**Full Stack Development with MERN**

**Project Documentation format**

**Book Store Project (BooksMall)**

**-Premasagar K**

**1. Introduction**

* **Project Title:** BooksMall

**Book Store (BooksMall)**is a production-ready full-stack e-commerce platform built with MERN stack featuring multi-role authentication (Customer/Admin/Operator), responsive Tailwind UI, JWT security, Redis caching, and MongoDB persistence. Deployed on Vercel (Backend) + GitHub Pages (Frontend).

**Live Demo:** <https://premasagark.github.io/BookStoreClient/>

* **Team Members:**

Premasagar K (Full Stack )

**2. Project Overview**

* **Purpose:**

BooksMall is a full-stack e-commerce platform designed to streamline book buying, selling, and management with multi-role authentication for Customers, Admins, and Book Operators. The primary objectives are:

1. **Solve fragmented book marketplace** - Centralized platform for Tamil Nadu students/book enthusiasts
2. **Multi-role workflow** - Customers shop, Admins manage, Operators supply books
3. **Production-ready deployment** - Live on Vercel + GitHub Pages.

**Core Value Proposition:** Secure, scalable bookstore with role-based dashboards replacing manual processes.

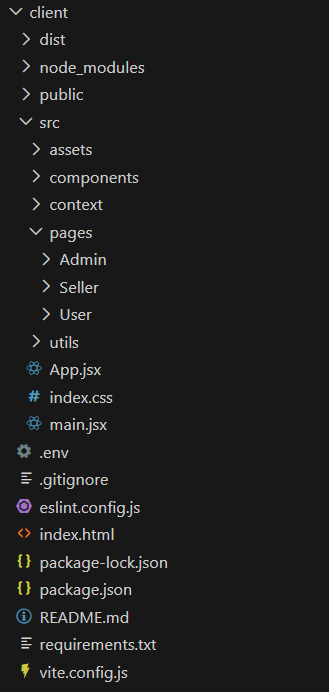
* **Features:**

| **Category** | **Features** |
| --- | --- |
| **Authentication** | Multi-role login/signup (Customer/Admin/Operator) |
|  | JWT Access/Refresh tokens (15m/7d) |
|  | Role-based route protection |
|  | Auto-login session refresh |
| **Customer Experience** | Book search + filtering |
|  | Shopping cart management |
|  | Order tracking/history |
|  | Responsive wishlist |
| **Admin Dashboard** | User approval/rejection workflow |
|  | Analytics cards (Users/Books/Orders) |
|  | Complete user management |
|  | Booking oversight |
| **Operator Dashboard** | Add/Edit/Delete books |
|  | Inventory level management |
|  | Assigned order notifications |
| **User Interface** | minimal design system |
|  | Mobile-first responsive layout |
|  | Hover animations + micro-interactions |
|  | Password visibility toggle |
| **Performance** | Redis caching (30-120s TTL) |
|  | MongoDB Atlas persistence |
|  | Code splitting + lazy loading |
| **Deployment** | Vercel serverless backend |
|  | GitHub Pages static frontend |
|  | Environment-based CORS |

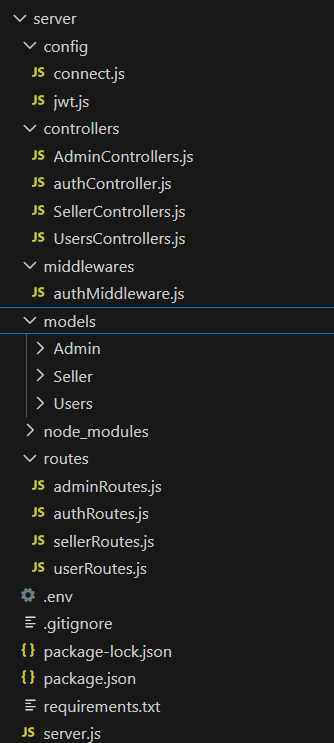
.

**3. Architecture**

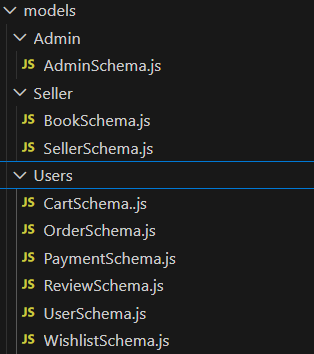
• **Frontend:** Describe the frontend architecture using React.



• **Backend:** Outline the backend architecture using Node.js and Express.js. •



• **Database:** Detail the database schema and interactions with MongoDB.



**4. Setup Instructions**

• **Prerequisites:**

* Node.js (version 18 or higher)
* npm package manager
* MongoDB Atlas account (free tier)
* Redis Labs account (free tier)
* Git version control
* Code editor (VS Code recommended)
* Web browser for testing

• **Installation:** Step-by-step guide to clone, install dependencies, and set up the environment variables.

**CLIENT Setup Instructions (React + Vite + Tailwind)**

**Step 1: Clone Frontend Repository**

Navigate to your directory and clone the client repository using Git.

**Step 2: Install Dependencies**

In the client directory, run the package manager install command to fetch all required dependencies listed in package.json.

**Step 3: Environment Configuration**

Create a .env file in the client root directory and add the following configuration:

VITE\_API\_URL=[https://localhost:5000]

**Step 4: Tailwind CSS Configuration**

Verify tailwind.config.js and vite.config.js files are properly configured for Tailwind integration and GitHub Pages deployment path.

**Step 5: Start Development Server**

Execute the development server command to start the React application on the default Vite port.

**Step 6: Production Build & Deploy**

For production, run the build command followed by the deployment script for GitHub Pages hosting.

**SERVER Setup Instructions (Node.js + Express + MongoDB)**

**Step 1: Clone Backend Repository**

Clone the server repository into a separate directory from the client.

**Step 2: Install Dependencies**

Navigate to the server directory and install all backend dependencies from package.json.

**Step 3: Database Configuration**

1. **MongoDB Atlas**: Create free M0 cluster, whitelist IP addresses, get connection string

**Step 4: Environment Variables**

Create server/.env file with these required variables:

MONGODB\_URI=[your-mongodb-connection-string]

JWT\_SECRET=[64-character-secret-key]

PORT=5000

CORS\_ORIGIN=[your-frontend-url]

**Step 5: Verify Database Connections**

Test MongoDB and Redis connectivity before starting the server.

**Step 6: Start Development Server**

Use the development script with nodemon for automatic restarts on file changes.

**Running Both Applications**

**Development Environment:**

Terminal 1: Backend server (port 5000)

Terminal 2: Frontend client (port 5173)

**Production Deployment References:**

* **Frontend**: GitHub Pages via gh-pages package
* **Backend**: Vercel serverless functions
* **Database**: MongoDB Atlas

**5. Running the Application**

Provide commands to start the frontend and backend servers locally.

* **Frontend:**

1. Navigate to client directory

2. Start development server

npm run dev

**Expected Output:**

Vite server running at http://localhost:5173

Local: http://localhost:5173/

Network: http://192.168.x.x:5173/

Frontend loads at: http://localhost:5173

* **Backend:**

1. Navigate to server directory

2. Start development server with auto-restart

npm run dev OR

npm run server

**Expected Output**:

Server is running on port 5000

Connected to MongoDB

Connected to Redis

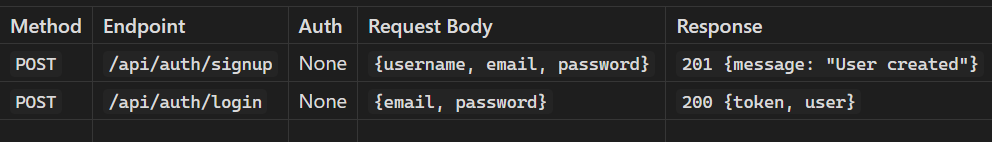
API Base URL: http://localhost:5000

**6. API Documentation**

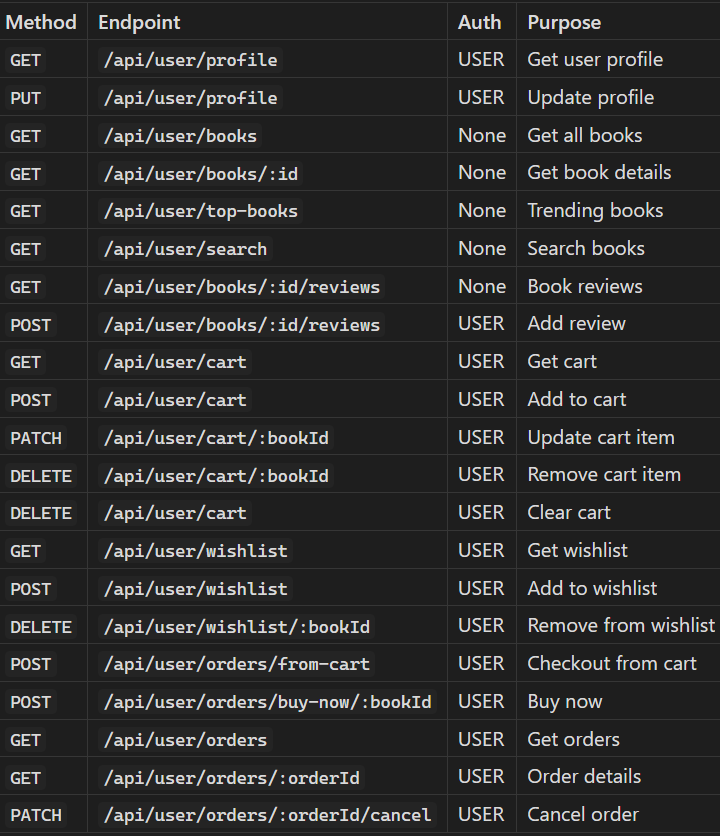
• **API BASE URL**:

