

BOOKNEST: WHERE STORIES NESTLE

INTRODUCTION :

BookNest is a full-stack MERN-based online bookstore application that allows users to discover, browse, and purchase books conveniently. This system offers functionalities for users to browse books, sellers to manage book listings, and admins to oversee the platform's smooth operation. The platform ensures a seamless and interactive experience for all user roles, supporting modern book shopping experiences.

KEY FEATURES :

USER FEATURES :

- Secure registration and login using email and password.
- Profile creation and management.
- Browse books by categories, authors, and ratings.
- Add books to cart and complete purchases.
- View order history and provide feedback.

SELLER FEATURES :

- Seller account registration and login.
- Manage book inventory: add, update, or delete books.
- Monitor orders placed by customers and fulfill them.

ADMIN FEATURES :

- Approve or reject seller registrations.
- Manage user and seller accounts.
- Monitor orders, inventory, and user activity.

DESCRIPTION :

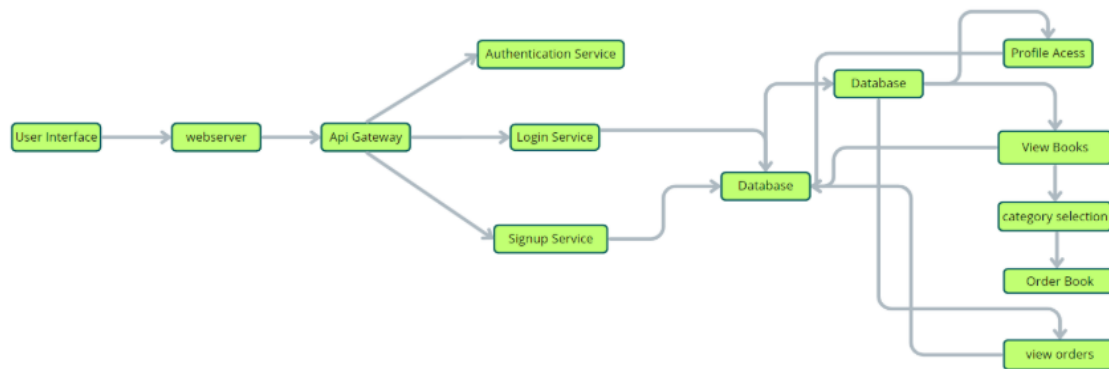
BookNest provides a user-friendly platform where book enthusiasts can register, browse books by various filters, and place orders. Sellers are provided with an interface to list and manage their book inventory. Admins have full control to oversee platform activity. Built using React.js for the frontend and Node.js with Express.js for the backend, the application uses MongoDB for efficient and scalable data storage.

SCENARIO-BASED CASE STUDY :

1. Sarah, an avid reader, registers on BookNest using her email.
2. She browses books by genre and ratings, adding her favorites to the cart.
3. Sarah completes the purchase and receives order confirmation.

4. A seller fulfills Sarah's order and updates its status.
5. The admin monitors new sellers and approves their registrations.

TECHNICAL ARCHITECTURE :



- **Frontend:** React.js, Bootstrap, Vite, Axios
- **Backend:** Node.js, Express.js REST API
- **Database:** MongoDB with Mongoose ODM
- **Authentication:** JWT for user sessions, bcrypt for password encryption

FRONTEND TECHNOLOGIES :

- React.js, Bootstrap, Axios, Vite

BACKEND FRAMEWORK :

- Node.js, Express.js, RESTful APIs

DATABASE AND AUTHENTICATION :

- MongoDB for data storage
- Mongoose for schema modelling
- JWT and bcrypt for authentication and password security

ADMIN PANEL & GOVERNANCE :

- Admin Dashboard for full platform control
- Role-based Access Control (Admin, Seller, User)

SCALABILITY AND PERFORMANCE :

- Horizontal scalability using MongoDB

TIME MANAGEMENT AND SCHEDULING :

