**PROJECT REPORT**

**BookNest: Where Stories Nestle**

## **Date : 27-06-2025**

Project Name : BookNest

Team ID: LTVIP2025TMID53034

Team Members:

Team Leader: Inkollu Loukika

Team member: Gunde Nithin Roy

Team member: Guntupalli Srivalli

Team member: Guntupalli Venkata Sal Jyoshnavi

**Introduction**

The rise of digital platforms has transformed how books are sold and bought online. However, challenges still exist for both book sellers and readers in discovering relevant listings, ensuring transaction safety, and managing inventories or orders. BookNest is a user-centric e-commerce platform built to bridge this gap by simplifying book buying, selling, and browsing.

Built using the MERN Stack—MongoDB, Express.js, React.js, Node.js—BookNest delivers a dynamic and scalable web experience for independent sellers and avid readers alike.

**Project Overview**

* BookNest is a multi-role e-commerce platform tailored to buying and selling books online. With dedicated features for buyers, sellers, and administrators, the platform supports the full book commerce lifecycle from listing to delivery.

**User Roles and Key Functions**

**Sellers:** Upload book listings (title, author, price, cover), manage inventory, and view order statuses.

**Admins:** Approve seller accounts, monitor site activity, manage disputes and control listings.

**Core Features**

Real-time book browsing and search filters (genre, author, price)

Buyer & seller registration/login using JWT authentication

Role-based dashboards for each user category

Order placement, cart, and cancellation features

Admin approval of seller accounts

Notifications for order updates

Responsive UI for desktop/mobile/tablet

**Purpose**

The goal of BookNest is to empower local sellers and offer readers an organized, secure, and seamless book shopping experience.

**Additional goals include:**

Streamline book buying with filters and instant checkout

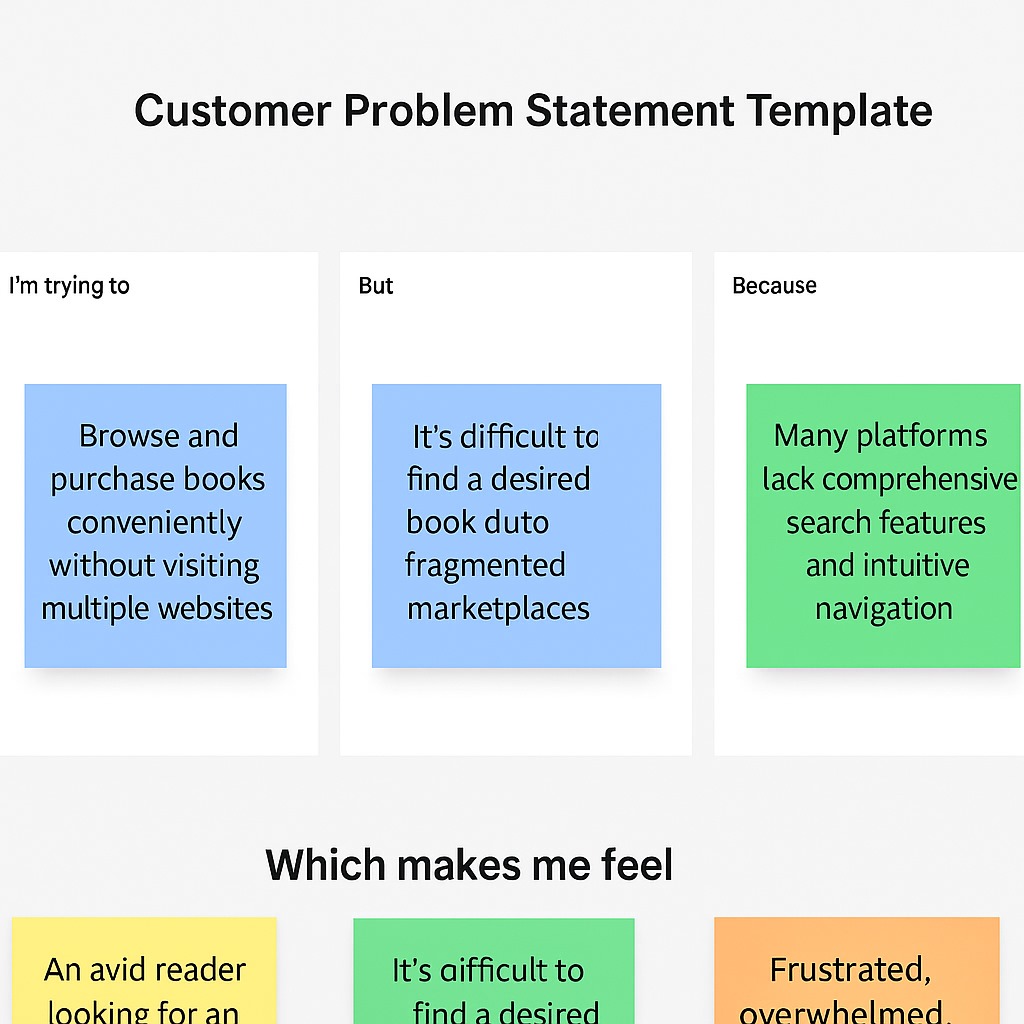
Provide user authentication and secure transactions

