Full Stack Development with MERN -- BOOKSTORE

FINAL REPORT

**Project Title: Bookstore App – BookNest**

**TEAM ID:-** SWTID1743519094

# Team Members:

# Tanya Maheshwari – 22BCE10169

# Neha Pal- 22BCE10367

# Akshit Singh:- 22BCE10386

# Shreyansh Sharma:- 22BCE11026

**1. INTRODUCTION**

* 1. Project Overview

The Bookstore App is a full-stack web application designed to facilitate digital book management. The system allows users to register, log in, and perform full CRUD operations on books in a responsive and secure environment.

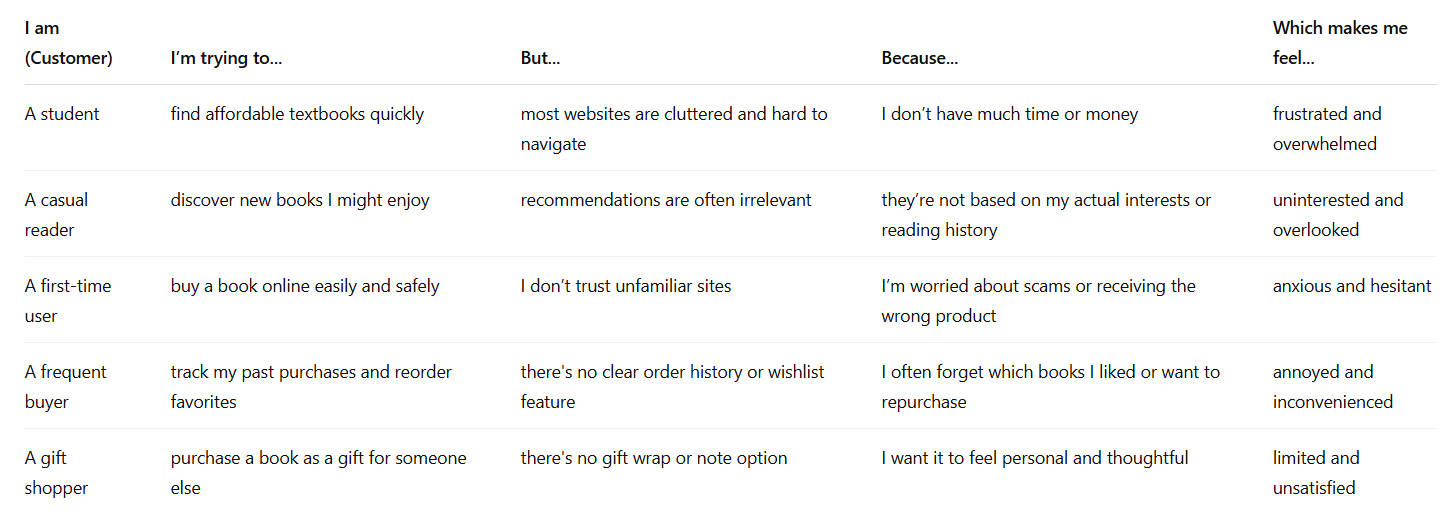
* 1. Purpose

The primary objective of this project is to digitize book management through an intuitive web interface, supporting operations such as searching, viewing, updating, and ordering books with secure authentication.

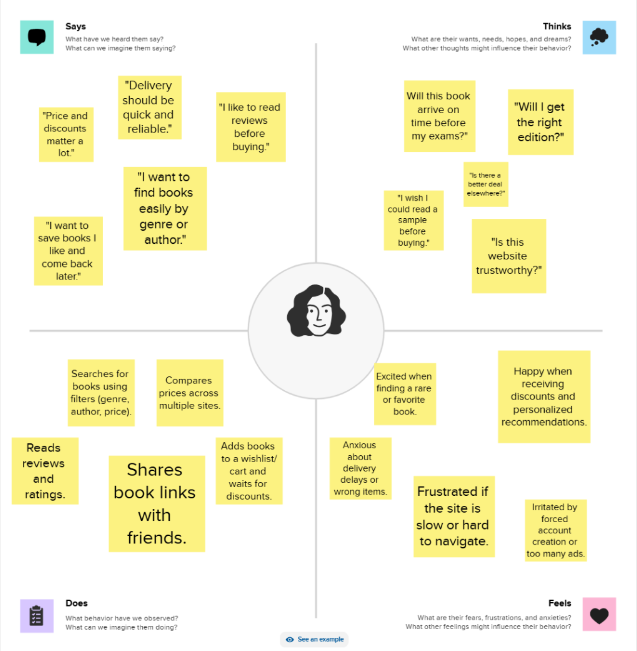
**2. IDEATION PHASE**

* 1. Problem Statement

Users need a reliable and easy-to-use platform to manage and purchase books online. The lack of centralized digital solutions results in poor user experience and book accessibility.

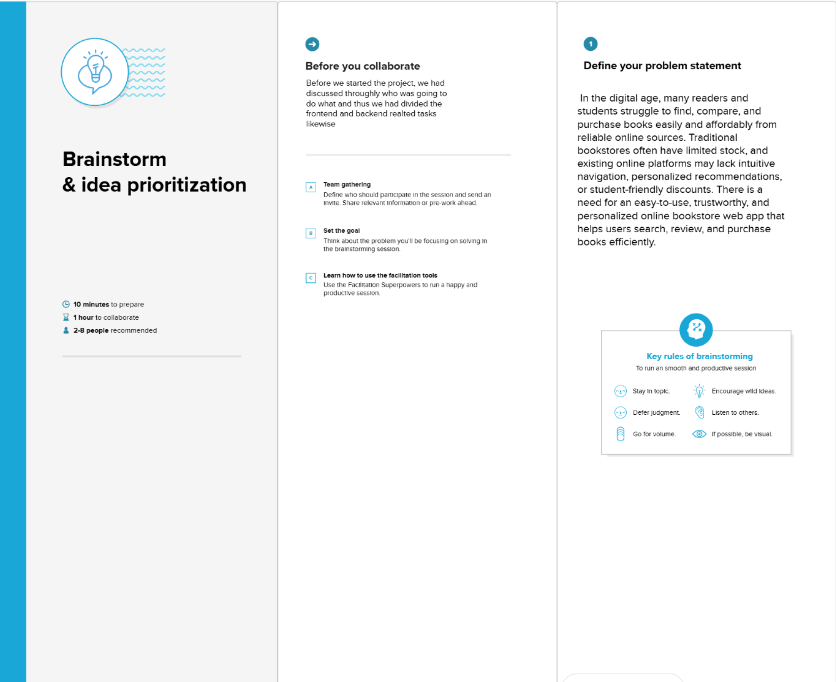


* 1. Empathy Map Canvas

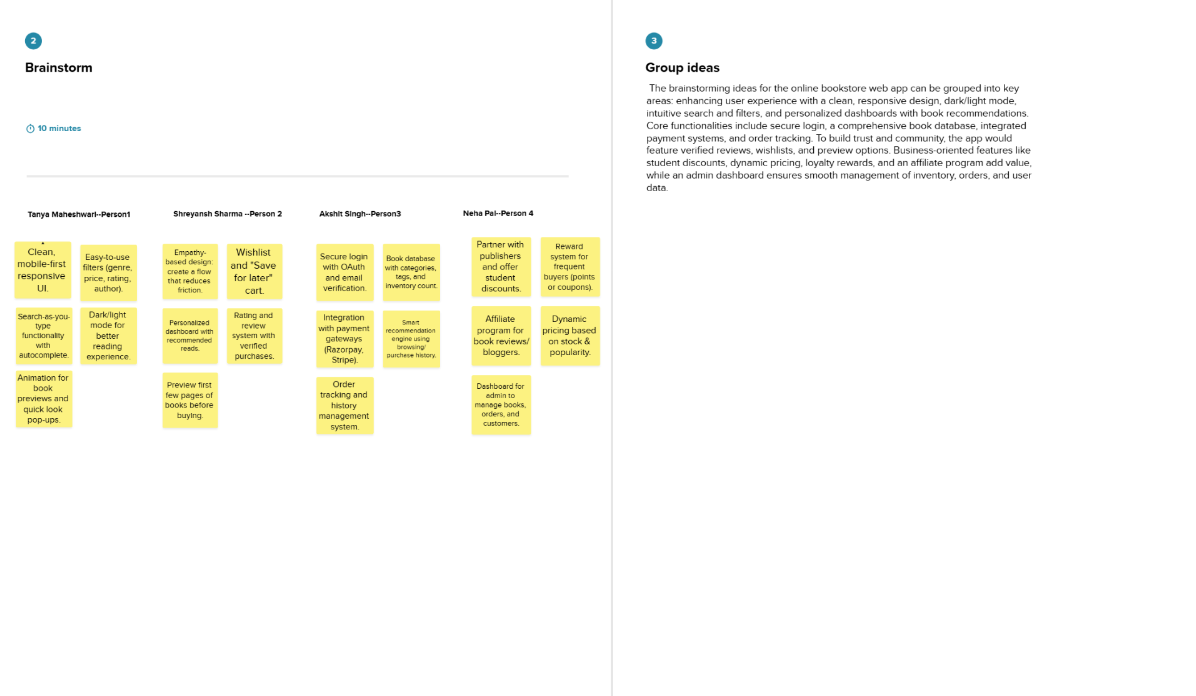


* + - Think & Feel: Users want fast access to books.
    - Hear: Recommendations influence user decisions.
    - See: Clean UI.
    - Say & Do: Add books, manage orders.
    - Pain: Inaccurate suggestions.
    - Gain: Seamless ordering
  1. Brainstorming