- Real-time updates for book availability and order tracking

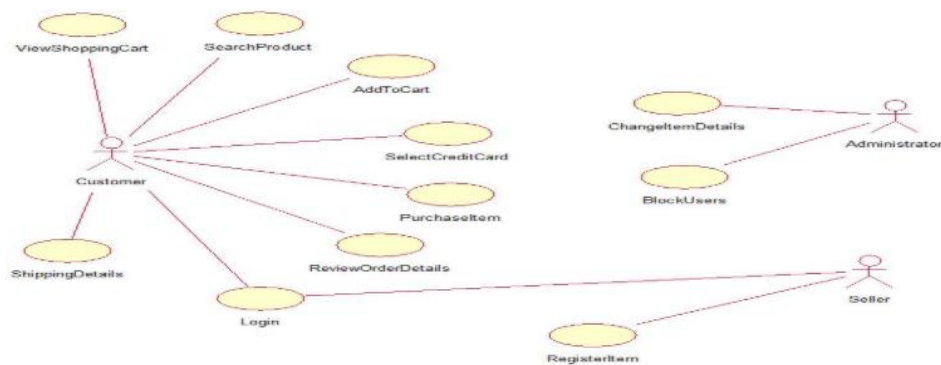
SECURITY FEATURES :

- JWT Authentication for secure access
- bcrypt encryption for password safety
- HTTPS encryption in production environments

NOTIFICATIONS AND REMINDERS :

- Order confirmation and updates via in-app notifications

ER DIAGRAM :



- **Users:** Basic details, role (user/seller/admin), authentication data
- **Books:** Book details, seller reference, availability
- **Orders:** User, book, order status
- **Sellers:** Seller profile, listed books
- **Admins:** Full platform access and controls

PRE-REQUISITES :

- Node.js, npm
- MongoDB Atlas
- React.js

SETUP AND INSTALLATION INSTRUCTIONS :

- Clone the repository
Download the project files from GitHub or clone the repository using Git.
- Install Frontend:

cd frontend

npm install

npm run dev

- Install Backend:

cd backend

npm install

npm start

- Configure MongoDB connection and JWT secret in .env file

INSTALL DEPENDENCIES:

- Navigate to the frontend and backend directories and install all required dependencies for both parts of the application.

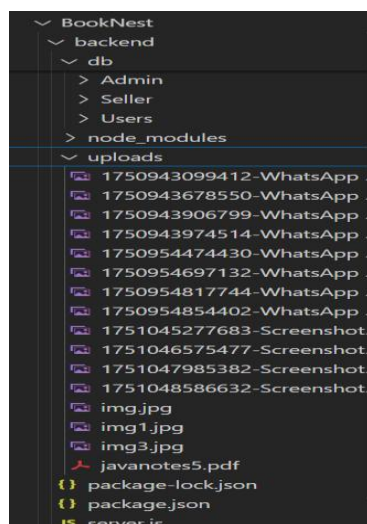
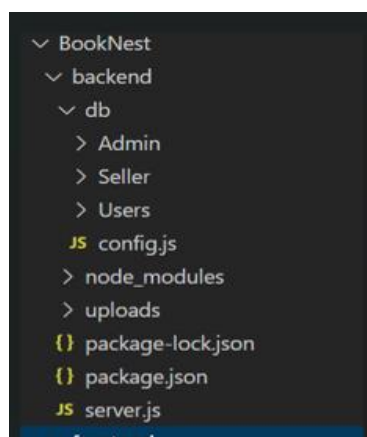
- Frontend:

- Navigate to the frontend directory and run npm install.

- Backend:

- Navigate to the backend directory and run npm install.

PROJECT FOLDER STRUCTURE :



PROJECT FLOW :

- **User:** Register → Browse Books → Cart → Order → Track Order
- **Seller:** Register → List Books → Fulfill Orders
- **Admin:** Manage Sellers → Manage Books → Platform Oversight

Frontend Part:

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.

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 **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.

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.

Application Flow

Start: Users open the BookEase app to explore a vast collection of books.

Home Page: Users land on the home page, which provides an overview of the book store's offerings. From here, they can navigate to various sections of the app.

Access Profile: Users have the option to access their profiles, allowing them to view or update personal information, preferences, and order history.

