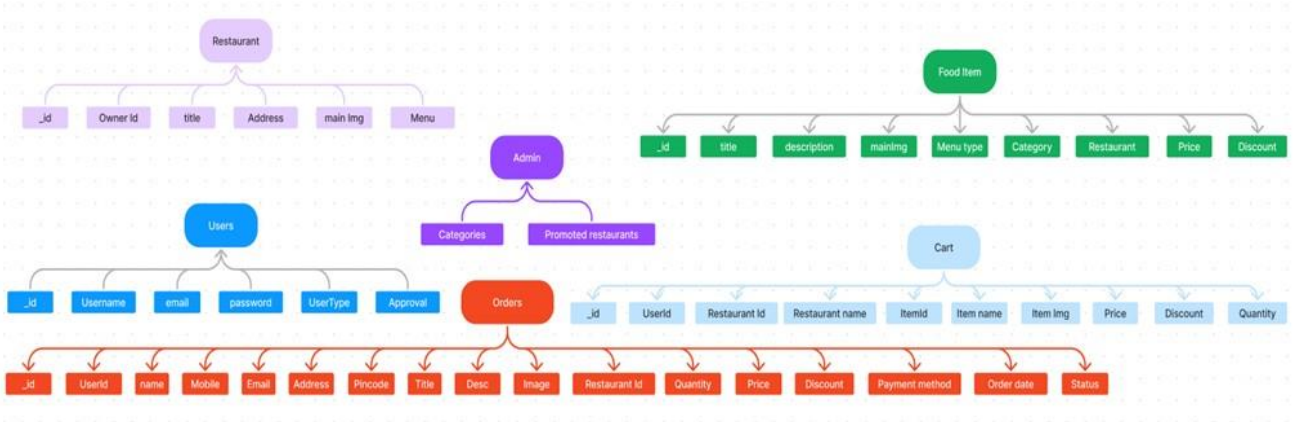
- The frontend is represented by the "Frontend" section, including user interface components such as User Authentication, Cart,

Products, Profile, Admin dashboard, etc.,

- The backend is represented by the "Backend" section, consisting of API endpoints for Users, Orders, Products, etc., It also includes Admin Authentication and an Admin Dashboard.
- The Database section represents the database that stores collections for Users, Admin, Cart, Orders, and products.

ER DIAGRAM:





The SB Foods ER-diagram represents the entities and relationships involved in an food ordering e-commerce system. It illustrates how users, restaurants, products, carts, and orders are interconnected. Here is a breakdown of the entities and their relationships:

User: Represents the individuals or entities who are registered in the platform.

Restaurant: This represents the collection of details of each restaurant in the platform. **Admin:** Represents a collection with important details such as promoted restaurants and Categories.

Products: Represents a collection of all the food items available in the platform.

Cart: This collection stores all the products that are added to the cart by users. Here, the elements in the cart are differentiated by the user Id.

Orders: This collection stores all the orders that are made by the users in the platform.

FEATURES:

1. **Extensive Book Catalog:** The bookstore offers a vast selection of books across various genres, including fiction, non-fiction, academic, children's books, and more. Each book listing provides details such as author, publication date, synopsis, ISBN, and customer reviews to help readers make informed choices.

2. **User-Friendly Search and Filter Options**: Customers can easily find books using intuitive search options and filters. They can search by title, author, genre, or ISBN and use filters like price range, rating, and publication year to refine their search results.

3. **Personalized Recommendations:** Based on users' previous purchases, browsing history, and interests, the system offers personalized book recommendations, enhancing the shopping experience and encouraging more purchases.

4. **User Account and Profile Management:** Registered users can create and manage profiles, view their purchase history, update personal information, and save preferred payment and shipping details for faster checkouts.

5. **Shopping Cart and Wishlist**: Users can add books to their shopping cart for purchase or save items to their wishlist to buy later. They can review and modify items in their cart or wishlist anytime before checkout.

6. **Secure Checkout and Payment**: The bookstore provides a secure checkout process with multiple payment options, including credit/debit cards, online banking, and digital wallets, ensuring a smooth and secure transaction.

7.

PREREQUISITES:

To develop a full-stack Book Store app using React JS, Node.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.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. To install React.js, a JavaScript library for building user interfaces, follow the installation guide:
https://reactjs.org/docs/create-a-new-react-app.html

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 Framework: Utilize Angular to build the user-facing part of the application, including product 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

To Connect the Database with Node JS go through the below provided link:

Link: https://www.section.io/engineering-education/nodejsmongoosejs-mongodb/

To run the existing SB Foods App project downloaded from github:

Follow below steps:

Clone the repository:

- Open your terminal or command prompt.
- Navigate to the directory where you want to store the e-commerce app.
- Execute the following command to clone the repository:

Git clone: https://github.com/harsha-vardhan-reddy-07/Food-Ordering-App-MERN Install

Dependencies:

- Navigate into the cloned repository directory: **cd Food-Ordering-App-MERN**
- Install the required dependencies by running the following command: **npm install**

Start the Development Server:

- To start the development server, execute the following command: **npm run dev or npm run start**
- The e-commerce app will be accessible at http://localhost:3000 by default. You can change the port configuration in the .env file if needed.

