

## **Documentation**

Project Title: **BOOKNEST: WHERE STORIES NESTLE**

Team size: 4

Team Leader: Kiran Sai Barugala

Team member: Alapati Kavya Sri

Team member: Burra Kireeti

Team member: Katevarapu Prasanna

### **Message to Nasscom Team From our side**

I am writing to express my heartfelt gratitude to the entire NASSCOM team for providing us with the incredible opportunity to participate in the internship program. Your valuable training and guidance have equipped us with the essential knowledge in machine learning and artificial intelligence, which has been instrumental in the successful completion of our project.

The insights and skills we gained through this experience have not only broadened our technical understanding but have also given us the confidence to apply these concepts in real-world scenarios. This internship has been a transformative experience for all of us, and we are incredibly grateful for the support and encouragement provided by your team.

Thank you once again for believing in us and for giving us the tools to succeed. We are excited to carry forward everything we've learned and apply it in our future

endeavours.

## **Project Overview**

### **Purpose**

The primary purpose of this project is to develop a comprehensive and efficient online shopping book store that caters to both book buyers and sellers. In today's digital era, online platforms have become the go-to solution for shopping due to their convenience, accessibility, and variety. This project aims to harness these advantages by creating a dedicated platform where book enthusiasts can seamlessly browse through an extensive collection of books, search for specific titles or authors, and make purchases from the comfort of their homes. The platform will offer a wide range of genres, including fiction, non-fiction, educational materials, self-help, biographies, and more, ensuring that it appeals to a diverse audience with varying reading preferences.

This project aims to create a comprehensive, secure, and scalable online book store that offers a seamless shopping experience for customers and an efficient sales platform for sellers. By focusing on usability, security, and performance, the project aims to establish a trusted and widely-used marketplace that promotes accessibility and convenience for book enthusiasts worldwide.

### **Features and Goals**

The core goals of this project revolve around creating a functional, scalable, and user-friendly online book store that meets the needs of both buyers and sellers. One

of the primary goals is to develop a seamless and intuitive user interface (UI) that prioritizes ease of navigation, aesthetic appeal, and functionality. The platform will feature a clean design, clear categorization, and efficient search and filter options, allowing customers to find their desired books quickly and effortlessly.

To enhance the overall shopping experience, the project aims to incorporate key e-commerce features such as a shopping cart, checkout system, and secure payment gateway. The shopping cart will allow customers to add, remove, and modify book quantities before finalizing their orders. Integration with trusted payment processors will guarantee secure transactions and protect customer data through encryption protocols. Customers will also have access to their order history, making it easy to reorder favourite books or reference past purchases.

## **Backend**

### **1. admin.js**

The `admin.js` file handles the admin functionalities of the bookstore platform, providing routes for managing users, sellers, books, and orders. Admins can log in, log out, and manage authentication cookies. The file provides CRUD operations, including adding, fetching, and deleting users, sellers, and books. It also allows admins to view and modify orders. The routes ensure that only logged-in admins can access or modify data, enhancing security.

### **2. book.js**

The `book.js` file defines the book-related routes, allowing users to browse and filter books. The file supports query parameters for filtering by title, author, genre, description, and price range. It also allows pagination by defining start and limit

parameters. The GET route retrieves books based on the applied filters.

### **3. seller.js**

The seller.js file manages functionalities specific to book sellers. Sellers can sign up, log in, and log out. The authentication uses cookies for session management. Sellers can add, modify, and delete books. The file also contains a route for fetching seller statistics, including the number of books, orders, and total sales.

### **4. user.js**

It includes routes for signup, login, logout, and order placement. During signup, user passwords are hashed using **CryptoJS** with salt for security. The login route verifies credentials and issues an authentication cookie upon successful login. Users can place orders, which include book details, seller information, and shipping addresses.

## **Database**

The backend uses Mongoose to connect to the MongoDB database, with schemas defining the structure of documents. The mongoose-setup.js file defines the schemas and models for the database, including User, Seller, Admin, Book, Order, and WishlistItem.

### **About Schemas**

The admin.js file manages administrative tasks such as logging in, logging out, and performing CRUD operations on users, sellers, books, and orders. It ensures only authenticated admins can access sensitive data, using authentication cookies for

security. The `book.js` file handles book browsing, enabling users to filter by title, author, genre, and price range. It also supports pagination and excludes unnecessary fields like images and internal IDs from responses for efficiency. The `seller.js` file manages seller-specific operations, allowing them to sign up, log in, add, modify, and delete books, and view statistics, including total books, orders, and sales. It uses `mongoose.ObjectId` for proper association with sellers' records. The `user.js` file handles user-related functionalities, including signup, login, logout, and order placement.