* Development: <https://premasagark.github.io/BookStoreClient/>
* Production: https: [//bookstore-server.vercel.app/api](https://bookdstoreserver.vercel.app/)

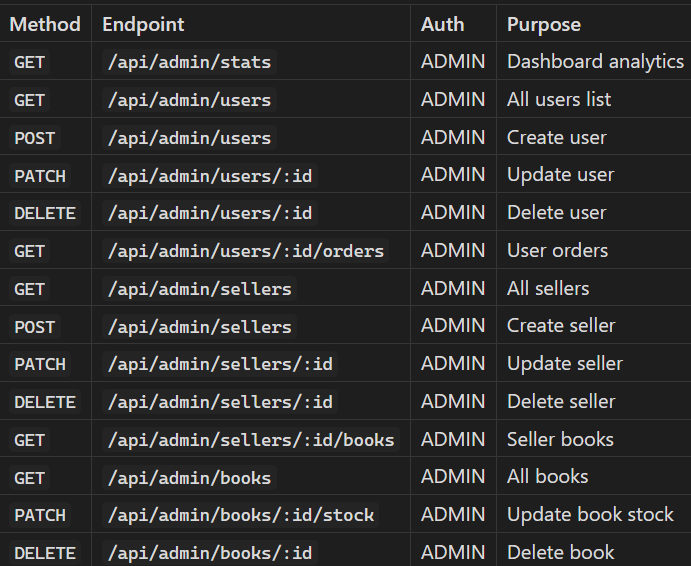
• **Authentication Endpoints**

****

• **User Endpoints (/api/user) [Public + Auth]**

****

• **Admin Endpoints (/api/admin) [ADMIN Only]**

****

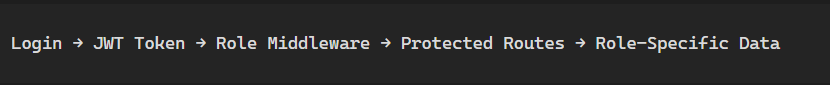
• **Seller Endpoints (/api/admin) [ADMIN Only]**

****

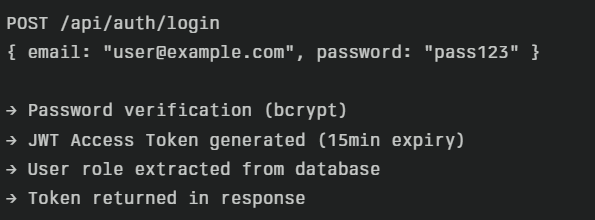
**7. Authentication**

• **Authentication & Authorization Overview**

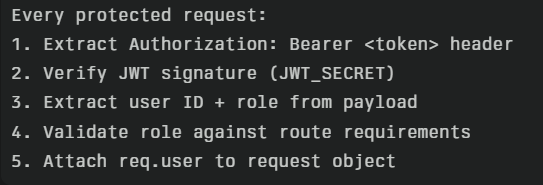
BooksMall implements a secure JWT-based system with role-based access control (RBAC) supporting three user roles: USER, ADMIN, and SELLER.



