**BookNest: Where Stories Nestle** 

Team ID: LTVIP2025TMID53966

**Team Members:** 

Vangara Bharath Kumar Vaddi Gayathridevi Veeramallu Deekshitha Maithreyi Vakkalagaddda Likitha

#### INTRODUCTION

BookNest is a cutting-edge digital bookstore application tailored for the modern bibliophile. Built using the MERN Stack (MongoDB, Express.js, React, Node.js), BookNest offers a seamless and immersive literary browsing and shopping experience. Whether you're discovering new releases or revisiting classics, BookNest redefines how readers connect with literature in the digital age.

## **Key Highlights:**

Immersive and responsive UI

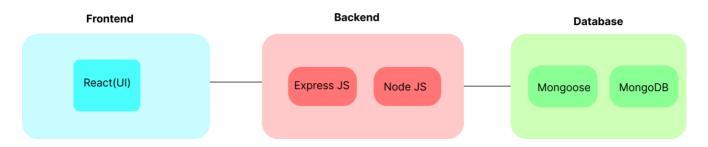
Personalized recommendations

Scalable and efficient backend

Secure and intuitive purchase experience.

## **PROJECT OVERVIEW**

# **TECHNICAL ARCHITECTURE:**



In this architecture diagram:

**Frontend (React)**: User-facing interface including user authentication, book listings, cart, profile, and admin dashboard.

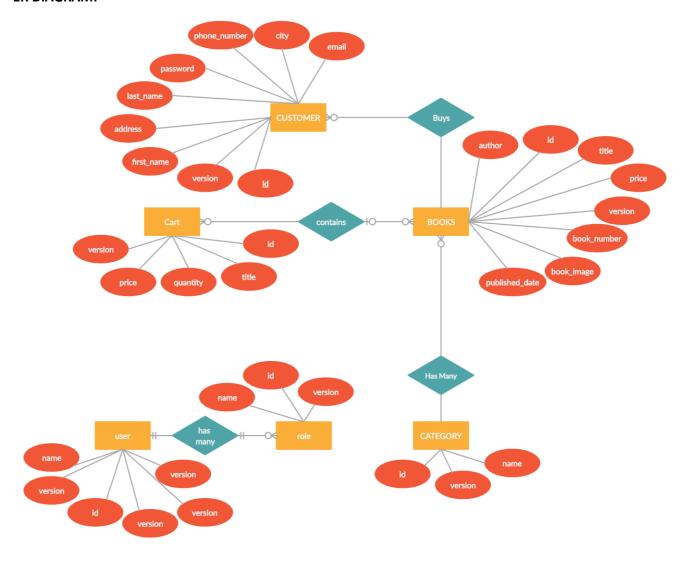
Backend (Node.js + Express.js): RESTful APIs for users, books, orders, admin controls.

Database (MongoDB): Collections for Users, Books, Cart, Orders, and Admin data.

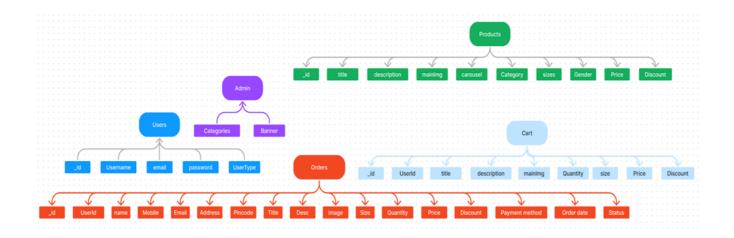
API Gateway: Central entry point for client requests, routing to appropriate services.

Authentication Service: Handles registration, login, JWT-based sessions, and role-based access.

# **ER DIAGRAM:**



• The Database section represents the database that stores collections for Users, Admin, Cart, Orders and products.



The BookNest ER diagram illustrates the relationships between users, books, carts, and orders, ensuring seamless tracking of user interactions and transaction flows across the application.

User: User registration and profile

Admin: Manages books, orders, and categories

**Books**: Title, author, genre, price, description, stock

Cart: Tracks user-selected books before purchase

**Orders**: Stores completed transactions with details

## **PURPOSE:**

BookNest is designed to provide an online bookstore where users can:

- 1. Easily register and authenticate
- 2.Discover and browse books by genre, author, or rating
- 3.Add books to cart and complete secure purchases
- 4. Track orders and review purchase history
- 5. Admins can manage books, users, and orders

#### **FEATURES:**

- **1.Comprehensive Book Catalog**: Explore detailed listings with title, author, price, genre, reviews, and availability.
- **2.Personalized Discovery**: Filter and search books by user preferences.
- **3.Secure Checkout**: Streamlined order placement with secure payment integrations.
- **4.Order Confirmation**: Post-purchase details including order ID, book list, and shipping info.
- 5. User Dashboard: View current and past orders, rate purchases, and track shipments.

**6.Admin Dashboard**: Add/edit/remove books, view sales data, manage users and orders.

## **PREREQUISITES:**

To develop a full-stack e-commerce 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: <a href="https://nodejs.org/en/download/package-manager/">https://nodejs.org/en/download/package-manager/</a>

**MongoDB:** Set up a MongoDB database to store hotel and booking information. Install MongoDB locally or use a cloud-based MongoDB service.

- Download: <a href="https://www.mongodb.com/try/download/community">https://www.mongodb.com/try/download/community</a>
- Installation instructions: <a href="https://docs.mongodb.com/manual/installation/">https://docs.mongodb.com/manual/installation/</a>

**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: <a href="https://gitscm.com/downloads">https://gitscm.com/downloads</a>

**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 <a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a>
- Sublime Text: Download from <a href="https://www.sublimetext.com/download">https://www.sublimetext.com/download</a>
- WebStorm: Download from https://www.jetbrains.com/webstorm/download

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

Link: <a href="https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/">https://www.section.io/engineering-education/nodejs-mongoosejs-mongodb/</a>

## To run the existing BookNest 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 BookNest app.
- Execute the following command to clone the repository:

Git clone: https://github.com/yaswanthpuritipati/ShopEZ-e-commerce-MERN

## **Install Dependencies:**

• Navigate into the cloned repository directory:

## cd SBookNest-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:4000 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:4000.
- You should see the flight booking app's homepage, indicating that the installation and setup were successful.