Access the App:

- Open your web browser and navigate to http://localhost:3000.
- You should see the flight booking app's homepage, indicating that the installation and setup were successful.

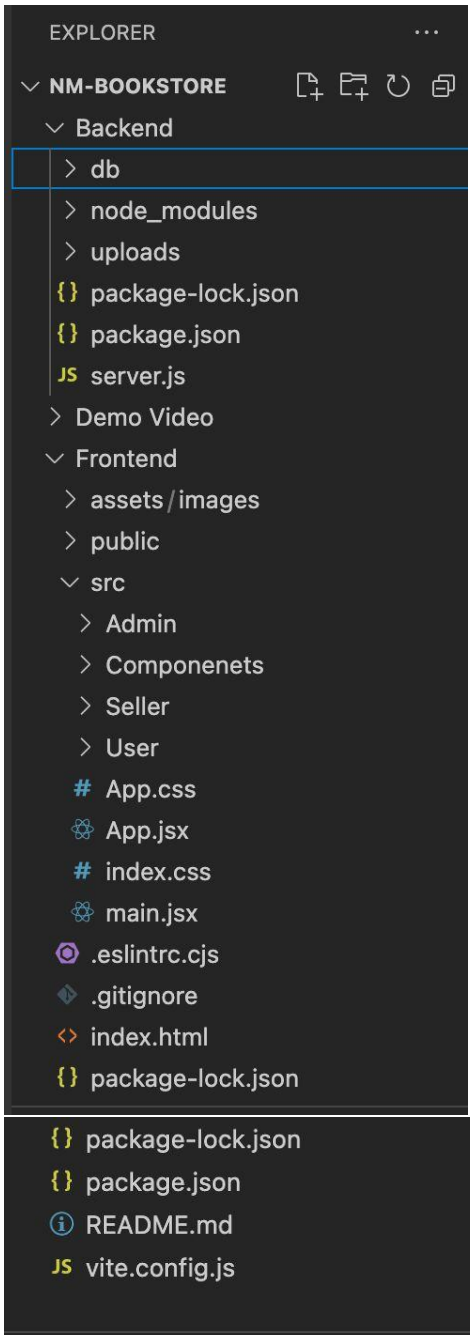
You have successfully installed and set up the SB Foods app on your local machine. You can now proceed with further customization, development, and testing as needed.

USER & ADMIN FLOW:

1. **User Flow:**
    - Register and log in.
    - Browse available books.
    - Add selected books to the cart.
    - Enter payment details and confirm the order.
    - Track orders in the user profile section.

2. **Admin Flow:**
   - Log in with admin credentials.
   - Manage book listings and user orders.
   - Access all books and users in the admin dashboard.

PROJECT STRUCTURE



This structure assumes a React app and follows a modular approach. Here's a brief explanation of the main directories and files:

- src/components: Contains components related to the application such as, register, login, home, etc.,
- src/pages has the files for all the pages in the application.

PROJECT SETUP AND CONFIGURATION:

Install required tools and software:

- Node.js.

Reference Article: https://www.geeksforgeeks.org/installation-of-node-js-on-windows/

- Git.

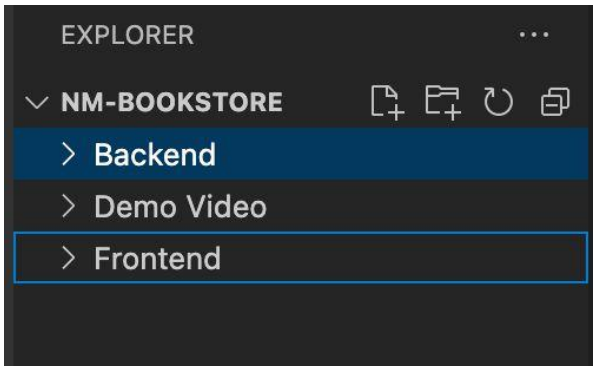
Reference Article: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Create project folders and files:

- Client folders. • Server folders

Referral Video Link:
https://drive.google.com/file/d/1uSMbPIAR6rfAEMcb_nLZAZd5QIjTpnYQ/view?usp=sharin g

Referral Image:



DATABASE DEVELOPMENT:

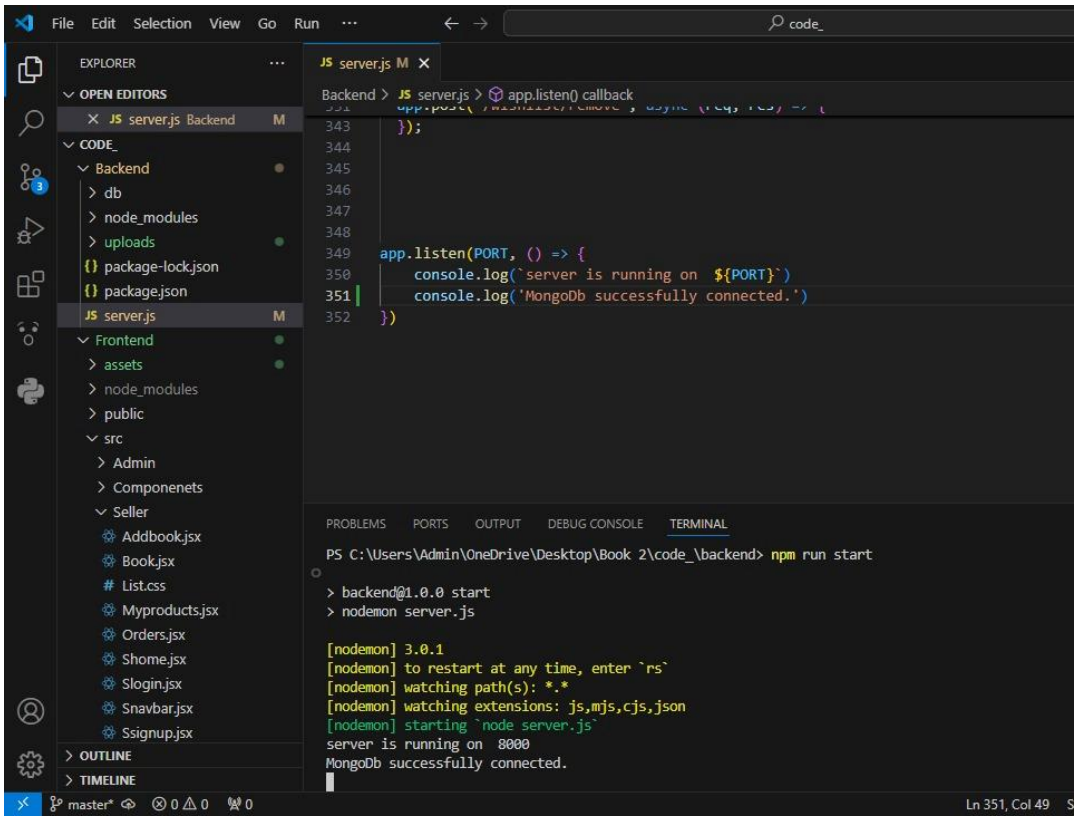
Create database in cloud video link:https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view

- Install Mongoose.
- Create database connection.

Reference Video of connect node with mongoDB database: https://drive.google.com/file/d/1cTS3_-EOAAvDctkibG5zVikrTdmoY2Ag/view?usp=sharing Reference Article:

https://www.mongodb.com/docs/atlas/tutorial/connect-to-your-cluster/

Reference Image:



Schema use-case:

1. **User Schema:**
    - Schema: userSchema
    - Model: ‘User’
    - The User schema represents the user data and includes fields such as username, email, and password.
2. **Product Schema:**
    - Schema: productSchema
    - Model: ‘Product’
    - The Product schema represents the data of all the products in the platform.
    - It is used to store information about the product details, which will later be useful for ordering.

3. **Orders Schema:**

- Schema: ordersSchema
- Model: 'Orders'
- The Orders schema represents the orders data and includes fields such as userId, product Id, product name, quantity, size, order date, etc.,

3. **Cart Schema:**

- Schema: cartSchema
- Model: 'Cart'
- The Cart schema represents the cart data and includes fields such as userId, product Id, product name, quantity, size, order date, etc., • The user Id field is a reference to the user who has the product in cart.

4. **Admin Schema:**

- Schema: adminSchema
- Model: 'Admin'
- The admin schema has essential data such as categories, promoted, etc.,

Schemas: Now let us define the required schemas

BACKEND DEVELOPMENT:

Set Up Project Structure:

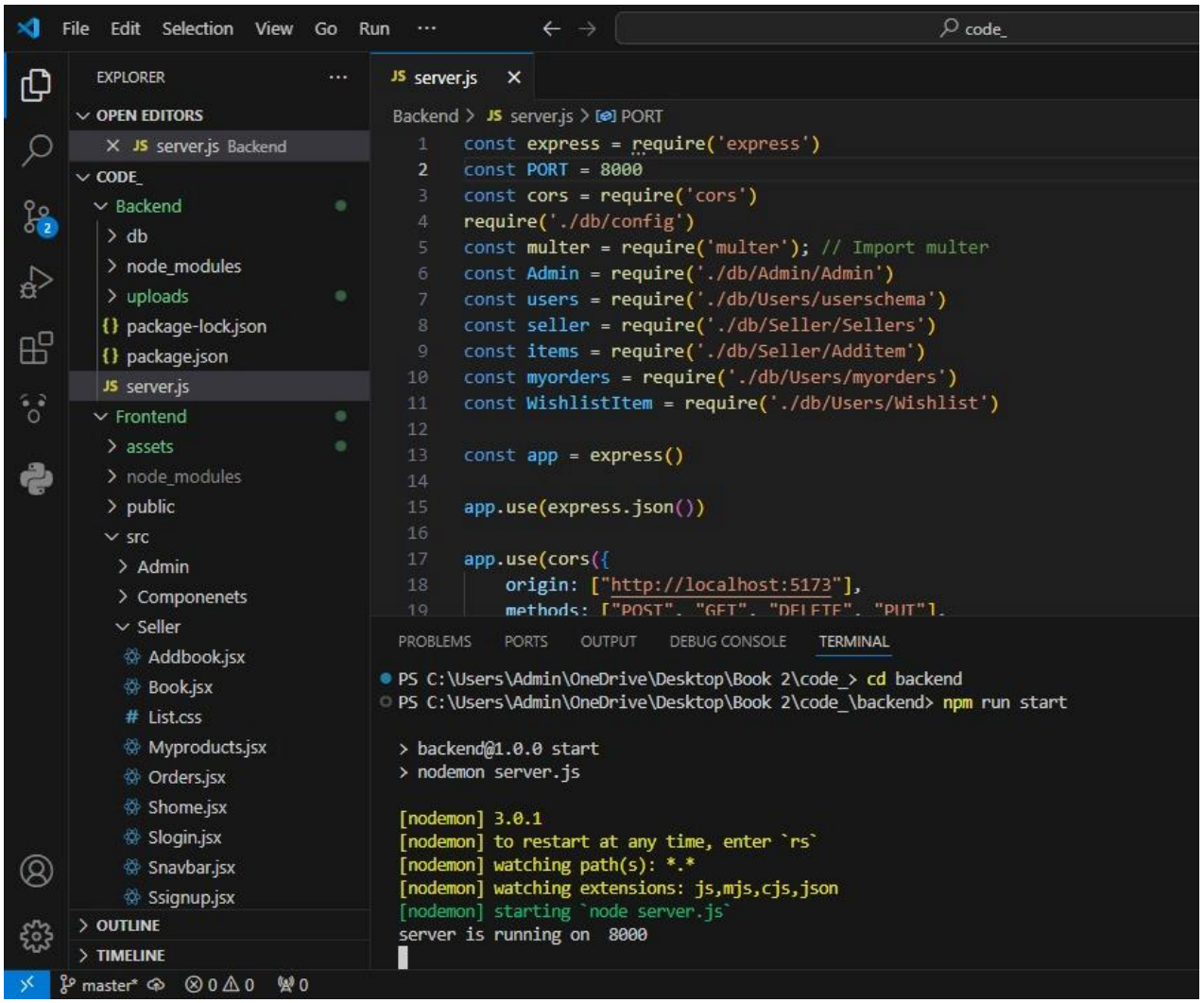
- Create a new directory for your project and set up a package.json file using the npm init command.
- Install necessary dependencies such as Express.js, Mongoose, and other required packages.

Reference Video: https://drive.google.com/file/d/19df7NU-gQK3DO6wr7ooAfJYIQwnemZoF/view?usp=sharing

1. Setup express server:

- Create index.js file.
- Create an express server on your desired port number.
- Define API's

Reference Video: https://drive.google.com/file/d/1-uKMIcrok_ROHyZl2vRORggrYRio2qXS/view?usp=sharing Reference Image:



2. Database Configuration:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas or use locally with MongoDB compass.
- Create a database and define the necessary collections for admin, users,
- Reference Video of connect node with mongoDB database:

  https://drive.google.com/file/d/1cTS3_-EOAAvDctkibG5zVikrTdmoY2Ag/view?usp=sharing

  Reference Article: https://www.mongodb.com/docs/atlas/tutorial/connect-to-your-cluster

3. 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 for handling cross-origin requests.

Reference Video:
https://drive.google.com/file/d/1-uKMIcrok_ROHyZl2vRORggrYRio2qXS/view?usp=s haring

4. **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.

5. **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.

6. **User Authentication:**
   - Create routes and middleware for user registration, login, and logout. • Set up authentication middleware to protect routes that require user authentication.

7. **Handle new products and Orders:**
   - Create routes and controllers to handle new product listings, including fetching products data from the database and sending it as a response.
   - Implement ordering(buy) functionality by creating routes and controllers to handle order requests, including validation and database updates.

8. **Admin Functionality:**
   - Implement routes and controllers specific to admin functionalities such as adding products, managing user orders, etc.
   - Add necessary authentication and authorization checks to ensure only authorized admins can access these routes.

9. **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.