Book Selection: After accessing their profiles, users proceed to browse and select books to purchase. The app presents a list of available books, along with details such as title, author, genre, and price.

Book Purchase: Users navigate through the available book options and specify the quantity of each book they wish to purchase. They can also choose additional options such as e-book format or special editions.

View Orders: Users have the option to view their current and past orders. This section provides details about ordered books, order status, and payment history.

Order Confirmation: For new purchases, users can initiate the ordering process. This involves selecting books, specifying quantities, confirming the order, and receiving an order confirmation.

End: The flow concludes as users have completed their desired actions within the BookEase app

SETUP & CONFIGURATION

Setting up BookNest involves configuring both the frontend using React.js and the backend using Node.js, Express.js, and MongoDB.

FRONTEND CONFIGURATION

- Clone the project repository using Git.
- Navigate to the frontend folder.
- Install all required dependencies using `npm install`. This includes React.js, React Router, Bootstrap, Axios, and Vite.
- Start the React development server using `npm run dev`.
- The frontend runs on `http://localhost:5173`.

BACKEND CONFIGURATION

- Navigate to the backend folder of the project.
- Install required backend dependencies using `npm install`. This includes Express.js.
- Set up MongoDB connection in a `.env` file using a local MongoDB connection string or MongoDB Atlas.
- Define environment variables like `PORT`

- Start the backend server using `node server.js`.
- The backend server runs on `http://localhost:4000`.

ACCESSING THE APPLICATION

- Frontend accessible at `http://localhost:5173`
- Backend API accessible at `http://localhost:4000/api/`
- Both frontend and backend must run simultaneously for full application functionality.

DATABASE

1. Configure MongoDB:

- Install **Mongoose** to interact with MongoDB.
- Create a **database connection** using Mongoose inside the backend configuration.
- Design **Schemas & Models** to structure data for:
 - Users
 - Sellers
 - Books
 - Orders

2. Connect Database to Backend:

Make sure the MongoDB connection is established before any API actions are performed in the backend.

3. Configure

Schemas are created in the backend `/db` folder:

- **User** includes user details like name, email, password, and role (user, seller, admin).
- **Seller** contains seller details and listed books.
- **Book** contains title, author, price, category, stock quantity.
- **Order** connects users and books with order date, status, and quantity.

FINAL CONFIGURATION & RUNNING THE APP

To run the **BookNest** application properly, both the frontend (React.js) and backend (Node.js, Express.js, MongoDB) servers must be running at the same time.

RUN BOTH SERVERS

- The **frontend** runs on React.js with Vite, and the **backend** runs on Node.js with Express.js.
- You can run both servers using two terminal windows or use a tool like **concurrently** to start both together from the root folder.
- In the root `package.json`, a script is created to run both servers at once. This ensures that both the frontend (on port 5173) and backend (on port 4000) run simultaneously.

- After setup, running the command `npm start` from the root directory will start both servers.

VERIFYING THE APP

- **Frontend Verification:** Open the browser and visit `http://localhost:5173` to check if the React application loads with book listings and user features.
- **Backend Verification:** Use API testing tools like **Postman** to test backend endpoints for user registration, login, book management, and order placement.

FOLDER STRUCTURE

The BookNest project is organized into two main folders:

- **Frontend Folder:** Contains the React.js application code.
- **Backend Folder:** Contains the Node.js and Express backend code, including API routes, database models, and server configuration.
- **PROJECT ROOT STRUCTURE**
- **Create the Main Folders:**
In your project's root directory, create two main folders: `frontend` and `backend`
- BookNest/
 - ├── frontend/
 - └── backend/

PROJECT FLOW : Project Demo :

- Before diving into development, you can view a demo of the project to understand its functionality and user interactions.

- Project Demo Video :

<https://drive.google.com/file/d/1oelpgm2CgiktENJFLRhHZ5HcH2j5k2nA/view?usp=sharing>

MILESTONE 1:PROJECT SETUP AND CONFIGURATION :

- Setting up a structured environment is crucial for any application. By organising the project into separate folders for the frontend and backend, we ensure that the codebase is manageable and scalable.