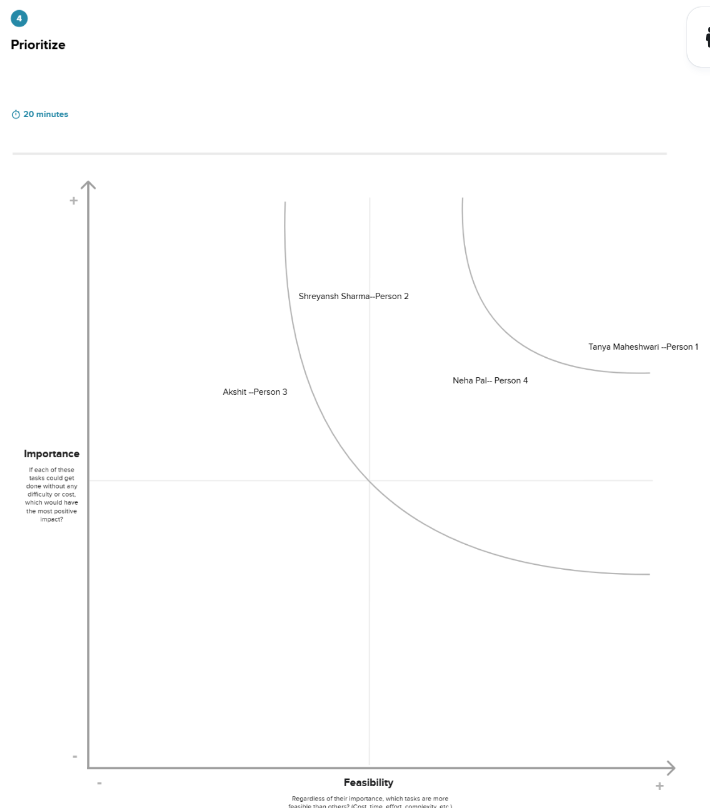
**Step-1: Team Gathering, Collaboration and Select the Problem Statement**



**Step-2: Brainstorm, Idea Listing and Grouping**



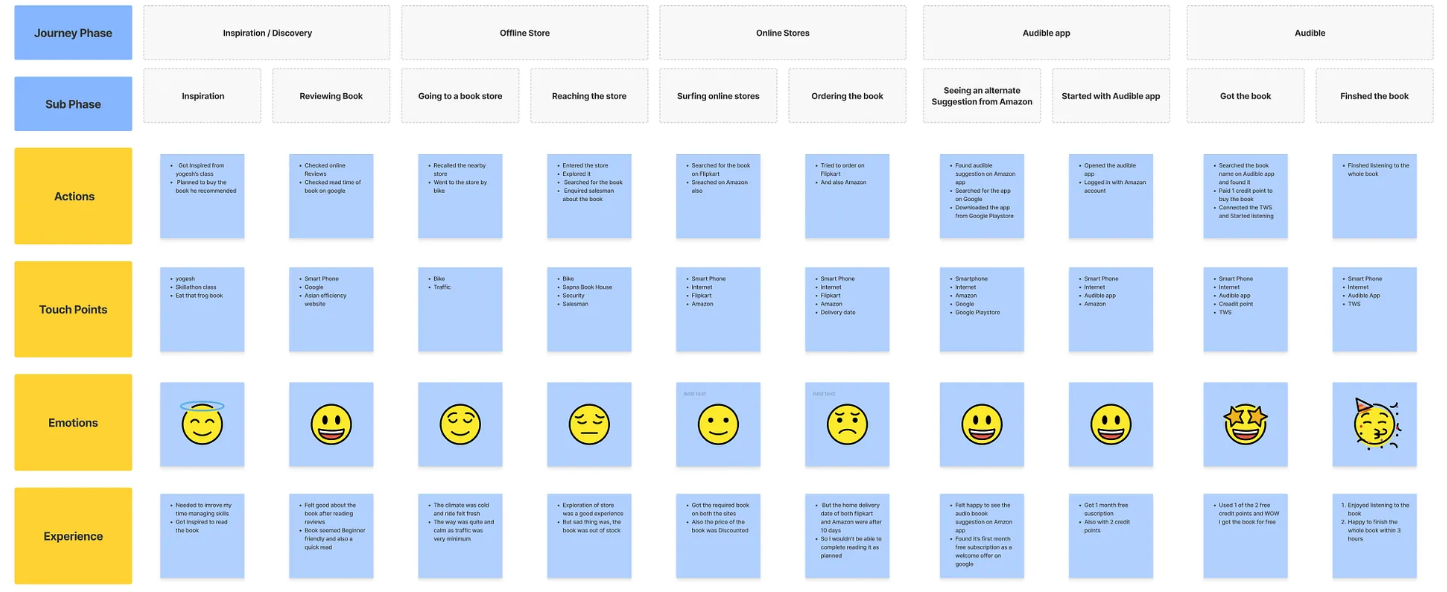
**Step-3: Idea Prioritization**



Explored multiple stack options. MERN was chosen for its flexibility, ease of use, and scalability.

**3. REQUIREMENT ANALYSIS**

* 1. Customer Journey Map



* 1. Solution Requirement

**Functional Requirements:**

Following are the functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | Registration through Form  Registration through Gmail  Registration through LinkedIN |
| FR-2 | User Confirmation | Confirmation via Email  Confirmation via OTP |
| FR-3 | Book Management | Add New Book (Admin)  Update Book Details (Admin)  Remove Book from Catalog (Admin)  View Book Details (User)  Search and Filter Books (User) |
| FR-4 | Order and Cart Management | Add Book to Cart  Remove Book from Cart  Modify Quantity in Cart  Place Order  View Order History |

**Non-functional Requirements:**

Following are the non-functional requirements of the proposed solution.