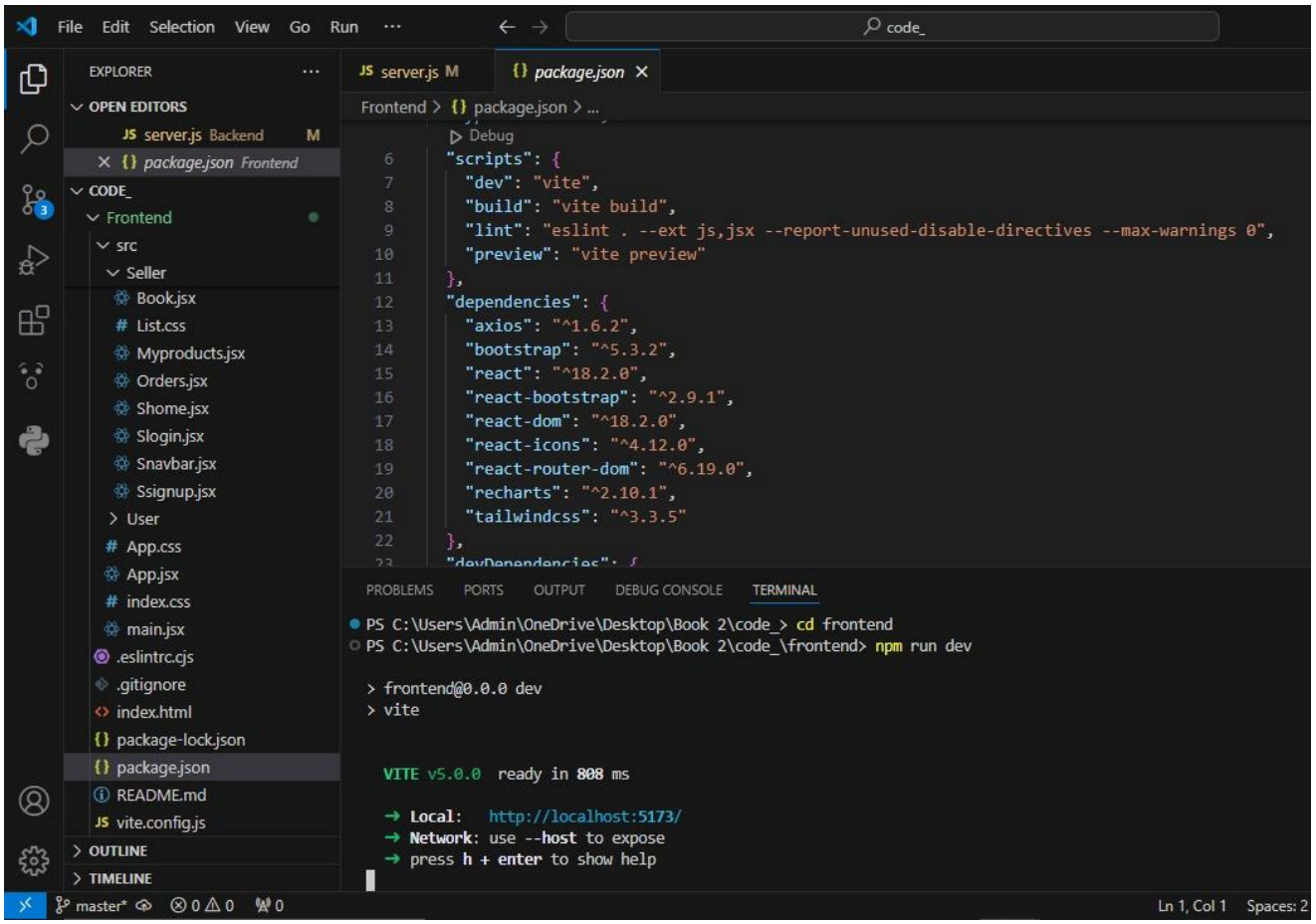
FRONTEND DEVELOPMENT:

1. **Setup React Application:**
    - Create a React app in the client folder.
    - Install required libraries
    - Create required pages and components and add routes.
2. **Design UI components:**
    - Create Components.
    - Implement layout and styling.
    - Add navigation.
3. **Implement frontend logic:**
    - Integration with API endpoints.
    - Implement data binding.

Reference Video Link:
https://drive.google.com/file/d/1EokogagcLMUGiIluwHGYQo65x8GRpDcP/view?usp=sharin g

Reference Article Link: https://www.w3schools.com/react/react_getstarted.asp

Reference Image:



CODE EXPLANATION

Server setup:

Let us import all the required tools/libraries and connect the database.

User Authentication:

- **Backend**

Now, here we define the functions to handle http requests from the client for authentication.

All Products (User):

- Frontend

In the home page, we'll fetch all the products available in the platform along with the filters.

- **Backend**

In the backend, we fetch all the products and then filter them on the client side.

Add product to cart:

- **Frontend**

  Here, we can add the product to the cart and later can buy them.

