



PROJECT REPORT ON

BookNest : Where Stories Nestle

Read Smarter. Publish Faster

SUBMITTED BY

Team ID : LTVIP2025TMI053625

Team Size : 4

Team Leader :Shaik.Alsaba

Team member : Shaikmohammad Shareef

Team member : Sibyala Reddy Parthasaradhi

Team member : Snehalatha Jerrg

Submitted To



1. INTRODUCTION

1. Project Overview

As part of our group summer internship project, we designed and developed a full-stack web application called **“BookNest: Where Stories Nestle”**. This project aims to simplify the discovery, reading, and sharing of stories by bringing readers, authors, and admins onto a unified digital platform.

In this system, readers can easily register, explore stories by genre or emotion, and create personalized collections called “Nests.” Authors get their own dashboard to upload stories, view engagement metrics, and interact with readers, while admins oversee author approvals and ensure content quality and platform integrity.

We built the application using the MERN stack – **MongoDB, Express.js, React.js, and Node.js** – along with tools like **Axios** for API communication, **Moment.js** for content scheduling, and **Material UI/Bootstrap** for an engaging and responsive interface. To safeguard user data and ensure secure access, we implemented authentication using **JWT tokens** and password encryption through **Bcrypt**.

This project deepened our understanding of full-stack development in real-world applications and showed us how technology can be used to elevate storytelling, improve discoverability, and foster creative communities online.

1. Purpose

The main purpose of this project is to address common challenges faced by readers and writers – such as poor discoverability of quality content, limited interaction between authors and their audience, and lack of personalization in digital reading platforms.

We aimed to build a system where:

- 📖 **Readers** can explore curated stories based on genre, emotion, and popularity, and create custom collections.
- ✍️ **Authors** can publish stories easily, track engagement, and receive meaningful feedback.
- 🛡️ **Admins** can manage platform content, approve author profiles, and maintain platform integrity.

Apart from the technical learning aspect, our goal was to create something meaningful that could **potentially be used in real-world settings**, especially in today's digital-first healthcare environment. It also helped us get hands-on experience with **project planning, teamwork, backend APIs, frontend design, and database management** during our internship.

1. IDEATION PHASE

1. Problem Statement

In today's digital era, discovering impactful stories or connecting with meaningful content remains surprisingly difficult. Most platforms offer generic feeds with limited personalization, while new authors struggle to find visibility without marketing budgets or publisher support. Readers face challenges like:

- 📺 Bland and repetitive recommendations
- 🔍 Poor tools to filter by emotion, theme, or genre
- 💬 Limited interaction with content creators
- 📖 Lack of community-based curation and engagement

Authors, meanwhile, often lack a space to share stories, build their audience, and receive feedback. Admins face challenges in moderating content, verifying creators, and ensuring platform quality.

We identified a gap in the storytelling ecosystem where a digital solution could make story exploration more personalized, emotionally resonant, and interactive—especially at a time when digital reading and independent publishing are on the rise.

Problem: There is a lack of a centralized, user-friendly storytelling platform where readers can explore personalized content and authors can engage meaningfully with their audience, while admins maintain

content integrity and user trust.

1. Empathy Map Canvas

To Understand user needs, we created an Empathy Map by stepping into the mindset of the Reader, Author and Admin.

| User | Says | Thinks | Does | Feels |
|--------|--|--|-------------------------------------|------------------------|
| Reader | "I want stories that move me." | "Will I find anything that matches my mood?" | Browses generic platforms | Disconnected, unsure |
| Author | "I need a space to share and grow." | "How will readers discover my story?" | Publishes on blogs or closed groups | Invisible, discouraged |
| Admin | "This platform should maintain quality." | "Are all authors and stories verified?" | Reviews uploads, monitors activity | Responsible, cautious |

This empathy map helped our team identify the frustrations and emotional expectations of our key users, guiding our design priorities and platform features.

1. Brainstorming

During early planning sessions, we shared ideas freely and used whiteboarding to visualize what a truly reader-friendly and author-supportive platform could look like. Our motto: Every idea has potential.

Some early questions we explored:

- Should readers be able to group stories based on emotion?
- Can AI recommend content based on recent mood or genre trends?
- How do we safeguard story originality and avoid plagiarism?
- Should authors have analytics showing reader engagement?
- Can non-tech users navigate the app effortlessly?

We narrowed our scope to core MVP features that would make BookNest functional and valuable for all stakeholders:

- 📥 Login/Signup for readers, authors, and admins
- 🔍 Emotion-based and genre-based story filtering
- ✍️ Story upload with formatting tools for authors
- 📊 Author dashboard with engagement metrics
- ✅ Admin approvals and content moderation
- 🔔 Real-time notifications and story recommendations

We used tools like Figma for UI design, Notion for organizing development tasks, and Miro to sketch workflows during our collaborative internship journey.

1. REQUIREMENT ANALYSIS

1. Customer Journey Map

To understand how users interact with the BookNest app, we mapped out the customer journey focusing on three main users: Reader, Author, and Admin.

- **Reader Journey:**

1. Registers and creates a personalized profile with genre/emotion preferences.
2. Explores stories filtered by emotion, genre, or popularity.
3. Adds favorite stories to personalized collections called “Nests.”
4. Follows favorite authors and engages via likes and comments.

5. Receives content recommendations and notifications.
6. Tracks reading history and revisits saved content.
7. Shares feedback and reviews to help improve story discovery.

- **Author Journey:**

1. Registers on the platform and submits profile for admin approval.
2. Uploads stories with optional emotion or genre tags.
3. Views engagement metrics and reader comments.
4. Updates stories, manages personal content library.
5. Receives feedback and insights on reader behavior.
6. Communicates with followers via updates or replies.

- **Admin Journey:**

1. Reviews and approves new author profiles.
2. Monitors story uploads and platform engagement metrics.
3. Resolves content disputes or user reports.
4. Enforces publishing guidelines and community standards.

Mapping this journey helped us understand platform interactions and improve feature alignment for each user role

1. **Solution Requirement**

Based on the problem and journey mapping, the solution requirements were divided into functional and non-functional.