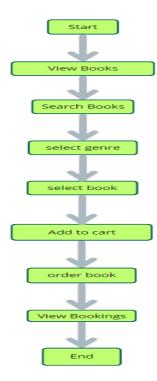
You have successfully installed and set up the BookNest app on your local machine. You can now

proceed with further customization, development, and testing as needed.

## **USER & ADMIN FLOW:**

## 1. User Flow:

- Users start by registering for an account.
- After registration, they can log in with their credentials.
- Once logged in, they can check for the available books in the platform.
- Users can add the books they wish to their carts and order.
- They can then proceed by entering address and payment details.
- After ordering, they can check them in the profile section.



## 2. Admin Flow:

- Admins start by logging in with their credentials.
- Once logged in, they are directed to the Admin Dashboard.
- Admins can access the users list, books, orders, etc.,

## **PROJECTSTRUCTURE:**



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.

# ✓ CODEV2 ∨ codev2\code ∨ Backend > db > node\_modules > uploads {} package-lock.json {} package.json JS server.js Frontend > node\_modules > public > src eslintrc.cjs .gitignore index.html {} package-lock.json {} package.json README.md vite.config.js

## 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

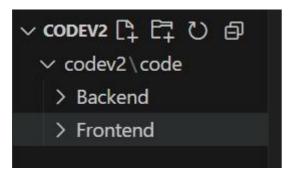
# Create project folders and files:

- Client folders.
- Server folders

#### Referral Video Link:

https://drive.google.com/file/d/1uSMbPIAR6rfAEMcb nLZAZd5QIjTpnYQ/view?usp=sharing

## Referral Image:



## **DATABASE DEVELOPMENT:**

Create database in cloud video link:- <a href="https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view">https://drive.google.com/file/d/1CQil5KzGnPvkVOPWTLP0h-Bu2bXhq7A3/view</a>

- Install Mongoose.
- Create database connection.

Reference Video of connect node with mongoDB database: <a href="https://drive.google.com/file/d/1cTS3">https://drive.google.com/file/d/1cTS3</a> - EOAAvDctkibG5zVikrTdmoY2Ag/view?usp=sharing

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

# Reference Image:

```
XI File Edit Selection View Go Run ···
                                                                                                                                                                                                          88 ~
                                                                                                                                                                                                                                         ○ codev1
          EXPLORER ... JS server.js X
                                                                                                                                                                                                                                                                                  □ ...
凸
         codev1 code codev1 code > Backend > Js serveris > ...

codev1 code const express = require('express')

2    const PORT = 4000

3    const cors = require('cors')

4    require('./db/config')

5    const multer = require('multer'); // Import multer

6    const Admin = require('./db/Admin/Admin')

7    const users = require('./db/Seller/Additem')

8    const seller = require('./db/Seller/Additem')

7    const users = require('./db/Seller/Additem')

8    const myorders = require('./db/Users/myorders')

10    const wishlistItem = require('./db/Users/myorders')
                                                  11 const WishlistItem = require('./db/Users/Wishlist')
                                                   13 const app = express()
               .gitignore
                                                            app.use(cors(
                                                            origin: ["http://localhost:5173"],
methods: ["POST", "GET", "DELETE", "PUT"],
credentials: true
               {} package-lock.json
              {} package.json
                                                             const storage = multer.diskStorage({
                                                                   destination: 'uploads', // The directory where uploaded files will be stored
filename: function (req, file, callback) {
                                                                          callback(null, Date.now() + '-' + file.originalname); // Set the file name
         > OUTLINE
```

## Model use-case:

#### 1. User Model:

• Model: 'User'

- The User schema represents the user data and includes fields such as username, email, and password.
- It is used to store user information for registration and authentication purposes.
- The email field is marked as unique to ensure that each user has a unique email address

#### 2. Book Model:

- Model: 'Book'
- The Book schema represents the data of all the books in the platform.
- It is used to store information about the book details, which will later be useful for ordering .

#### 3. Order Model:

- Model: 'Order'
- The Orders schema represents the orders data and includes fields such as userId, book Id, book name, quantity, size, order date, etc.,
- It is used to store information about the orders made by users.
- The user Id field is a reference to the user who made the order.