PROJECT FOLDERS:

- **Frontend Folder:** Contains all code related to the user interface, written in JavaScript using frameworks and libraries like React, Material UI, and Bootstrap. This setup helps maintain a clear boundary between UI logic and server logic.
- **Backend Folder:** Manages the server, API routes, and database interactions, typically handled through Node.js and Express.js. Using separate folders enables a modular structure, allowing changes in one area without affecting the other.

LIBRARY AND TOOL INSTALLATION: Backend Libraries:

- **Node.js:** Provides a runtime environment to run JavaScript code on the server side.
- **MongoDB:** A NoSQL database, perfect for flexible and schema-less data storage, ideal for applications needing frequent updates and various data types.
- . ● **Frontend Libraries:**
 - **React.js:** Manages component-based UI creation, providing the flexibility to build reusable UI components.
 - **Axios:** Facilitates easy HTTP requests, allowing the frontend to communicate with the backend effectively.

MILESTONE 2: BACKEND DEVELOPMENT : The backend forms the core logic and data management for the project. A well-structured backend ensures efficient data handling, security, and scalability.

BACKEND DEVELOPMENT

Express Server Setup:

The backend for BookNest is developed using **Node.js** and **Express.js**. The server is initialized in a central file (server.js) located inside the backend folder. A .env file is used to define environment variables such as the port number and MongoDB connection string, allowing global access to configuration settings.

The Express server is configured with:

- **cors** to handle cross-origin requests
- **body-parser** (now part of Express) to handle incoming request data
- Connection to MongoDB using **Mongoose**

User Authentication:

User authentication is implemented using **JWT (JSON Web Tokens)**:

- Routes are created for **user registration, login, and logout**.
- **bcryptjs** is used to hash user passwords before storing them in the database.
- Middleware is used to protect private routes, ensuring only authenticated users can access certain endpoints.
- Token-based authentication maintains secure sessions without the need for server-side sessions.

API Routes:

- Routes are modularized by functionality:
 - **User Routes:** for user account actions
 - **Seller Routes:** for managing book listings and inventory
 - **Admin Routes:** for overall platform management
 - **Order Routes:** for handling user orders and purchases
- Each route contains handlers for HTTP methods like GET, POST, PUT, DELETE.
- Controllers are used to separate route logic from database operations, following clean coding practices.

Data Models:

- **Mongoose Schemas** are defined for:
 - **Users:** Storing personal details, authentication info, and user roles (user/seller/admin)
 - **Books:** Book details like title, author, genre, price, stock quantity
 - **Orders:** Connecting users to books with order status and payment information
 - **Sellers:** Information about sellers and their listed books
- **Mongoose Models** are created to perform CRUD operations on each schema.

Error Handling:

- A centralized **error-handling middleware** is used to catch errors during API execution.
- Appropriate **HTTP status codes** and **descriptive error messages** are returned to the frontend to maintain a smooth user experience and easier debugging.

DATABASE

1. Configure MongoDB

In the **BookNest** application, **MongoDB** is used as the database to store all essential data like users, sellers, books, and orders.

- **Mongoose** is installed to simplify interaction between Node.js and MongoDB.
- A **database connection** is created in the backend (config.js) using Mongoose.
- Multiple **schemas and models** are created for each major entity in the system:
 - **User Schema** to store user credentials, personal information, and roles.
 - **Seller Schema** to manage seller profiles and their book listings.
 - **Book Schema** to store book details such as title, author, genre, price, and stock.
 - **Order Schema** to track user orders, statuses, and payment details

2. Connect Database to Backend

- Before any backend actions are performed, the server establishes a connection to the **MongoDB database** using Mongoose.
- The database connection is initialized as soon as the Express server starts, ensuring smooth communication between the backend APIs and the database.
- The connection setup ensures that **BookNest** APIs only proceed when a successful connection to MongoDB is established, maintaining data integrity.

3. Configure Schemas

- Schemas are designed by referring to the **ER Diagram** of the BookNest project.
- The **User, Seller, Book, and Order Schemas** define how data is structured inside MongoDB.
- These schemas help enforce a clear and consistent data format, making queries and updates efficient.
- **Mongoose Models** built from these schemas are used in the backend routes to perform all CRUD (Create, Read, Update, Delete) operations.

