**Full Stack Development with MERN Project**

**Documentation**

# **1.introduction**

**Project Title:** BookNest: Where StoriesNestle **Team ID : LTVIP2025TMID55058Team Members :**

Kokkiligadda Rangith kumar(Team Leader)-Backend developer

Katuri Gayathri (Member 2)-Full Stack Developer

Katta Kanchana(Member 3)-Frontend Developer

Katuri Susmitha (Member 4)-Database &Testing

# **2. Project Overview**

**Purpose:** BookNest is a MERN stack online bookstore designed to allow users to browse, discover, and purchase books online seamlessly. It bridges the gap between the traditional bookstore experience and the convenience of modern digital access.

**Features:**

* User registration and JWT-based authentication.
* Add to cart and secure checkout with order generation.
* Order history tracking for past purchases.
* Responsive design for desktop, tablet, and mobile.
* Real-time inventory updates after purchases.

# **3. Architecture**

**Frontend(react):**

* Built with React.js for a dynamic, component-based UI. ▪ Uses React Router for navigation between Home, Login, ▪ Register, Cart, Orders, and Admin pages.
* Axios is used for making HTTP requests to the backend APIs.
* Uses Bootstrap/Tailwind CSS for a responsive, clean interface.

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

* RESTful API architecture for handling CRUD operations on books, users, and orders.
* JWT for authentication and protected routes.
* Organized with controllers, models, and routes for scalability and maintainability.

|  |  |
| --- | --- |
| **Database(MongoDB):** |  |
| ▪ | MongoDB stores user profiles, book collections, and order |
| ▪ | histories. |
| ▪ | Uses Mongoose for schema definition and data validation. |

**Relationships:**

|  |  |
| --- | --- |
| ▪ | Each Order references User and Book documents. |
| ▪    **4. Setup Instructions**  **Prerequisites:** | Book collection stores title, author, genre, description, price, and stock availability. |
| ▪ | Node.js (v18+ recommended) |
| ▪ | MongoDB (local or Atlas) |
| ▪ | Git |

**Installation:**

|  |
| --- |
| # Clone the repository  git clone https://github.com/yourusername/booknest.git cd booknest    # Install server dependencies cd server  npm install    # Install client dependencies  cd ../client  npm install    # Set up environment variables # In server/.env:  PORT=5000  MONGO\_URI=mongodb+srv://bookuser:Book%401234@myatlasclusteredu .iispjpa.mongodb.net/bookstore?retryWrites=true&w=majority&appName= myAtlasClusterEDU  JWT\_SECRET= myVeryStrongSecret12345 |

**5. Folder Structure**:

**Client:**

client/

├── public/

│ └── index.html

├── src/

│ ├── components/ # Reusable components (Navbar, BookCard, etc.)

│ ├── pages/ # Page components (Home, Cart, Orders, etc.)

│ ├── App.js # Route definitions

│ ├── index.js # Entry point

│ └── api/ # Axios API calls

**Server:**

server/

├── controllers/ # Request handling logic

├── models/ # Mongoose schemas (User, Book, Order)

├── routes/ # Express routes

├── middleware/ # Authentication middleware

├── server.js # Entry point

└── config/ # DB connection

# **6. Running the Application**

• Provide commands to start the frontend and backend servers locally. **Frontend:** cd client

npm start

**Backend:** cd server

npm start

# **7. API Documentation**

**Base URL: http://localhost:5000/api**

**Authentication:**

* POST /api/auth/register: Register a new user.
* POST /api/auth/login: Login and receive JWT.

**Books:**

* GET /api/books: Retrieve all books.
* POST /api/books: Add a new book (Admin/Seller).
* GET /api/books/:id: Get book details.
* PUT /api/books/:id: Update book details.
* DELETE /api/books/:id: Delete a book.