## **Code Explanation:**

```
∨ CODEV2
                                codev2 > code > Frontend > src > User > 🎡 Login.jsx > 😰 Login > 😰 handleSubmit > 😚 then() callback
                              import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import Home from '../Componenets/Home';

∨ Backend

    JS server.js

→ Frontend

    > Admin
                                          const navigate = useNavigate();
      > Componenets
                                           axios.defaults.withCredentials = true;
      > Seller
                                          const handleSubmit = (e) => {
       Login.jsx
                                           e.preventDefault();
setLoading(true);
       Myorders.jsx
       ☼ OrderItem.jsx
                                let payload = { email, password };

axios

.post("http://localhost:4000/login", payload)

.then((res) => {

console.log("login: " + res.data.Status);

if (res.data.Status === "Success") {

.console.log(res.data.user);
       Products.jsx
       🥸 Signup.jsx
       # uhome.css

♥ Uhome.jsx

       Uitem.jsx
       Unavbar.jsx
                                                   console.log(res.data.user);
localStorage.setItem('user', JSON.stringify(res.data.user));
navigate('/uhome');
alert("login_successful");

⇔ Wishlist.jsx

      # App.css
                                                       alert("Login successful");
      App.jsx
```

```
	imes File Edit Selection View Go Run \cdots \leftarrow 	o
                                                                                                           83 ~
                                                                                                                            ... JS server.js
(D)
                          app.post('/items', upload.single('itemImage'), async (req, res) => {

∨ codev1\code

       ✓ Backend
                                  const item = new items({ itemImage, title, author, genre, description, price, userId, userName });
await item.save();
ည္
       > node_modules
                                res.status
} catch (err)
        > uploads
                                       res.status(201).json(item);
       {} package-lock.json
                                       res.status(400).json({ error: 'Failed to create item' });
       {} package.json
Js server.js
        > node_modules
                                app.get('/getitem/:userId', async (req, res) => {
                                  const userId = req.params.userId;
        > src
                                  const tasks = await items.find({ userId }).sort('position');
res.json(tasks);
        eslintrc.cjs
        gitignore
                                  } catch (err)
        index html
                                       res.status(500).json({ error: 'Failed to fetch tasks' });
        {} package-lock.json
        {} package.json

    README.md

        🔻 vite.config.js
                                app.delete('/itemdelete/:id', (req, res) => {
                                    const { id } = req.params;
                                   items.findByIdAndDelete(id)
                                           res.sendStatus(200);
(2)
                                        .catch((error) => {
> outline
                       codev2 > code > Frontend > src > User > 😘 Orderitem.jsx > 🗘 Orderitem

∨ codev2\code

                       import React, { useState, useEffect } from 'react';
import axios from 'axios';
 ∨ Backend
 JS server.js
                       206 import { useNavigate, useParams } from 'react-router-dom';
 ∨ Frontend
                                       '../Seller/List.css';
                              import
                       208 import Unavbar from './Unavbar';
  > node_modules
  > public
                              function OrderItem() {
  ∨ src
                              const [item, setItem] = useState({});
   > Admin
                               const [formData, setFormData] = useState({
  > Componenets
                                  flatno: '',
  > Seller
                                city: '',
                                pincode: '',

∨ User

                                  state: '',
   Login.jsx
  Myorders.jsx
                                const fee= 99;
    OrderItem.jsx
                                const { id } = useParams();
                                const navigate = useNavigate();
   Signup.jsx
   # uhome.css
                               useEffect(() => {
                                axios.get(`http://localhost:4000/item/${id}`)
   ₩ Uhome.jsx
                                  .then((resp) => {
    Uitem.jsx
                                      setItem(resp.data);

☆ Unavbar.jsx

                                  .catch((error) => {
   Wishlist.jsx
   # App.css
                                     console.log("Failed to fetch item data:", error);
   App.jsx
OUTLINE
TIMELINE
```

# **BACKEND DEVELOPMENT:**

## Setup express server:

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

Reference Video: <a href="https://drive.google.com/file/d/1-uKMIcrok\_ROHyZI2vRORggrYRio2qXS/view?usp=sharing">https://drive.google.com/file/d/1-uKMIcrok\_ROHyZI2vRORggrYRio2qXS/view?usp=sharing</a>

## **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

Reference Images:

```
ы раскаде.json
∨ CODEV2 📭 📴 🖔 🗊 💮 codev2 > code > Backend > 🚯 package.json > ...
 ∨ codev2\code
                2 "name": "backend",
3 "version": "1.0.0",

→ Backend

   > db
   > node_modules 5
> uploads
                                  "description": "",
                          5 "main": "index.js",
                                    ▶ Debug
  {} package-lock.json 6 "scripts": {

{} package.json 7 "test": "ed

Js server.js 8 "start": "r
                                     "test": "echo \"Error: no test specified\" && exit 1",
                                    "start": "nodemon server.js"
                                  },
"author": "",
"se": "I
  > Frontend
                                  "license": "ISC",
                                   "dependencies": {
                                     "cors": "^2.8.5",
                                     "express": "^4.18.2",
                                     "mongoose": "^8.0.1",
                                 "multer": "^1.4.5-lts.1",
"nodemon": "^3.0.1"
}
```

## 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, products, orders and other relevant data.

## 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.

# 4. Define API Routes:

- Create separate route files for different API functionalities such as users, orders, and authentication.
- Define the necessary routes for listing books, 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 books, 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 book listings, including fetching book 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 books, 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.

## **WEB 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/1EokogagcLMUGilluwHGYQo65x8GRpDcP/view?usp=sharing

#### Reference Article Link:

https://www.w3schools.com/react/react\_getstarted.asp

## Reference Image:

```
codev2 > code > Frontend > src >  App.jsx > ...
27    import Items from "./Admin/Items
v CODEV2 [森 日 ひ 日
∨ codev2\code 28

→ Backend

                             <div className="app">
                                    <Route path='/' element={<Home/>} />
  JS server.js
                            ∨ Frontend
  > node_modules 37
> public 38
  ∨ src
   > Admin
   > Admin 41
> Componenets 42
   > Seller
   > User
                                    <Route path='/slogin' element={<Slogin/>} />
   # App.css
                                   <Route path='/ssignup' element={<Ssignup/>} />
   App.jsx
                                    <Route path='/shome' element={<Shome/>} />
   # index.css
                                    <Route path='/myproducts' element={<Myproducts/>} />
   🏶 main.jsx
                                    <Route path='/addbook' element={<Addbook/>} /3
  eslintrc.cjs
                                   <Route path='/book/:id' element={<Book/>} />
                                   <Route path='/orders' element={<Orders/>} />
  .gitignore
  index.htmlpackage-lock.json5354
                                    <Route path='/login' element={<Login/>}/>
OUTLINE
                                    <Route path='/signup' element={<Signup/>} />
> TIMELINE
                                    <Route path='/nav' element={<Unavbar/>}/>
```

## PROJECT IMPLEMENTATION & EXECUTION:

## **User Authentication:**

Backend

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

```
₃ auth.js ×
server > routes > ₃ auth.js > ♡ router.post('/register') callback > ੴ error
               router.post('/login', async (req, res) => {
    30
    31
                      const { email, password } = req.body;
    32
                      const { email, password } = req.body;
const user = await User.findOne({ email });
if (!user) return res.status(400).json({ error: 'Invalid credentials' });
const match = await bcrypt.compare(password, user.password);
if (!match) return res.status(400).json({ error: 'Invalid credentials' });
const token = jwt.sign({ id: user._id, isAdmin: user.isAdmin }, JWT_SECRET, { expiresIn: '7d' });
res.json({ token, user: { id: user._id, name: user.name, email: user.email, isAdmin: user.isAdmin } });
    33
    34
    38
                  } catch (err) {
    res.status(500).json({ error: err.message });
    39
   40
   42
   43
               module.exports = router;
```