In summary, MongoDB in **BookNest** serves as a flexible, schema-based storage solution that supports the project's multi-role architecture and transactional processes.

MILESTONE 5: PROJECT IMPLEMENTATION AND TESTING

VERIFY FUNCTIONALITY

After completing development, it is important to thoroughly test the **BookNest** application to identify any bugs or functional issues.

- Running the complete application ensures that the **frontend (React)**, **backend (Node.js, Express.js)**, and **database (MongoDB)** are properly integrated and functioning together.
- Testing key user flows like:
 - **User registration and login**
 - **Browsing and filtering books**
 - **Adding books to cart and placing orders**
 - **Seller book management and order handling**
 - **Admin approving sellers and managing users**
- This confirms that all features of the system work as expected across different user roles.

USER INTERFACE ELEMENTS

Testing the user interface involves:

- Verifying the **design and responsiveness** of key pages including:
 - Landing Page
 - Login and Registration Pages

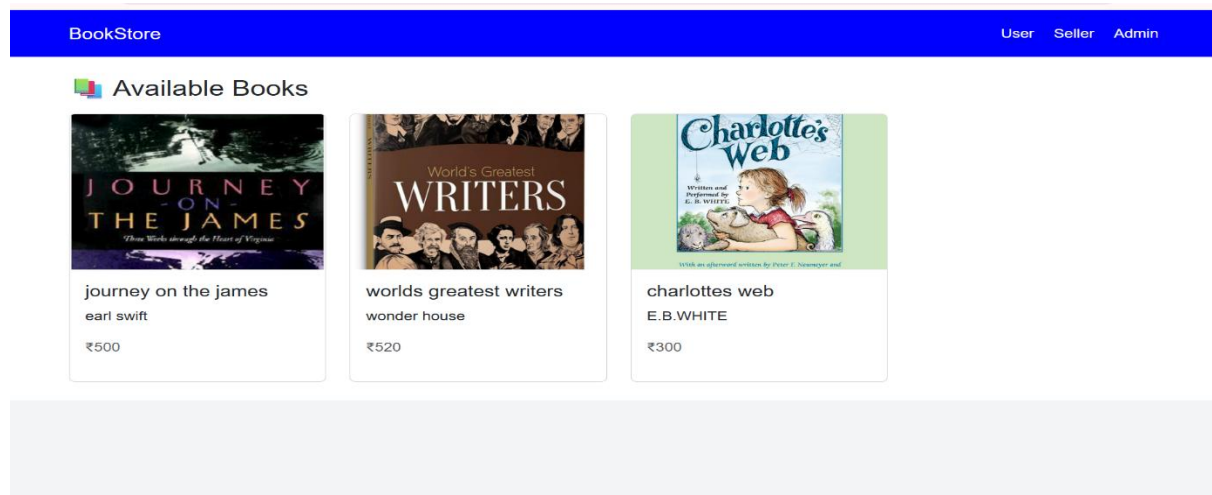
- User Dashboard (Orders and Cart)
- Seller Dashboard (Book Listings)
- Admin Dashboard (User and Seller Management)
- Ensuring a **smooth user experience** across multiple devices like desktops, tablets, and mobiles.
- Checking that navigation, forms, and action buttons work intuitively for all users (users, sellers, and admins).