Enhance discoverability of books for buyers

Manage book inventories and orders for sellers

Equip admins with tools for oversight and moderation

**Ideation Phase**

****

**Problem Statement:**

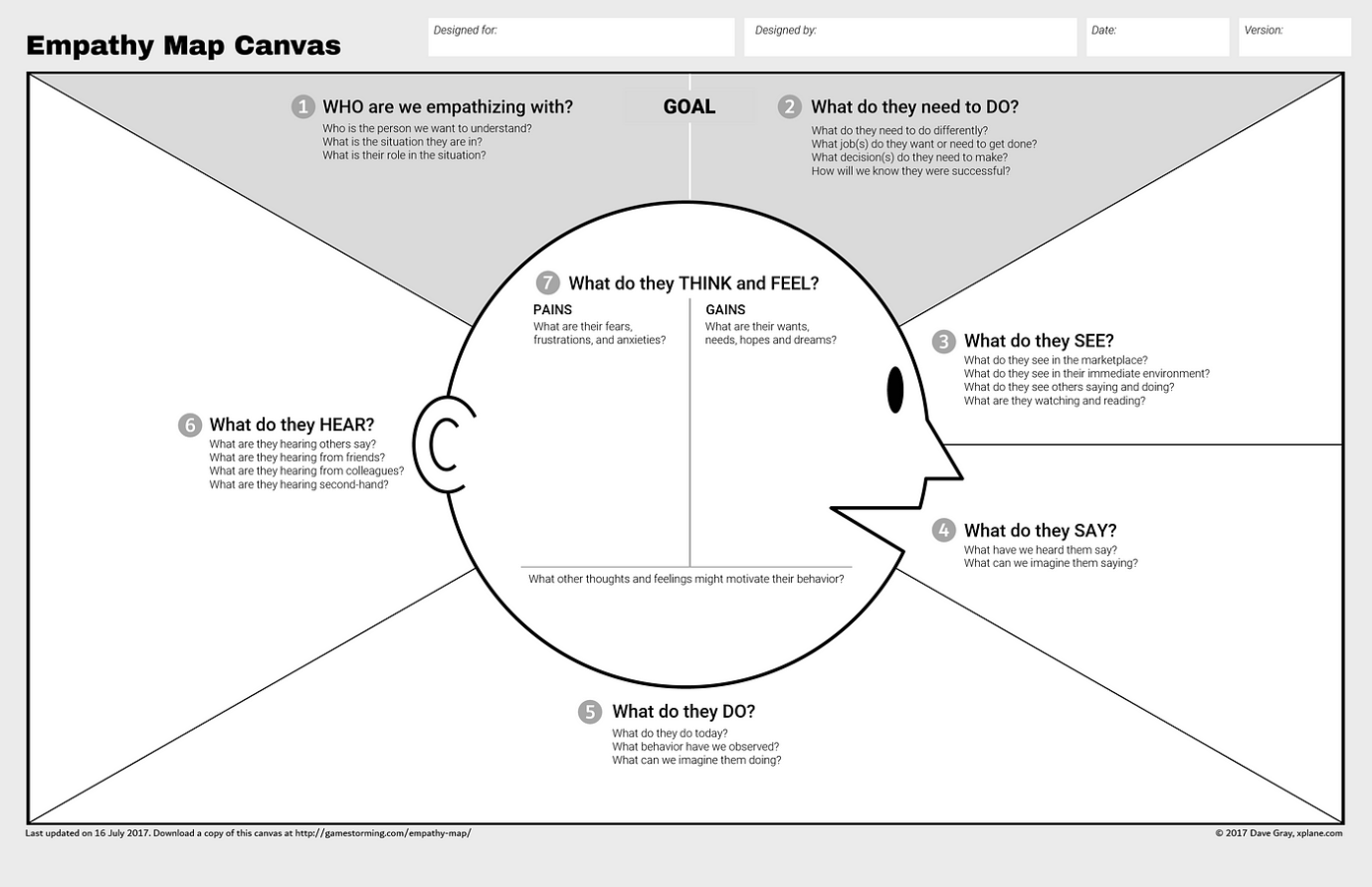
1. Book lovers find it hard to discover niche or second-hand books in crowded marketplaces.
2. Independent sellers lack a dedicated platform for organized book listings.
3. Manual or disorganized order management leads to delayed deliveries and bad experiences.

**Proposed Solution:**

1. Develop a dedicated book-selling platform using the MERN stack.
2. Features for:
3. Buyers: Register, browse, search, order, track, and cancel.
4. Sellers: Upload books, manage listings, check orders.
5. Admins: Approve sellers, manage users, resolve issues.
6. Add real-time UI updates, role-based dashboards, and responsive design.

**Empathy Map Canvas**

This empathy map is based on the primary user: **the customer:**

****

### ****Brainstorming & Idea Prioritization****

### **Objective:**

### **The objective of this phase was to explore, design, and prioritize innovative ideas to build a seamless, user-friendly online bookstore platform. Through collaborative brainstorming and strategic planning, the team focused on addressing common issues in book purchasing and inventory management while creating a scalable digital foundation for future e-commerce expansion.**

### **Key Activities:**

**Step 1: Team Collaboration and Problem Statement Selection**

Team Collaboration and Problem Statement Selection

The team assembled on [Insert Start Date] to analyze gaps in traditional and online book-selling platforms.