#### **Frontend**

#### Login:

```
CODEV2 [] [] [] []
                       codev2 > code > Frontend > src > User > ∰ Login.jsx > [ o] Login > [ o] handleSubmit > ∯ then() callback
                              import React, { useState } from 'react';
∨ codev2\code
                              import { useNavigate } from 'react-router-dom';
 > Backend
                              import axios from 'axios';

→ Frontend

                              import Home from '../Componenets/Home';
  > node_modules
  > public
                             const Login = () => {
  ∨ src
                              const [email, setEmail] = useState('');
                               const [password, setPassword] = useState('');
   > Admin
                               const [loading, setLoading] = useState(false);
   > Componenets
                              const navigate = useNavigate();
   > Seller

∨ User

                                axios.defaults.withCredentials = true;
    Login.jsx
    Myorders.jsx
                               const handleSubmit = (e) => {
   OrderItem.jsx
                                e.preventDefault();
                                 setLoading(true);
   Products.jsx
   Signup.jsx
                                let payload = { email, password };
    # uhome.css
                                 axios
   W Uhome.jsx
                                   .post("http://localhost:4000/login", payload)
.then((res) => {
   W Uitem.jsx
   W Unavbar.jsx
                                     console.log("login: " + res.data.Status);
                                     if (res.data.Status === "Success") {
   Wishlist.jsx
                                      console.log(res.data.user);
localStorage.setItem('user', JSON.stringify(res.data.user));
   # App.css
   App.jsx
                                        navigate('/uhome');
   # index.css
                                        alert("Login successful");
   main.isx
                        28
                                      alert("Wrong credentials");
```

## Register:

```
codev2 > code > Frontend > src > User > 🏶 Signup.jsx >
                      1 import React, { useState } from 'react';
 ∨ codev2\code
                         2 import { useNavigate } from 'react-router-dom';
  > Backend
                               import axios from 'axios';
import Home from '../Componenets/Home';
  ∨ Frontend
   > node_modules
   > public
                         const [name, setName] = useState('');
const [email, setEmail] = useState('');
const [password, setPassword] = useState('');
const [loading, setLoading] = useState(false);
   ∨ src
   > Admin 8
> Componenets 9
    > Seller
    ∨ User
                                const navigate = useNavigate();
    Login.jsx
     Myorders.jsx
                                const handleSubmit = (e) => {
     e.preventDefault();
                                 setLoading(true);
    ⇔ Products.jsx
     Signup.jsx
                                  let payload = { name, email, password };
     # uhome.css

♥ Uhome.jsx

                                  axios

₩ Uitem.jsx

                                    .post("http://localhost:4000/signup", payload)
    ⇔ Unavbar.jsx⇔ Wishlist.jsx
                                     alert('Account created successfully!');
console.log(result);
navigate(',');
                                     .then((result) => {
    # App.css
                                        navigate('/login');
                                   App.jsx
    # index.css
   main.isx
                                      console.log(err);
OUTLINE
                                         alert("Failed to create an account. Please try again.");
 TIMELINE
```

## logout:

# All books (User):

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

## **Fetching books:**

```
∨ codev2\code
✓ backend
                        1 import React, { useState, useEffect } from 'react';
  ∨ Frontend
                             import Unavbar from './Unavbar';
                            import { Link } from 'react-router-dom';
   > node_modules
   > public
                       6 function Products() {
   ∨ src
                             const [items, setItems] = useState([]);
    > Admin
                              const [wishlist, setWishlist] = useState([]);
    > Componenets
                             const [loading, setLoading] = useState(true);
    > Seller
                             const [searchTerm, setSearchTerm] = useState('');

∨ User

                              const [selectedGenre, setSelectedGenre] = useState('all');
    Login.jsx
                             useEffect(() => {
    Myorders.jsx
  OrderItem.jsx
     Products.jsx
                               .get(`http://localhost:4000/item`)
.then((response) => {
    Signup.jsx
                                  const taskData = response.data;
setItems(taskData);
    # uhome.css

⊕ Uhome.jsx

                              setLoading(ra:
})
.catch((error) => {
  console.error('Error fetching tasks: ', error);
  setLoading(false);
    Uitem.jsx

₩ Unavbar.jsx

    Wishlist.jsx
    # App.css
   App.jsx
   # index.css
   e main.jsx
                                 const user = JSON.parse(localStorage.getItem('user'));
> OUTLINE
                                 if(user){
> TIMELINE
                                 axios.get(`http://localhost:4000/wishlist/${user.id}`)
```

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

```
∨ CODEV2 [‡ 閏 ひ 自
                         codev2 > code > Frontend > src > User > @ Uitem.jsx > ...
 ∨ codev2\code
                         1 import axios from 'axios';
                               import React, { useEffect, useState } from 'react';
import { Link, useParams } from 'react-router-dom';
  ∨ Frontend
   > node_modules
                              import Unavbar from './Unavbar';
   > public
                                const Uitem = () => {
   ∨ src
                                 const [item, setItem] = useState(null);
    > Admin
                                  const [loading, setLoading] = useState(true);
const [wishlist, setWishlist] = useState([]);
const [isInWishlist, setIsInWishlist] = useState(false);
    > Componenets 8
    > Seller
    ∨ User
                                    const { id } = useParams();

⇔ Login.jsx

     Myorders.jsx
                                   useEffect(() => {
     OrderItem.jsx
     Products.jsx
                                         axios.get(`http://localhost:4000/item/${id}`)
     Signup.jsx
                                            .then((resp) => {
     # uhome.css
                                                 console.log(resp);
  Uhome.jsx
                                                 setItem(resp.data);
                                                 setLoading(false);
     Uitem.jsx
     Unavbar.jsx
                                             .catch(() => {

₩ishlist.jsx

                                                 console.log("Did not get data");
    # App.css
                                                  setLoading(false);
    App.jsx
    # index.css
    🐡 main.jsx
                                         const user = JSON.parse(localStorage.getItem('user'));
> OUTLINE
                                         if (user) {
> TIMELINE
                                             axios.get(`http://localhost:4000/wishlist/${user.id}`)
```

