**Full Stack Development with MERN**

# 1. Introduction

* **Project Title:** BookNest-Where Stories Nestle
* **Team Members:** Jakke VeeraShankar [Team Leader]

Dulla SivaKalyani[Team member]

Duggana MahaLakshmi[Team member] Battivalasala Sri Lalitha devi[Team member]

# 2. Project Overview

* **Purpose:**  The purpose of the BookNest application is to redefine the traditional reading experience by offering a digital platform where users can explore, discover, and purchase books with ease. The app is tailored for modern readers who prefer a seamless, online experience that doesn't compromise on the joy of browsing and collecting literature.
* **Key Features:** 
  + Users can securely register and log in to the platform using their email and password. The system ensures data privacy by encrypting passwords and using token-based authentication to manage sessions. This feature allows users to safely access personalized features like order history and saved preferences..
  + The application provides a comprehensive listing of books available for purchase. Each book entry displays relevant details such as the title, author, genre, description, price, rating, and current stock status. This helps users make informed decisions while browsing the catalog..
  + Books can be added to a shopping cart with specified quantities. Once ready, users can proceed through a secure checkout process to finalize their purchases. This triggers backend operations to generate an order and update the inventory accordingly.
  + After a successful transaction, users receive an order confirmation that includes a unique order ID, the list of books purchased, the total cost, and estimated delivery information. This assures users that their order was successfully placed.
  + Users can view a detailed history of all past and current orders from their profile. Each order record includes purchase details and shipment tracking (if integrated). Users can also review books and rate their shopping experience, contributing to community feedback and recommendations..
  + Admins can manage books, users, and orders through a secure backend interface. They can add/edit/delete book entries, monitor sales, and view analytics.

# 3. System Architecture

* **Client Layer (Frontend)**

**Technology:** React.js

**Users:** Readers (Customers), Admin

**Role:**

* Interactive book browsing and purchasing.
* Responsive UI for all devices.
* Axios-based API calls.
* User session management with local storage.
* **Server Layer (Backend API)**

**Technology:** Node.js + Express.js

**Responsibilities:**

* API handling for authentication, book inventory, and orders.
* Middleware for validation and security.
* Separate routes for books, users, and orders**.**
* **API Gateway**

**Technology:** Node.js with Express.js (or a dedicated gateway service like NGINX or API Gateway libraries for microservices)

**Function:** Acts as a single entry point for all incoming client requests, helping to decouple the frontend from individual backend services.

* **Authentication Service**

**Technology:** MongoDB

**Role:**

* + Registers new users and stores hashed passwords.
  + Authenticates login attempts using bcrypt and issues JWT tokens.
  + Verifies user identity for protected routes.
* Maintains session validity via tokens, enabling secure, stateless access to user .

# 4. Setup Instructions

* **Install Prerequisites :** Node.js, npm, MongoDB (local/Atlas)
* **CloneRepository :**  git clone <https://github.com/your-username/booknest.git>
* **Start MongoDB :** mongod or connect to MongoDB Atlas
* **Setup Backend:** cd server npm install npm start
* **Setup Frontend:** cd client npm install npm start
* **Frontend API:** <http://localhost:3000>
* **Backend API:** <http://localhost:5000>

# 5. Folder Structure

**Client (React Frontend) :**

/client

/public

/src

/components

/pages

/context

/styles

App.jsx

package.json

**Server (Node.js + Express Backend):**

/server

/models

/routes

/controllers

index.js

package.json

6. **Running the Application**

**• Frontend:** cd client

npm start

# Runs on <http://localhost:3000>

• **Backend**: cd server

npm start

# Runs on <http://localhost:5000>

## 7. API Documentation

* **User APIs**:
* POST /api/regist Register new user
* POST /api/login – Authenticate user

•**Book APIs:**

* GET /api/books – Fetch all books
* GET /api/books/:id – Fetch single book details
* POST /api/books – (Admin) Add new book
* PUT /api/books/:id – (Admin) Update book
* DELETE /api/books/:id – (Admin) Delete book

•**Order APIs:**

* POST /api/orders – Place an order
* GET /api/orders – View all orders for user
* GET /api/orders/:id – View order details