**Through research and shared user experiences, the final problem statement was defined:**

"Traditional bookstore systems lack real-time inventory, user convenience, and personalized experience, resulting in poor engagement and missed sales opportunities."

**Step 2: Brainstorming, Idea Listing, and Grouping**

**All team members contributed ideas openly, which were then categorized into three core focus areas:**

**User-Centered Features:**

**Role-based registration/login (Admin, Seller, Buyer)**

**Book browsing with filters (category, price, author, rating)**

**Add to wishlist/cart functionality**

**Order history and tracking**

**Reviews and ratings for books**

**Seller-Centered Features:**

**Add/update/remove books with images and details**

**View order requests and manage stock**

### **Access analytics for top-selling books and customer trends**

### **Admin-Centered Features:**

### **Approve or reject seller registrations**

### **Manage all users and books**

### **Monitor transactions and user feedback**

### **Generate platform-wide reportsStep 3: Idea Prioritization**

The team used an impact–effort matrix to identify features for the Minimum Viable Product (MVP).

Priority was given to high-impact, low-effort functionalities to ensure a quick and effective launch.

**Selected for MVP (Phase-1):**

Role-based registration/login (Admin, Seller, Buyer)

* Book listing with category filters
* Add to cart and checkout
* Admin approval of seller accounts
* Order tracking for users and sellers
* Dashboards for all user

**Deferred for Future Phases:**

Recommendation system based on purchase history

* Real-time chat with sellers
* AI-based personalized book suggestions
* Loyalty points and discounts
* Return/refund management system

### ****Customer Journey Map****

**Purpose:** To visualize and improve the user experience when browsing, reserving, and borrowing books online, whether from a public library, campus collection, or community-sharing system.

**Key Components:**

1. **User Actions:**
   * Registering and logging into the platform.
   * Browsing books by category, author, availability.
   * Reserving or borrowing books online.
   * Viewing borrowing history and due dates.
   * Returning books and giving ratings/reviews.
2. **User Thoughts and Emotions:**
   * **Before:** Frustration with unavailable books or unclear availability.
   * **During:** Joy in discovering and reserving desired titles quickly.
   * **After:** Satisfaction or dissatisfaction based on timely returns and reminders.
3. **Touchpoints:**

* Home page, category filters, book detail pages.
* Reservation form and confirmation pages.
* Email/SMS alerts for due dates and reminders.
* User dashboard for tracking status and history.

1. **Pain Points and Opportunities:**

* Difficulty finding books → Implement smart search & filters.
* Overdue penalties → Enable timely reminders.
* Limited availability → Waitlist/reservation feature.

**Insight:** Mapping this journey highlights where users get stuck and helps streamline the borrowing/reservation process with personalization, reminders, and better navigation.

### ****Solution Requirements****

### ****1. Functional Requirements:****

These define what the system should do.

* User Registration/Login with role-based access (Reader, Admin).
* Browse/search books with filters (genre, availability, title, author).
* Book reservation system with calendar-based pickup/return slot.
* Reader dashboard to view current and past borrows, due dates.
* Admin dashboard to manage books, returns, and users.
* Notification system for reminders and overdue alerts.
* Feedback system for book reviews and ratings.

### ****2. Non-Functional Requirements:****

* Responsive interface (mobile/tablet/desktop).
* Optimized search and minimal API response times.
* Secure data handling and role-based access control.
* Easy maintainability and scalability.

### ****3. System Requirements:****

* **Frontend:** React.js with Bootstrap/Tailwind.
* **Backend:** Node.js + Express.js.
* **Database:** MongoDB with Mongoose.
* **Authentication:** JWT + bcrypt.js.
* **Hosting:** Render or Vercel with GitHub Actions CI/CD.
* **Notifications:** NodeMailer or Twilio.

**Data Flow :**

A **Data Flow Diagram (DFD)** illustrates how data moves within the Freelance Finder platform. It captures how users (freelancers and clients) interact with the system, how information flows between different components, and where the data is stored.

**As a User...**

* I can log in securely and access my dashboard.
* I can browse or search for books by different criteria.
* I can reserve books and view my borrowing queue.
* I get reminders before due dates to avoid penalties.
* I can give feedback and rate books I’ve read.

**As an Admin...**

* I can add, edit, or delete book records.
* I can approve/track user reservations and returns.
* I can manage late returns and generate reports.
* I can view usage analytics and system logs.

**Technology Stack:**

| **S.No** | **Component** | **Technology Used** |
| --- | --- | --- |
| 1 | User Interface | React.js, Bootstrap, Tailwind CSS |
| 2 | Routing | React Router DOM |
| 3 | API Calls | Axios |
| 4 | Auth & Role Management | JWT, bcrypt.js |
| 5 | Backend | Node.js, Express.js |
| 6 | Database | MongoDB, Mongoose |
| 7 | Notifications | NodeMailer / Cron jobs |