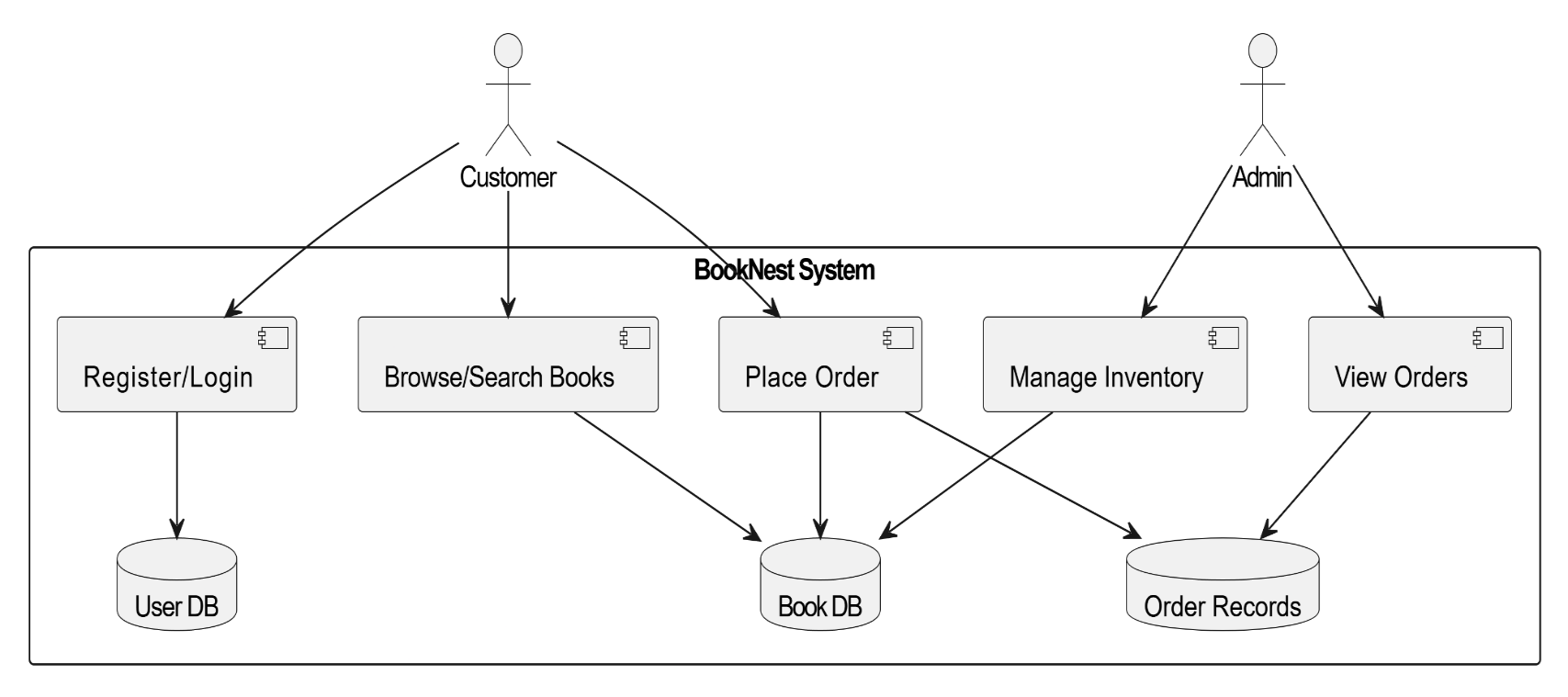
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The website should have a user-friendly interface that is easy to navigate, with intuitive features such as a search bar, filters for books by genre/author, and a clear checkout process to enhance the user experience. |
| NFR-2 | **Security** | User data must be protected through secure authentication mechanisms (e.g., JWT), data encryption, and secure APIs. Payment details should be handled through trusted third-party gateways. |
| NFR-3 | **Reliability** | The application must function consistently without failures. It should ensure that book listings, cart data, and orders are processed and saved accurately, even under load. |
| NFR-4 | **Performance** | The website should have fast response times, with optimized database queries, efficient frontend rendering using React, and backend performance tuning to handle concurrent users smoothly. |
| NFR-5 | **Availability** | The system should ensure high uptime through cloud deployment with redundancy. Users should be able to access the bookstore at any time with minimal disruptions. |
| NFR-6 | **Scalability** | The system should support horizontal scaling to accommodate growing numbers of books, users, and transactions using technologies like Node.js clusters and MongoDB sharding. |

The application requires:

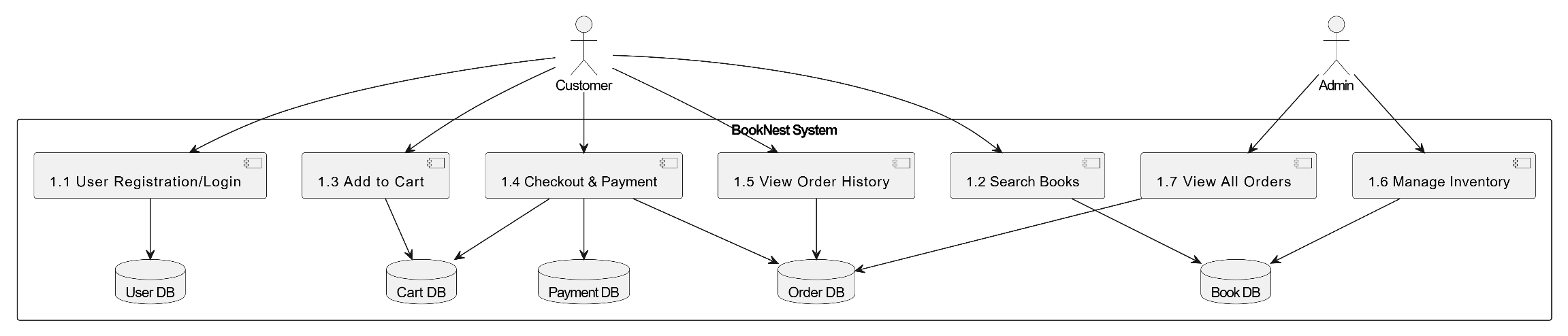
* + - Secure authentication
    - CRUD APIs
    - MongoDB persistence
    - Role-based access (future scope)
    - Responsive design
  1. Data Flow Diagram

**Data Flow Diagrams:**

DFD Level 0

****

**DFD level 1**

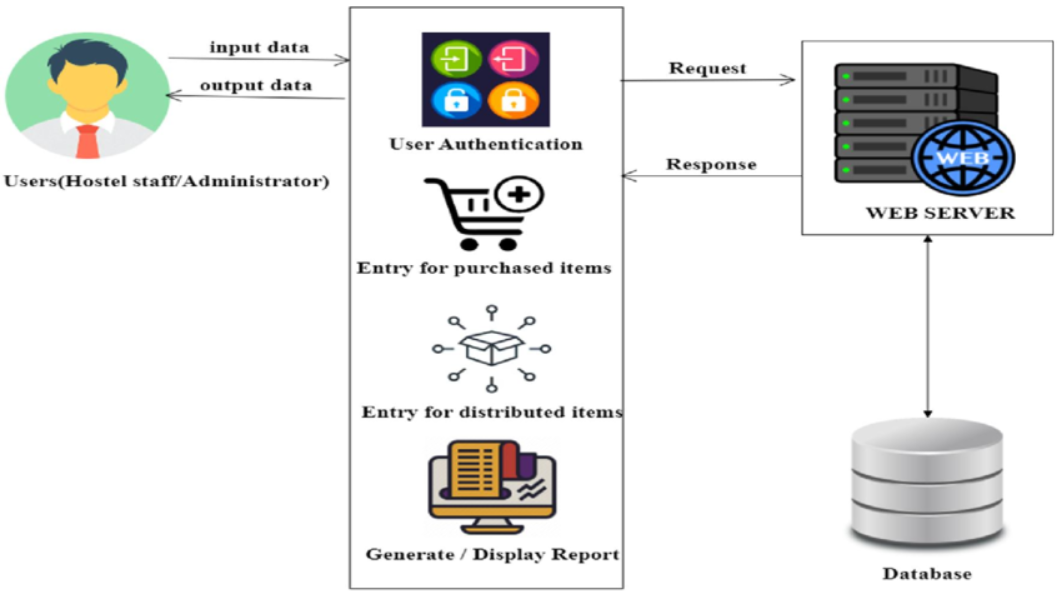
****

**User Stories**

| **User Type** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Acceptance Criteria** | **Priority** | **Release** |
| --- | --- | --- | --- | --- | --- | --- |
| Customer (Web) | Registration | USN-1 | As a user, I can register with email and password. | I can access my account after successful registration. | High | Sprint-1 |
| Customer (Web) | Registration | USN-2 | As a user, I receive a confirmation email after registering. | I can confirm my email by clicking the link. | High | Sprint-1 |
| Customer (Web) | Login | USN-3 | As a user, I can login with valid credentials. | I can access my dashboard. | High | Sprint-1 |
| Customer (Web) | Browse/Search Books | USN-4 | As a user, I can browse and search for books by title or author. | I can view relevant book listings. | High | Sprint-1 |
| Customer (Web) | Cart & Checkout | USN-5 | As a user, I can add books to cart and proceed to checkout. | I can place an order. | High | Sprint-2 |
| Customer (Web) | Order History | USN-6 | As a user, I can view my previous orders. | I can see a list of past orders. | Medium | Sprint-2 |
| Admin | Manage Inventory | USN-7 | As an admin, I can add, update, or delete books from inventory. | I can maintain the catalog. | High | Sprint-1 |
| Admin | Manage Users | USN-8 | As an admin, I can view and manage registered users. | I can access and modify user information. | Medium | Sprint-2 |
| Admin | View Orders | USN-9 | As an admin, I can view all orders placed by users. | I can track all sales and order records. | High | Sprint-2 |

* 1. Technology Stack

**Technical Architecture:**

****

**Table-1 : Components & Technologies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Component** | **Description** | **Technology** |
|  | User Interaction Layer | Mode through which end-users engage with the system, such as web platforms, mobile applications, or conversational agents | HTML, CSS, JavaScript / Angular Js / React Js etc. |
|  | Logic Layer-1 | Code responsible for handling specific functions of the application | Java / Python |
|  | Logic Layer-2 | Module managing speech input conversion | IBM Watson STT service |
|  | Logic Layer-3 | Handles chatbot and interaction-based tasks | IBM Watson Virtual Agent |
|  | Database | Database schema and data structuring approaches | MySQL, NoSQL, etc. |
|  | Cloud Database | Managed database platforms on cloud | IBM DB2, IBM Cloudant etc. |
|  | File Handling | Managing and maintaining user files and documents | IBM Block Storage or Other Storage Service or Local Filesystem |
|  | External API-1 | External service integration required in the solution | IBM Climate API or equivalent |
|  | External API-2 | Another external system linked for validation or identity servicesn | Aadhar API, etc. |
|  | Machine Learning Model | AI component purpose within the system | Object Recognition Model, etc. |
|  | Infrastructure (Server / Cloud) | Application hosting, deployment environments | Local, Cloud Foundry, Kubernetes, etc. |