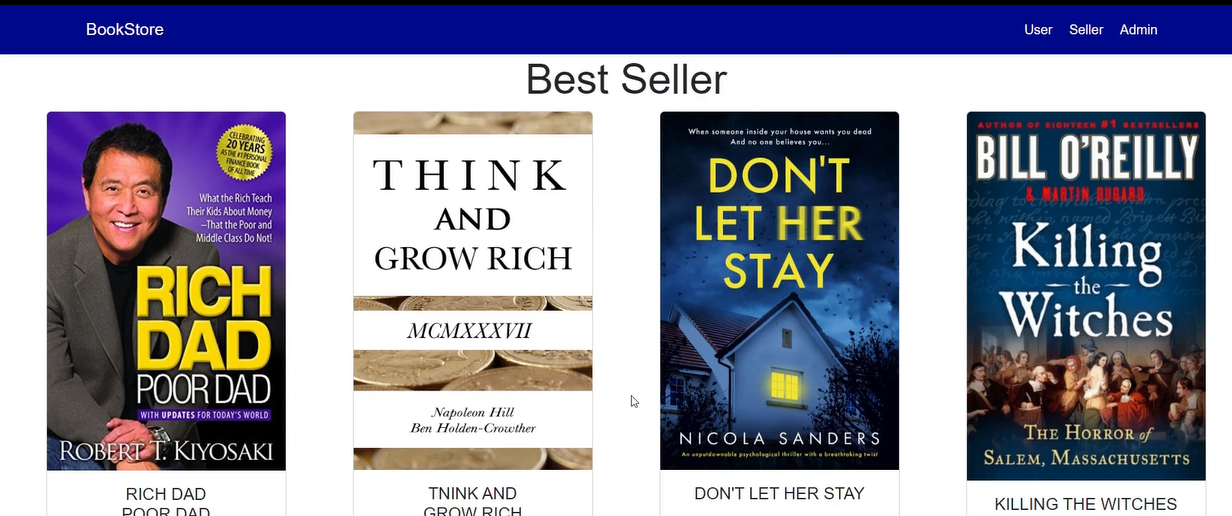
**Cart/Orders:**

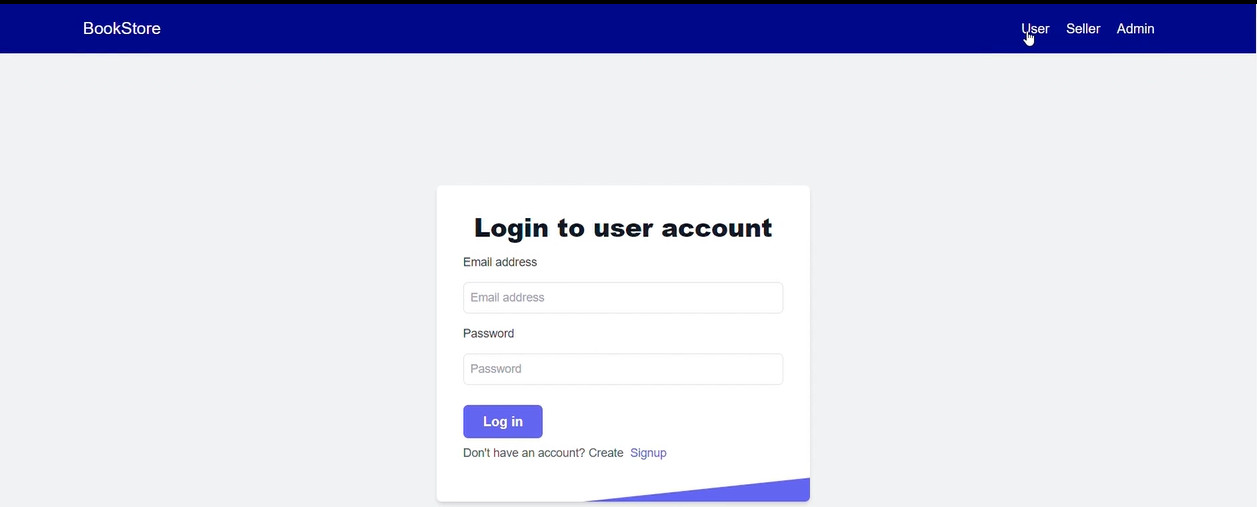
* POST /api/orders: Place a new order.
* GET /api/orders/user/:userId: Get user’s order history.
* GET /api/orders/:id: Get order details by order ID