FINAL DEPLOYMENT

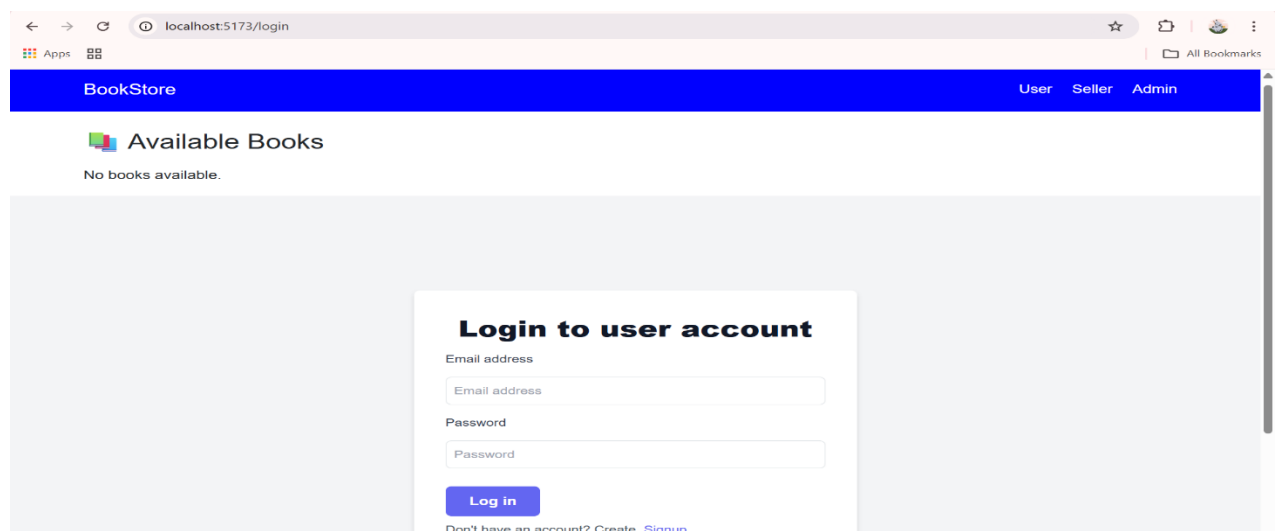
Once all tests are passed and functionality is verified:

- The **MongoDB database** can be hosted on **MongoDB Atlas**.
- This makes **BookNest** accessible to end-users with a stable, production-ready environment.

LANDING PAGE :



LOGIN PAGE :



REGISTRATION PAGE :