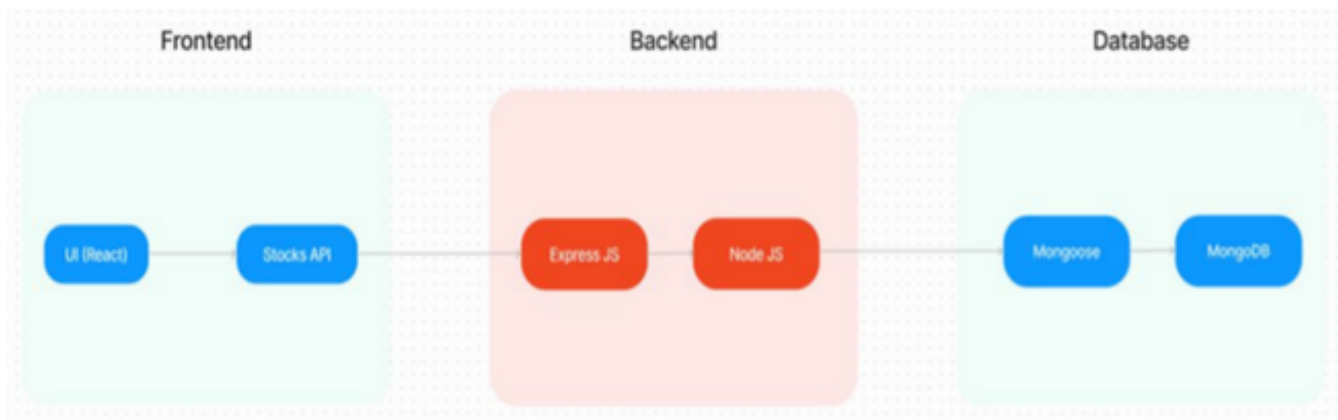
- **Functional Requirements:**

- User authentication with secure login/signup.
- Doctor browsing with filters (specialty, location, availability).
- Appointment booking interface with date/time selection.
- Document upload support for medical records.
- Real-time appointment confirmation and notifications.
- Doctor dashboard for managing availability and appointments.
- Admin panel for verifying doctors and overseeing the platform.

■ Non-Functional Requirements:

- Secure data storage and encrypted communication (SSL/TLS).
- Fast response time and smooth UI interactions.
- Role-based access control for different user types.
- Scalability to handle growing users and data.
- Cross-device compatibility (responsive design)

3.3 Data Flow Diagram



The Data Flow Diagram (DFD) depicts how data moves within the Book a Doctor App:

- **Patients** send registration data and appointment requests to the backend server.
- The **backend** validates, stores, and processes data in MongoDB.
- **Doctors** access appointment requests and update statuses.
- The **Admin** reviews doctor registrations and manages user data.
- Notifications are sent to users via email/SMS when appointments are confirmed or updated.

This DFD ensures clear understanding of data movement and responsibilities between components, which helped us during backend and API design.

1. Technology Stack

To build a reliable and scalable app, we selected the following technologies:

FRONTEND TECHNOLOGIES :

- **Bootstrap and Material UI:** Provide a responsive and modern UI that adapts to various devices, ensuring a user-friendly experience.
- **Axios:** A promise-based HTTP client for making requests to the backend, ensuring smooth data communication between the frontend and server.

BACKEND FRAMEWORK :

- **Express.js:** A lightweight Node.js framework used to handle server-side logic, API routing, and HTTP request/response management, making the backend scalable and easy to maintain.

DATABASE AND AUTHENTICATION :

- **MongoDB:** A NoSQL database used for flexible and scalable storage of user data, doctor profiles, and appointment records. It supports fast querying and large data volumes.
- **JWT (JSON Web Tokens):** Used for secure, stateless authentication, allowing users to remain logged in without requiring session storage on the server.
- **Bcrypt:** A library for hashing passwords, ensuring that sensitive data is securely stored in the database.

ADMIN PANEL & GOVERNANCE :

- **Admin Interface:** Provides functionality for platform admins to approve doctor registrations, manage platform settings, and oversee day-to-day operations.
- **Role-based Access Control (RBAC):** Ensures different users (patients, doctors, admins) have appropriate access levels to the system's features and data, maintaining privacy and security.

SCALABILITY AND PERFORMANCE :

- **MongoDB:** Scales horizontally, supporting increased data storage and high user traffic as the platform grows.

Load Balancing: Ensures traffic is evenly distributed across servers to optimise performance, especially during high traffic periods.

- **Caching:** Reduces database load by storing frequently requested data temporarily, speeding up response times and improving user experience.

TIME MANAGEMENT AND SCHEDULING

- **Moment.js:** Utilised for handling date and time operations, ensuring precise appointment scheduling, time zone handling, and formatting.

SECURITY FEATURES :

- **HTTPS:** The platform uses SSL/TLS encryption to secure data transmission between the client and server.
- **Data Encryption:** Sensitive user information, such as medical records, is encrypted both in transit and at rest, ensuring privacy and compliance with data protection regulations.
-

NOTIFICATIONS AND REMINDERS : FRONTEND TECHNOLOGIES :

- **Bootstrap and Material UI:** Provide a responsive and modern UI that adapts to various devices, ensuring a user-friendly experience.
- **Axios:** A promise-based HTTP client for making requests to the backend, ensuring smooth data communication between the frontend and server.

BACKEND FRAMEWORK :

- **Express.js:** A lightweight Node.js framework used to handle server-side logic, API routing, and HTTP request/response management, making the backend scalable and easy to maintain.

DATABASE AND AUTHENTICATION :

- **MongoDB:** A NoSQL database used for flexible and scalable storage of user data, doctor profiles, and appointment records. It supports fast querying and large data volumes.

- **JWT (JSON Web Tokens):** Used for secure, stateless authentication, allowing users to remain logged in without requiring session storage on the server.
- **Bcrypt:** A library for hashing passwords, ensuring that sensitive data is securely stored in the database.

ADMIN PANEL & GOVERNANCE :

- **Admin Interface:** Provides functionality for platform admins to approve doctor registrations, manage platform settings, and oversee day-to-day operations.
- **Role-based Access Control (RBAC):** Ensures different users (patients, doctors, admins) have appropriate access levels to the system's features and data, maintaining privacy and security.

SCALABILITY AND PERFORMANCE :

- **MongoDB:** Scales horizontally, supporting increased data storage and high user traffic as the platform grows.

Load Balancing: Ensures traffic is evenly distributed across servers to optimise performance, especially during high traffic periods.

- **Caching:** Reduces database load by storing frequently requested data temporarily, speeding up response times and improving user experience.

TIME MANAGEMENT AND SCHEDULING

- **Moment.js:** Utilised for handling date and time operations, ensuring precise appointment scheduling, time zone handling, and formatting.

SECURITY FEATURES :

- **HTTPS:** The platform uses SSL/TLS encryption to secure data transmission between the client and server.
- **Data Encryption:** Sensitive user information, such as medical records, is encrypted both in transit and at rest, ensuring privacy and compliance with data protection regulations.
-

NOTIFICATIONS AND REMINDERS :

- **Email/SMS Integration:** Notifications for appointment confirmations, reminders, cancellations, and updates are sent to users via email or SMS, ensuring timely communication.

1. PROJECT DESIGN

1. Problem Solution Fit

After analyzing the core problems faced by readers and storytellers, we ensured that our proposed BookNest application directly addresses those pain points. The main issues identified were poor content discoverability, limited feedback for authors, and lack of emotional personalization in story platforms. Our solution provides a centralized, reader-friendly platform where users can search stories by emotion or genre, curate content into themed collections (“Nests”), and interact meaningfully with authors.

By focusing on intuitive browsing, structured publishing, and real-time engagement features, the app bridges the gap between readers’ storytelling expectations and authors’ visibility needs, creating a vibrant, emotionally resonant digital reading experience. This alignment shows a strong problem-solution fit that benefits both user groups and supports content growth.