#### Wishlist:

```
∨ CODEV2 [‡ 閏 ひ 卣
                        codev2 > code > Frontend > src > User > ∰ Wishlist.jsx > ...
                          1

∨ codev2\code

✓ Frontend

                               import React, { useState, useEffect } from 'react';
   ∨ src
                               import axios from 'axios';
    > Componenets
                               import { Link } from 'react-router-dom';
    > Seller
                               import Unavbar from './Unavbar';

∨ User

                              function Wishlist() {

    ⇔ Login.jsx

                              const [wishlist, setWishlist] = useState([]);
     const [loading, setLoading] = useState(true);
     OrderItem.jsx
     Products.jsx
                                useEffect(() => {
     Signup.jsx
                                   const user = JSON.parse(localStorage.getItem('user'));
                                   if (user) {
     # uhome.css
                                     axios

⇔ Uhome.jsx

                                       .get(`http://localhost:4000/wishlist/${user.id}`)

₩ Uitem.jsx

                                       .then((response) => {
    ♥ Unavbar.jsx
                                         const wishlistData = response.data;
     Wishlist.jsx
                                         setWishlist(wishlistData);
    # App.css
                                         setLoading(false);
    App.jsx
                                       .catch((error) => {
    # index.css
                                         console.error('Error fetching wishlist items: ', error);
    🯶 main.jsx
                                         setLoading(false);
   .eslintrc.cjs
   .gitignore
                                   } else {
   index.html
                                     console.log('ERROR');
                                     setLoading(false);
> OUTLINE
> TIMELINE
```

#### Order Item:

Here, we can add the product to the cart or can buy directly.

```
∨ CODEV2 [th 日 ひ 日
                        codev2 > code > Frontend > src > User > ♥ OrderItem.jsx > ♥ OrderItem

∨ codev2\code

→ Frontend

                               import React, { useState, useEffect } from 'react';
                               import axios from 'axios';
   ∨ src
                               import { useNavigate, useParams } from 'react-router-dom';
    > Componenets
                               import '../Seller/List.css';
     > Seller
                               import Unavbar from './Unavbar';
    ∨ User
                               function OrderItem() {
     🔅 Login.jsx
                                 const [item, setItem] = useState({});

₩ Myorders.jsx

                                 const [formData, setFormData] = useState({
     OrderItem.jsx
                                   flatno: '',
     Products.jsx
                                   city: '',
     Signup.jsx
                                   pincode: '',
     # uhome.css
                                   state: '',
                                 });

⇔ Uhome.jsx

                                 const fee= 99;
     ₩ Uitem.jsx
                                 const { id } = useParams();
     Unavbar.jsx
                                 const navigate = useNavigate();
     ₩ Wishlist.jsx
    # App.css
                                 useEffect(() => {
    App.jsx
                                    axios.get(`http://localhost:4000/item/${id}`)
                                      .then((resp) => {
    # index.css
                                        setItem(resp.data);
    main.jsx
                                      })
   eslintrc.cjs
                                      .catch((error) => {
   .gitignore
                                        console.log("Failed to fetch item data:", error);
   index.html
                                 }, [id]);
> OUTLINE
                         231
> TIMELINE
                                    net handleChange - (e)
```

· Backend: In the backend, if we want to buy, then with the address and payment method, we process buying. If we need to add the product to the cart, then we add the product details along with the user Id to the cart collection.

#### Add Book:

```
CODEV2 [1] [2] [3] Codev2 > code > Frontend > src > Seller > ∰ Addbook.jsx > ...
                        1 import React, { useState } from 'react';
 ∨ codev2\code
                         2 import axios from 'axios';
  > Backend
                         3 import { useNavigate } from 'react-router-dom';

∨ Frontend

                         4 import Snavbar from './Snavbar';
   > node_modules
   > public
                        6 function Addbook() {
                               const [formData, setFormData] = useState({
    > Admin
                                  description: ",
                                description
title:
author:
    > Componenets

✓ Seller

                                genre: '',
     Addbook.jsx
                                price: '',
itemImage: null

⇔ Book.jsx

     # List.css
     const [loading, setLoading] = useState(false);
     Orders.jsx
                                 const navigate = useNavigate();
     Shome.jsx
                                 const user = JSON.parse(localStorage.getItem('user'));
     Slogin.jsx
     ⇔ Snavbar.jsx⇔ Ssignup.jsx
                               const handleChange = (e) => {
                                if (e.target.name === 'itemImage') {
    setFormData({ ...formData, [e.target.name]: e.target.files[0] });
    > User
    # App.css
                                  const { name, value } = e.target;
setFormData({ ...formData, [name]: value });
    App.jsx
    # index.css
    main.jsx
   eslintrc.cis
> OUTLINE
                                const handleSubmit = async (e) => {
> TIMELINE
                                e.preventDefault();
```