**Table-2: Application Characteristics:**

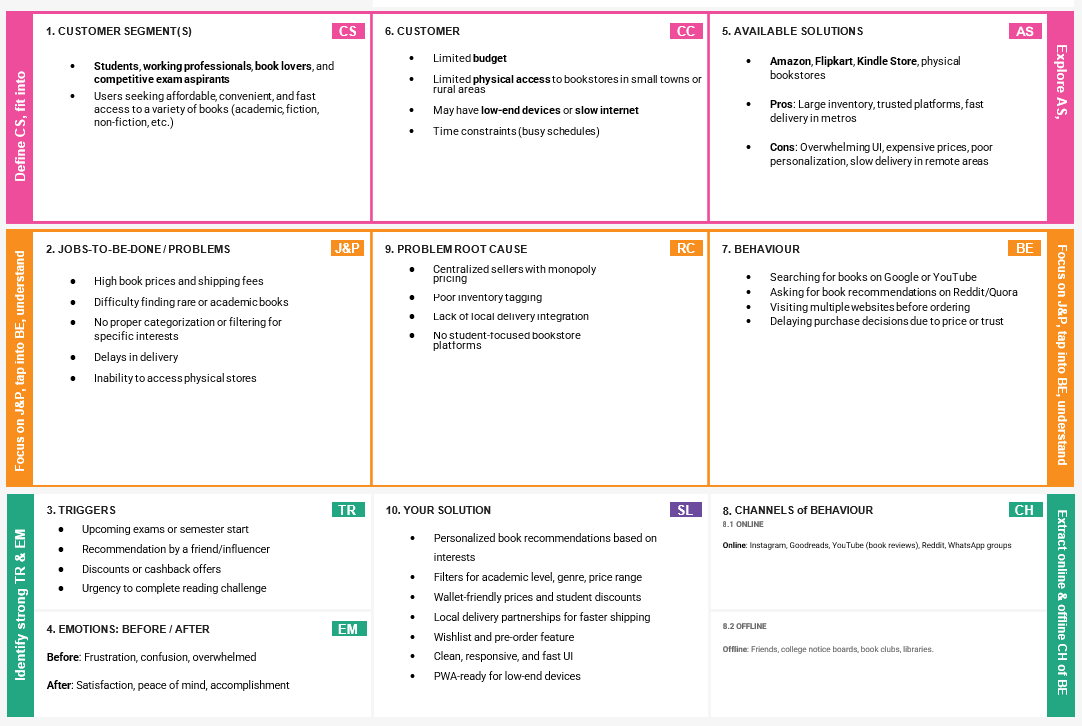
| **S.No** | **Characteristics** | **Description** | **Technology** |
| --- | --- | --- | --- |
|  | Community-Driven Tools | Open-source tools or ecosystems leveraged during development | Technologies from open-source ecosystems |
|  | Security Implementations | List all the security / access controls implemented, use of firewalls etc. | Techniques such as SHA-256, IAM Policies, Data Encryption, OWASP Compliance |
|  | Expandable Design | Explain the mechanisms ensuring system uptime (e.g., load balancers, redundancy setup) | Technology adopted for modular scaling |
|  | Availability | Justify the availability of applications (e.g. use of load balancers, distributed servers etc.) | Tools and infrastructure ensuring high availability |
|  | Efficiency & Speed | Design measures taken to improve response rate (e.g., caching, CDN, request handling) | Performance optimization technologies |

Frontend: React.js

Backend: Node.js + Express.js Database: MongoDB + Mongoose Auth: Firebase

**4. PROJECT DESIGN**

* 1. Problem Solution Fit



The application addresses the need for a seamless digital bookstore by combining a robust backend with an intuitive frontend.

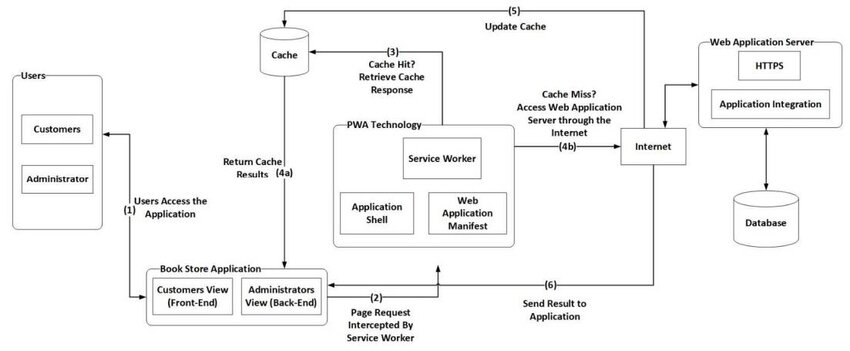
* 1. Proposed Solution

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Parameter** | **Description** |
| 1. | Problem Statement (Problem to be solved) | Many students, readers, and professionals struggle to find, compare, and purchase books online due to scattered platforms, lack of personalized recommendations, and limited regional access to titles. Local bookstores often lack digital presence, making it harder for users to buy or explore easily. |
|  | Idea / Solution description | We propose a full-featured online bookstore using the MERN stack (MongoDB, Express, React, Node.js). The platform will provide:  • User authentication  • Book browsing and search  • Wishlist and cart management  • Online payment integration  • Admin panel for inventory and orders  • Delivery tracking & notifications. |
|  | Novelty / Uniqueness | • Seamless user experience with fast, modern UI using React  • Personalized book suggestions  • Integration of delivery & tracking APIs  • Responsive Progressive Web App (PWA) design  • Optional student-focused features like textbook exchanges or study bundles.  • Cloud-hosted scalable backend with auto-deployment pipelines. |
|  | Social Impact / Customer Satisfaction | • Promotes reading by making books more accessible  • Supports local/regional bookstores with digital listing opportunities  • Offers a clutter-free, fast and mobile-friendly shopping experience  • Student-friendly pricing options and easy book discoverability  • Eco-friendly impact with digital management over physical stores. |
|  | Business Model (Revenue Model) | • Commission-based model on each book sale  • Subscription plan for premium members (discounts, early access)  • Ad space for publishers or featured listings  • Seller plans for local bookstore owners to list inventory |
|  | Scalability of the Solution | • Built on scalable cloud infrastructure (MongoDB Atlas, Render/Vercel)  • Easily supports horizontal scaling as traffic grows  • Microservice-friendly backend architecture  • Expandable into mobile apps, audiobook marketplace, or eBook reader support |

The web-based platform supports user login, personalized recommendations, cart management, and secure order handling.

* 1. Solution Architecture

**Solution Architecture Diagram:**



*Figure 1: Architecture and data flow of the Bookstore MERN Application*

**5. PROJECT PLANNING & SCHEDULING**

5.1 Project Planning

**Product Backlog, Sprint Schedule, and Estimation**

| **Sprint** | **Functional Requirement (Epic)** | **User Story Number** | **User Story / Task** | **Story Points** | **Priority** | **Team Members** |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | Registration | USN-1 | As a user, I can register using email and password. | 2 | High | Aryan Sharma, Ansh Bansal |
| Sprint-1 | Registration | USN-2 | As a user, I receive confirmation email after registration. | 1 | High | Aryan Sharma |
| Sprint-1 | Registration (Gmail) | USN-4 | As a user, I can register through Gmail. | 2 | Medium | Shivam Pant |
| Sprint-1 | Login | USN-5 | As a user, I can login with email and password. | 1 | High | Ansh Bansal |
| Sprint-1 | Search/Browse Books | USN-6 | As a user, I can search or browse books. | 3 | High | Aryan Sharma,  Shivam Pant |
| Sprint-2 | Cart | USN-7 | As a user, I can add books to cart. | 2 | High | Shivam Pant,  Ansh Bansal |
| Sprint-2 | Checkout | USN-8 | As a user, I can place orders from my cart. | 3 | High | Aryan Sharma,  Ansh Bansal |
| Sprint-2 | Order Confirmation | USN-9 | As a user, I receive confirmation of my order. | 1 | Medium | Ansh Bansal |
| Sprint-2 | Order History | USN-10 | As a user, I can view my order history. | 2 | Medium | Shivam Pant |
| Sprint-3 | Admin Inventory | USN-11 | As admin, I can add/edit/delete books. | 3 | High | Aryan Sharma |
| Sprint-3 | Admin Order Management | USN-12 | As admin, I can view and manage all user orders. | 2 | Medium | Ansh Bansal,  Aditya Poduval |
| Sprint-4 | Wishlist | USN-13 | As a user, I can add books to wishlist. | 2 | Low | Shivam Pant |
| Sprint-4 | Customer Support Request | USN-14 | As a user, I can raise support requests for order or delivery issues. | 2 | Medium | Aryan Sharma,  Aditya Poduval |