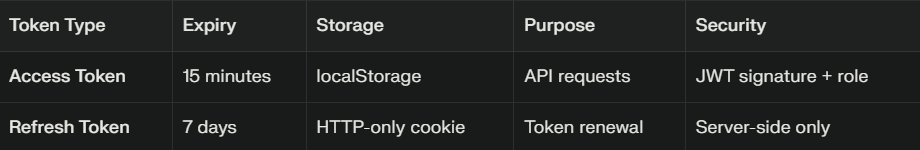
• **Login process**



• **Requests protected**

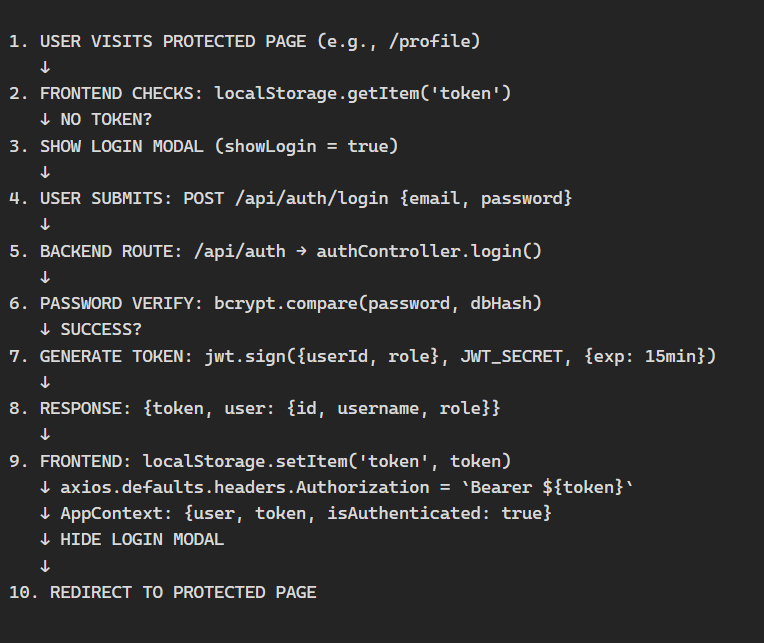


• **Token Security and validity**

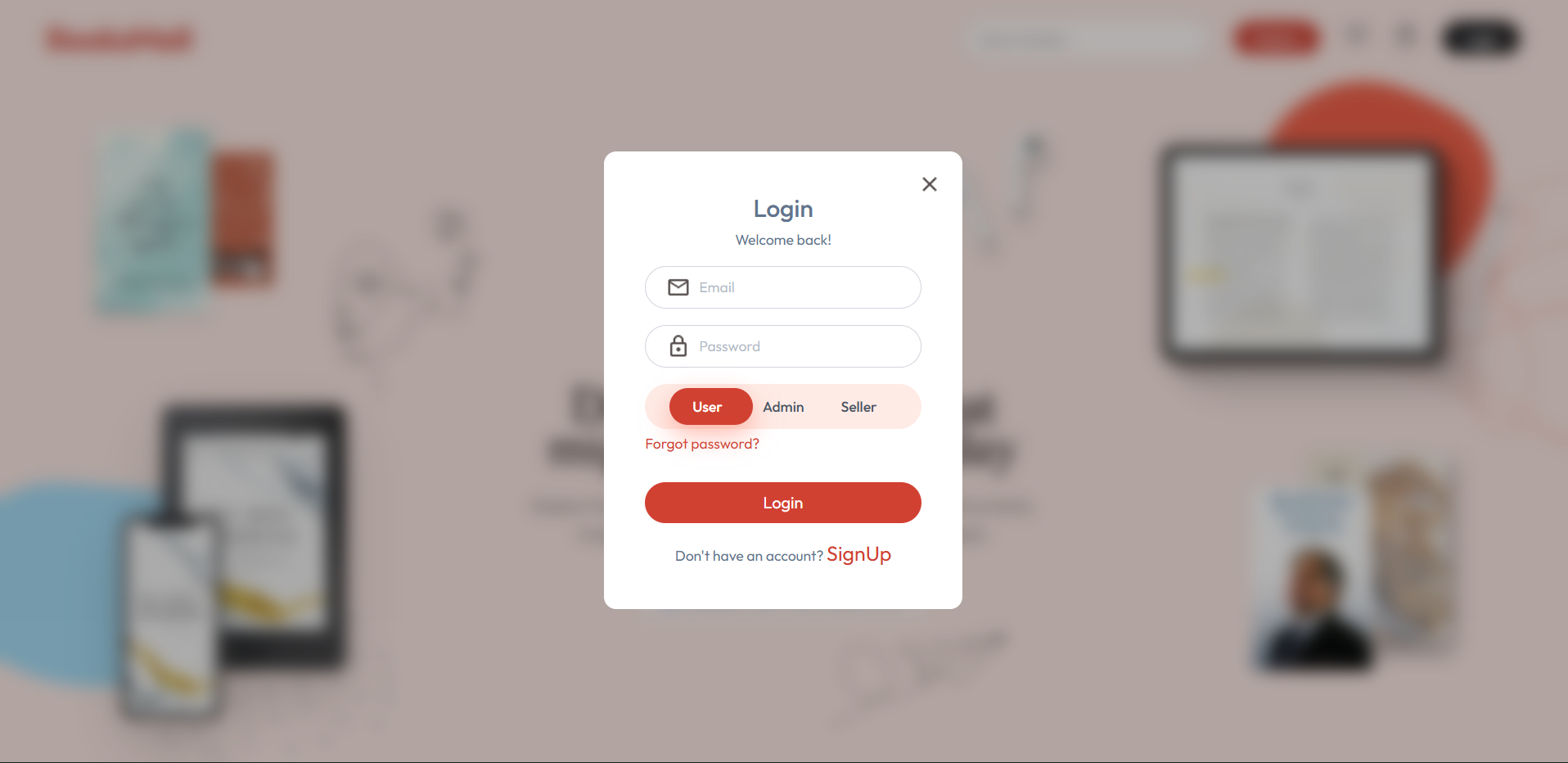
****

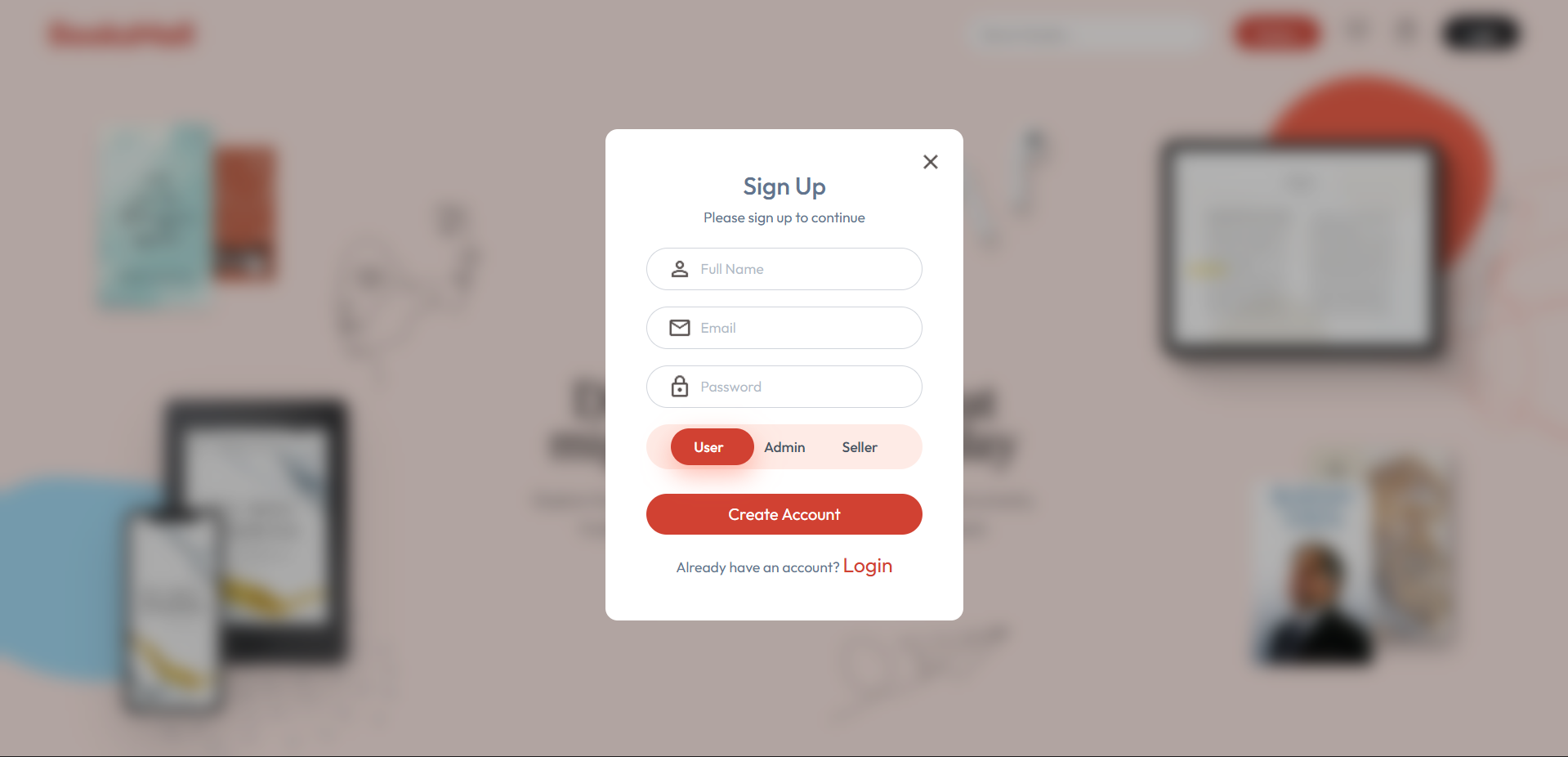
• **Auth Flow**

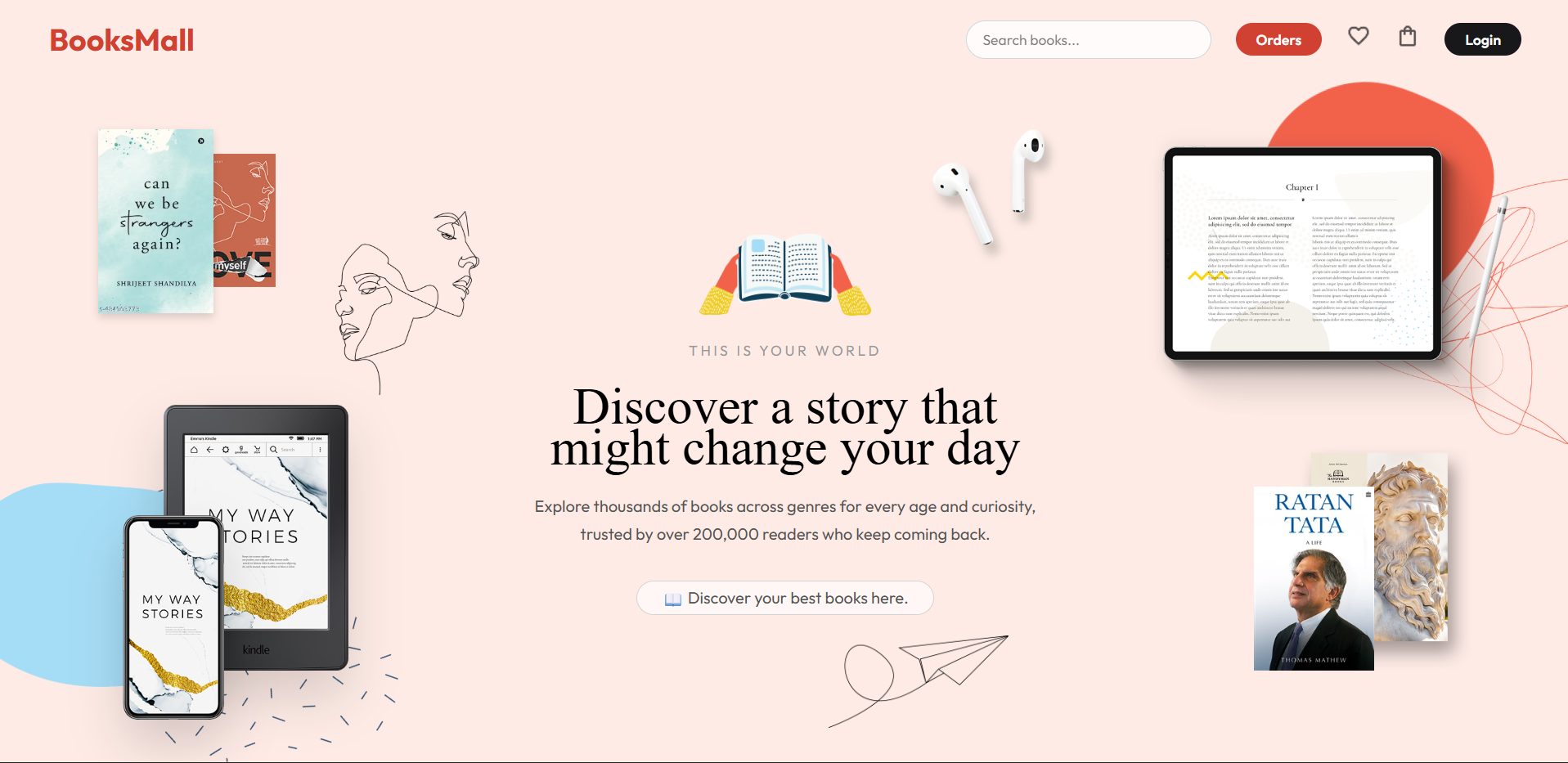
**Frontend Request → Backend Server → Auth Middleware → Controller → Response**