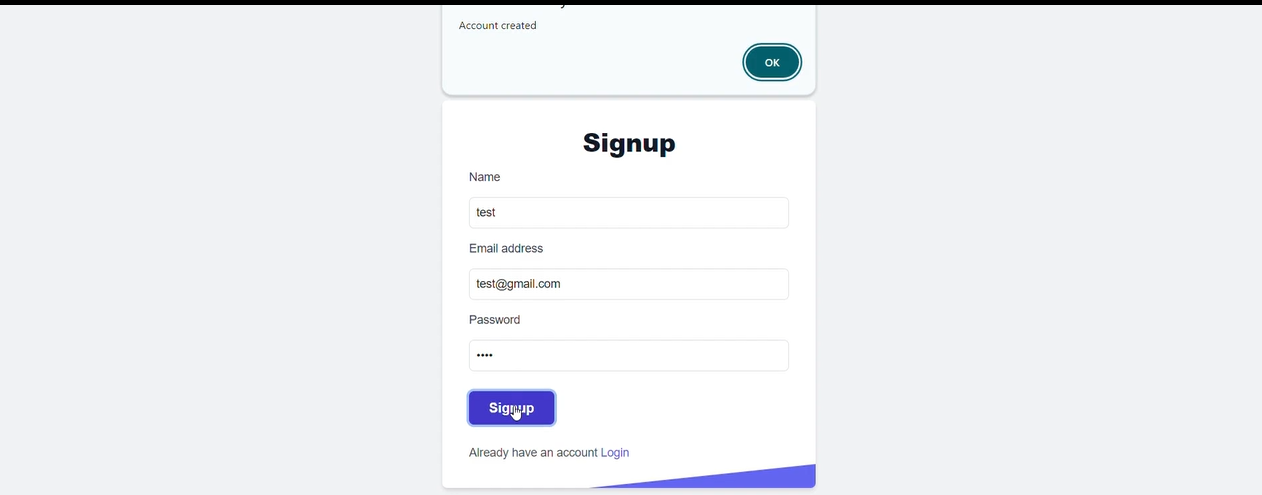
# **8. Authentication**

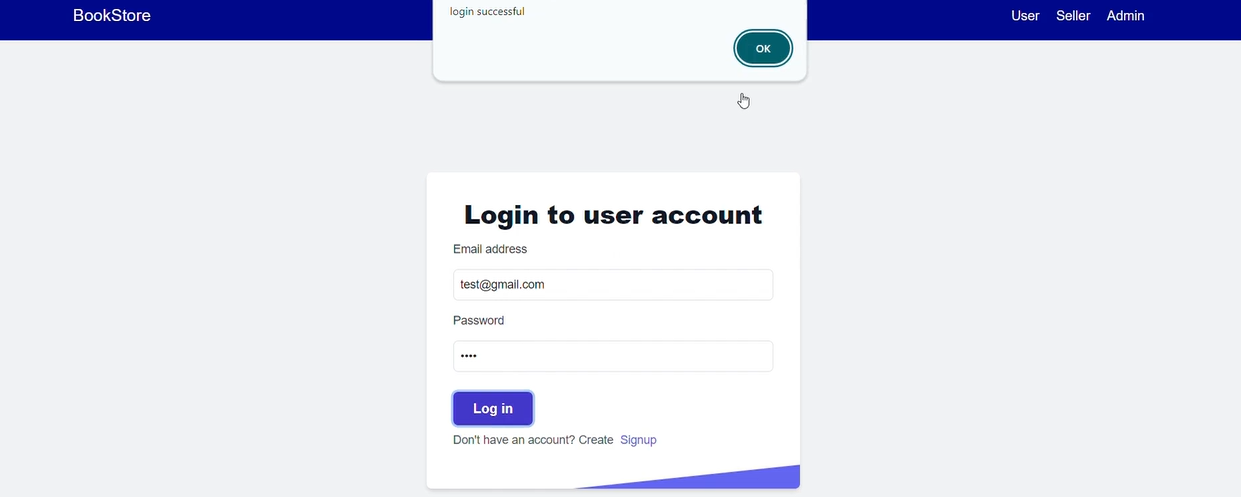
* JWT tokens are generated on login and stored in localStorage on the client.
* Protected routes on the server check the validity of the token using middleware.
* Passwords are hashed with bcryptjs before storage in MongoDB.
* Users can only access their own order history.
* Include details about tokens, sessions, or any other methods used.