**Project Tracker, Velocity & Burndown Chart:**

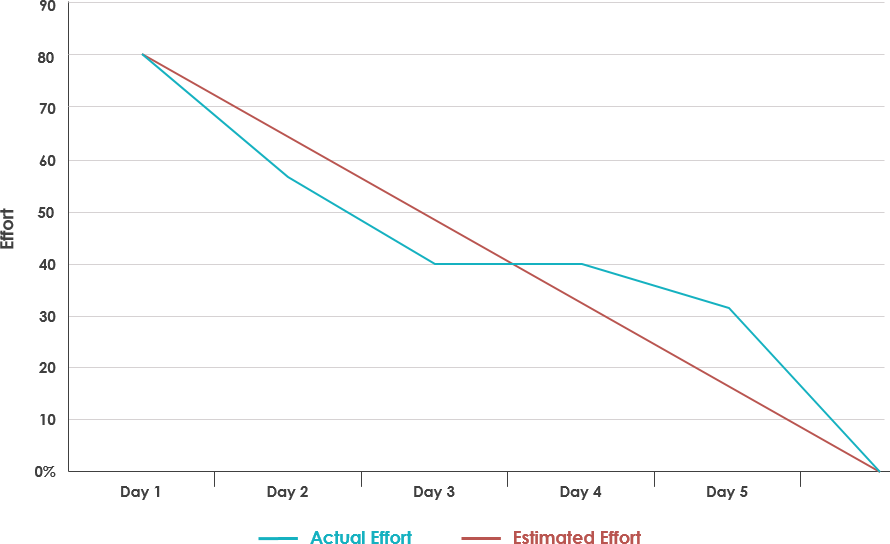
| Sprint | Total Story Points | Duration | Sprint Start Date | Sprint End Date (Planned) | Story Points  Completed | Sprint Release Date |
| --- | --- | --- | --- | --- | --- | --- |
| Sprint-1 | 9 | 6 Days | 12 Mar 2025 | 17 Mar 2025 | 9 | 17 Mar 2025 |
| Sprint-2 | 8 | 6 Days | 19 Mar 2025 | 24 Mar 2025 | - | - |
| Sprint-3 | 5 | 6 Days | 26 Mar 2025 | 31 Mar 2025 | - | - |
| Sprint-4 | 4 | 6 Days | 03 Apr 2025 | 08 Apr 2025 | - | - |

**Velocity:**



**Burndown Chart:**

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile[software development](https://www.visual-paradigm.com/scrum/what-is-agile-software-development/) methodologies such as [Scrum](https://www.visual-paradigm.com/scrum/scrum-in-3-minutes/). However, burn down charts can be applied to any project containing measurable progress over time.



The project followed an agile methodology with 4-week sprints. Each sprint focused on setup, core feature development, integration, and testing.

**6. FUNCTIONAL AND PERFORMANCE TESTING**

* 1. Performance Testing

Project Description: Book Nest is a feature-rich online bookstore web application designed to offer users a seamless, personalized, and secure book-buying experience. The platform provides an intuitive UI/UX with essential functionalities like user authentication and authorization, a personalized dashboard, cart and order management, and an order history page. Users can explore curated sections such as Top Sellers and Personalized Recommendations tailored to their reading habits. Designed for both casual readers and frequent buyers, Book Nest ensures quick navigation, trustworthy transactions, and smart book discovery — making it a one-stop solution for all book lovers.

Testing Period: 28-03-2025 to 30-04-2025

**Testing Scope:**

**List of Features and Functionalities to be tested**

1. **User Authentication & Authorization**
   * Sign up, login, logout
2. **User Dashboard**
   * Personalized recommendations
3. **Book Browsing**
   * Search functionality
   * View book details
4. **Cart & Checkout**
   * Add to cart, remove from cart, update quantity
   * Checkout and place order
5. **Order Management**
   * View order history
   * Track current orders
6. **Top Sellers Page**
   * Display best-selling books
   * Sorting and filtering
7. **Recommendation System**
   * Suggestions based on user history or preferences
8. **Admin Panel (if applicable)**
   * Manage books, users, and orders
   * Add/edit/delete books
9. **Responsive Design**
   * Compatibility with different screen sizes and browser

**Testing Environment:**

* **Frontend Stack**: React.js
* **Backend Stack**: Node.js / Express.js with MongoDB
* **Testing Tools**: Postman, Chrome DevTools
* URL/Location: localhost

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Case ID | Test Scenario | Test Steps | Expected Result | Actual Result | Pass/Fail |
| TC-001 | User Registration | 1. Navigate to the sign-up page2. Enter valid details3. Click 'Register' | Account should be created and user redirected to dashboard | Account created and redirected successfully | Pass |
| TC-002 | User Login with Valid Credentials | 1. Go to login page2. Enter valid email and password3. Click 'Login' | User should be logged in and directed to dashboard | Login successful and dashboard loaded | Pass |
| TC-003 | Add Book to Cart | 1. Search/select a book2. Click 'Add to Cart' | Book should appear in cart | Book added to cart and shown in cart page | Pass |
| TC-004 | View Order History | 1. Log in2. Go to 'Order History' from dashboard | List of previous orders should be displayed | Order history displayed | Pass |
| TC-005 | View Personalized Recommendations | 1. Log in2. Visit 'Recommendations' page | Books based on user history should be displayed | Irrelevant books shown | Fail |

**Bug Tracking:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Bug ID** | **Bug Description** | **Steps to Reproduce** | **Severity** | **Status** | **Additional Feedback** |
| BG-001 | Irrelevant books in recommendations section | 1. Log in with test user2. Go to 'Recommendations'3. Observe book list | Medium | Open | Books shown are not related to user's genre |
| BG-002 | Cart not updating item quantity | 1. Add book to cart2. Increase quantity3. Click 'Update Cart' | High | In Progress | Quantity stays the same after clicking update |
| BG-003 | Broken link in footer | 1. Scroll to footer2. Click ‘Contact’ link | Low | Open | Leads to 404 page |

Testing was conducted using Postman and Chrome DevTools. Test cases included user login, adding books to cart, and viewing order history.

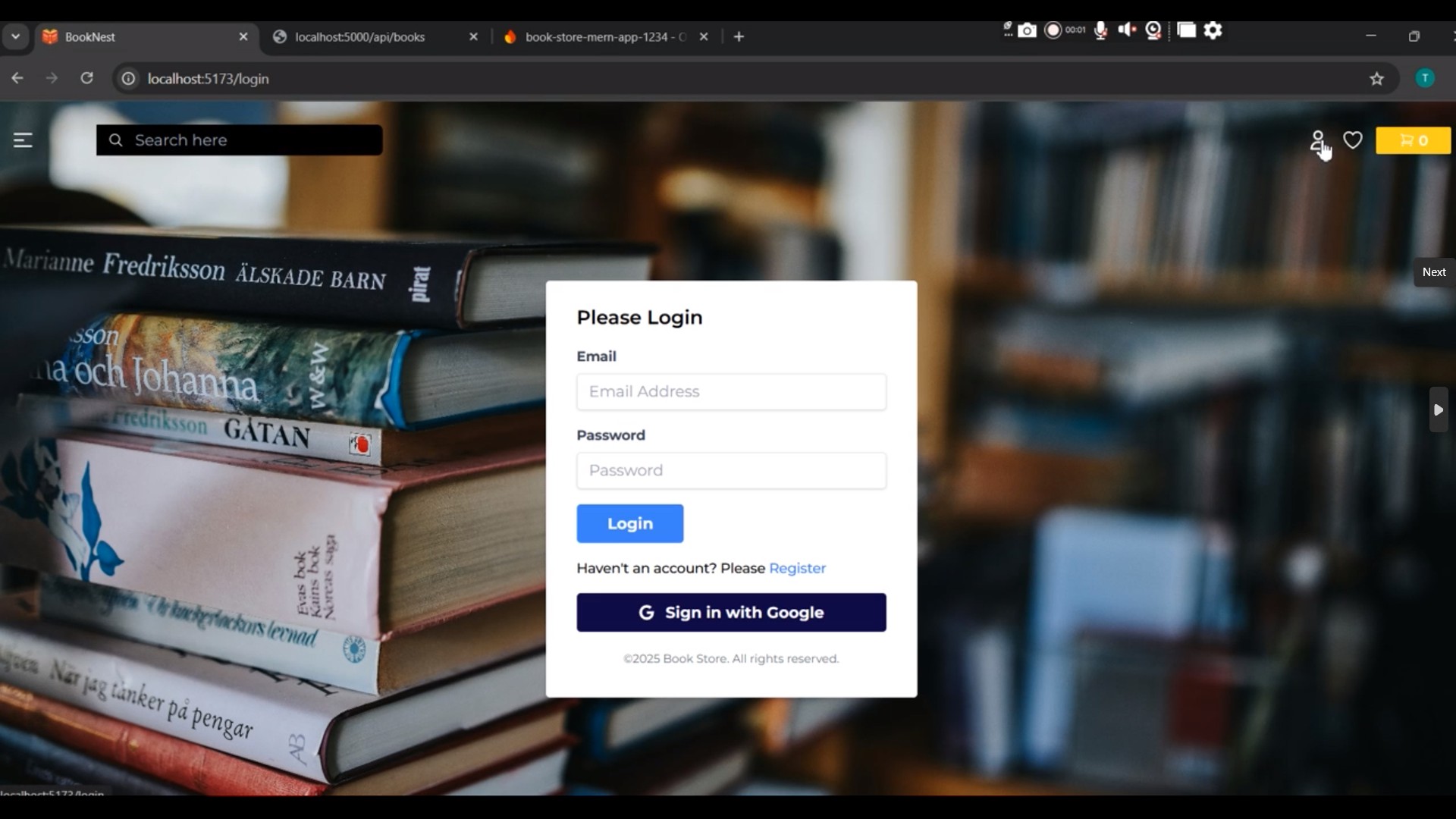
Bug examples:

* + - Broken footer links
    - Redundant re-renders
    - Cart update issues

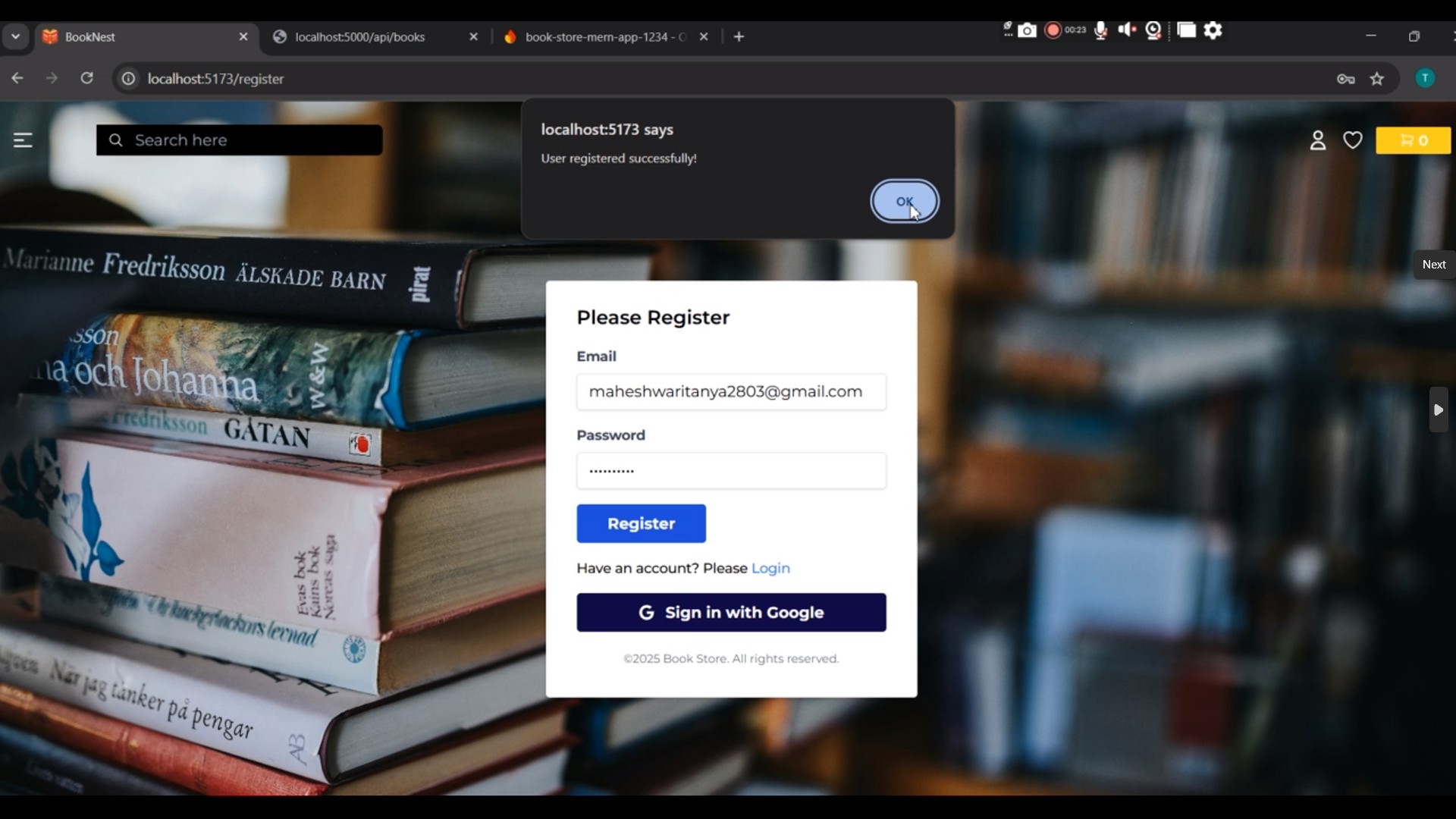
**7. RESULTS**

* 1. Output Screenshots

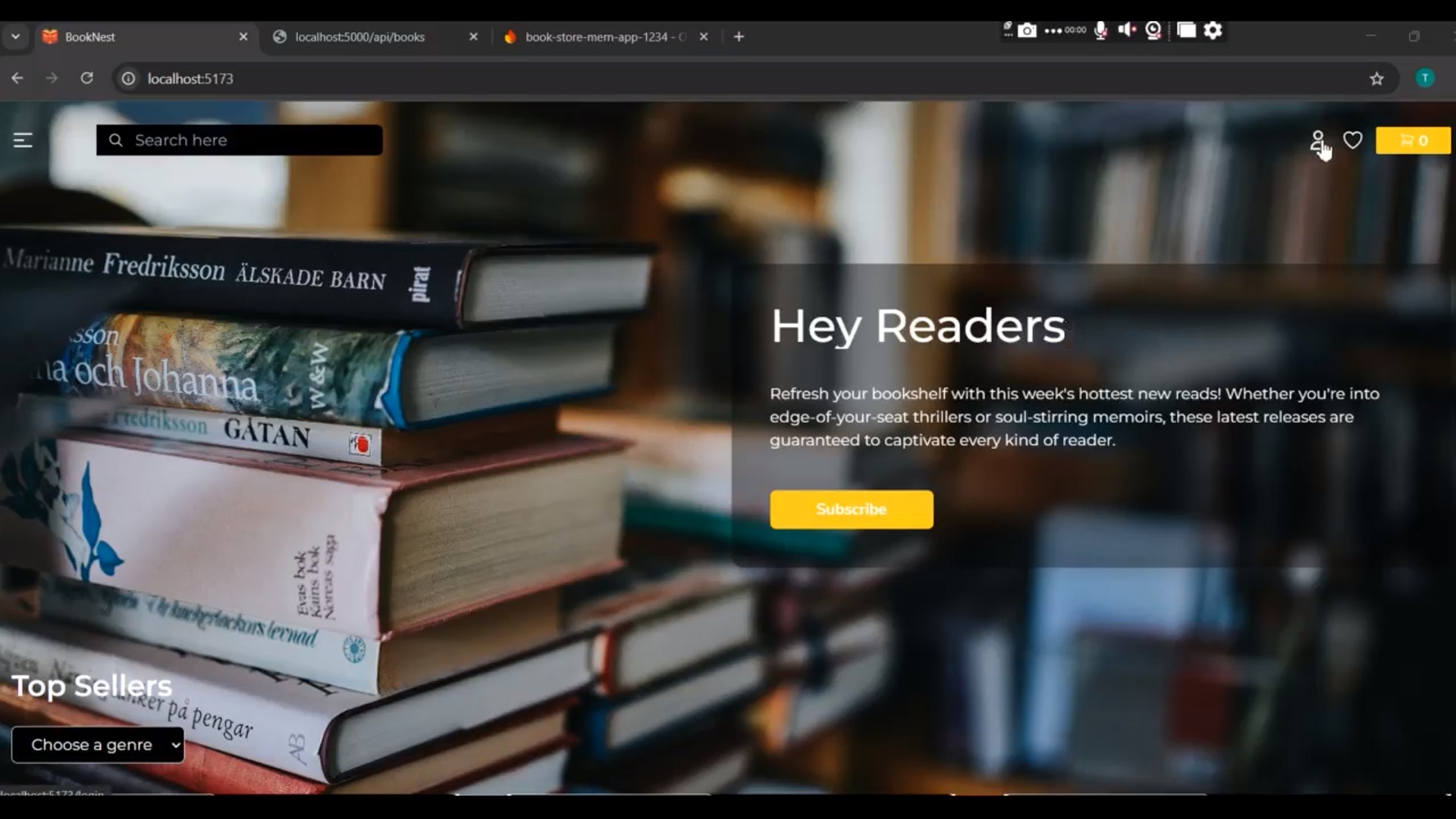
1. Initial Login/ Registration Page :