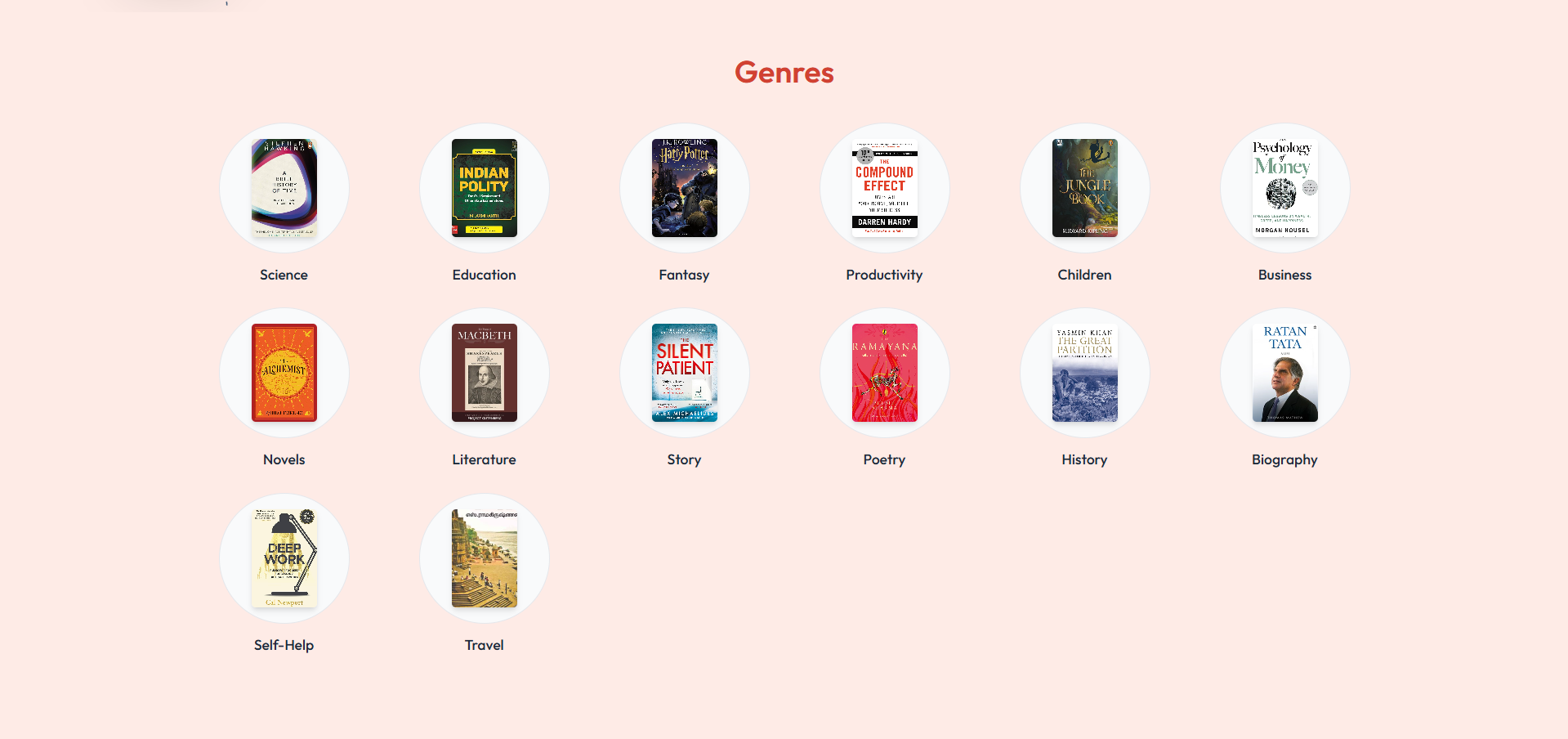
****

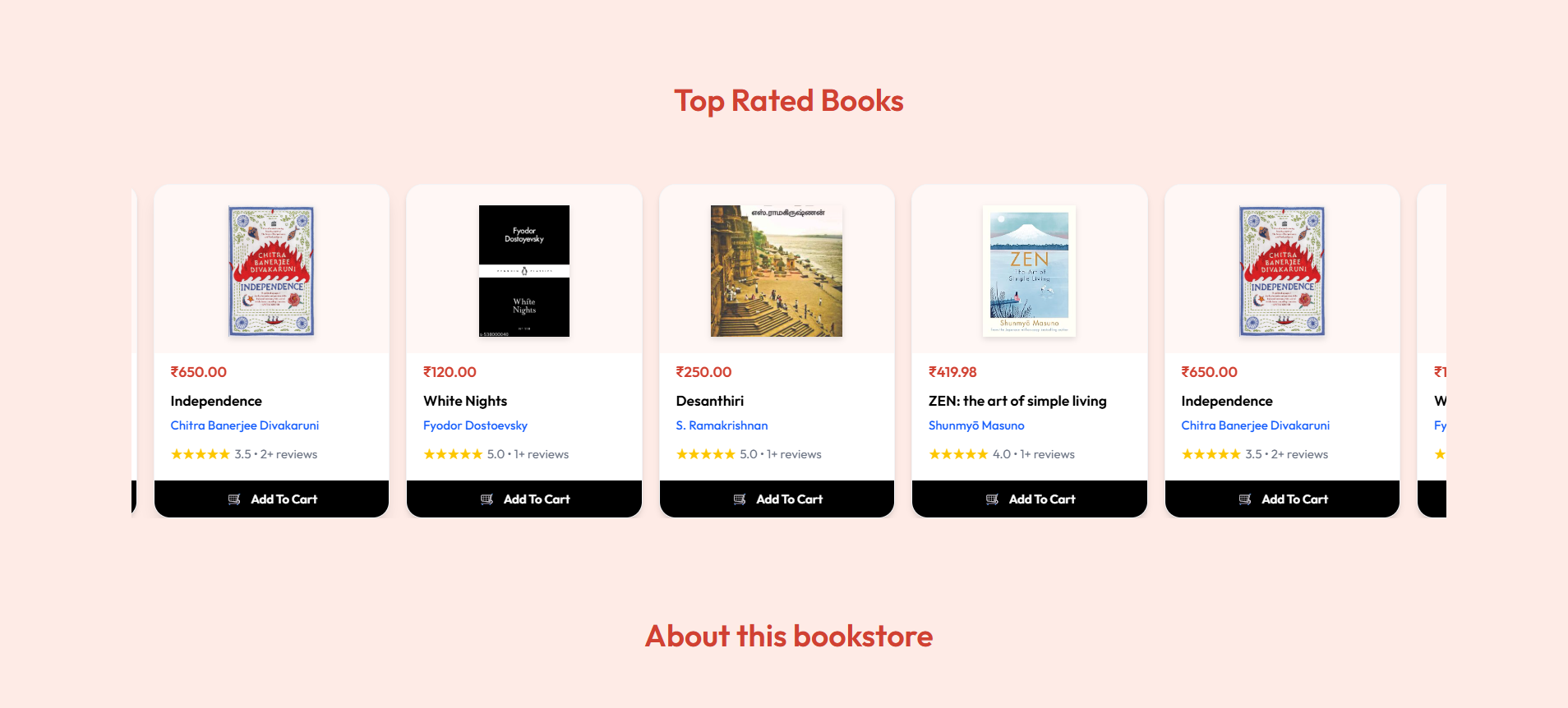
**8. User Interface**

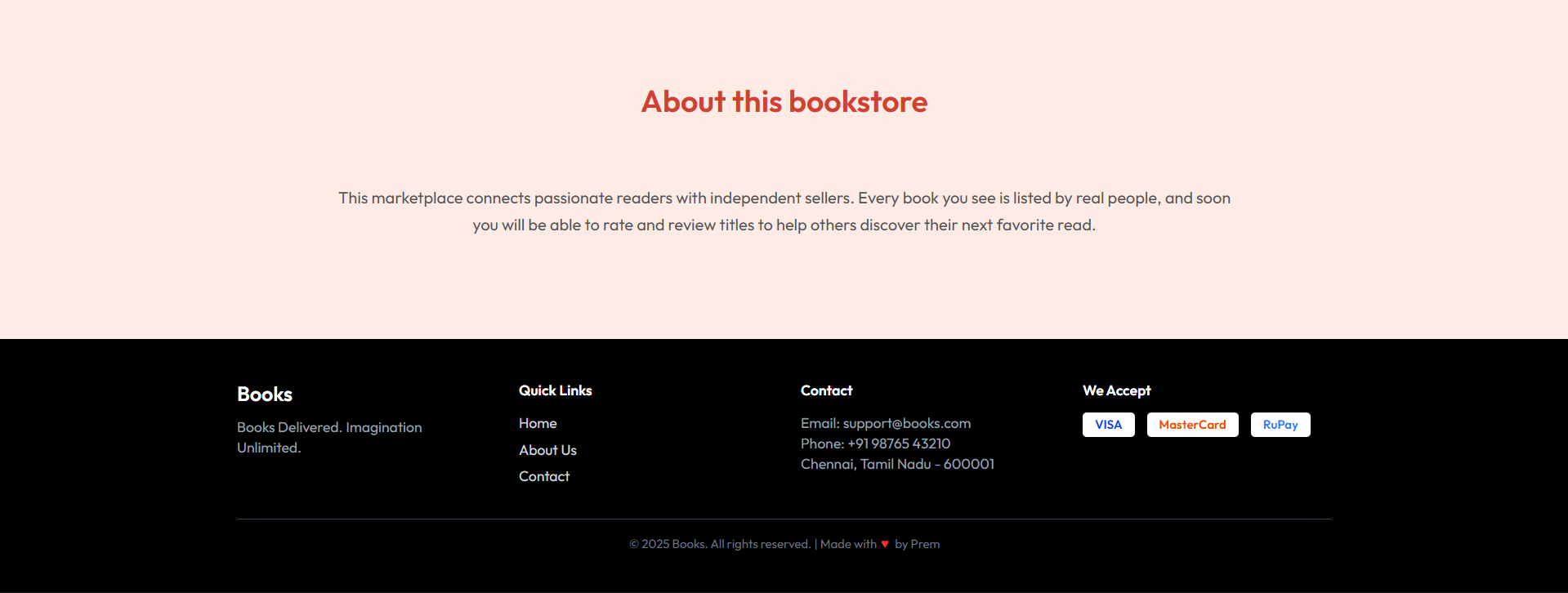
****

****

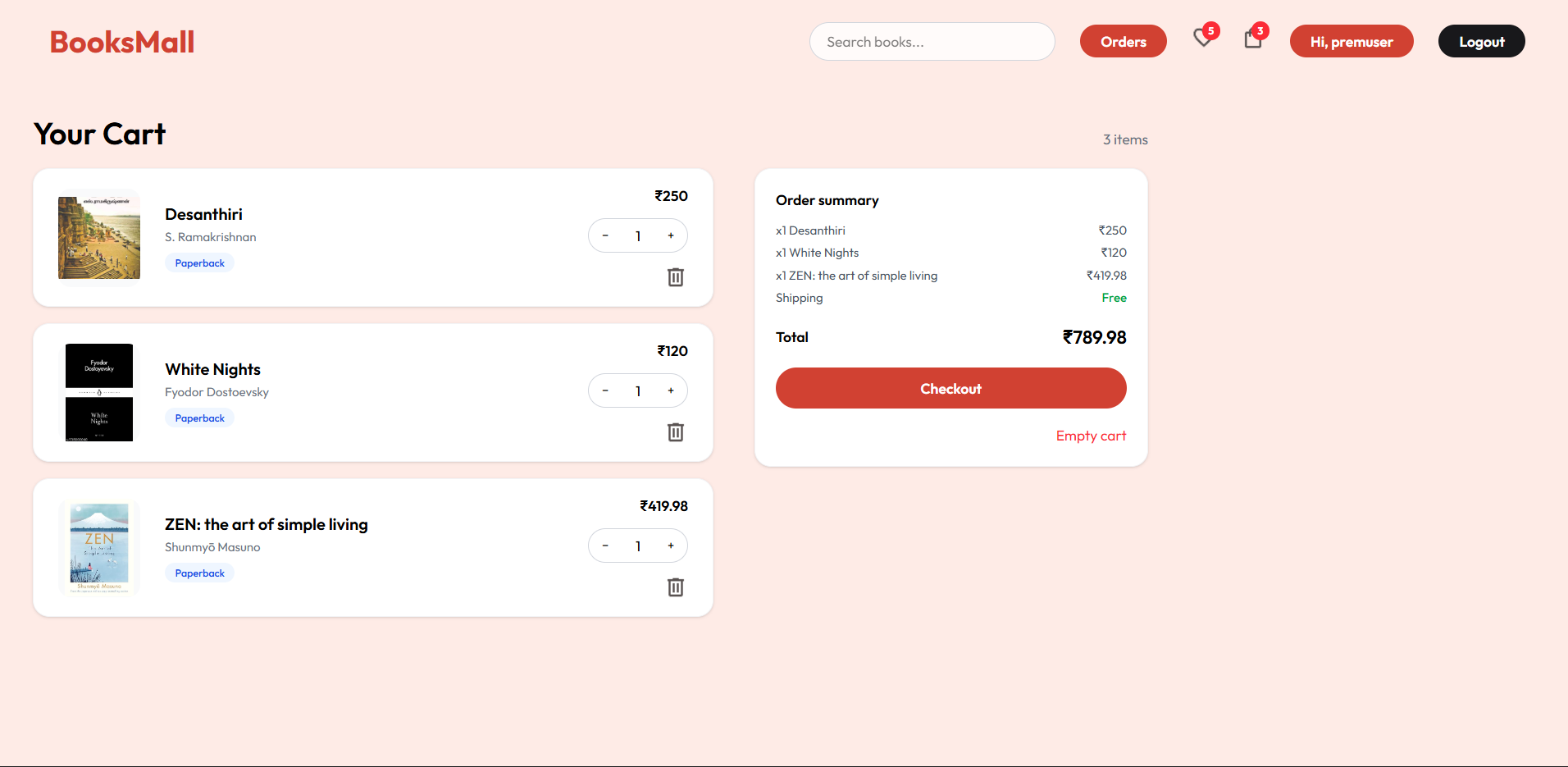
****

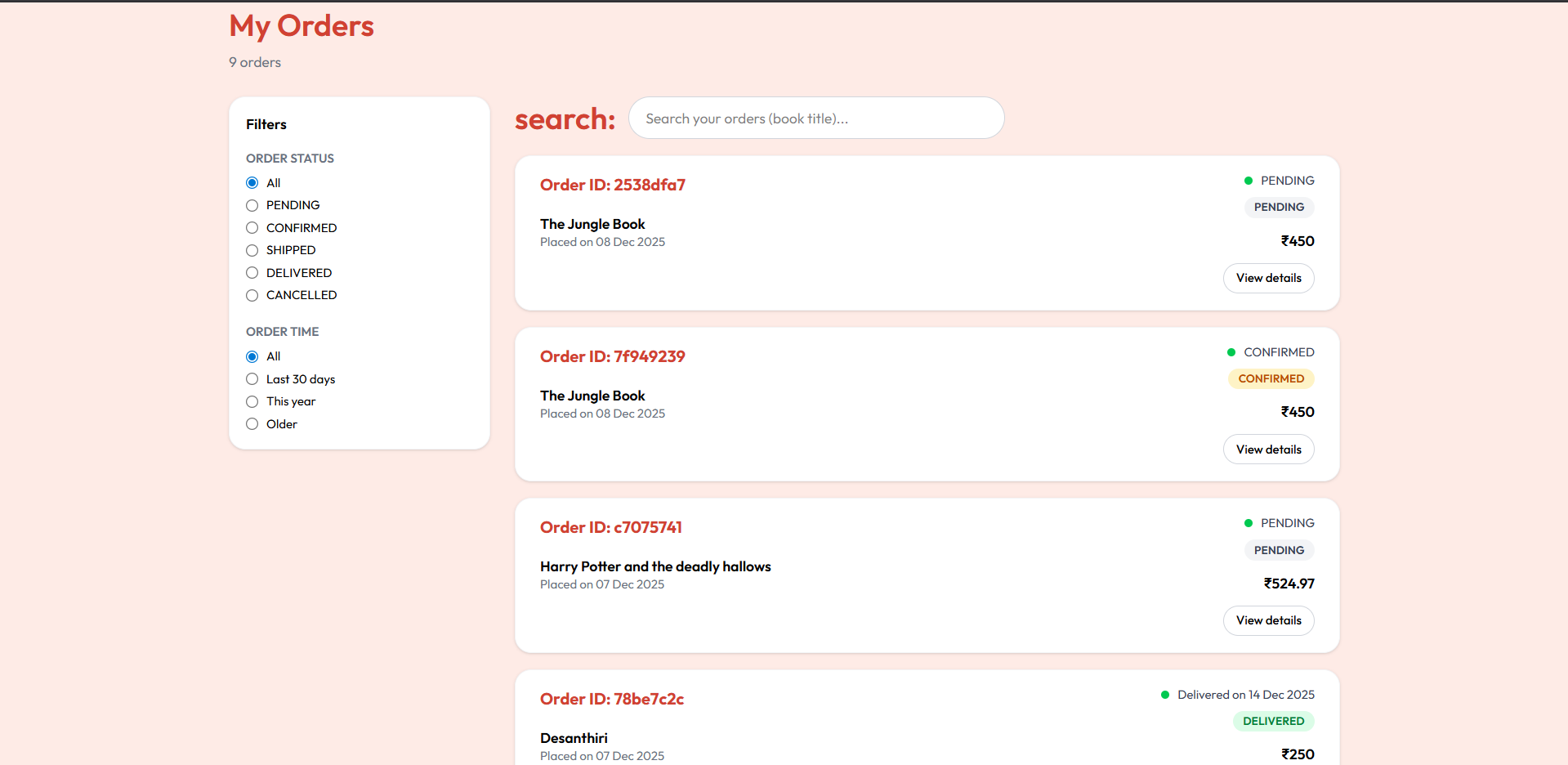
****

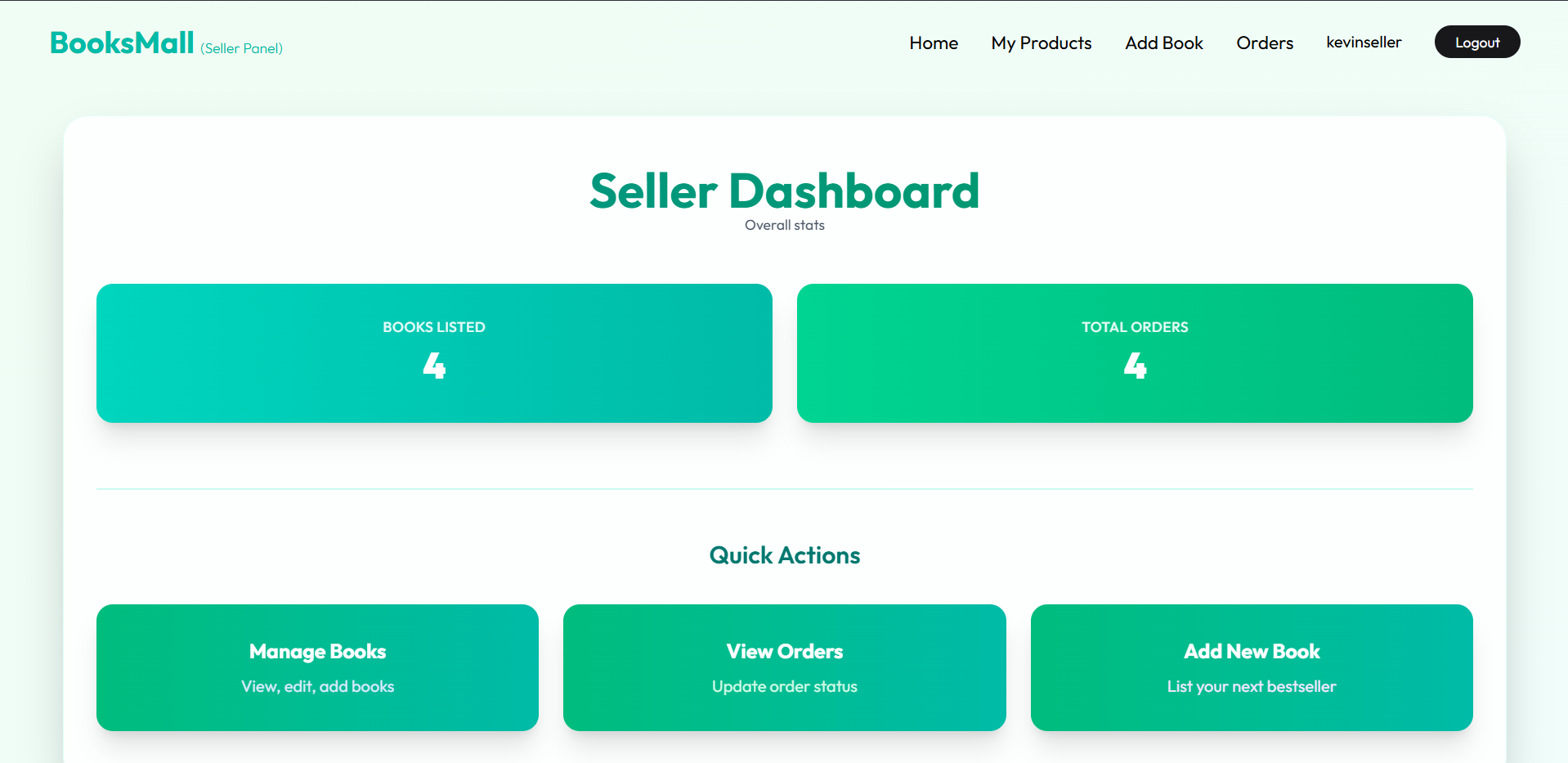
****

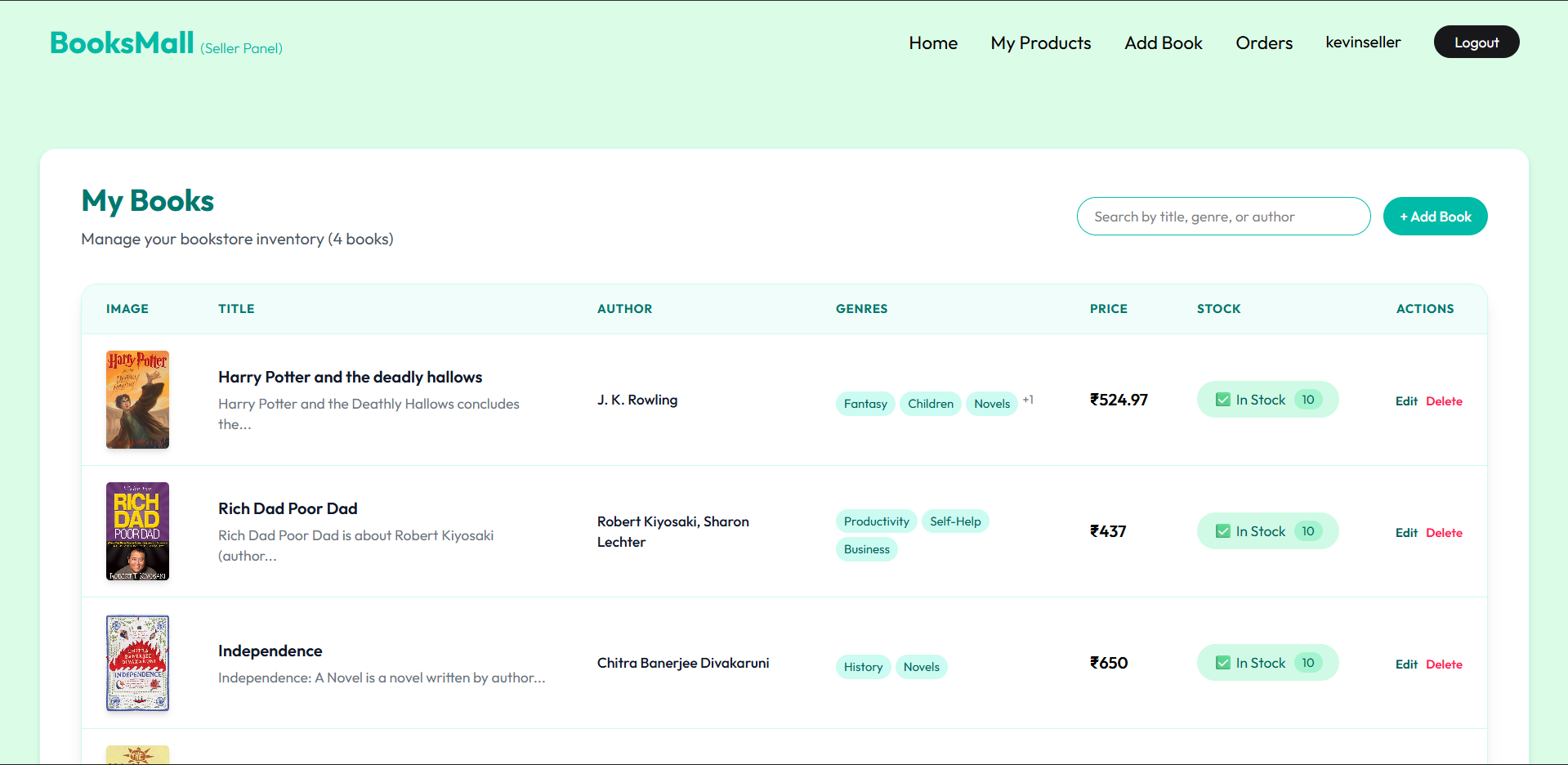
****

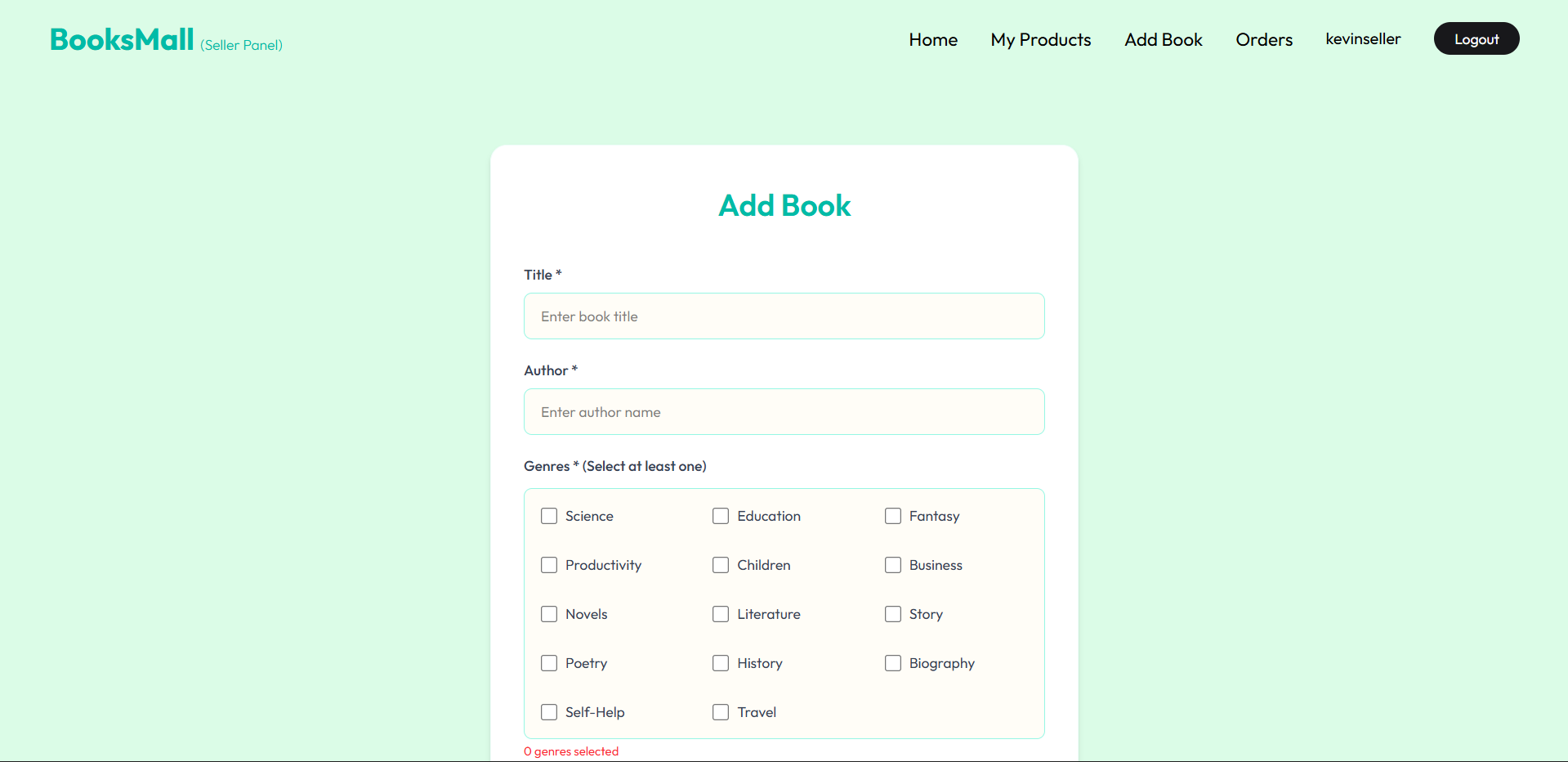
****

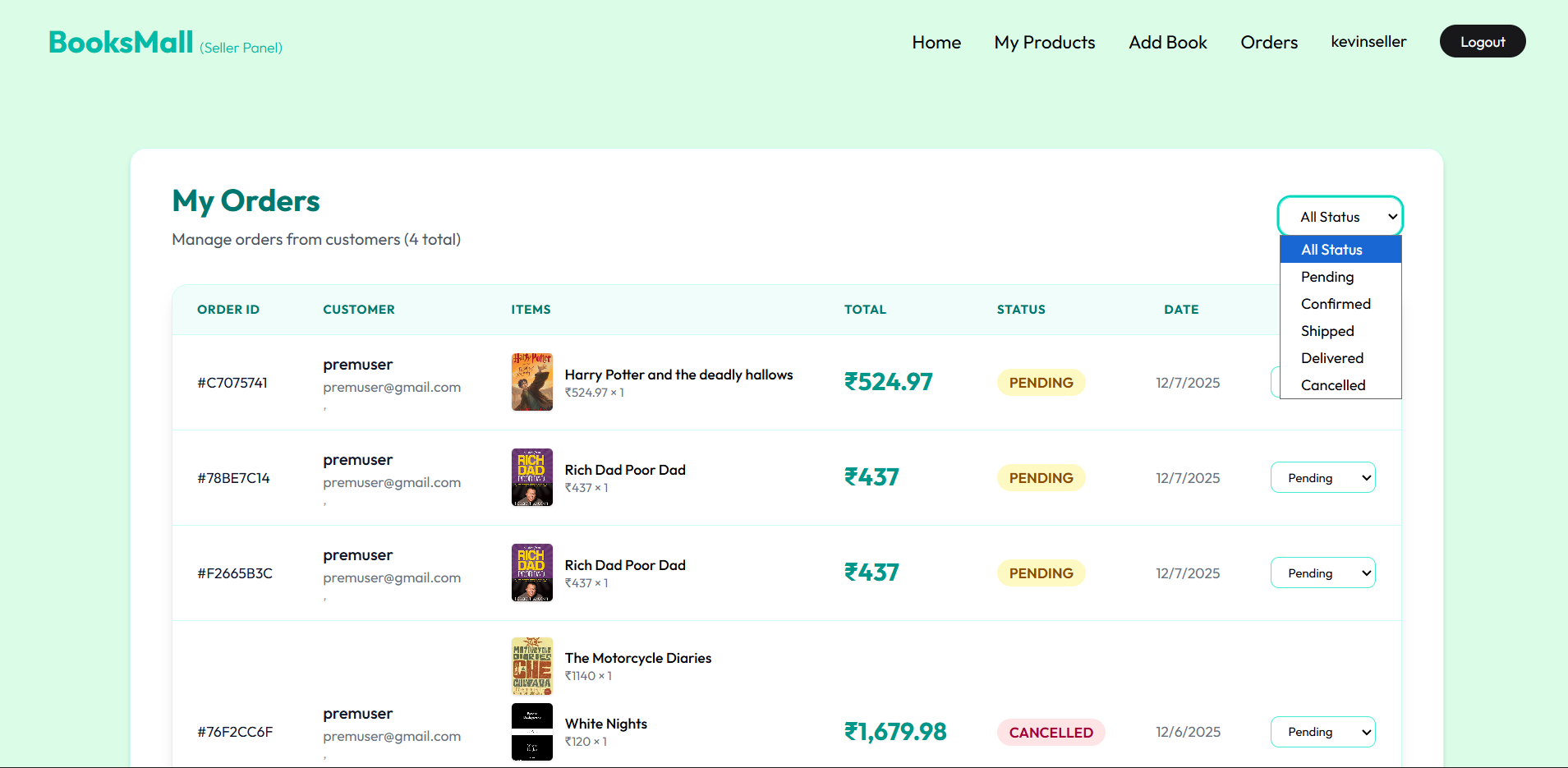
****

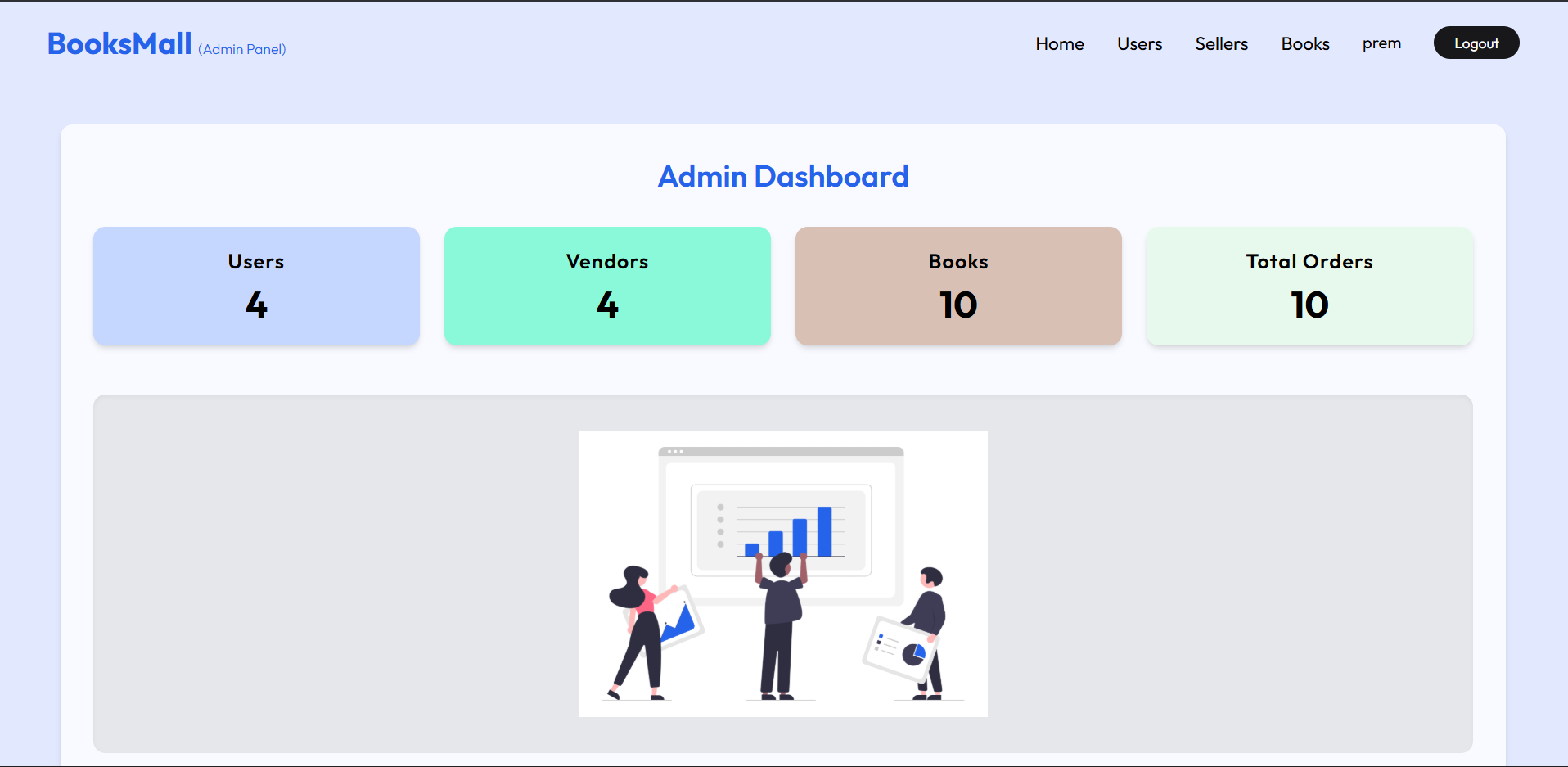
****

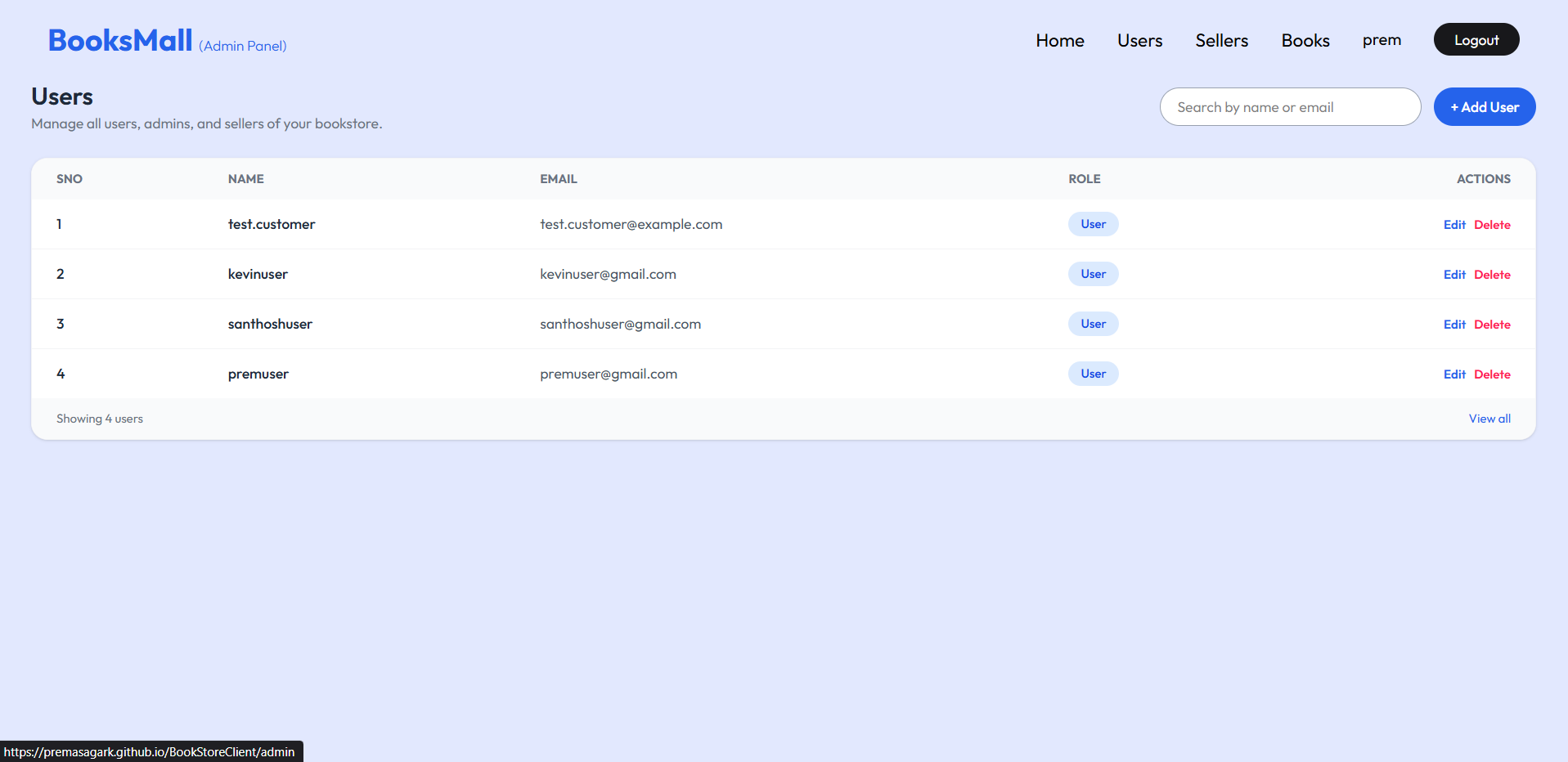
****

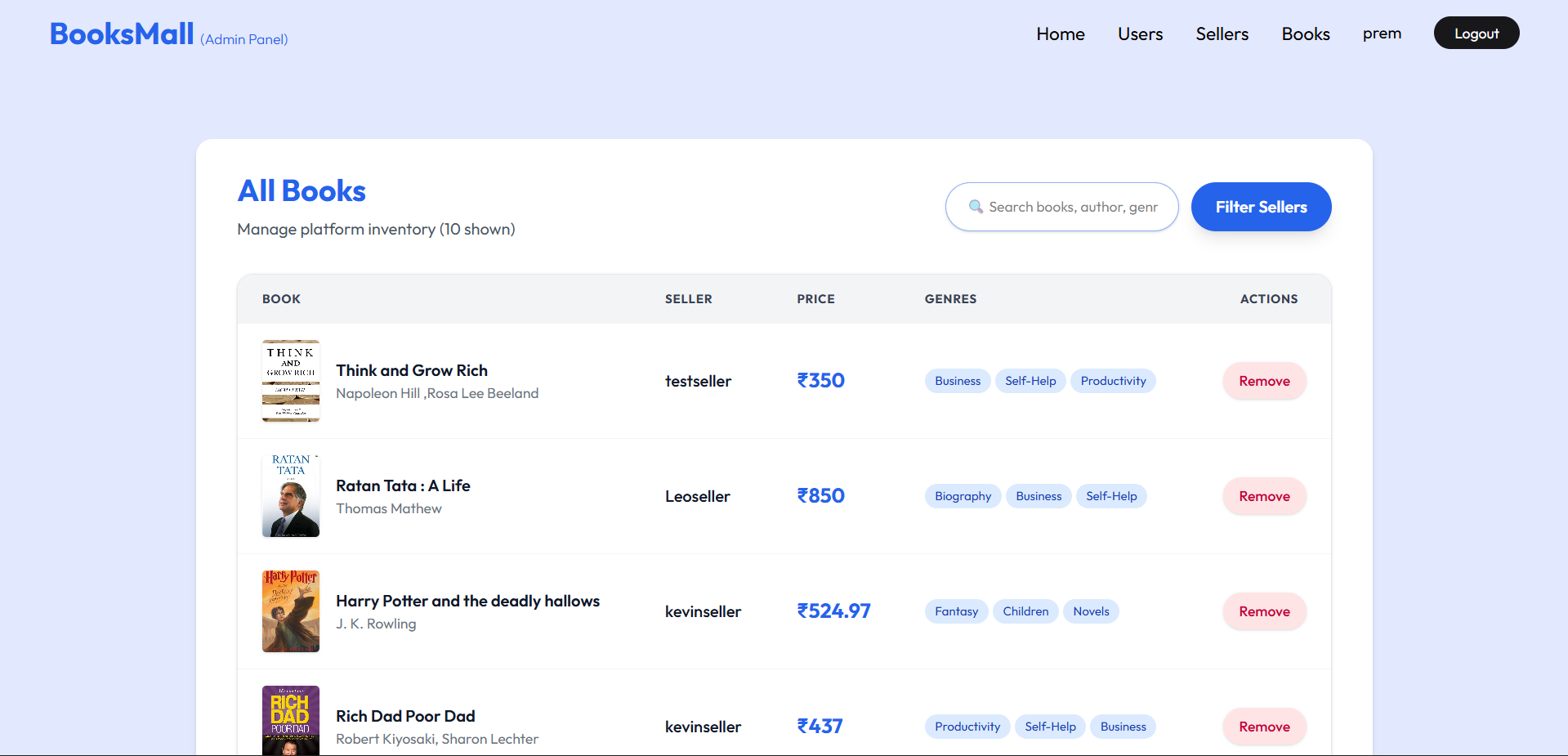
****

****

****

****

****

****

**9. Testing**

Testing was performed to ensure the Book Store application works smoothly, reliably, and meets all functional requirements. The following testing methods were applied:

**9.1 Functional Testing**

Functional testing was carried out to validate individual features of the system.

Modules Tested

* User Authentication
  + Login, registration, password validation
* Book Catalogue
  + View books, search books, filter by category
* Book Details
  + View full details, ratings, reviews
* Cart Management
  + Add to cart, remove from cart, update quantity
* Order Placement
  + Checkout, order summary, payment integration (if any)
* Admin Panel
  + Add/update/delete books, manage categories, view orders

Results

All modules were tested and validated to ensure expected outputs.

**9.2 UI/UX Testing**

Tested for:

* Mobile responsiveness
* Smooth navigation
* Consistent layouts
* Card scrolling behavior
* Button and icon functionality
* Infinite scroll behavior (Top Rated Section)

**9.3 Performance Testing**

Checked:

* Page load time
* Scroll performance for large book lists
* API response time
* Image loading and caching

**9.4 Compatibility Testing**

Application was tested on:

* Browsers: Chrome, Firefox, Edge, Safari
* Devices: Desktop, Tablet, Mobile

No major issues were found.

**9.5 Integration Testing**

Validated interactions between:

* Frontend ↔ Backend APIs
* Cart ↔ Checkout
* Admin panel ↔ Database
* Search engine ↔ Book catalogue

**9.6 Security Testing**

* SQL injection testing
* Input validation
* Password hashing verification
* Admin access restriction
* Token/session validation