1. Proposed Solution

The BookNest app is a full-stack web-based platform designed to empower storytelling and personalized reading experiences. It offers:

4.2.1 A reader-friendly interface to browse stories based on filters like emotion, genre, and trending tags

4.2.2 A personalized recommendation system suggesting stories based on user preferences and recent activity

4.2.3 Secure story publishing module allowing authors to submit and manage their content with formatting tools

4.2.4 An author dashboard to track reader engagement, manage uploads, and respond to feedback

4.2.5 An admin panel that verifies author profiles, moderates content, and ensures community guidelines are followed

4.2.6 Automated notification system that delivers updates on new stories, saved Nests, and community interactions via email or alerts

This feature-rich solution uses modern design patterns and thoughtful user flows to make storytelling more accessible and rewarding for all stakeholders.

1. Solution Architecture

The BookNest application also follows a client-server architecture with a modular design separating frontend, backend, and data layers:

4.3.1 The frontend is built using React.js with Tailwind CSS and Material UI for an engaging, responsive reading interface. Story grids and interactive Nests enhance content discovery.

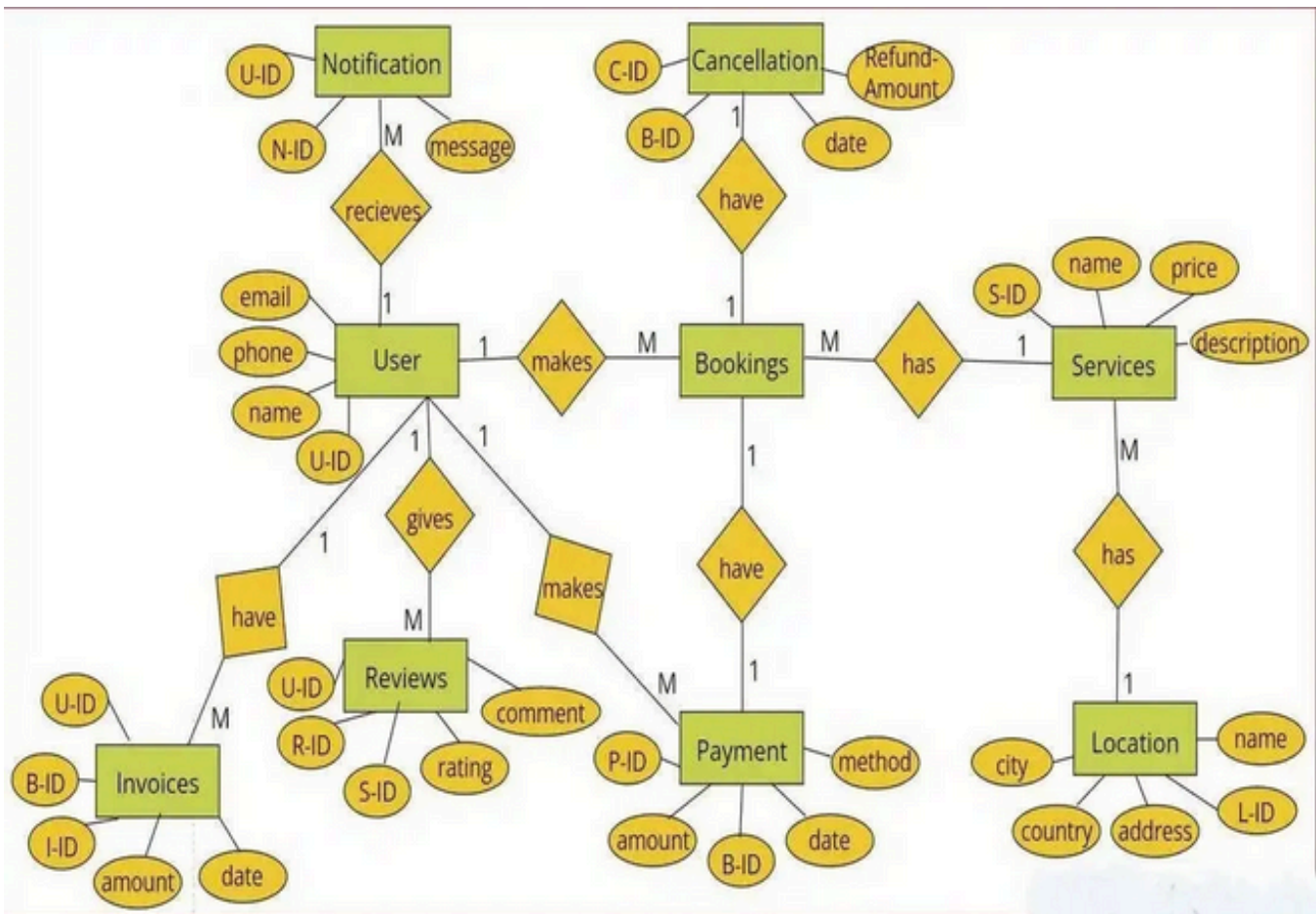
4.3.2 The backend is powered by Node.js and Express.js, handling publishing workflows, user authentication, story tagging logic, and analytics generation.

4.3.3 MongoDB stores user profiles, story metadata, feedback interactions, and Nest collections with scalable structure.

4.3.4 JWT tokens are used for secure, role-based session handling across readers, authors, and admins.

4.3.5 Notifications are handled via integrated email systems and in-app alerts to ensure timely communication and reader retention.

This flexible, component-driven architecture supports scalability, modularity, and future feature expansion such as multilingual content and mobile accessibility.



1. PROJECT PLANNING & SCHEDULING

1. Project Planning

For this group project, we followed a structured planning approach to ensure timely delivery and efficient teamwork during our summer internship period. The project was divided into clear phases, with each team member assigned specific responsibilities based on their strengths and interests.

The key stages of planning included:

1. **Requirement Gathering:** We collectively researched and discussed the problem domain, user needs, and technical feasibility.
2. **Design and Architecture:** The team collaborated on designing the system architecture, user flows, and database schema to ensure a robust foundation.
3. **Development Tasks:** The work was divided between frontend and backend development. Some members focused on building React components and UI, while others developed API endpoints, database models, and authentication.

4. **Testing and Integration:** After development, we conducted functional and performance testing, fixing bugs and refining features.
5. **Documentation and Presentation:** We prepared project documentation, reports, and demo presentations to showcase the app.

To manage our progress, we used tools like Trello for task tracking and GitHub for version control. Regular team meetings ensured communication and problem-solving, while setting realistic deadlines helped us stay on schedule.

Overall, this planning approach helped us coordinate effectively, balance workload, and deliver a functional BookNest: Where Stories Nestle App by the end of the internship.

1. FUNCTIONAL AND PERFORMANCE TESTING

1. Performance Testing

We tested each feature based on the user stories and requirements to ensure accurate functionality.

Key modules included user registration and login, story browsing and filtering, story upload, Nest creation, reader-author interactions, notifications, and admin moderation controls.

Testing was conducted manually by team members simulating different user roles—readers, authors, and admins. Bugs were identified in story upload validation, content filtering, and comment submission. These were resolved to enhance platform stability and ensure smooth user experiences across the board.