**Components & Technologies – BookNest**

|  |  |  |  |
| --- | --- | --- | --- |
| S.No | Component | Description | Technology Used |
| 1 | User Interface | Web-based responsive UI for patients, doctors, and admins | React.js, HTML, CSS, JavaScript, Bootstrap, Tailwind CSS |
| 2 | Routing & Navigation | Handles client-side page switching and protected routes | React Router DOM, Context API |
| 3 | API Communication | Facilitates frontend-backend interaction | Axios (HTTP client) |
| 4 | Authentication System | Secure login and role management | JSON Web Tokens (JWT), bcrypt.js |
| 5 | Appointment Logic | Booking, status management, and slot scheduling | Node.js, Express.js |
| 6 | Admin Panel | Doctor approval, user management, and appointment oversight | React.js (admin views), Express.js (backend logic) |

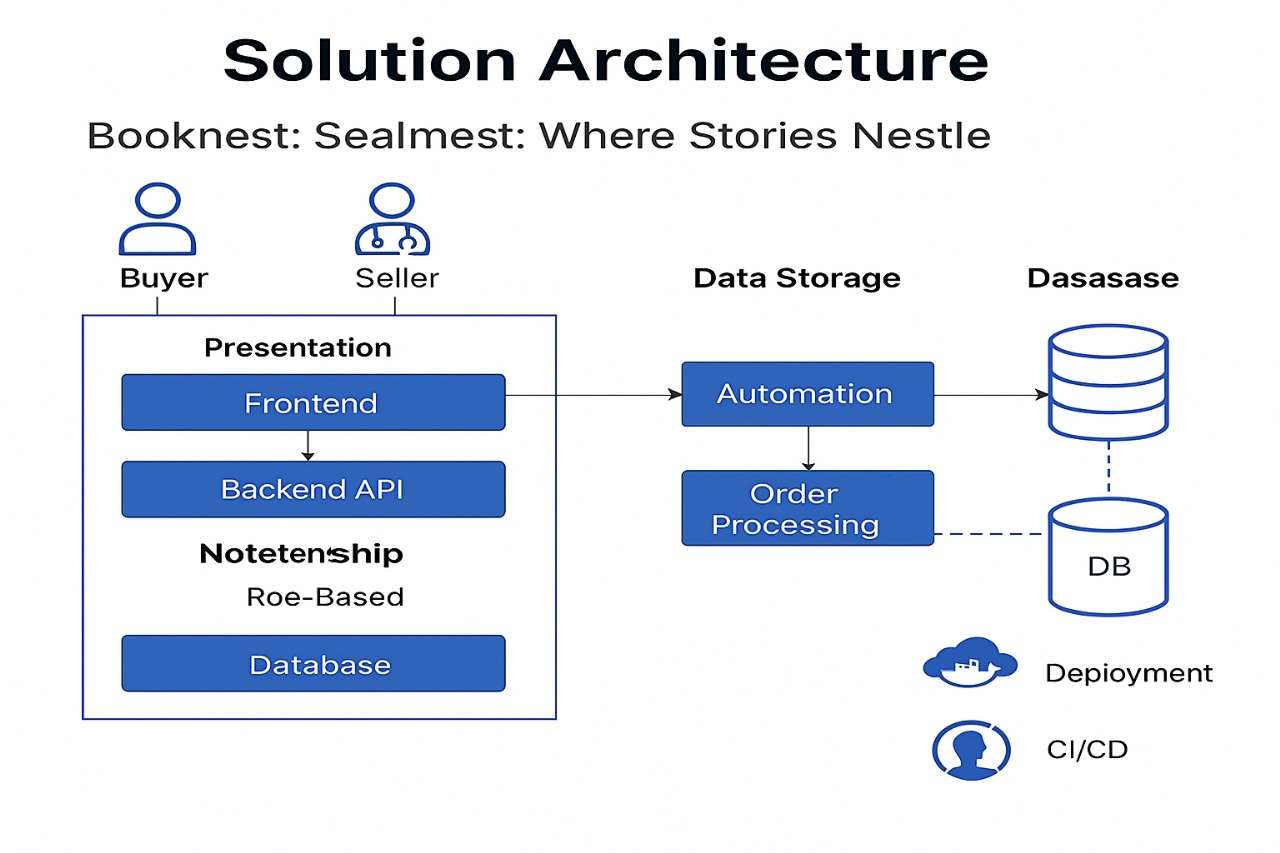
**Problem–Solution Fit Overview**

| **Parameter** | **Description** |
| --- | --- |
| Problem Statement | Users struggle with finding available books and face issues with returns. |
| Idea / Solution | A centralized platform for browsing, reserving, and managing book lending. |
| Novel Features | Smart search, waitlist handling, rating system, email reminders. |
| Social Impact | Encourages reading, improves library access, reduces missed returns. |
| Revenue Model | Freemium library integrations, subscription for advanced features. |
| Scalability | Modular design for multi-branch library support and multi-language support. |

**Proposed Solution – DocSpot App**

|  |  |  |
| --- | --- | --- |
| S. No. | Parameter | Description |
| 1 | **Problem Statement** | Book readers and library-goers struggle to find, reserve, or track books efficiently. Admins face issues with managing book inventory, user access, and overdue tracking. |
| 2 | **Idea / Solution** | A full-stack MERN web app for real-time book reservation, user-role access (readers, librarians, admins), return reminders, inventory management, and analytics. |
| 3 | **Novel Features** | Real-time book availability, multi-role system, return date reminders, overdue penalties, review system, digital borrowing for eBooks. |
| 4 | **Social Impact** | Encourages reading habits, digitizes library access, improves resource sharing in underserved or rural areas. |
| 5 | **Revenue Model** | Freemium for users, subscriptions for libraries, pay-per-book for rare content, affiliate eBook sales. |
| 6 | **Scalability** | Supports multilingual content, responsive mobile-first design, microservice-ready backend for future modules (e.g., eBook store, community). |