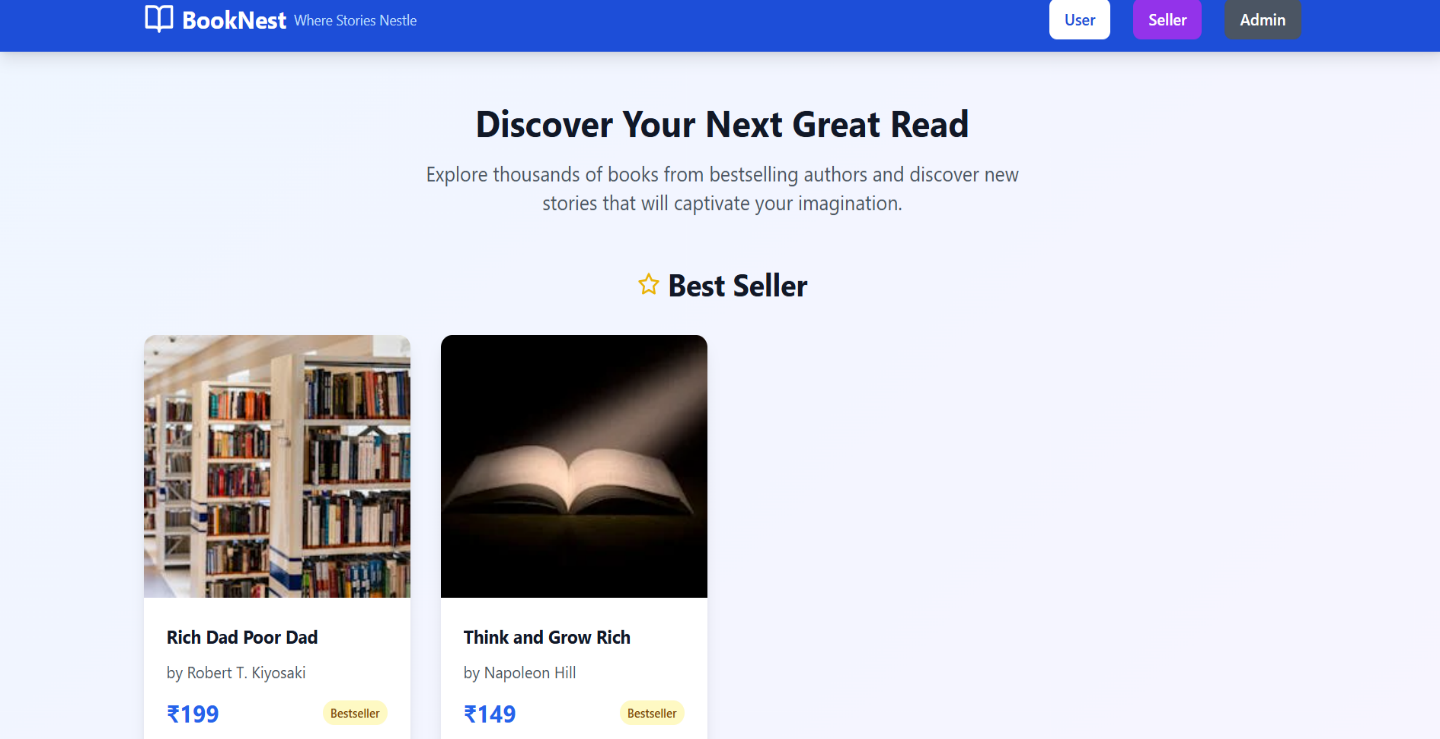
# 8. Authentication Flow

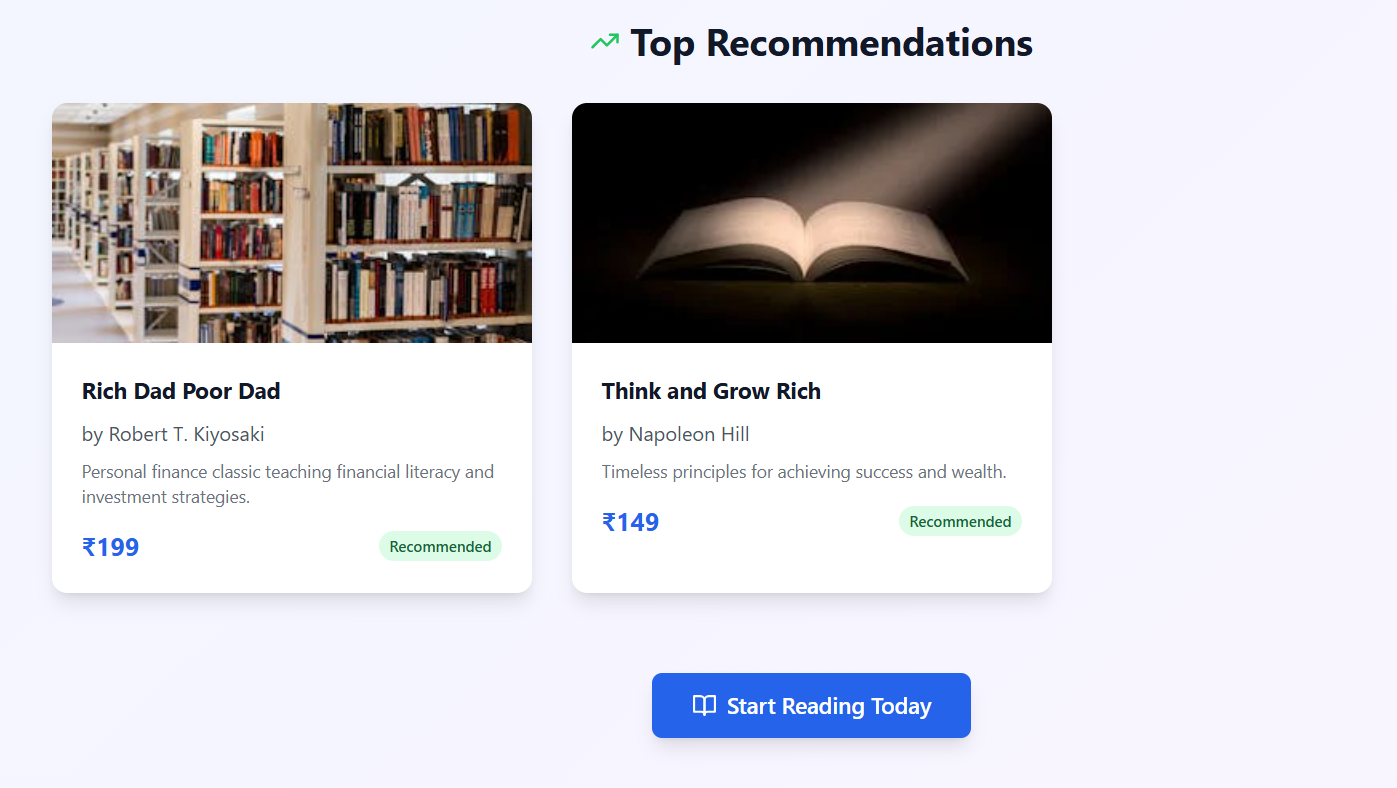
* **User Registration (POST /register):** Registers name, email, password (hashed), saved in DB.
* **User Login (POST /login):** Authenticates credentials → Returns JWT → Stored on client.