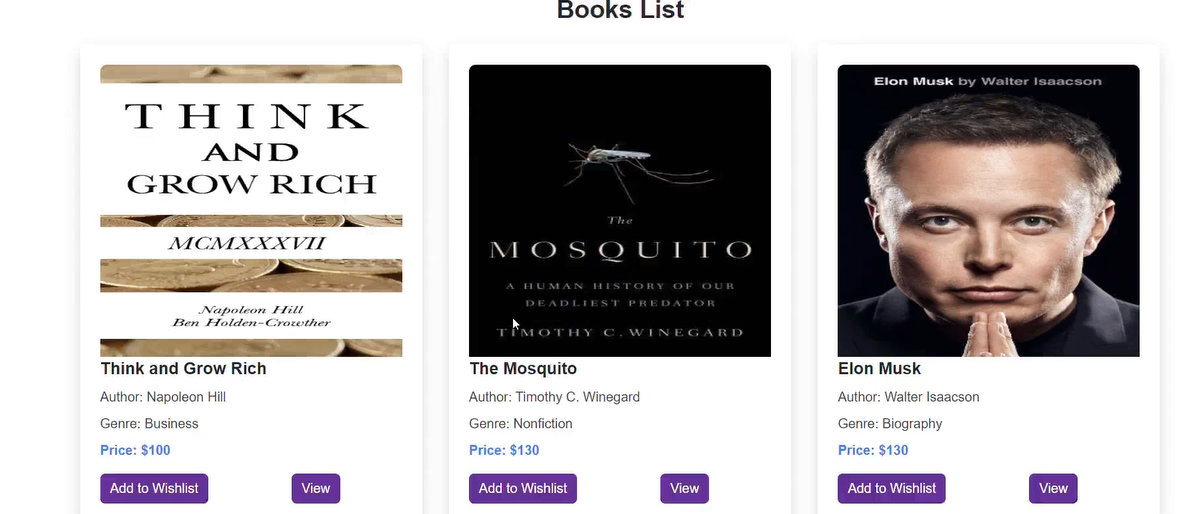
# **9. User Interface**

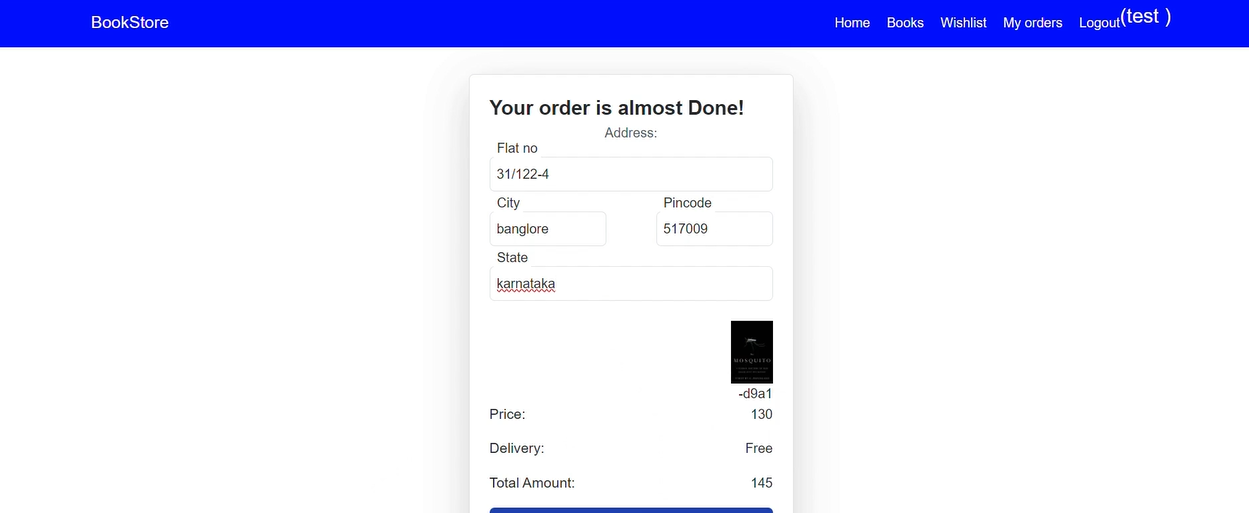