- **Backend**

Add product to cart:

Order products:

Now, from the cart, let's place the order

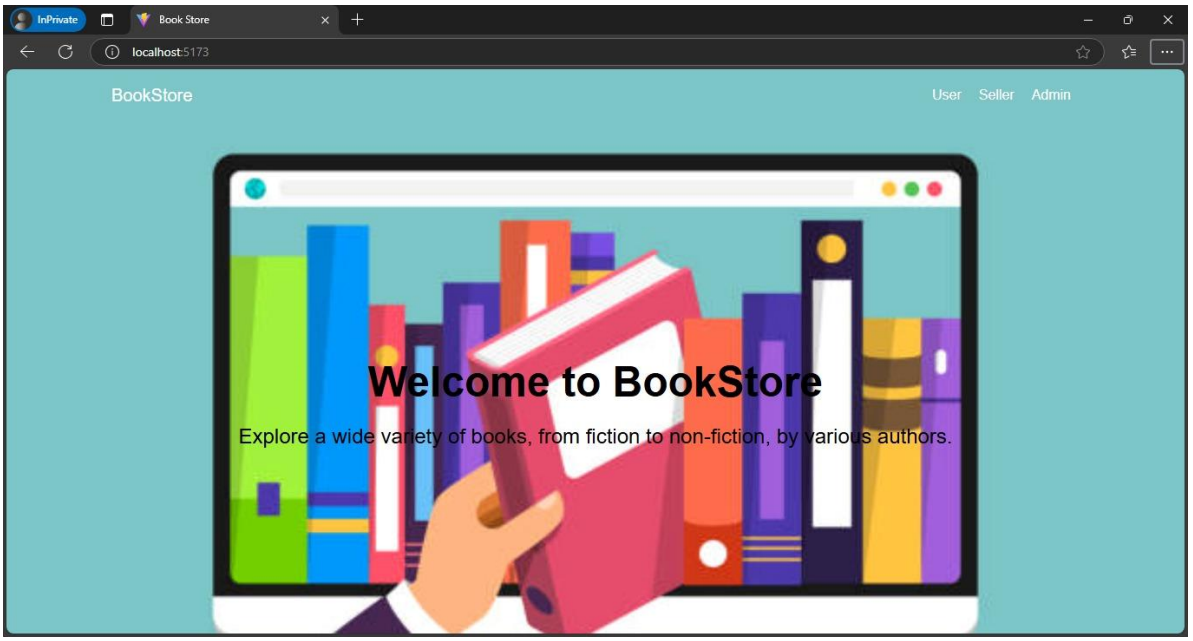
Add new product:

Here, in the admin dashboard, we will add a new product.

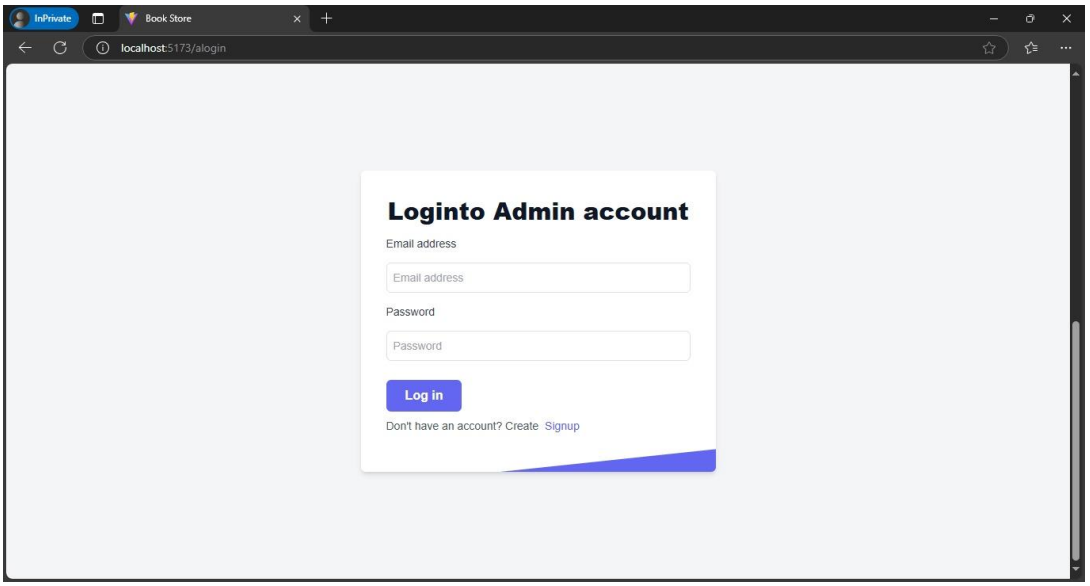
Along with this, implement additional features to view all orders, products, etc., in the admin dashboard.