We also verified:

- Reader content recommendations based on genre/emotion preferences
- Author dashboard metrics rendering correctly
- Admin workflows for verifying and publishing stories

Performance Testing:

To validate responsiveness and stability, we evaluated system behavior under various workloads. Tools like Postman and browser dev tools were used to monitor API response times, database queries, and rendering speeds.

Simulated traffic included multiple readers browsing and liking stories while authors uploaded content concurrently. MongoDB queries related to story metadata and user preferences were optimized for faster retrieval, and lazy loading strategies were implemented to enhance performance during large data operations.

The platform consistently returned API responses under 300 milliseconds, with the frontend maintaining smooth interaction even under high concurrency. These optimizations ensured that the app could scale efficiently while delivering a seamless reading and storytelling experience.

Conclusion:

Through extensive functional and performance testing, we confirmed that BookNest meets usability, scalability, and reliability standards—offering readers and authors an engaging, responsive platform to explore and share stories.

1. RESULTS

1. Output Screenshots

During the development of the BookNest application, we successfully built and tested key features that fulfill the platform's storytelling and engagement objectives. Below are several conceptual output screenshots that highlight the working modules of the application:

7.1.1 Reader Registration and Login: Secure sign-up and login screens for readers, authors, and admins with role-based access.

7.1.2 Story Browsing and Filtering: Reader interface showcasing stories with filters for genre, emotion, popularity, and author.

7.1.3 Nest Creation and Management: UI that allows readers to group favorite stories into custom collections called “Nests.”

7.1.4 Story Interaction Panel: Screens for commenting, liking, and saving stories, along with real-time recommendations.

7.1.5 Author Dashboard: Authors manage uploaded stories, view engagement analytics, and respond to feedback.

7.1.6 Admin Panel: Admins approve author profiles, moderate content uploads, and oversee platform activity with metrics.

These screenshots demonstrate that the application meets the design and user interaction goals, presenting a user-centric layout with smooth navigation across all roles.

During frontend and backend testing, the system delivered consistent performance, maintaining quick rendering times and accurate data flow. Notifications were dispatched reliably, and user actions across stories and collections were reflected without delays.

Overall, the output results confirm that BookNest fulfills its mission to connect readers with storytellers, promote emotional engagement through curated content, and support platform governance—aligning closely with the project’s original vision.