**Solution Architecture**

**Architecture Principles:** Modularity, Security, Scalability, Maintainability

* **Client-Server Model:**
  + **Frontend (Client):** React-based responsive UI for searching, booking, reviewing books, and profile management.
  + **Backend (Server):** Node.js with Express for routing, authentication, business logic, and handling API calls.
* **Component Interaction:**
  + REST APIs connect frontend to backend.
  + Real-time reminders via WebSockets or third-party APIs (email/SMS).
  + Admin portal interacts with book database, inventory, and users.
* **Data Storage:**
  + MongoDB stores user data, book records, reservations, penalties, reviews.
  + Encryption used for sensitive data (e.g., credentials, payment details).
* **Extensibility:**
  + Clearly defined services (user, book, booking, review, notifications).
  + Clean code structure to allow team scaling and feature expansion.

**PROJECT PLANNING & SCHEDULING:**

**Project Planning**

* **Scope Definition:** Book discovery, reservations, reviews, and library admin dashboard.
* **Task Breakdown:**
  + Module 1: User Authentication
  + Module 2: Book Search & Reservation
  + Module 3: Admin & Inventory Management
  + Module 4: Notifications & Penalties
* **Resource Allocation:** MERN stack developers, QA testers, UI/UX designer.
* **Timeline Estimation:** 4–6 weeks for MVP.
* **Milestones:**
  + Week 1: UI & Backend Setup
  + Week 2: Auth Complete
  + Week 3: Book Flow Complete
  + Week 4: Admin Dashboard
  + Week 5: Testing & Polishing
* **Risks & Mitigation:**
  + Book data volume → Use pagination
  + UI delay → Ready-made UI kits (e.g., Material UI)
* **Communication Plan:**
  + Weekly Scrum
  + Jira/Trello for tracking
  + Slack/Email for async communication

**FUNCTIONAL AND PERFORMANCE TESTING:**

**Performance Testing**

**Performance Testing Goals**

* Ensure users can browse and reserve books without delay.
* Optimize for < 1.5s response for book search.
* Handle concurrent users during peak hours.

**Testing Types**

* **Load Testing:** 1000+ concurrent users reserving books.
* **Stress Testing:** Simulate mass returns & overdue generation.
* **Scalability Testing:** Add 50,000 books and test pagination, search filters.
* **Endurance Testing:** Run system continuously for 48 hours to detect memory leaks or failures.

**Agile Sprint Backlog**

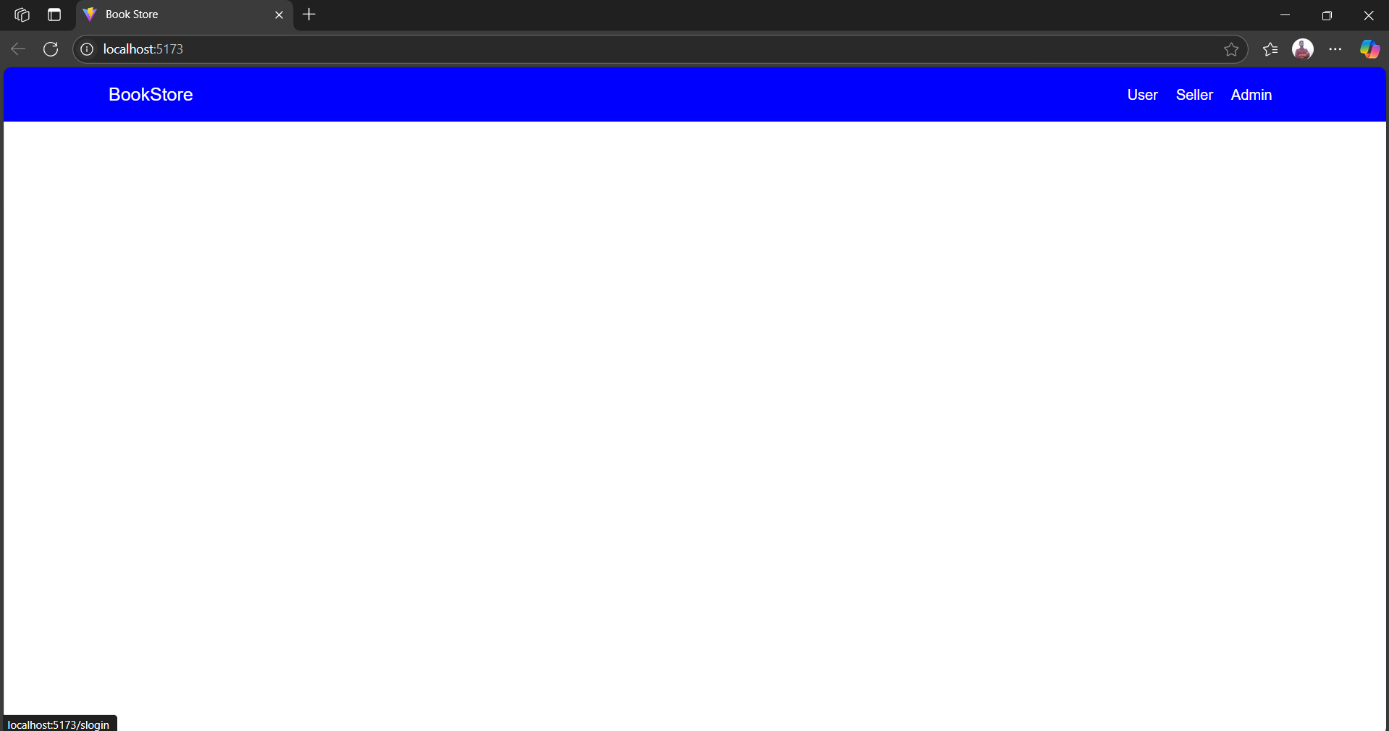
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sprint | Functional Requirement (Epic) | User Story Number | User Story / Task | Story Points | Priority |
| Sprint-1 | User Authentication | USN-1 | As a user, I can sign up and log in. | 3 | High |
|  |  | USN-2 | As a user, I can reset my password securely. | 2 | Medium |
| Sprint-2 | Book Discovery & Reservation | USN-3 | As a reader, I can search/filter books by genre, author, or title. | 2 | High |
|  |  | USN-4 | As a reader, I can reserve an available book. | 3 | High |
| Sprint-3 | Inventory & Admin Panel | USN-5 | As an admin, I can add/edit/delete books and set availability. | 3 | High |
|  |  | USN-6 | As an admin, I can view overdue books and fine users. | 2 | High |
| Sprint-4 | Notifications & Reviews | USN-7 | As a user, I receive reminders for book return deadlines. | 2 | Medium |
|  |  | USN-8 | As a user, I can rate/review books I’ve read. | 2 | High |
|  |  | USN-9 | As a user, I can download eBooks (if available). | 2 | Medium |
|  |  |
|  |  |