## **Authentication**

The `index.js` file acts as the entry point, initializing the server, connecting middleware, and defining routes for admins, users, sellers, and books. It uses `dotenv` for environment variables, `cors` for cross-origin requests, `body-parser` for JSON parsing, and `cookie-parser` for managing authentication cookies. It also sets up routes for serving book images and handles errors gracefully.

The `auth.js` file handles authentication logic, including functions for validating credentials, generating authentication cookies, and extracting credentials from cookies. It uses `CryptoJS` for password hashing and an `LRUCache` for caching credentials to optimize performance. The `middlewares.js` file defines two middleware functions: `setAuthenticatedRequestProperty`, which checks authentication cookies to determine if a user is logged in, and `blockIfNotLoggedIn`, which blocks access to protected routes for unauthenticated users.

Overall, the project is well-structured, with clear separation of concerns, proper authentication, and efficient data management, making it robust and production-

ready.

## Setup Instructions

### Prerequisites

- HTML
- CSS
- React js
- Axios
- Node js
- Express js
- MongoDB

### Installation

**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**

## Demo Screenshots

### User Login page

---

BookNest 

User

Seller

Admin

Login

Email

Enter your email

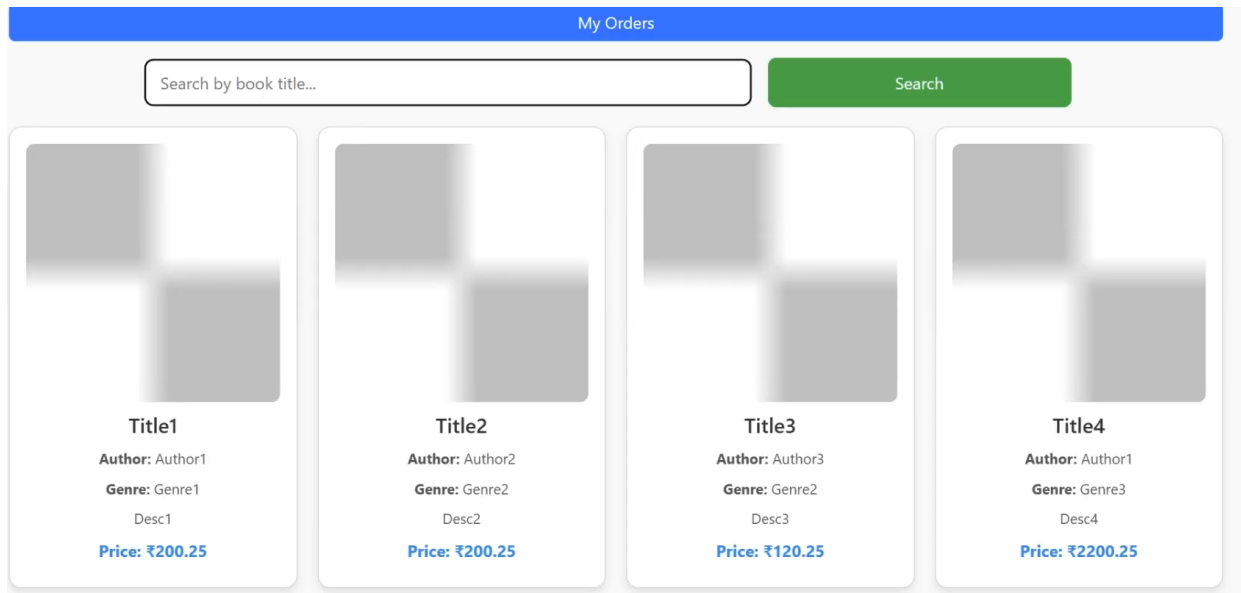
Password

Enter your password

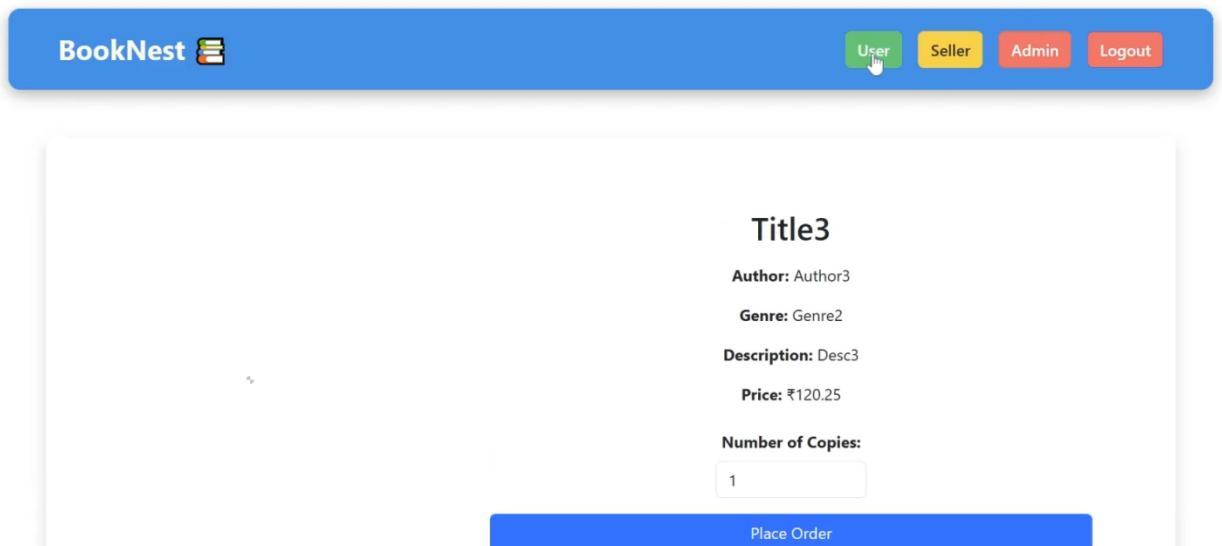
Login

[Sign Up](#)

## Users - My Orders page




## After Searching the specific book





## Seller Login Page

BookNest 

UserSellerAdmin

### Seller Login


Email / Username

Password

Login

Don't have an account? [Signup](#)

## Seller Dashboard

BookNest 


UserSellerAdminLogout

### Welcome, Seller!


Total Books Listed 3	Total Orders 0	Total Revenue \$0
-------------------------	-------------------	----------------------

Add New BookView My BooksView Orders

## Seller Management

BookNest 			User	Seller	Admin	Logout
Seller Management						
ID	Name	Email				
67e2d92151805bb89c62dc25	Seller1	seller1@gmail.com				