## My Orders:

```
✓ CODEV2
                        codev2 > code > Frontend > src > Seller > ∰ Orders.jsx > ∯ Orders > 🕪 calculateStatus
                         1 import React, { useState, useEffect } from 'react';

∨ codev2\code

                              import axios from 'axios';
  > Backend
                             import { useNavigate } from 'react-router-dom';

✓ Frontend

                              import Snavbar from './Snavbar';
   > node modules
   > public
                              function Orders() {
   ∨ src
                              const [orders, setOrders] = useState([]);
const navigate = useNavigate();
    > Admin
    > Componenets
                               useEffect(() => {
    ∨ Seller
     Addbook.jsx
                                 const user = JSON.parse(localStorage.getItem('user'));

⇔ Book.jsx

                                   console.log(user);
     # List.css
                                   axios.get(`http://localhost:4000/getsellerorders/${user.id}`)

⇔ Myproducts.jsx

                                       .then((response) => {
     Orders.jsx
                                         setOrders(response.data);

    ⇔ Shome.jsx

     Slogin.jsx
                                      .catch((error) => {
     Snavbar.jsx
                                       console.error('Error fetching orders: ', error);
    Ssignup.jsx
    > User
    # App.css
    App.jsx
                                // Function to calculate the status based on the delivery date
    # index.css
                         26
                                const calculateStatus = (Delivery) => {
    🐡 main.jsx
                                  const currentDate = new Date();
   eslintrc.cis
                                   const formattedDeliveryDate = new Date(Delivery);
> OUTLINE
> TIMELINE
                                   if (formattedDeliveryDate >= currentDate)
```

#### Book:

```
マ codev2 〉 codev2 〉 code > Frontend > src > Seller > ⇔ Book.jsx >...
      ∨ codev2\code
            > Backend

→ Frontend

                                                                                                                                              import axios from 'axios';
                                                                                                              4 import axios from 'axios ;
5 import React, { useEffect, useState } from 'react';
               > node_modules
               > public
                                                                                                                  6 import { useParams } from 'react-router-dom';
                                                                                                                    7 import Vnavbar from './Snavbar';
               ∨ src
                                                                                                                 8
9 const Book = () => {
    const [item, setItem]
                   > Admin
                   > Componenets
                                                                                                                                              const [item, setItem] = useState(null); // Initialize item as null

    ⇔ Addbook.jsx

                                                                                                                                                 const { id } = useParams();

    Book.jsx
    Book.jsx

                                                                                                                                       useEffect(() => {
   axios.get(`http://localhost:4000/item/${id}`)
                      # List.css
                      Consult Parishing of code | code | brontail | code | consult | code | consult | code | consult | code | cod
                        Shome, isx

1/ console.log(resp),

18 setItem(resp.data); // Set item to the fetched data (an object, not an array)
                       Sloain.isx
                                                                                                                                              })
.catch(() => {
    _consoie.iog("Did not get data");
});
         Snavbar jsx

Frontand

Signup jax

>> Ose, morning
                                                                                                                23 [id]); // include 'id' as a dependency to re-fetch data when the ID changes
               #_App.css
                   # index 23
                   main.jsx
                                                                                                               27 Vnevbar+>
              esiintro.cis
> outlinkere
                                                                                                                                                                  {item && (
> TIMETINE GODOORION
```

## Add Item:

```
CODEV2 [ ☐ C Odev2 > code > Backend > db > Seller > JS Additem.js >
∨ codev2\code
                   1 const mongoose = require('mongoose');
 ∨ Backend
                          const bookSchema = new mongoose.Schema({
  ∨ db
   ∨ Admin
                                 type: String,
   JS Admin.js
                                  required: true,
                               },
author: {
type: String,
 ∨ Seller
   JS Additem.js
  JS Sellers.js
                            },
genre: {
type: String,
required: tru
                                  required: true,

∨ Users

   JS myorders.js
  JS userschema.js 13
JS Wishlist.js 14
JS config.is 15
  required: true,
 {} package-lock.json
  {} package.json
                             userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
  JS server.js
                               userName:String,

∨ Frontend

  > node_modules
                     24 module.exports =mongoose.model('books',bookSchema)
  > public
```

## My orders:

```
∨ CODEV2 [‡ E‡ ひ 🗊
                        codev2 > code > Backend > db > Users > JS myorders.js > ...
                             const mongoose = require('mongoose');
 ∨ codev2\code
  ∨ Backend
                               const bookschema = new mongoose.Schema({
   ∨ db
                                   flatno:String,
    ∨ Admin
                                  pincode:String,
     JS Admin.js
                                 city:String,

∨ Seller

                                 state:String,
     JS Additem.js
                                 totalamount:String,
                                 seller:String,
sellerId:String,
     JS Sellers.js

∨ Users

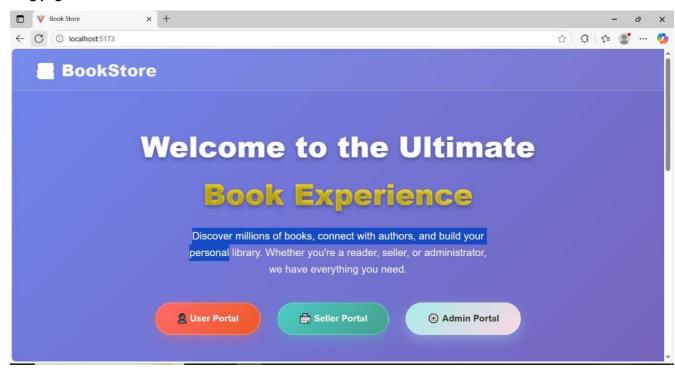
                                  booktitle:String,
     JS myorders.js
                                 bookauthor:String,
     Js userschema.js
                                 bookgenre:String,
     JS Wishlist.js
                                 itemImage:String,
   JS config.js
                                   description:String,
                                   userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
   > node_modules
                                   userName:String,
   > uploads
                                   BookingDate: {
   {} package-lock.json
                                       type: String, // Store dates as strings
   {} package.json
                                       default: () => new Date().toLocaleDateString('hi-IN') // Set the default value
   JS server.js

→ Frontend

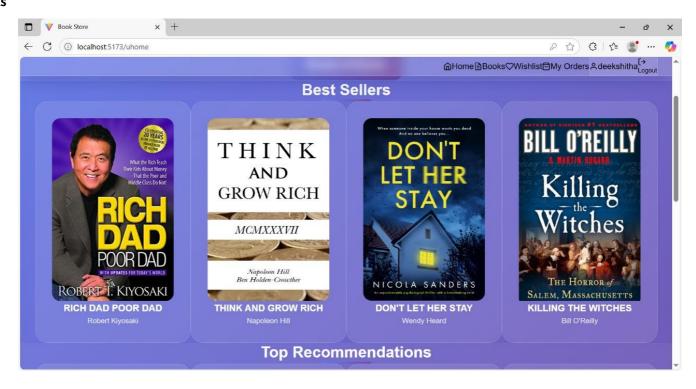
                                   Delivery: {
                                     type: String, // Store dates as strings
   > node_modules
                                     default: () => {
   > public
                                       // Set the default value to the current date + 7 days in "MM/DD/YYYY" format
   ∨ src
                                       const currentDate = new Date();
    > Admin
                                       currentDate.setDate(currentDate.getDate() + 7); // Add 7 days
    > Componenets
                                       const day = currentDate.getDate();
> OUTLINE
                                        const month = currentDate.getMonth() + 1; // Month is zero-based, so add 1
> TIMELINE
                                        const year = currentDate.getFullYear();
```