**RESULTS:**

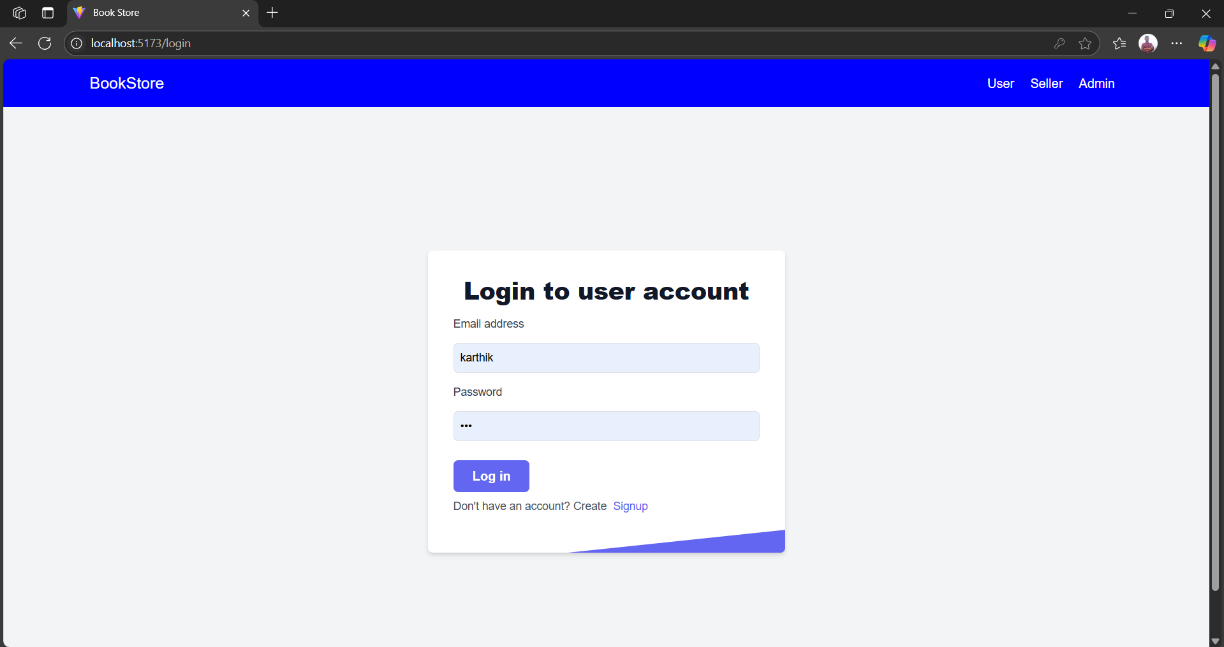
**Output Screenshots**

**Links :** https://docs.google.com/document/d/1eM6wTVcuQzAgSisKO7fi8Wf3tyAnT698/edit?usp=drive\_link&ouid=103014717462452408435&rtpof=true&sd=true