67e2d92151805bb89c62dc27	Seller2	seller2@gmail.com				

## Admin Login Page

BookNest 			User	Seller	Admin
----------------------------------------------------------------------------------------------	--	--	------	--------	-------

### Admin Login


Email

Password

Login

Don't have an account? [Signup](#)

# Admin Dashboard

BookNest 

User

Seller

Admin

Logout


Admin Dashboard

Total Users  
2

Total Sellers  
2

Total Orders  
0

# Admin - All Users Page

BookNest 

User

Seller


Admin

Logout

All Users

User ID	Name	Email	Delete
67e2d92151805bb89c62dc21	User1	user1@gmail.com	<div>Delete</div>
67e2d92151805bb89c62dc23	User2	user2@gmail.com	<div>Delete</div>

## Order Management Page

BookNest 		User	Seller	Admin	Logout
Order Management					
Order ID	User	Book	Total Amount		Status



## Known Issues

### 1. Authentication Issues

When the authentication cookie is invalid or tampered with, the server may return a vague error message.

### 2. Error Handling for MongoDB Operations

Some MongoDB operations, such as querying with invalid IDs, can cause the server to crash.

### 3. Inefficient Query Performance

Some MongoDB queries, especially for filtering books and orders, lack proper

indexing.

#### **4. Hardcoded Port in Server**

The server uses the port specified in `.env`, but if the variable is missing, it defaults to an undefined port.

#### **Future Enhancements**

Future enhancements for the online bookstore project include improving the user experience by adding advanced book search filters, book reviews, ratings, and personalized recommendations powered by collaborative filtering or AI models. Enhancing seller features with detailed analytics dashboards, real-time notifications, and bulk book uploads will improve efficiency. Security enhancements such as role-based access control (RBAC), two-factor authentication (2FA), and email verification will strengthen the platform's security. Finally, adding features like social media integration, subscription models, and AI-powered book suggestions will enhance engagement and retention, making the platform more competitive and user-friendly.