▼ **BOOK-STORE**

▼ Backend

▼ db

> Seller

> Users

JS config.js

> node_modules

> uploads

{} package-lock.json

{} package.json

JS server.js

▼ frontend

> node_modules

> public

▼ src


> Admin

> Componenets


> Seller


> User


App.css


 App.jsx

index.css

 main.jsx


 .eslintrc.cjs

 .gitignore

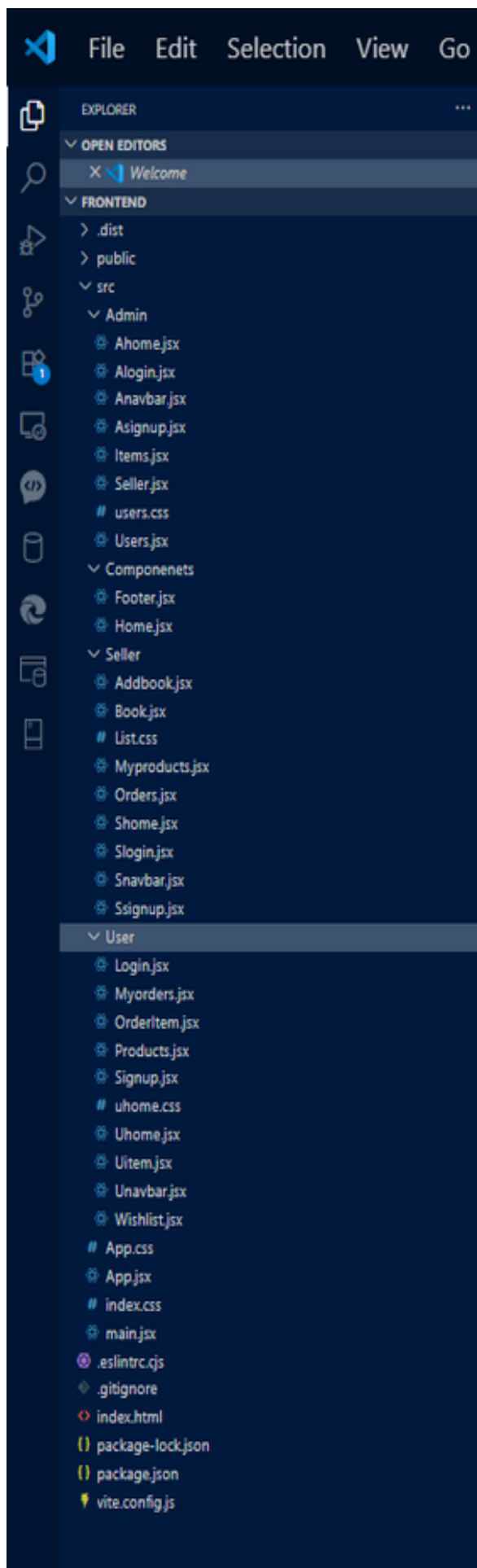
 index.html

{} package-lock.json

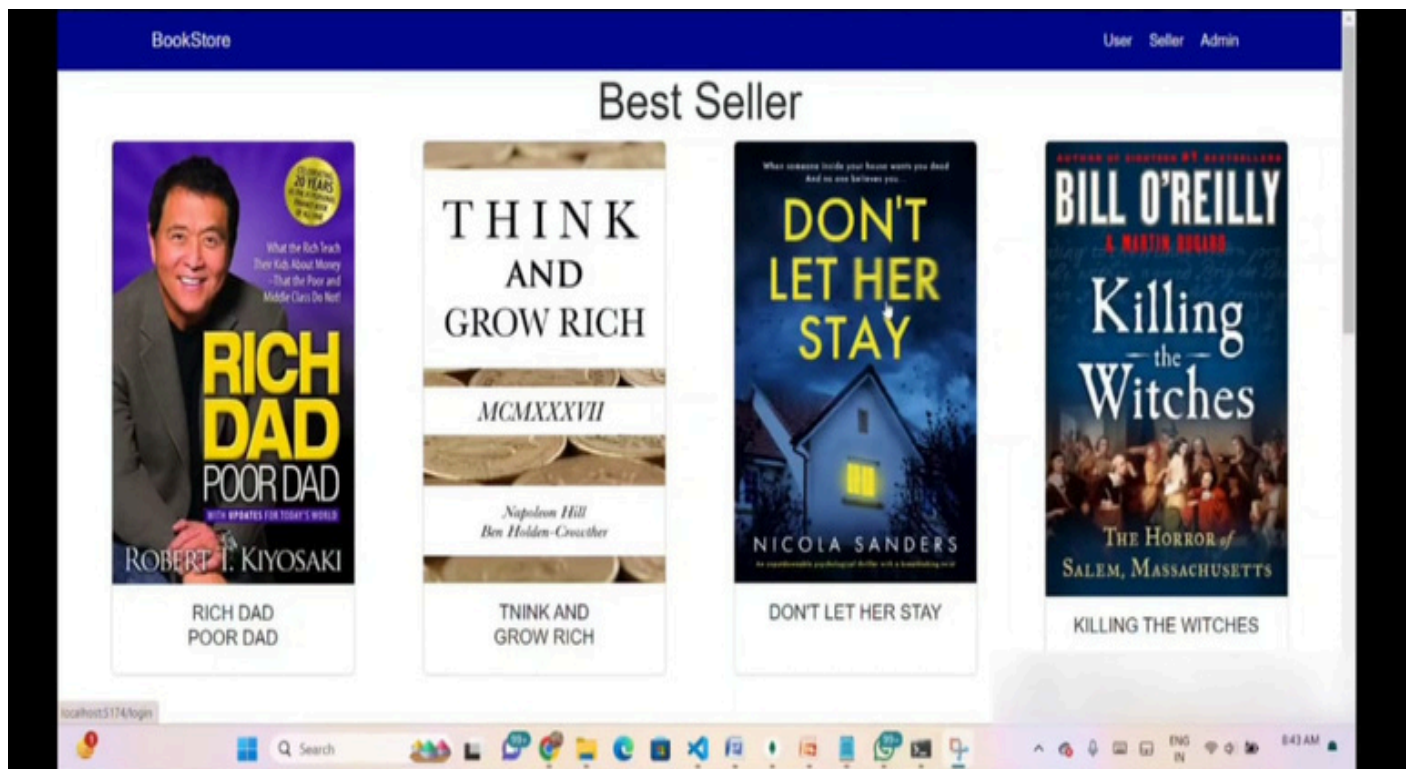
{} package.json

 README.md

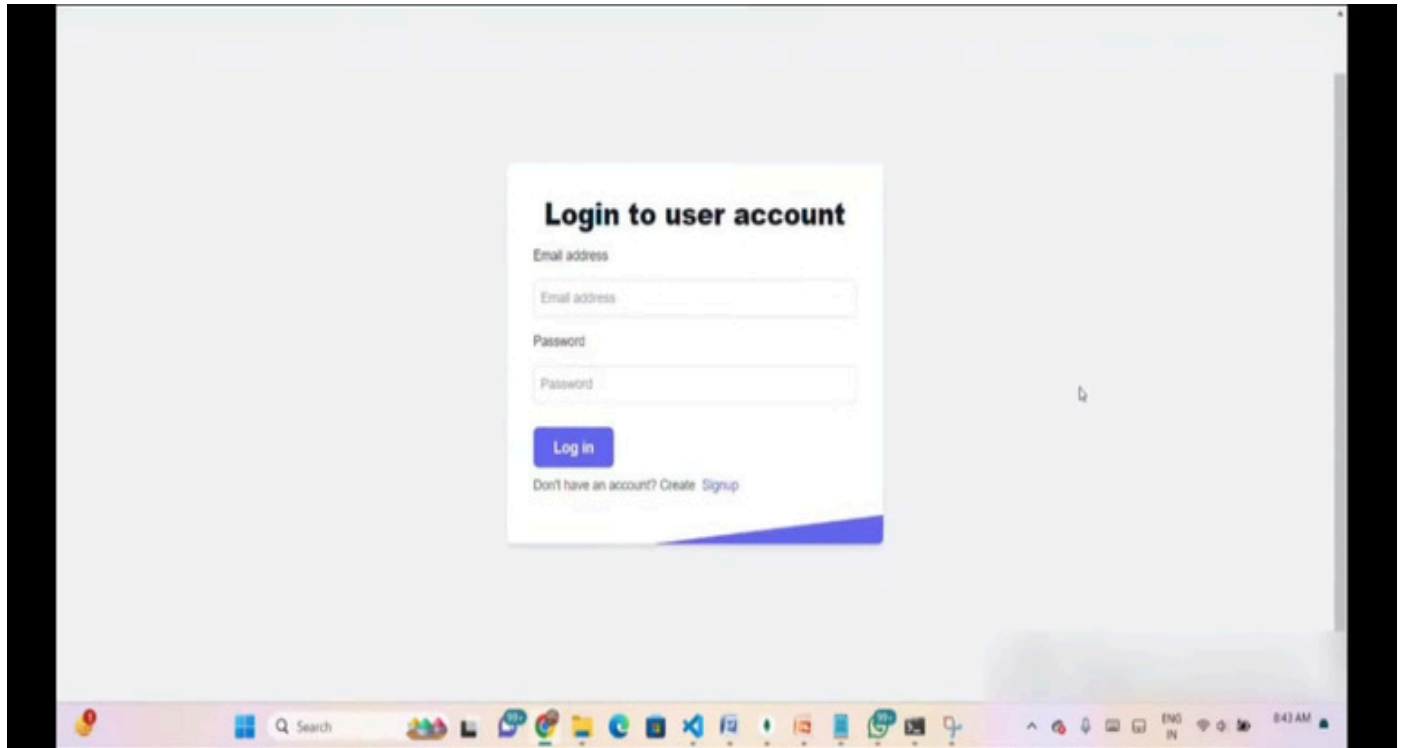
JS vite.config.js



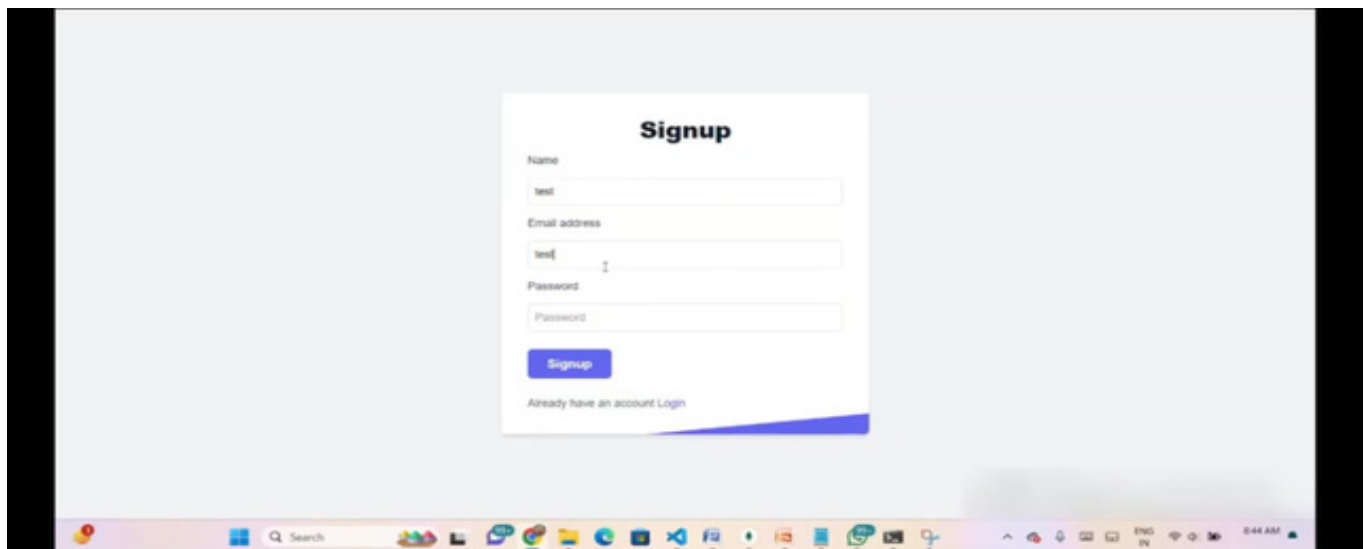
LANDING PAGE :