**LANDING PAGE :**

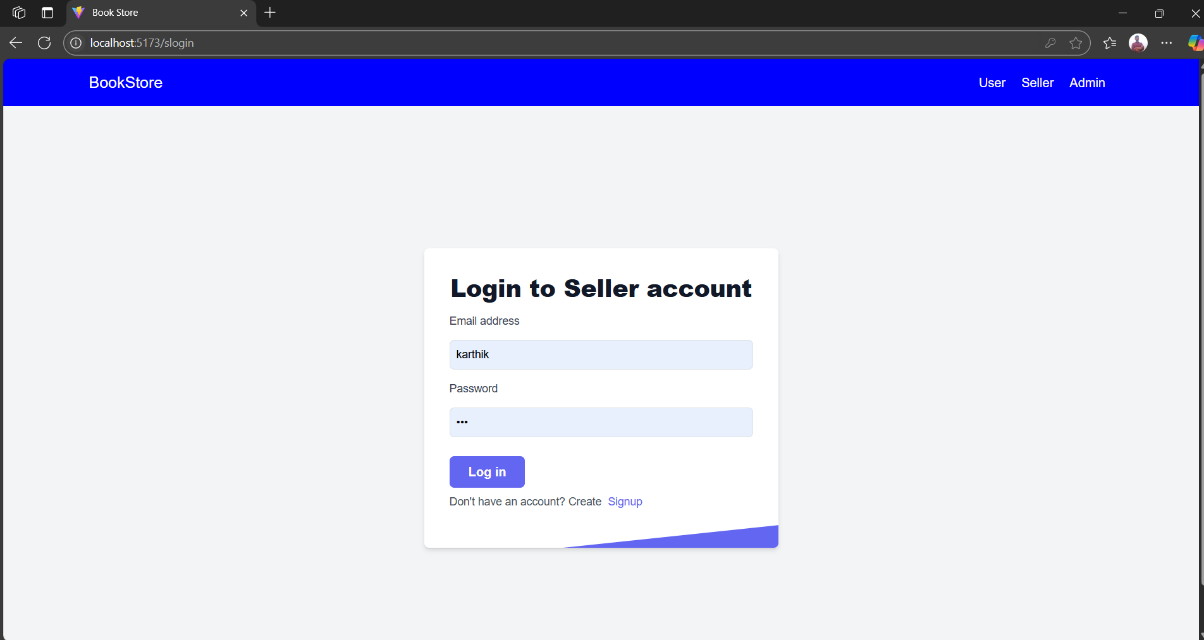
****

**LOGIN PAGE FOR USER:**

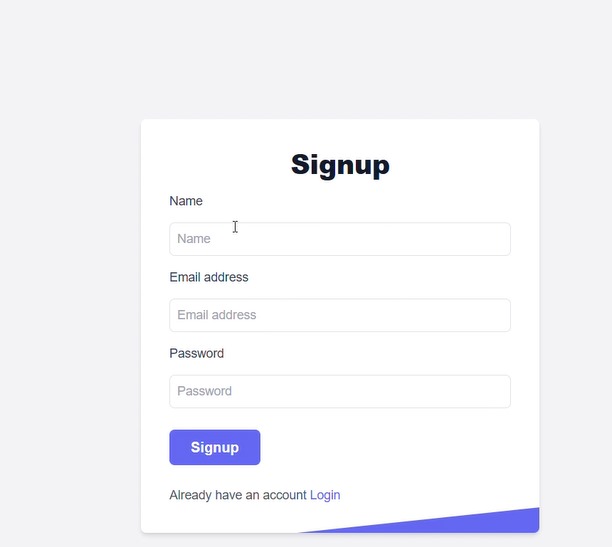
****

**LOGIN SUCCESS**

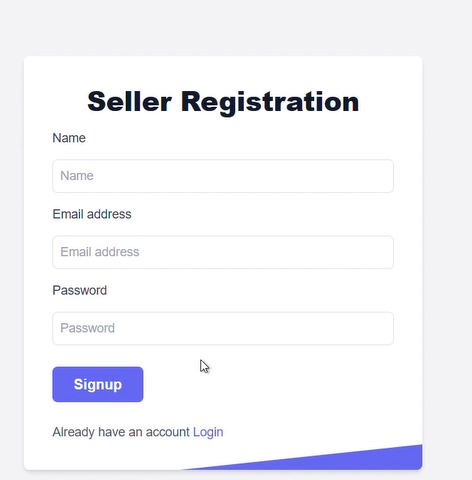
**LOGIN PAGE FOR SELLER:**

****

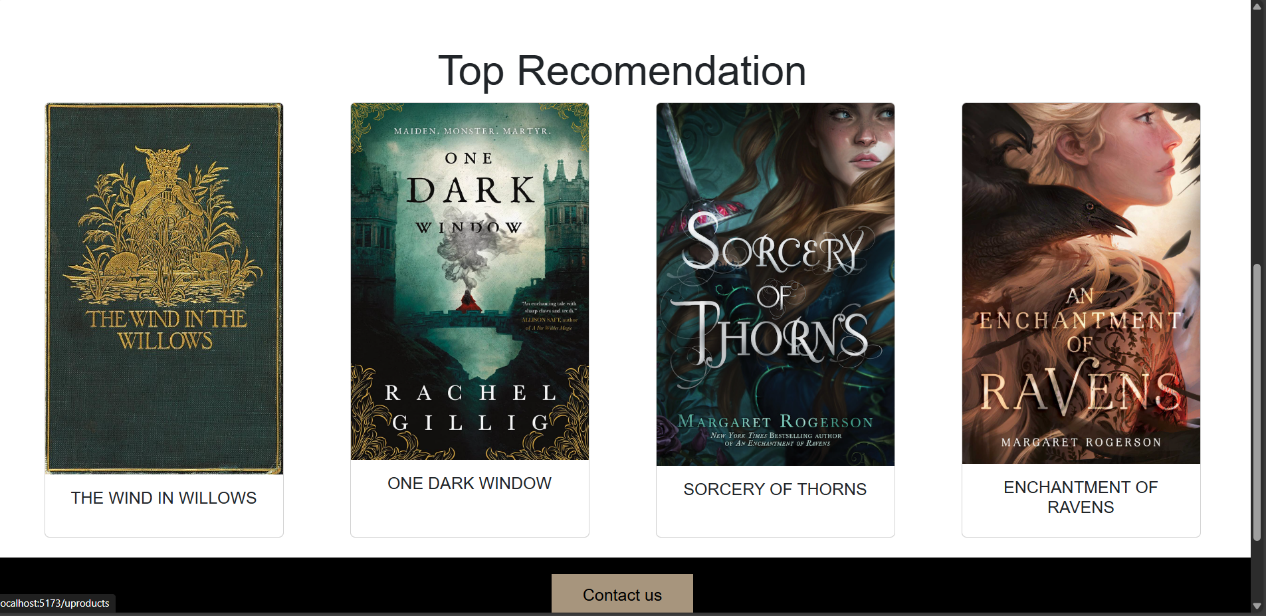
**REGISTER FOR USER**

****

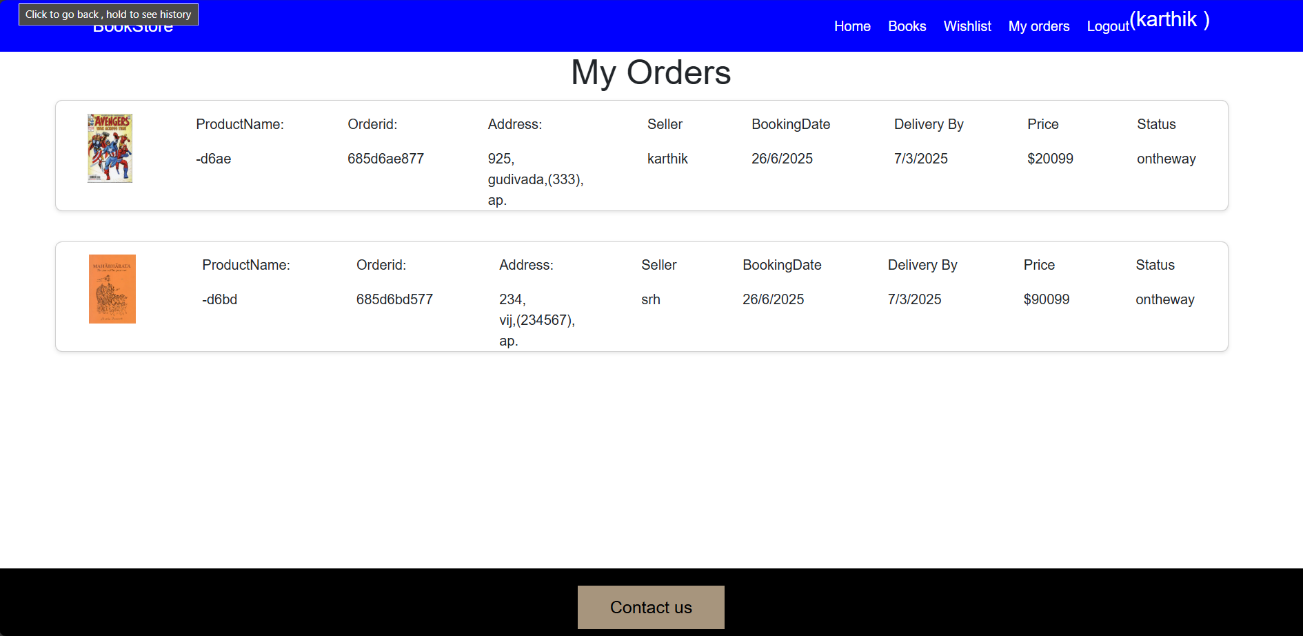
**REGISTER FOR SELLER**

****

**BACK TO DASHBOARD**

****

**APPLY NOW**

****

**ORDER APPROVED**

**CONCLUSION:**

**BookNest** demonstrates a robust, scalable library/bookstore booking and management system using the MERN stack. It supports a real-world use case with multiple user roles and features like book reservation, overdue alerts, and inventory control. Security is enforced using JWT and RBAC. With future upgrades like multilingual support and mobile versions, BookNest can serve institutions globally.This project serves as both a **learning tool** and a **practical application**, showcasing how to design and develop a complete web-based system using the MERN stack.

**FUTURE SCOPE:**

* Mobile apps for Android and iOS.
* AI-based book recommendations.
* QR code-based in-library borrowing.
* Support for library branches and inter-branch loans.
* Book clubs and reader communities.
* Offline sync and reading history analytics.

**APPENDIX:**

 **Source Code:** <https://github.com/Karthik-Immaneni-7/booknest>

 **Project Demo Link:** **https://drive.google.com/file/d/1apn4nEoCPD9EFaQpzlbCqIxseluSbr6M/view?usp=drive\_link**