**9.7 Bug Fixes**

Some issues encountered and resolved:

* Improper card looping in infinite scroll
* Slow loading of book list (fixed using lazy loading)
* Duplicate API calls during scroll
* Layout breaking on small screens
* Incorrect category filter results

**9.8 User Acceptance Testing (UAT)**

Final testing was done in the presence of sample users.  
Feedback included:

* UI improvements
* Faster navigation
* Better category grouping

Changes were applied where necessary.

**10. Known Issues**

• Document any known bugs or issues that users or developers should be aware of.

| **Issue** | **Description** | **Impact** | **Status** | **Workaround** |
| --- | --- | --- | --- | --- |
| **Role Redirect Failure** | When logging in as ADMIN or SELLER, the home route / loads successfully instead of redirecting to /admin or /seller | **Medium** - Users see wrong dashboard | **Open** | Manual navigation to /admin or /seller |
| **Protected Routes Work** | Routes inside ProtectedRoute (e.g., /admin/users, /seller/books) function correctly | **Low** - Core functionality intact | **Working** | Use protected sub-routes |

**11. Future Enhancements**

* **Short-term (1 month):** Real-time notifications, advanced search, book recommendations, image optimization.
* **Medium-term (6 weeks):** Seller analytics dashboard, bulk operations, seller verification, inventory alerts.
* **Long-term (3-6 months):** PWA support, Redis sessions, Docker deployment, CI/CD pipeline, multi-language, mobile app.