Signup

Name

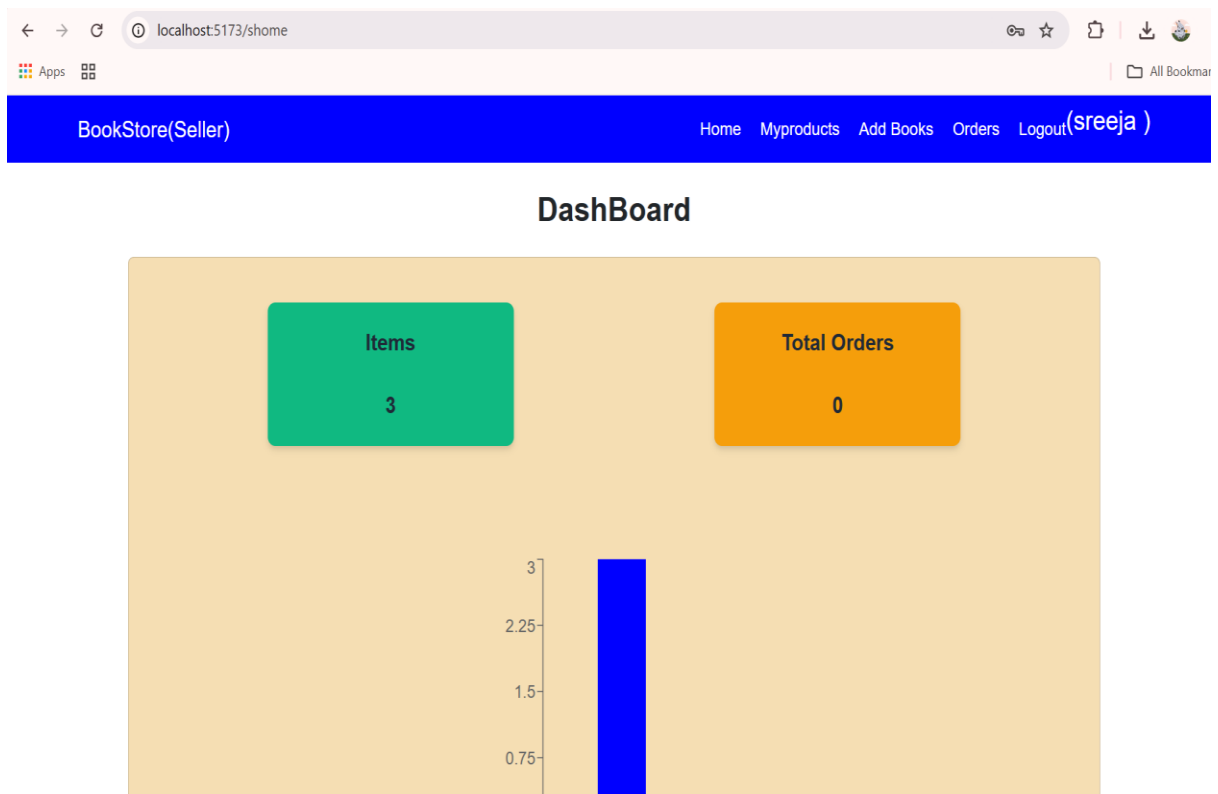
Email address

Password

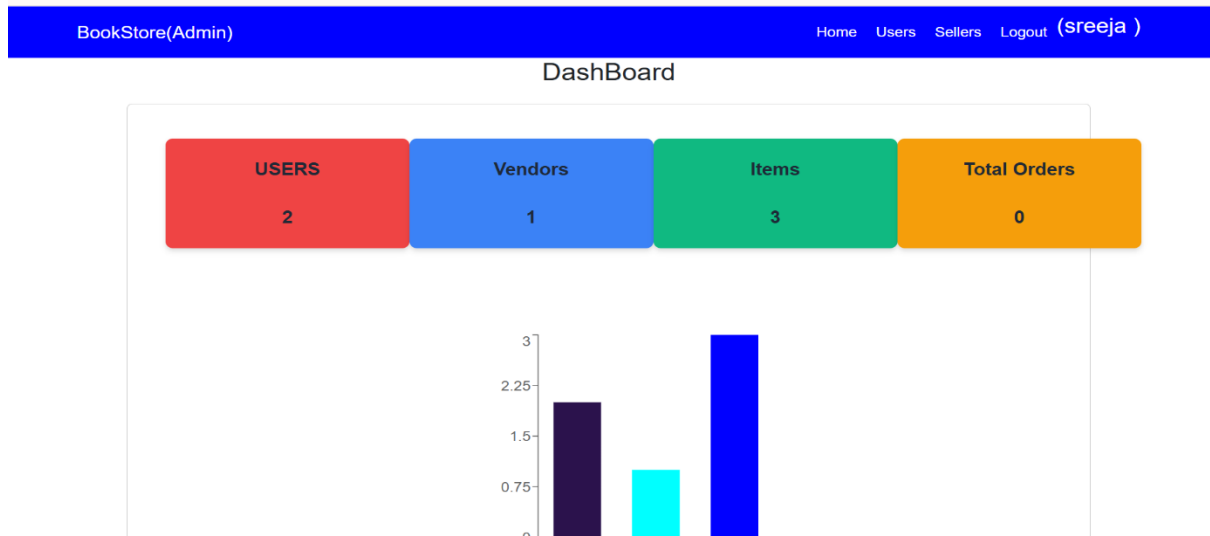
Signup

[Already have an account Login](#)

Seller PAGE :



ADMIN PAGE :



BOOK LISTING PAGE :

localhost:5173/uproducts

BookStore Home Books Wishlist My orders Logout(sreeja)

Books List



journey on the james
Author: earl swift
Genre: james
Price: \$500

worlds greatest writers
Author: wonder house
Genre: gons
Price: \$520

charlottes web
Author: E.B.WHITE
Genre: unabridged
Price: \$300

USERS HISTORY PAGE :

Users

sl/no	UserId	User name	Email	Operation
1	685e31366860f5b0aa735eec	sai	sai@gmail.com	  view
2	685e33736860f5b0aa735f24	sreeja	sreejasre689@gmail.com	  view

CONCLUSION :

BookNest is a scalable, responsive, and secure online bookstore application built on the MERN stack. It provides tailored functionalities for users, sellers, and admins while ensuring a smooth and interactive book shopping experience.