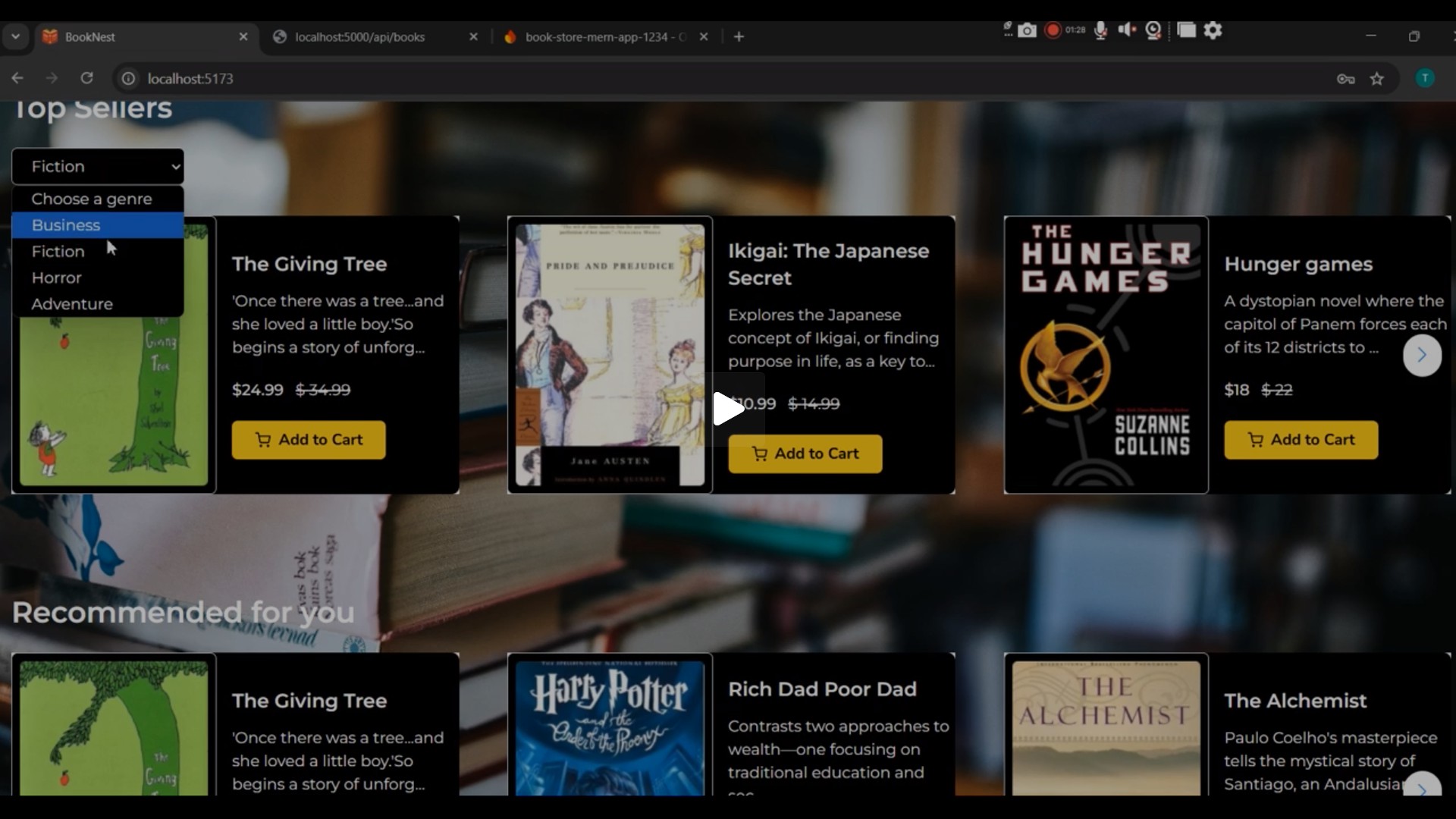
2. Upon Successful Registration:



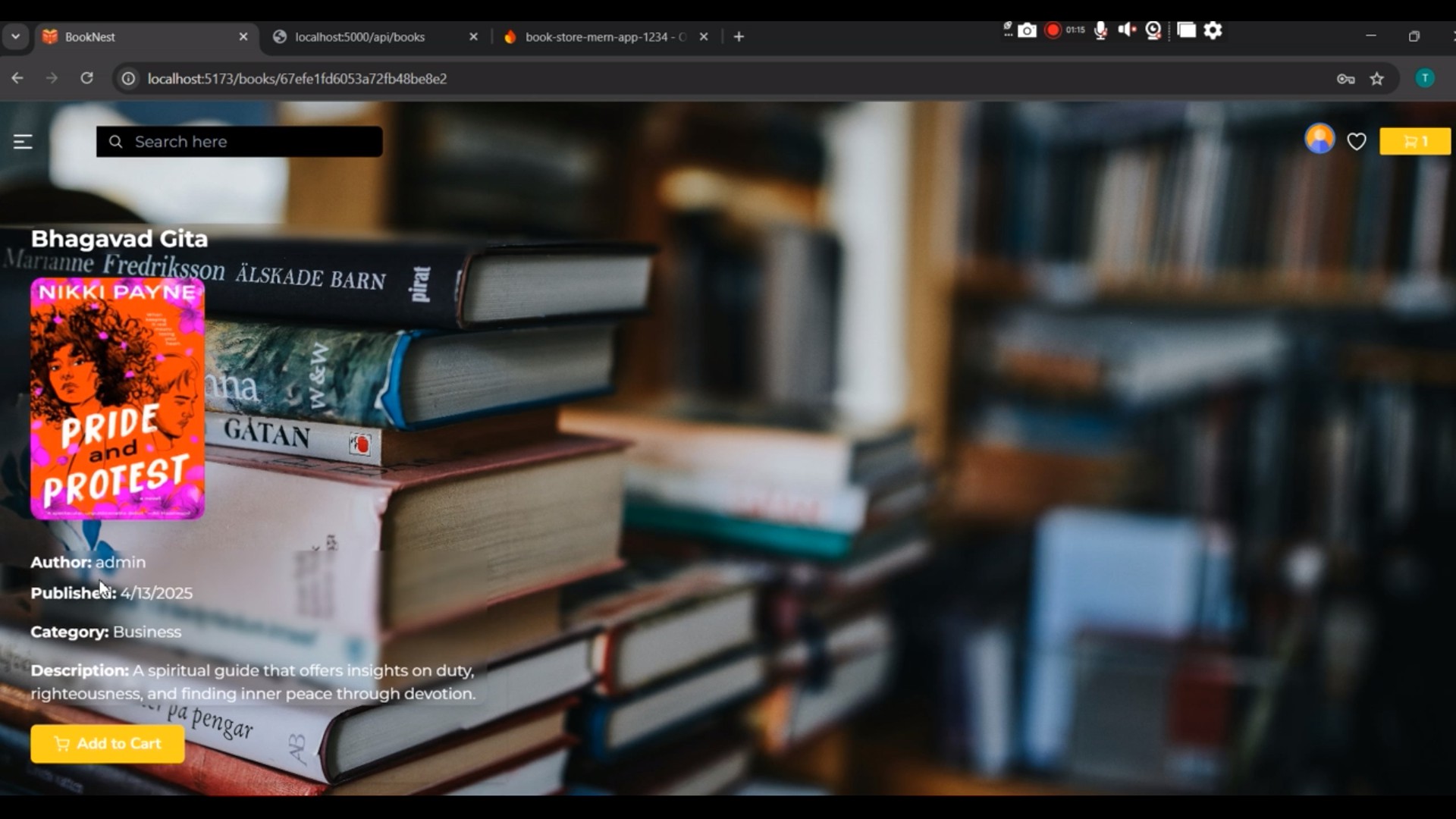
1. Home Page UI:



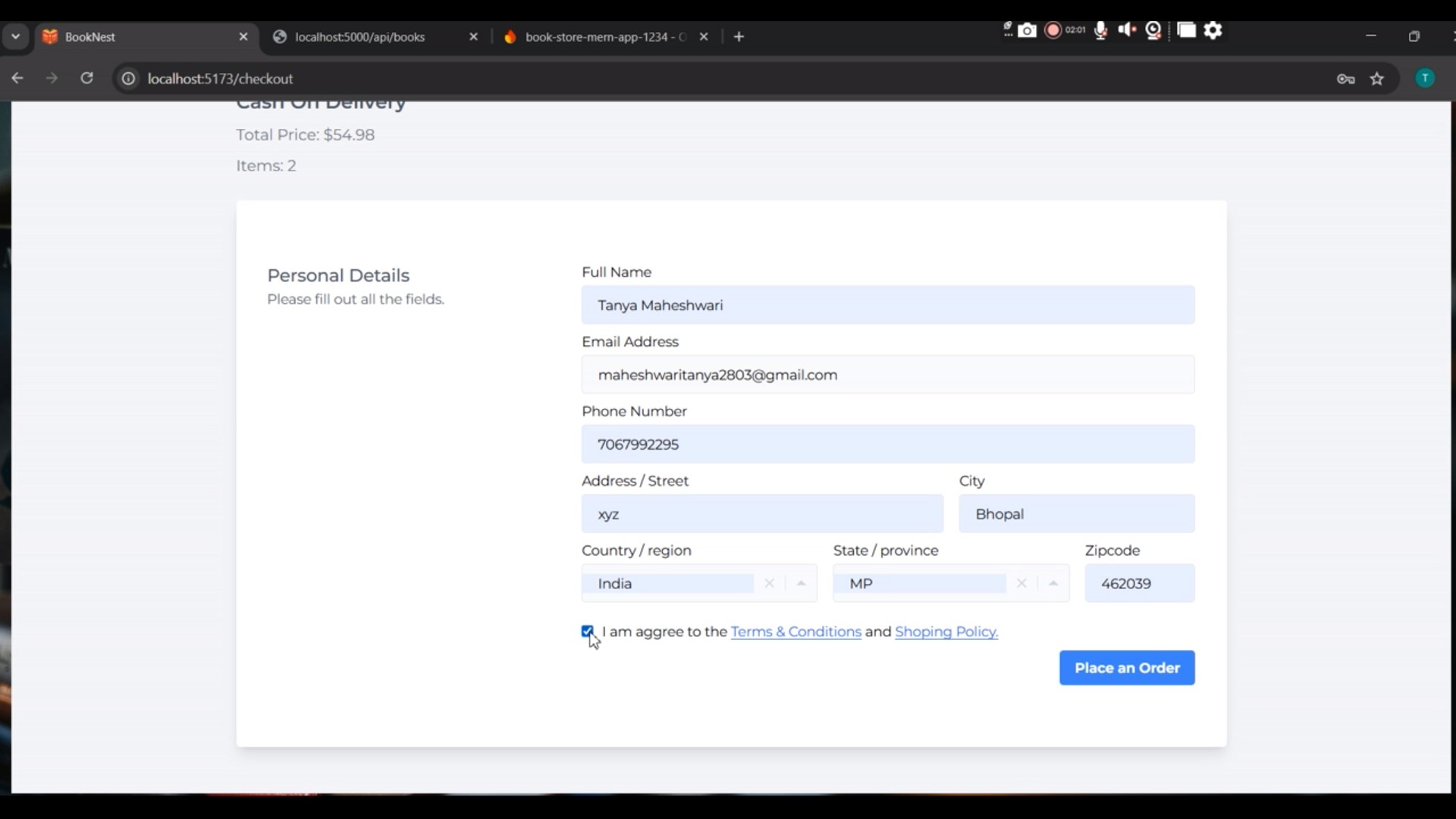
4. Book Library UI:

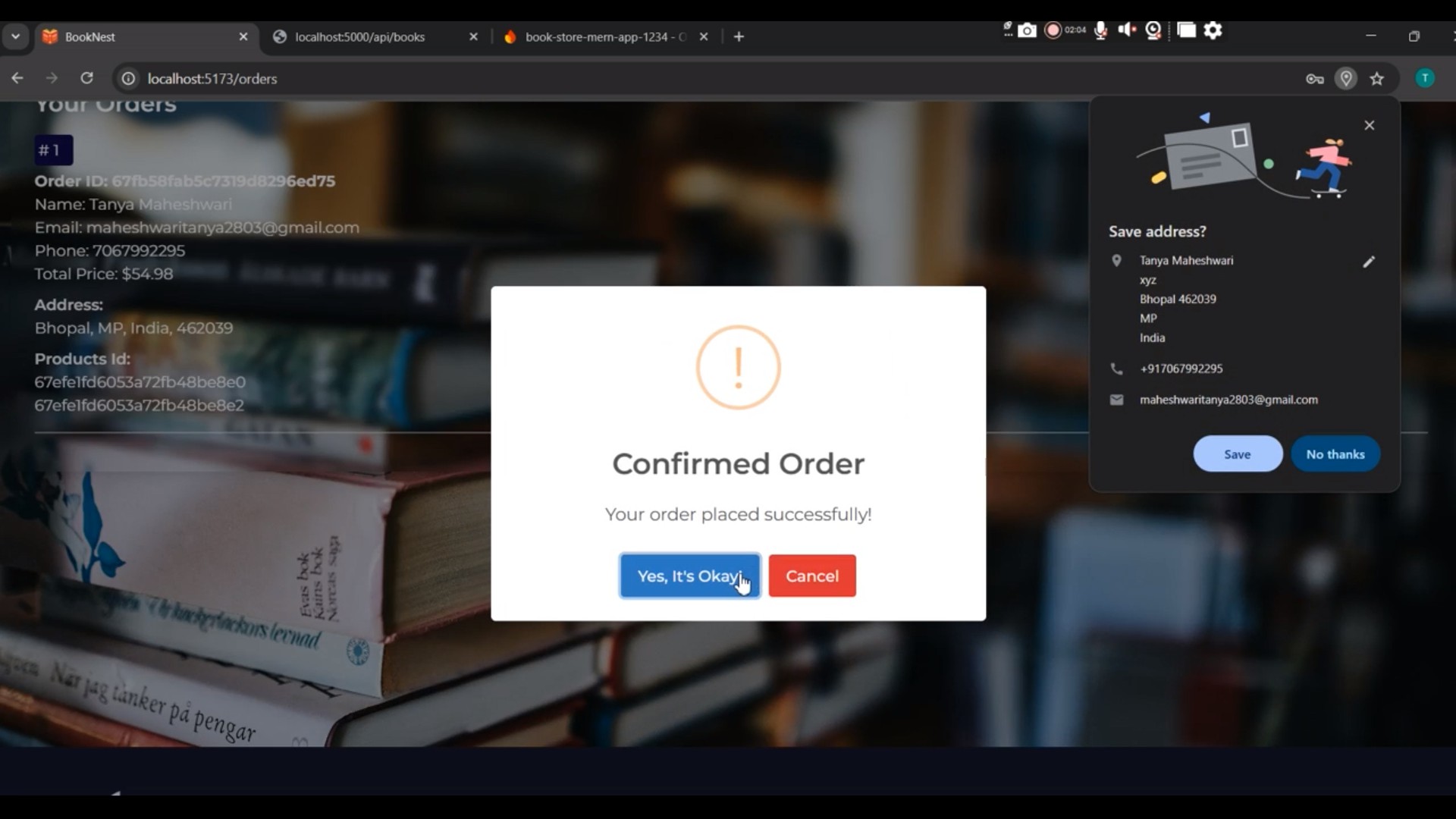


1. Choosing a book:



1. Ordering menu :



1. Confirming Order:

**8. ADVANTAGES & DISADVANTAGES**

Advantages:

* + - Full CRUD functionality
    - Firebase authentication
    - Responsive UI
    - Scalable architecture

Disadvantages:

* + - No offline support
    - No RBAC
    - Minimal error handling

**9. CONCLUSION**

The Book-Store application successfully demonstrates the development of a functional MERN-based bookstore. It meets all primary objectives while providing a foundation for future enhancements.

**10. FUTURE SCOPE**

* + - Role-based access
    - Social logins
    - Token refresh logic
    - Toast notifications
    - Two-factor authentication

**11. APPENDIX**

Source Code: <https://github.com/TanyaMaheshwari28/ookNest>

Demo Link:

https://drive.google.com/drive/folders/1P4LtrP0PdnP7KCFMG\_gQbsOX

vJIKGKRz?usp=sharing