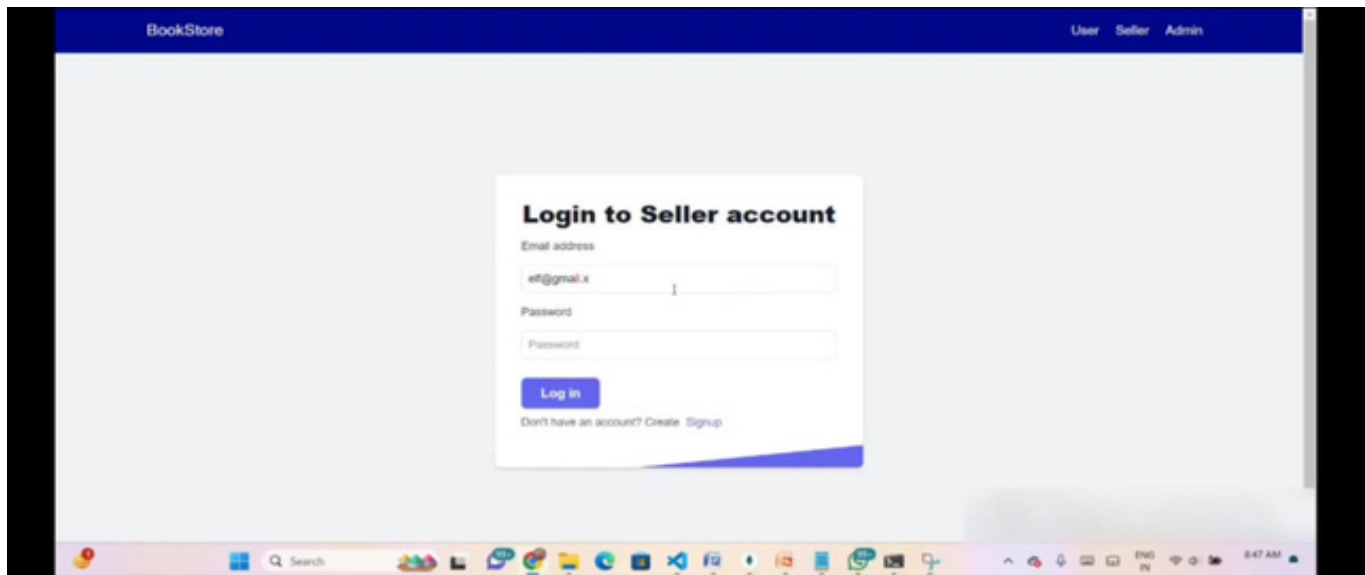
LOGIN PAGE (User):



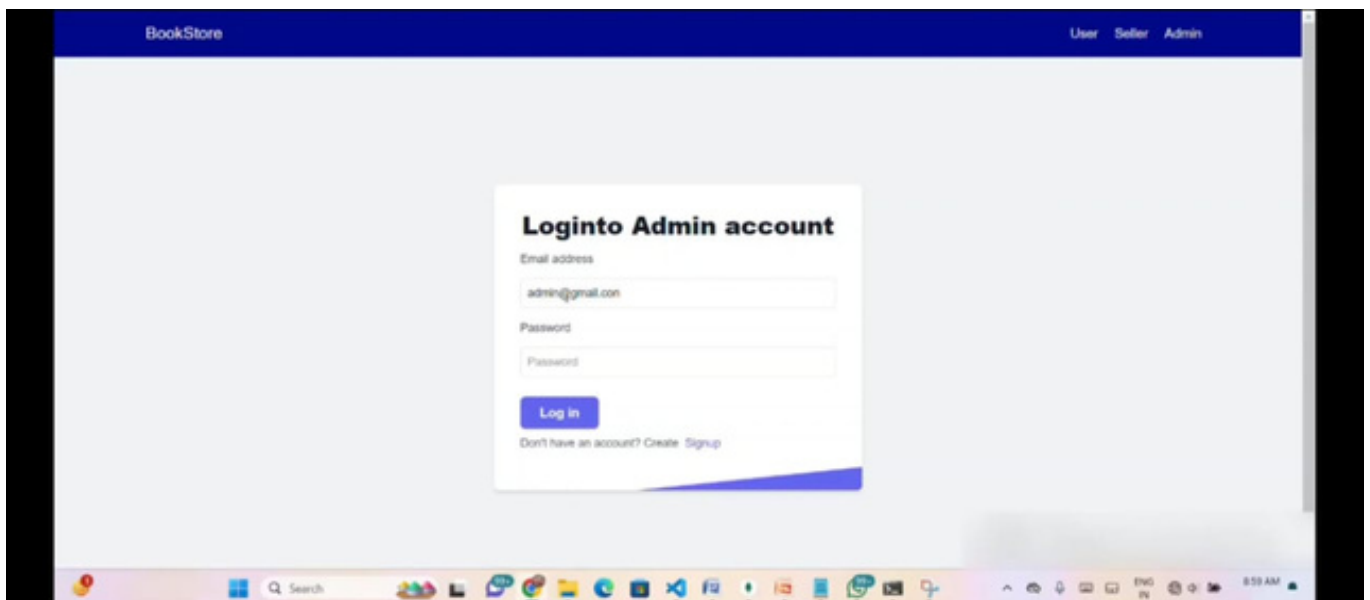
REGISTRATION PAGE (User):