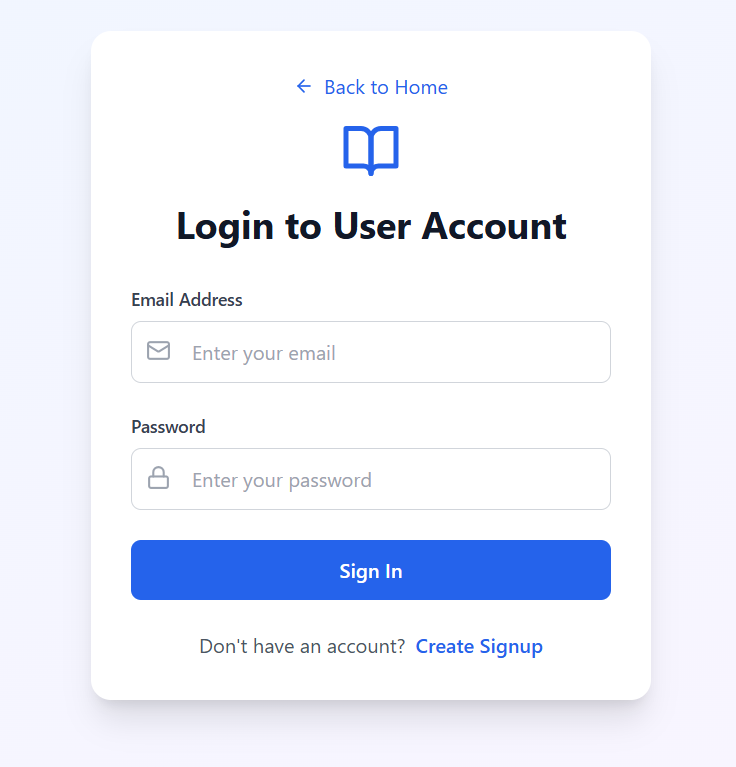
# 9.Testing

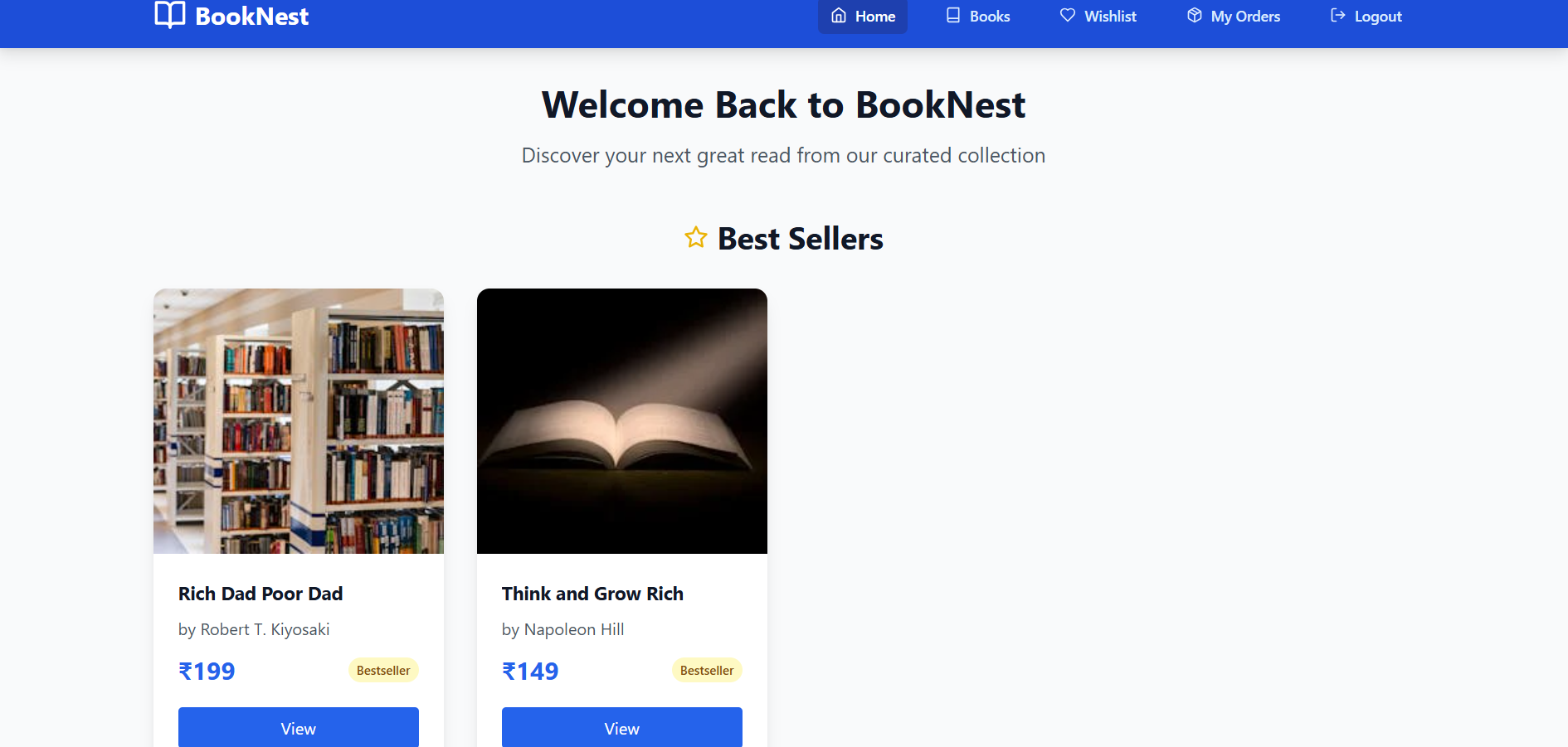
* **Unit Testing**: API functions and logic verified.
* **Integration Testing:** End-to-end testing for cart → order → confirmation.
* **UI Testing:** Responsive design tested across devices.
* **Security Testing**: JWT authentication and protected routes.