## **Demo UI images:**

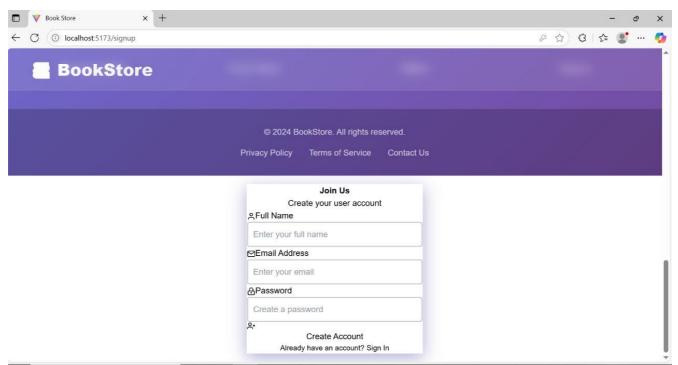
Landing page

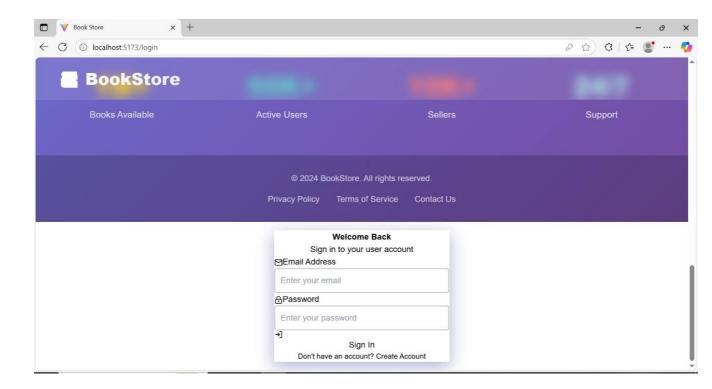


## books

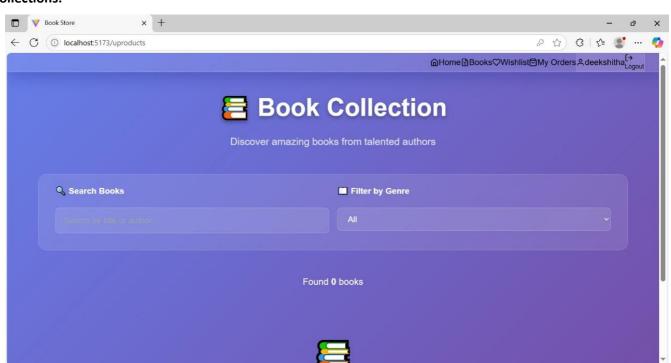


## **Authentication**

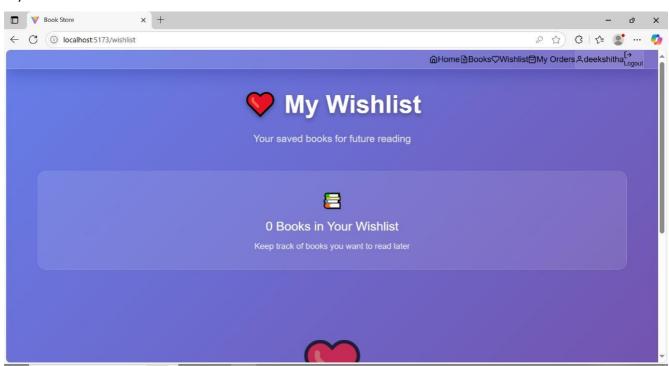




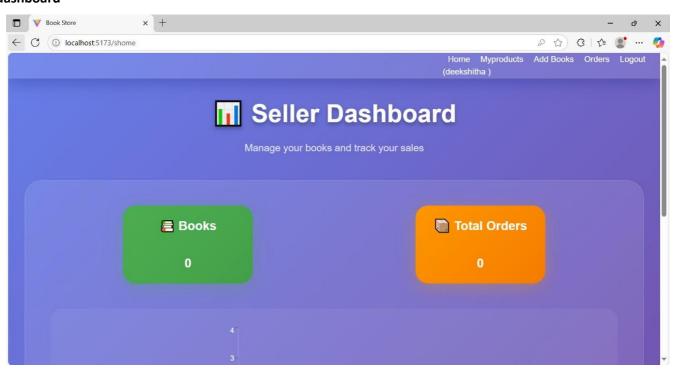
## **Book Collections:**



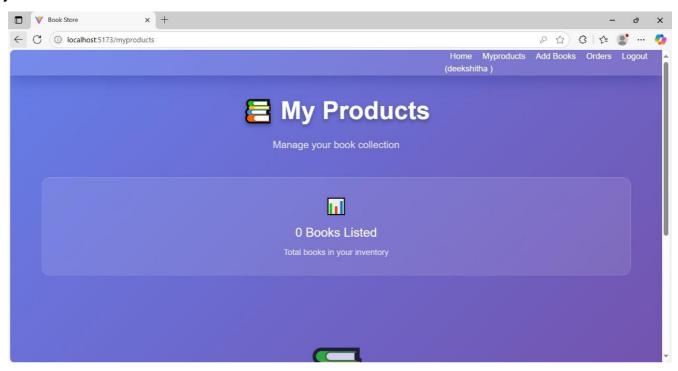
# Mywishlist:



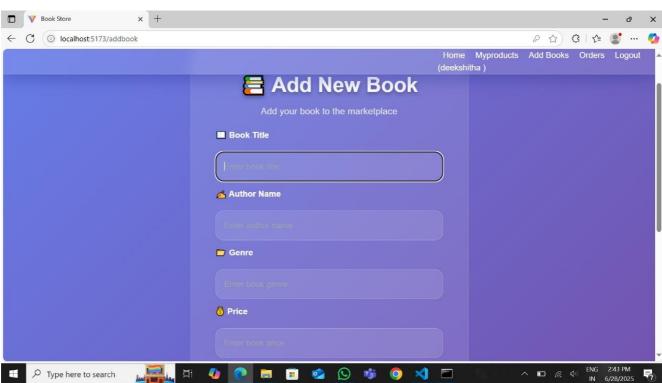
# Seller dashboard



# My Projects:

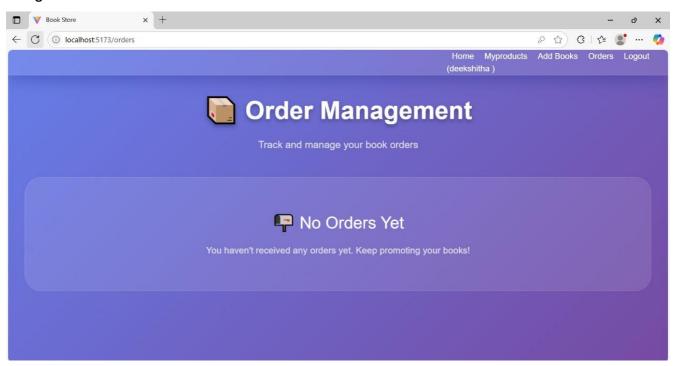


# **Add Books:**

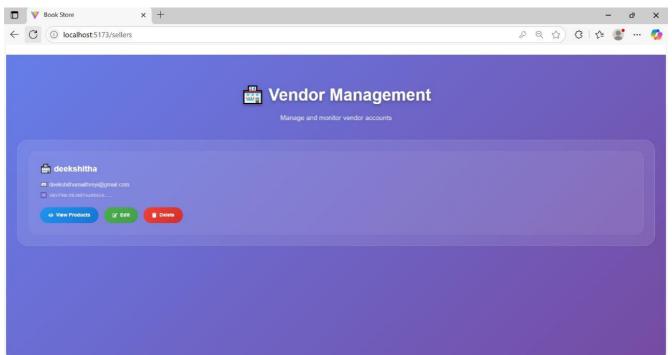


.

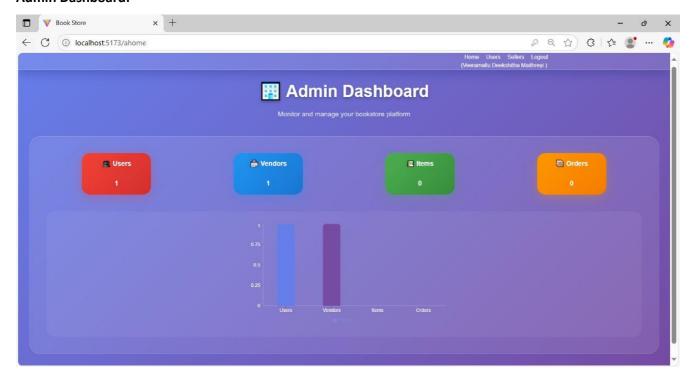
# **Order Management:**



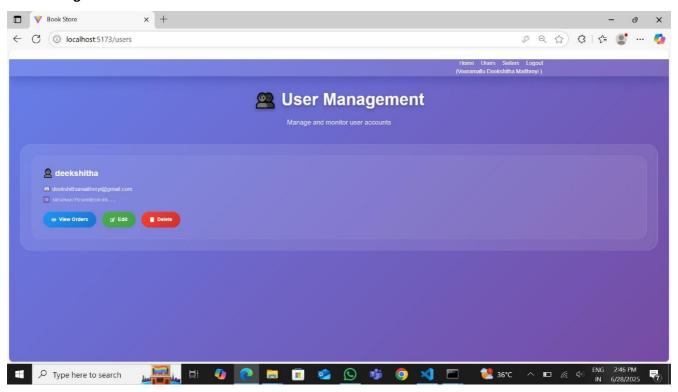
# **Vender Management:**



## **Admin Dashboard:**



# **User Management:**



For any further doubts or help, please consider the drive,

https://drive.google.com/drive/folders/13GfDZKAYzkzK9pwMStaaVUEgY3kNY9CO?usp=sharing

The demo of the app is available at:

https://drive.google.com/file/d/1LXRrIHydpSGso5S6fwyv3yYW7xmyQsm /view?usp=sharing

## **TESTING**

Manual testing done for all flows: register, login, cart, order, admin panel

# **Testing Scope**

## **Features and Functionalities to be Tested:**

- · User registration and login
- Product browsing and search
- Cart management
- · Order placement and order history
- Admin product/user/order management

## User Stories/Requirements to be Tested:

- USN-1: User registration
- USN-2: Registration confirmation
- USN-3: User login
- USN-4: Product browsing/search
- USN-5: Cart and order placement
- USN-6: Admin management

## **KNOWN ISSUES**

- book images might not appear if uploads folder is empty or missing
- Basic error messages need better UX design
- No support for password reset yet

#### **FUTURE ENHANCEMENTS**

- Integrate Stripe or Razorpay for payment gateway
- Allow book reviews and star ratings
- Enable email notifications on order status
- Multi-language support
- Add accessibility features and dark mode