Demo UI images:

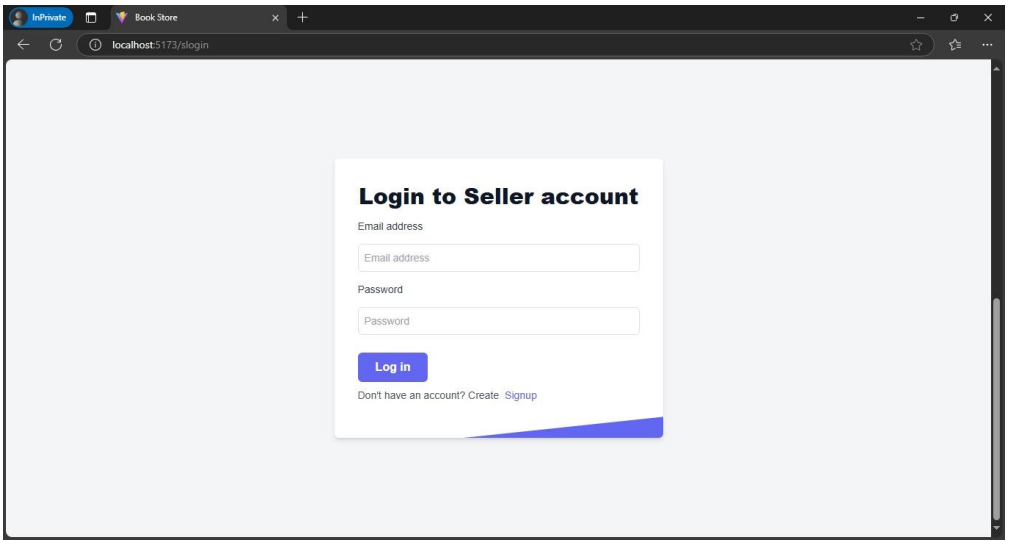
- **Landing page**



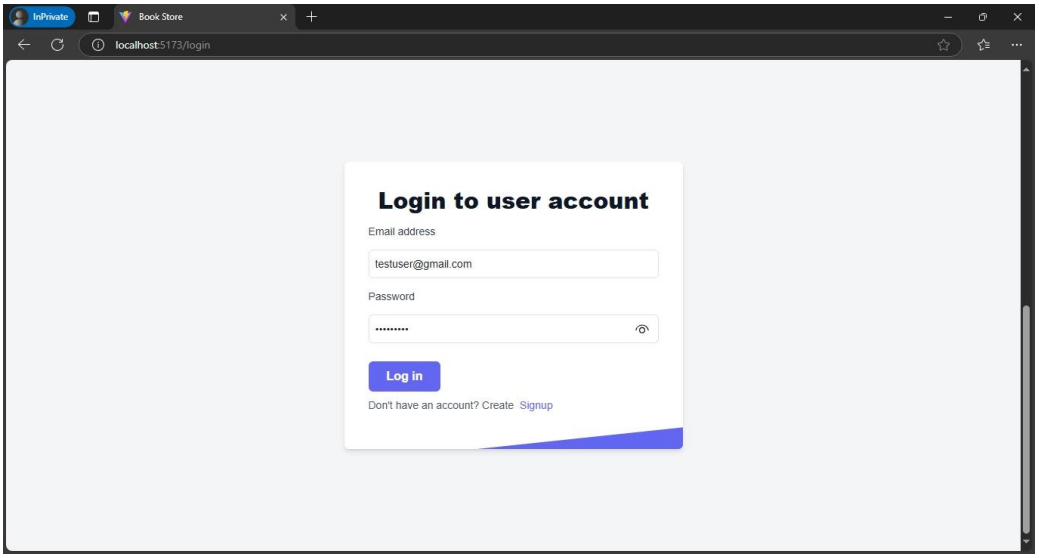
Admin Login



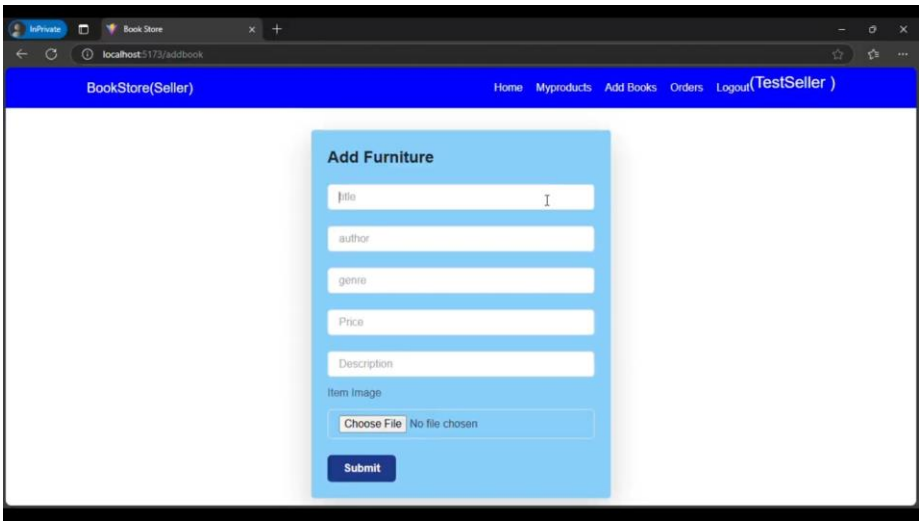
Seller Login



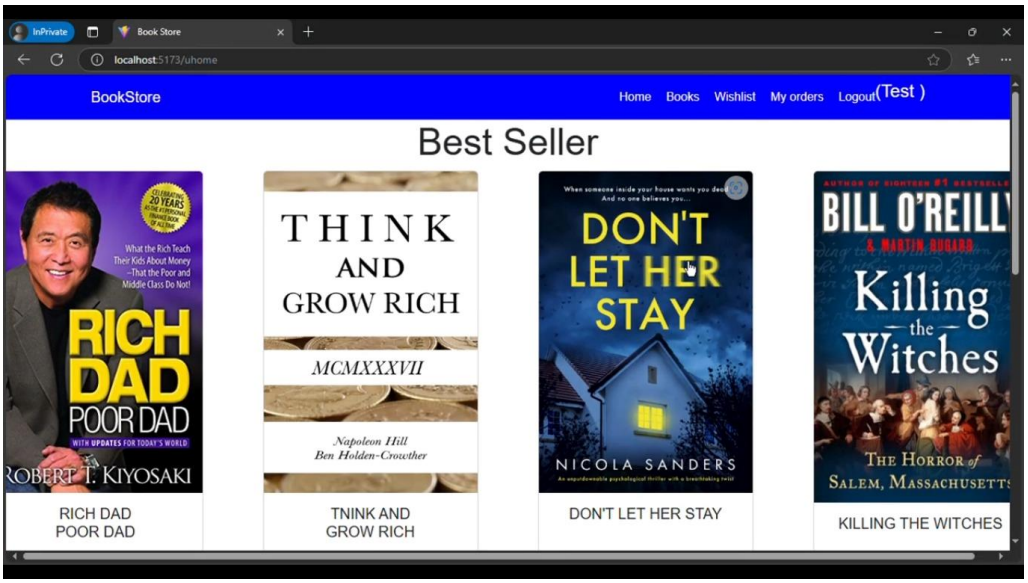
Authentication



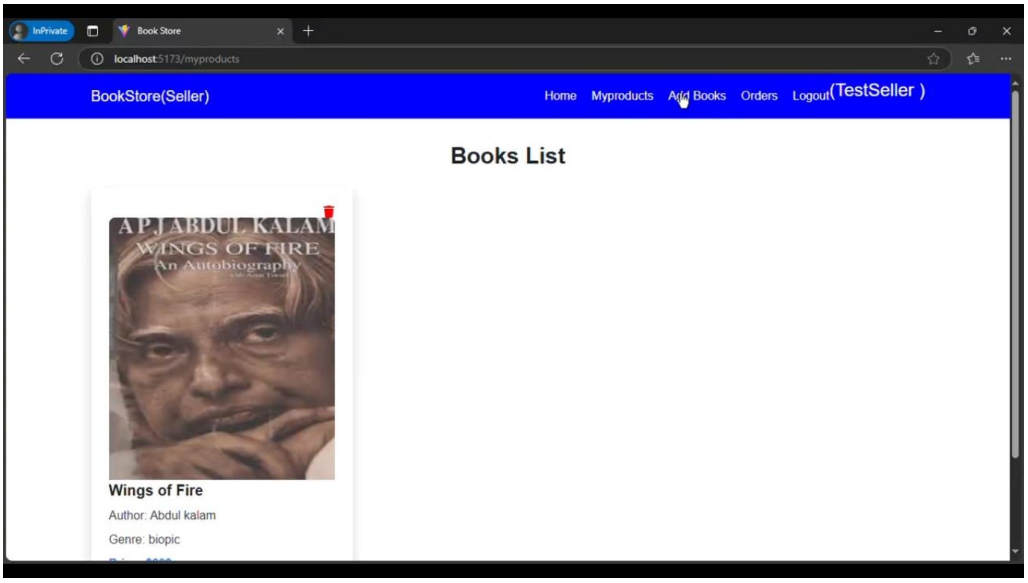
Seller



Best Seller



Admin dashboard



For any further doubts or help, please consider the GitHub repo, https://github.com/JEEVAfr/NM-BOOKSTORE

Bookstore The demo of the app is available at:

https://drive.google.com/file/d/1MNTmIRCROHI2YnLBqDjhqgmi96lBxJX3/view?userstoinvite=rajarams7200@gmail.com&sharingaction=manageaccess&role=writer&ts=67370d19 sharing

** Happy Coding **