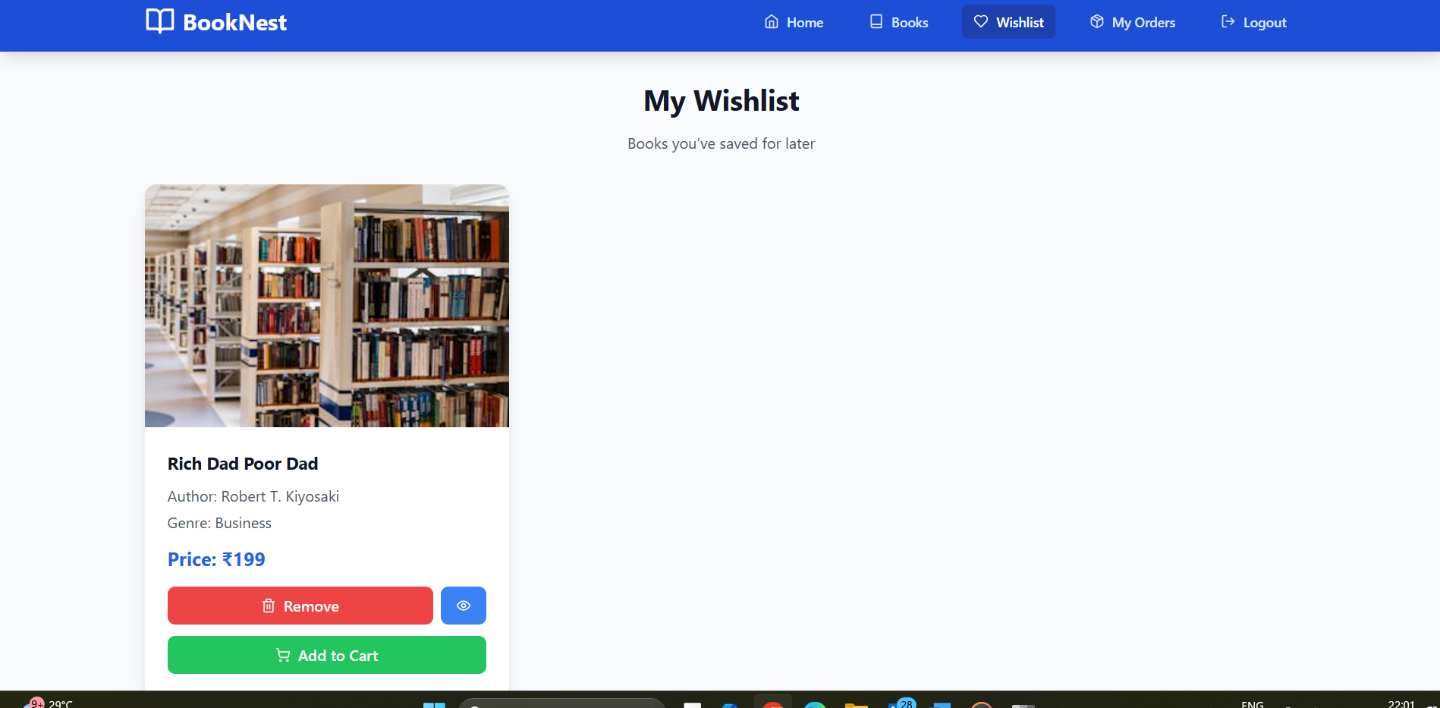
# 10. Screenshots / Demo

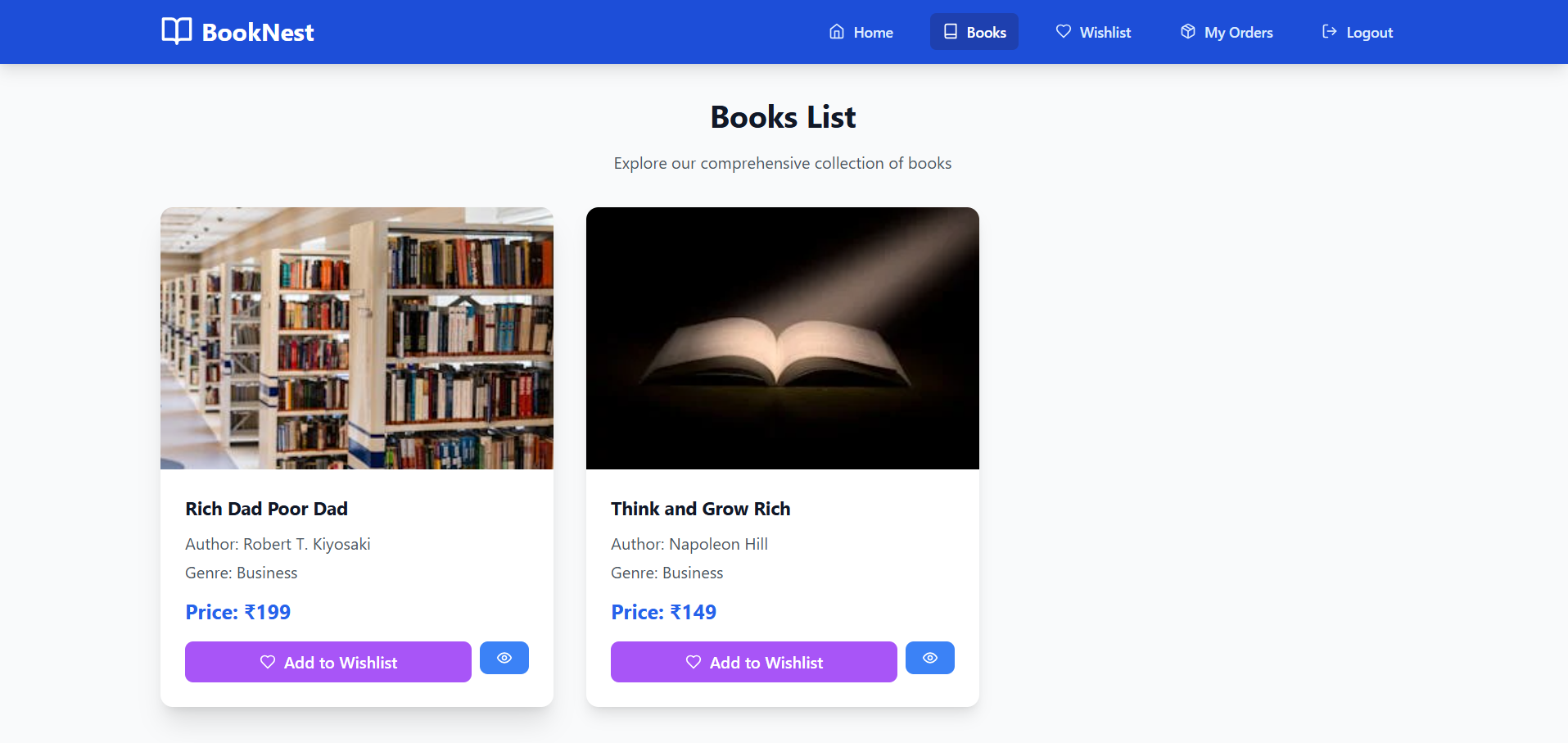


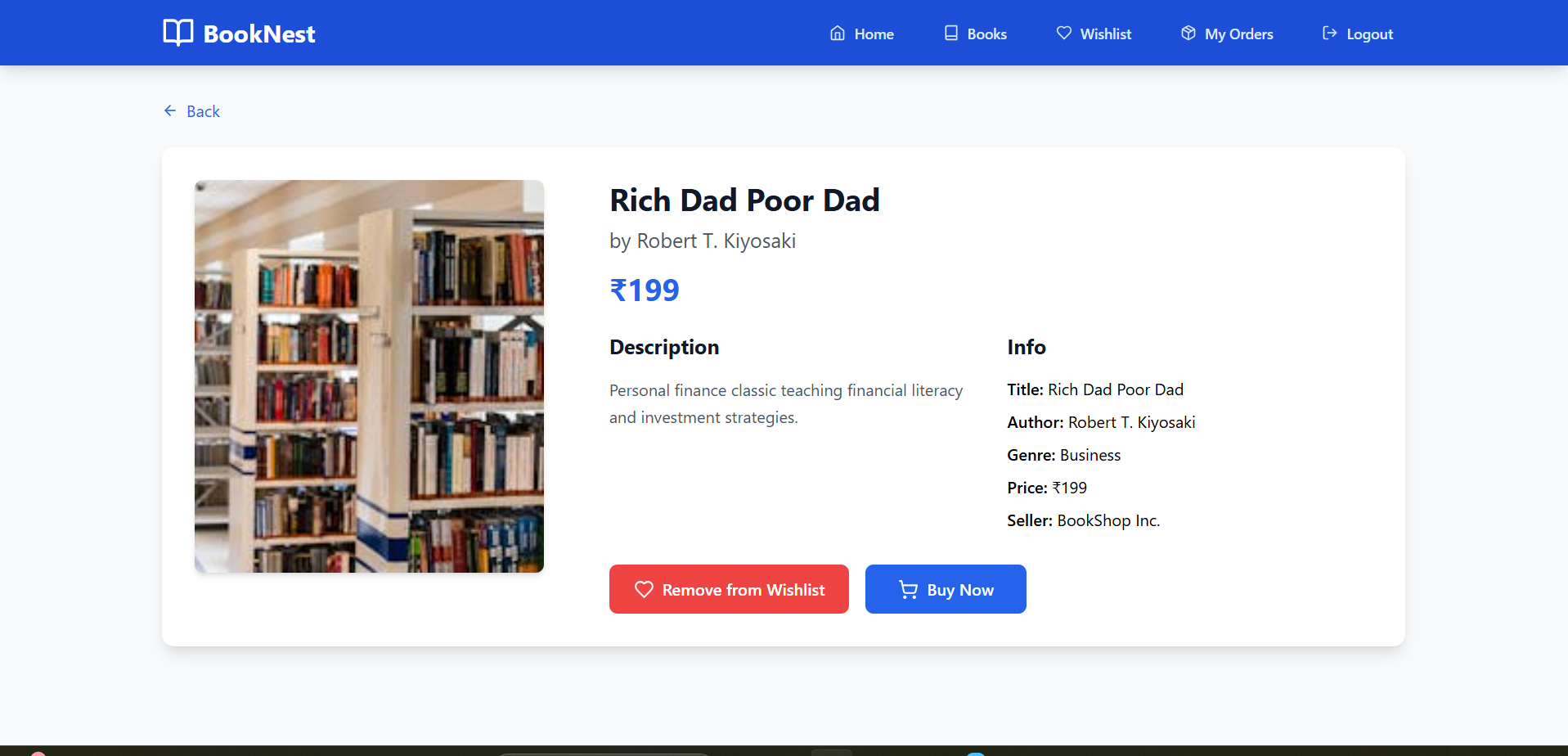


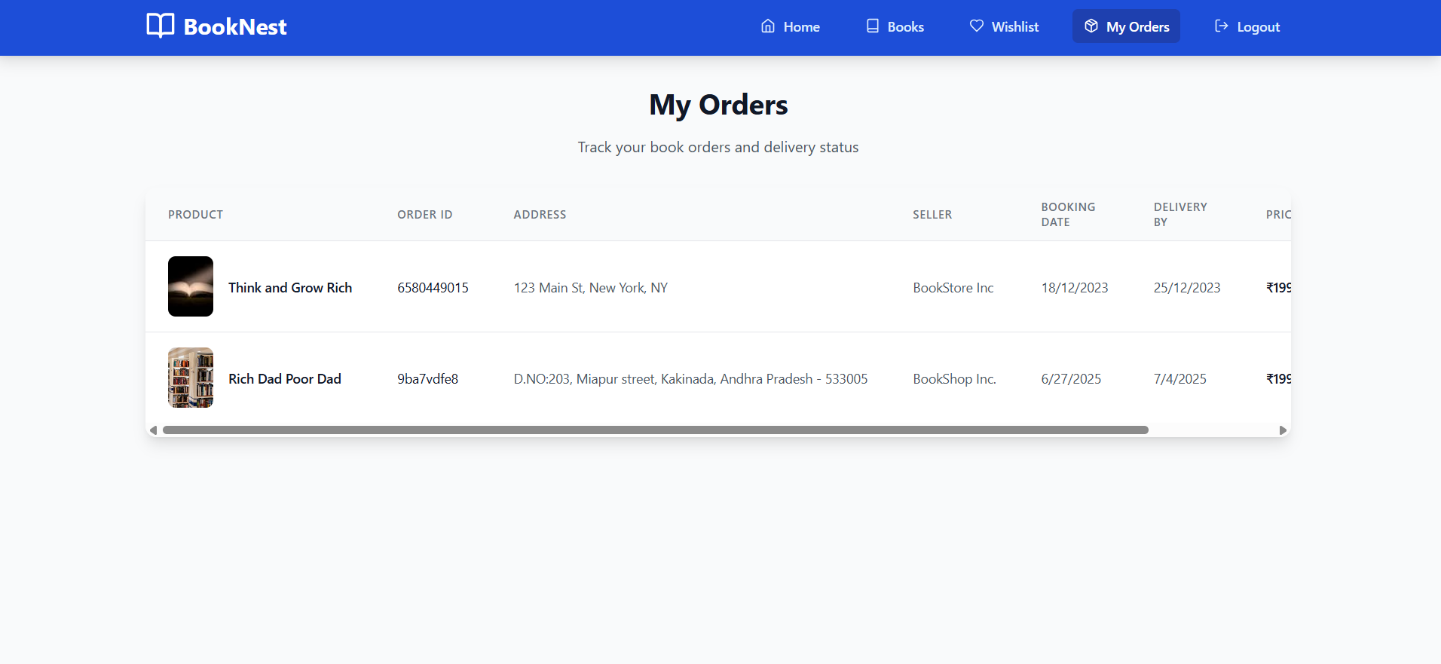