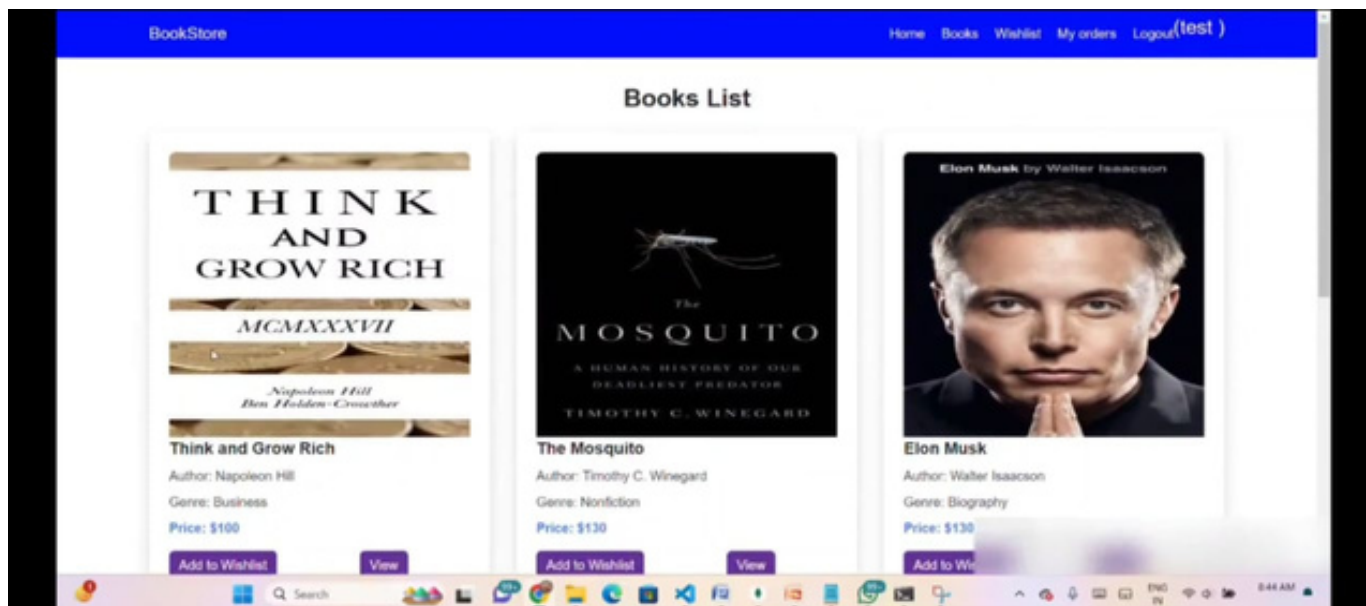
SELLER PAGE :



ADMIN PAGE




User Interface :



ADMIN DASHBOARDS :

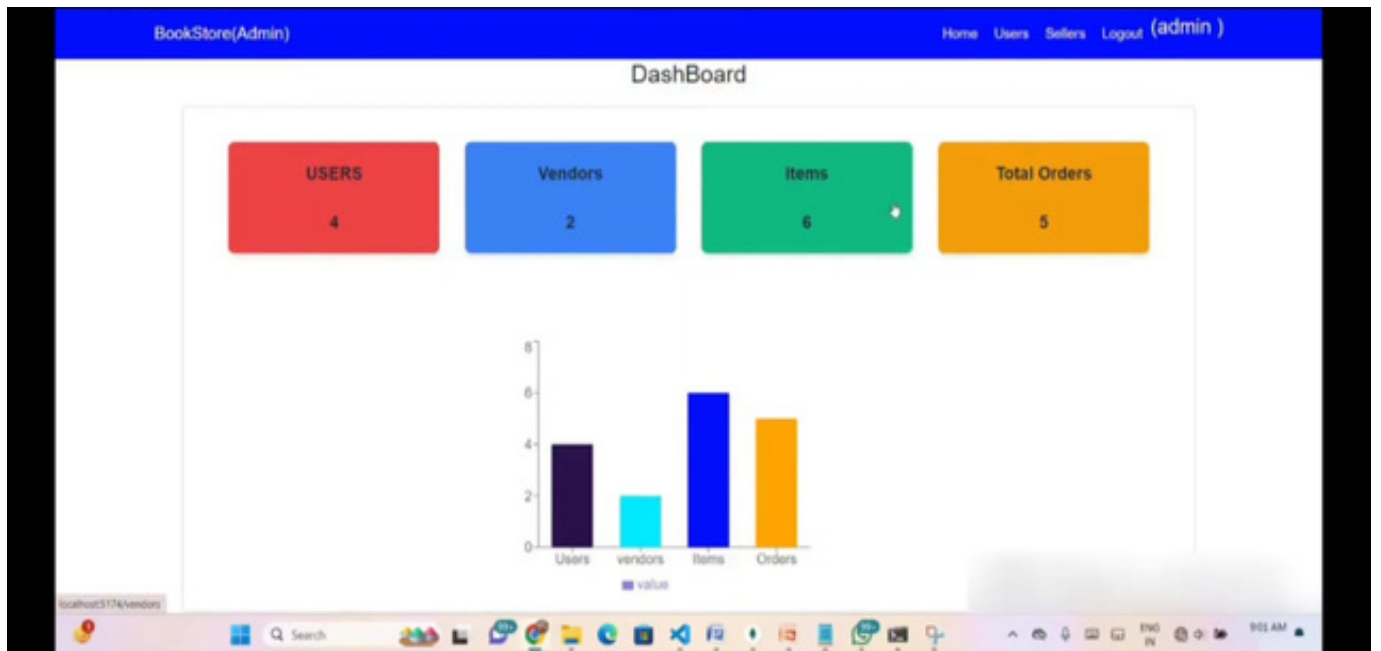
Users

| slno | Userid | User name | Email | Operation |
|------|--------------------------|-----------|-------------------|--|
| 1 | 655d964c4f4ace5f198b9abf | anshad | anshad@gmail.com |   view |
| 2 | 655e571802e8144c8a9cb27f | shivani | shivani@gmail.com |   view |
| 3 | 6580442915b2ba8f503a659 | syed | syed@gmail.com |   view |
| 4 | 6600ec03677e192e578b249b | test | test@gmail.com |   view |



Search





ORDERING BOOKS :

BookStore Home Books Wishlist My orders Logout(test)


Your order is almost Done!

Address:

Flat no: 31/122-4

City: I Pincode:

State:

 -dila1

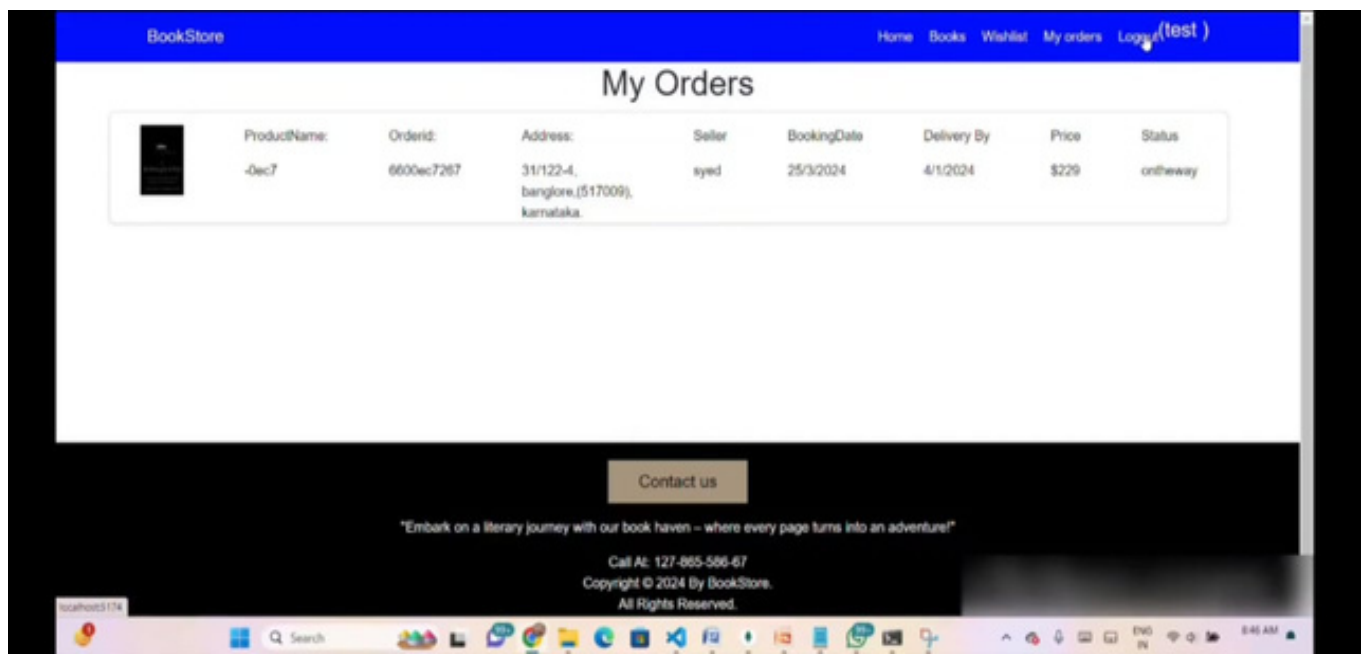
Price: 130

Delivery: Free

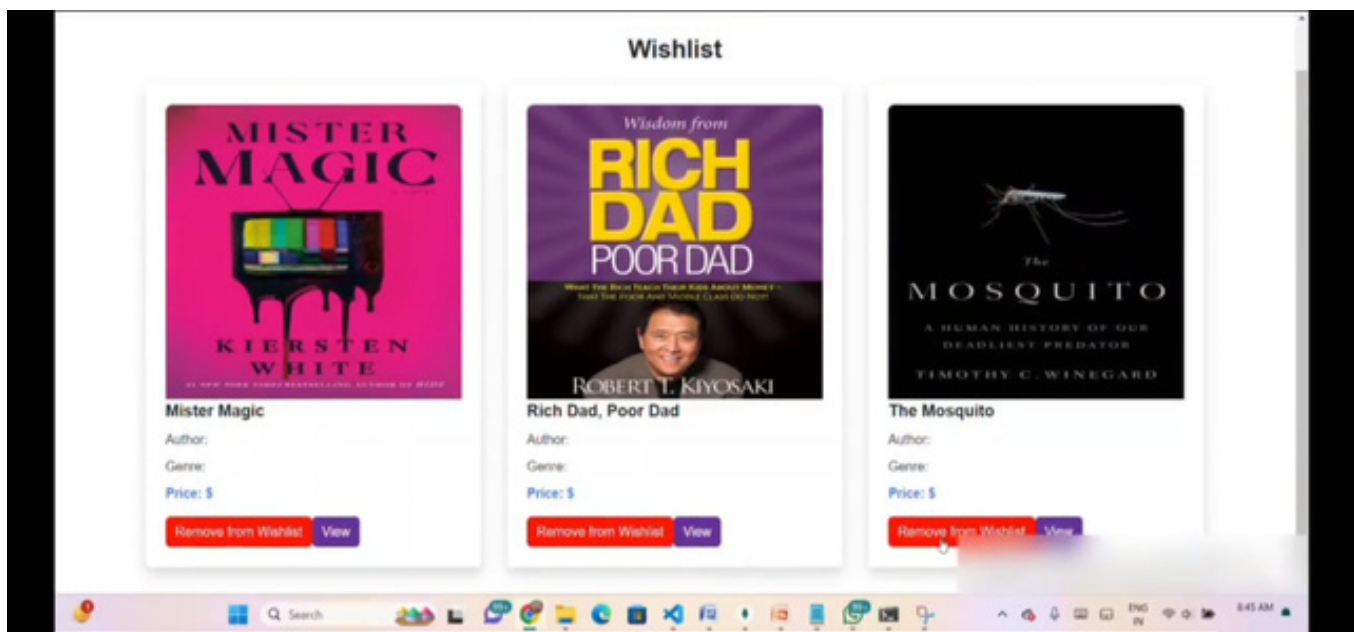
Total Amount: 145

Order

8:45 AM



WISHLIST :



8 ADVANTAGES & DISADVANTAGES

Advantages:

- **User-Friendly Interface:** The app offers an easy-to-navigate design, making appointment booking simple for patients of all ages.
- **Real-Time Availability:** Live updates of doctor availability reduce scheduling conflicts and improve user experience.
- **Secure Data Handling:** Use of JWT and bcrypt ensures user data and sensitive medical records are safely stored and transmitted.
- **Role-Based Access Control:** Different user roles (User, Seller, admin) get customized access, improving security and functionality.
- **Automated Notifications:** Email and SMS reminders help reduce missed appointments and improve communication.
- **Scalable Architecture:** Using MongoDB and Express.js supports future growth and increased user demand.

Disadvantages:

- **Internet Dependence:** The app requires stable internet connectivity, limiting access in areas with poor coverage.

- **Complex Backend Management:** Admins need technical knowledge to manage registrations and oversee platform governance.
- **Document Upload Limitations:** Currently, only limited file types and sizes are supported for document uploads.
- **No Offline Mode:** Users cannot access appointment information or book without an internet connection.

CONCLUSION

The BookNest project successfully addresses the challenges of modern storytelling and content engagement by offering a digital platform that connects readers and authors in meaningful ways. Throughout our summer internship, the team collaboratively ideated, designed, developed, and tested the system to ensure it is secure, scalable, and user-centric.

The platform's core features—emotion-based story discovery, personalized Nests, secure story uploads, real-time recommendations, and admin moderation—were implemented with attention to usability and creative freedom. By leveraging full-stack development tools and sentiment-driven recommendation techniques, the app enhances story visibility, reader satisfaction, and author growth.

Overall, this project provided valuable hands-on experience in user-centered design, database structuring, and community-driven application logic, equipping us for future roles in web development, digital publishing, and content technology.

1. FUTURE SCOPE

☒ **Mobile Application Development:** Build native Android and iOS apps to improve access and reading convenience across devices.

- ☒ **Audio Storytelling Feature:** Introduce voice narration for select stories to enhance accessibility and storytelling immersion.
- ☒ **AI-Based Story Recommendations:** Use machine learning algorithms to suggest stories based on user preferences, emotion patterns, and reading behavior.
- ☒ **Advanced Author Analytics Dashboard:** Enable authors to track reader engagement trends, feedback metrics, and story performance over time.
- ☒ **Multi-language Support:** Expand reach by supporting regional and global languages to promote diverse storytelling.
- ☒ **Story Monetization Options:** Allow verified authors to monetize popular content through reader subscriptions or sponsorships.
- ☒ **Offline Reading and Sync:** Provide offline story access with automatic data sync and reading progress updates when back online.

1. APPENDIX

- ☐ **Source Code:** The complete source code for the BookNest application will be uploaded and maintained in the designated GitHub repository once deployment is finalized.
- ☐ **Dataset Link:** The database schema and sample emotion-tagged story datasets used during development and testing are included in the same repository to support reproducibility and future improvements.

GitHub & Project Demo Link:

GitHub Repository: <https://github.com/alsaba18/BookNest-Where-Stories-Nestle>

Project Demo: <https://youtu.be/PG8lyu4RXuA?si=dbXaAaEoYLM84WdO>