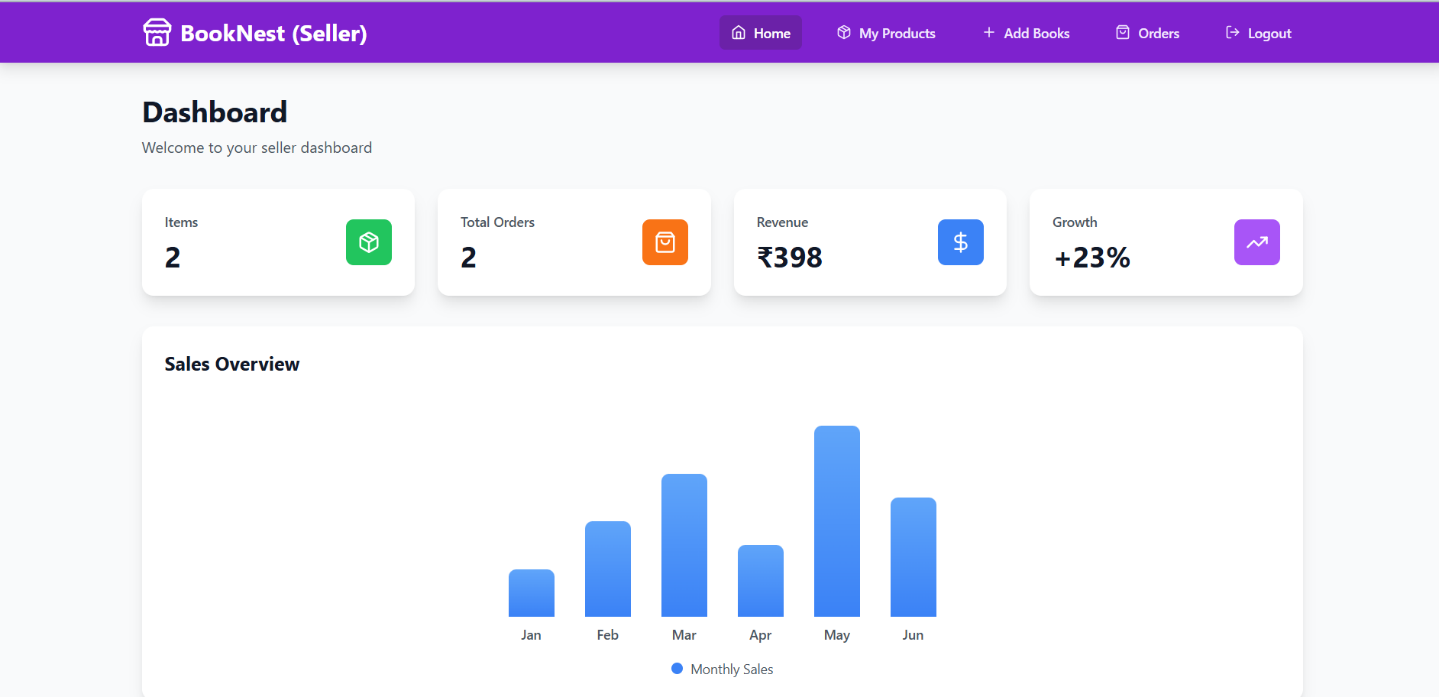


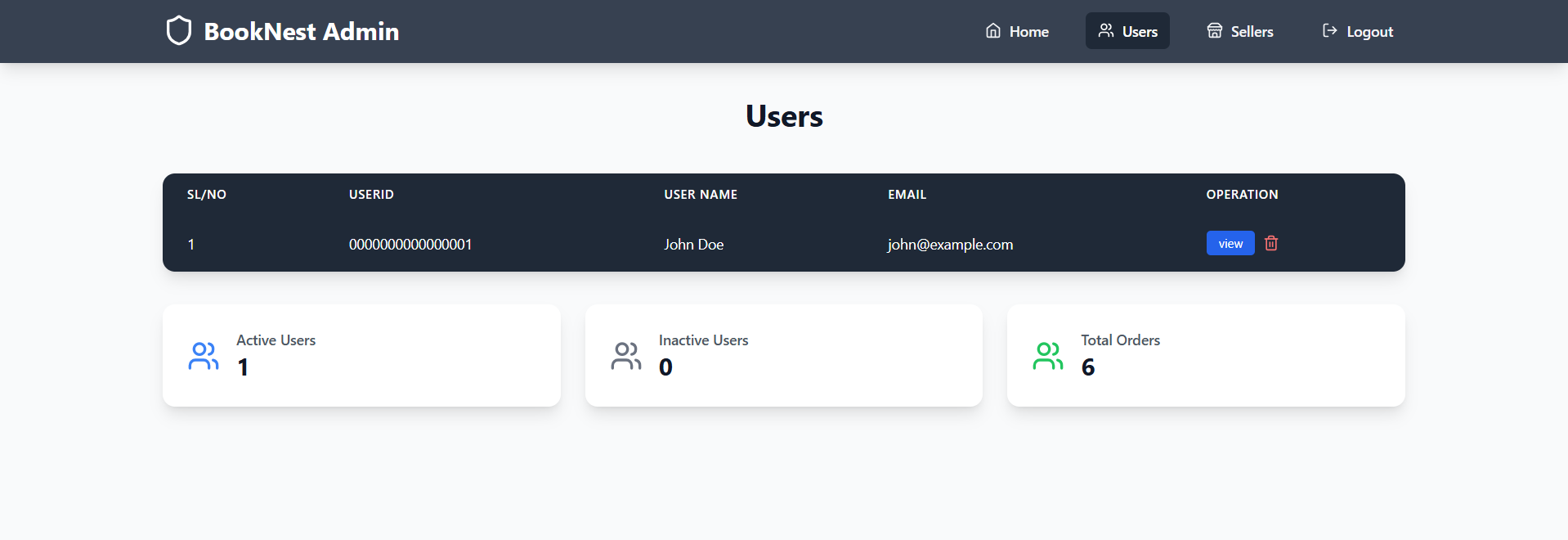


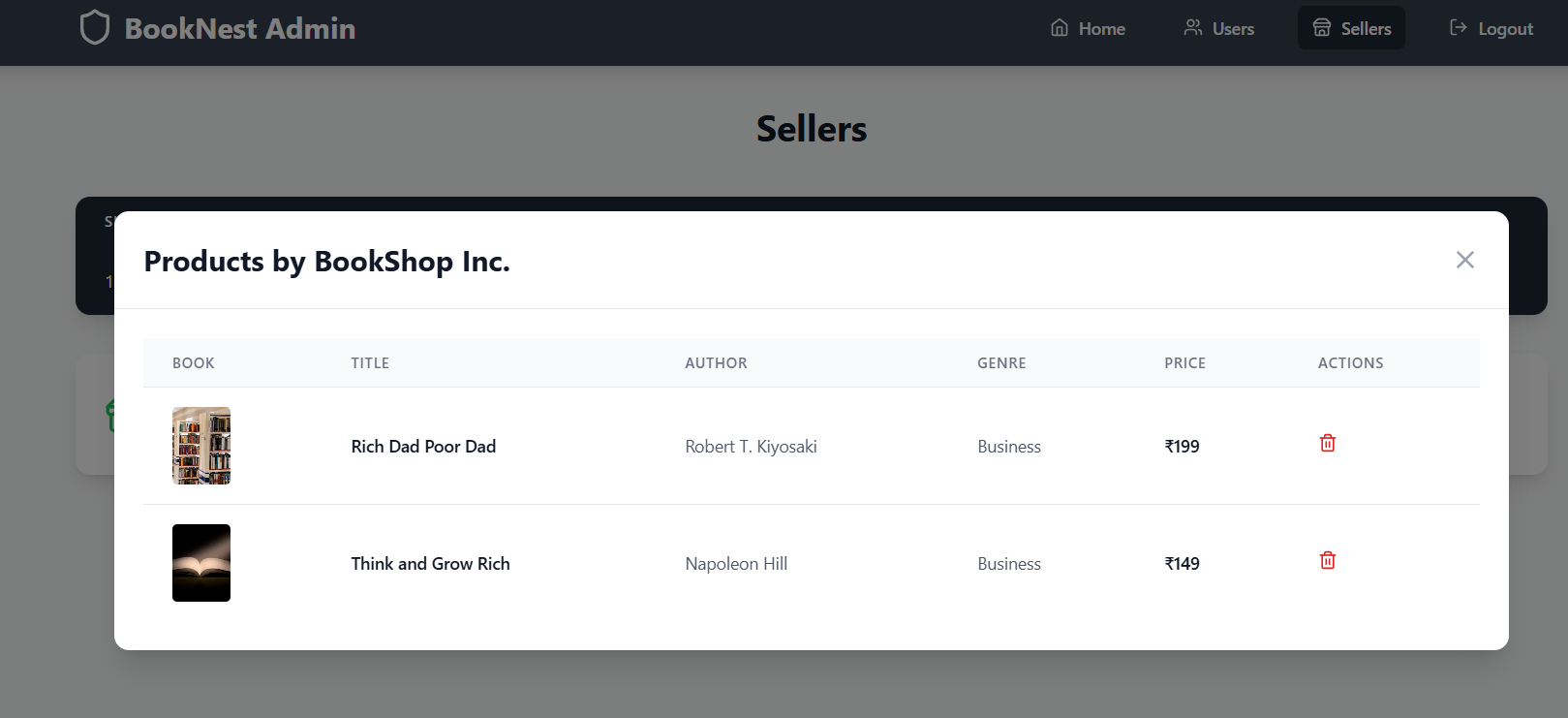












The demo of the app is available at:  [https://drive.google.com/file/d/1y7TPSSUC5krDNSxNI66VvYI4MTxPy8XV/view?usp=sharing](https://drive.google.com/file/d/1lVILTZxAW9KFQbsTs9qziVxPN7u5AyDU/view?usp=sharing)

# 11. Known Issues

* No Payment Gateway Yet – Purchases are simulated.
* Role-based admin access pending frontend enforcement.
* UI needs improvement on small screens.
* Real-time order tracking not yet implemented

# 12. Future Enhancements

* Integrate payment gateways (Stripe, Razorpay)
* Add analytics dashboard for admin
* Allow users to upload book reviews
* Mobile App version for Android/iOS
* Real-time in-app notifications
* Implement recommendation engine (based on reading behavior)
* Real-time stock alert system