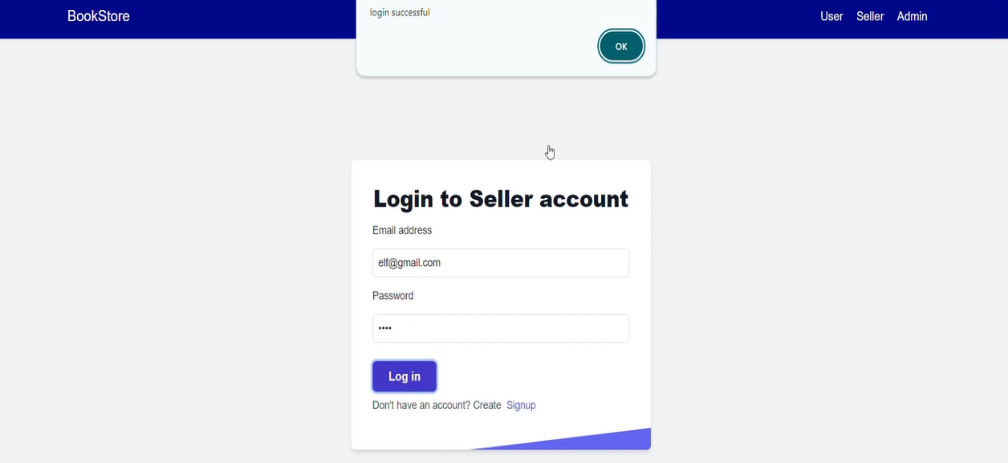


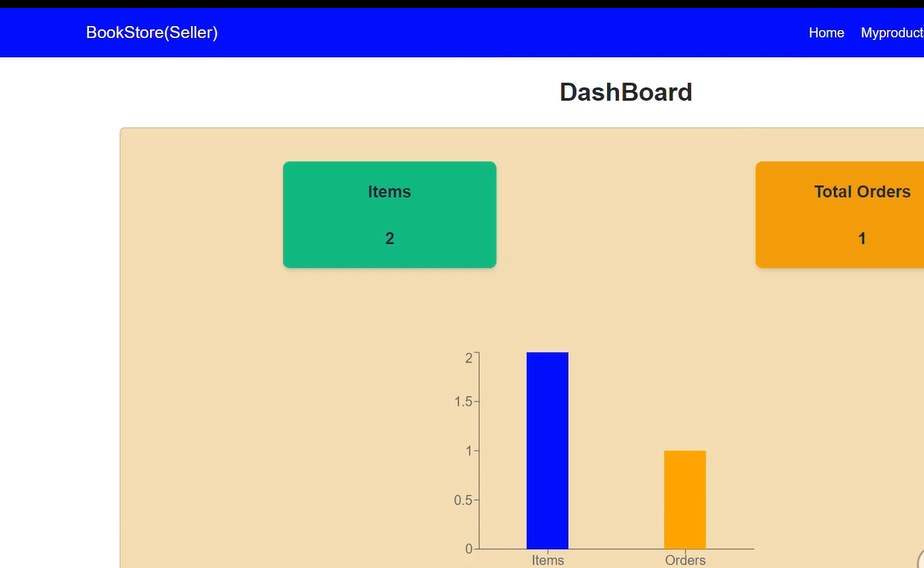


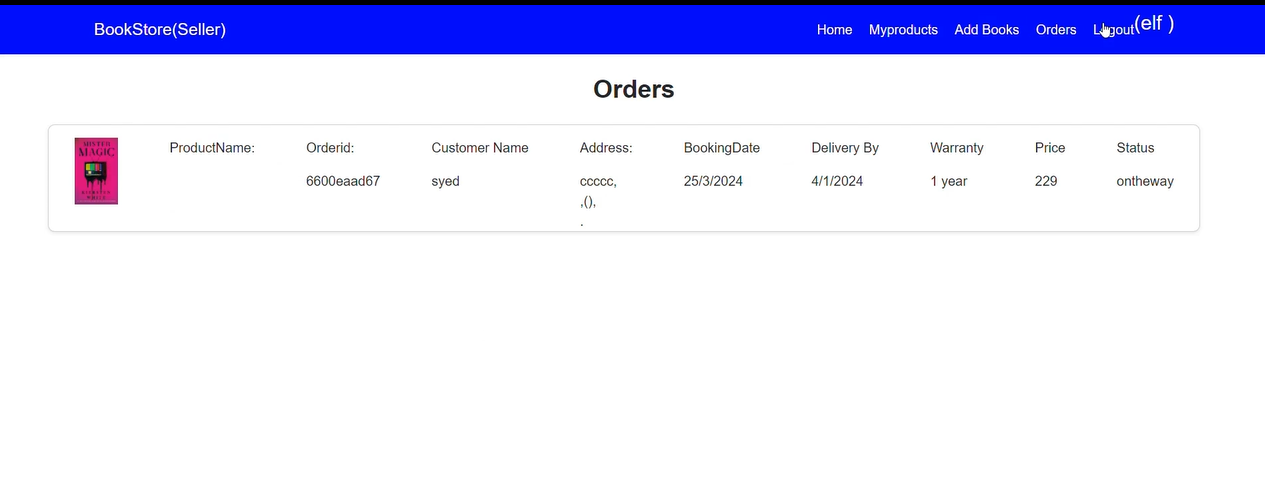


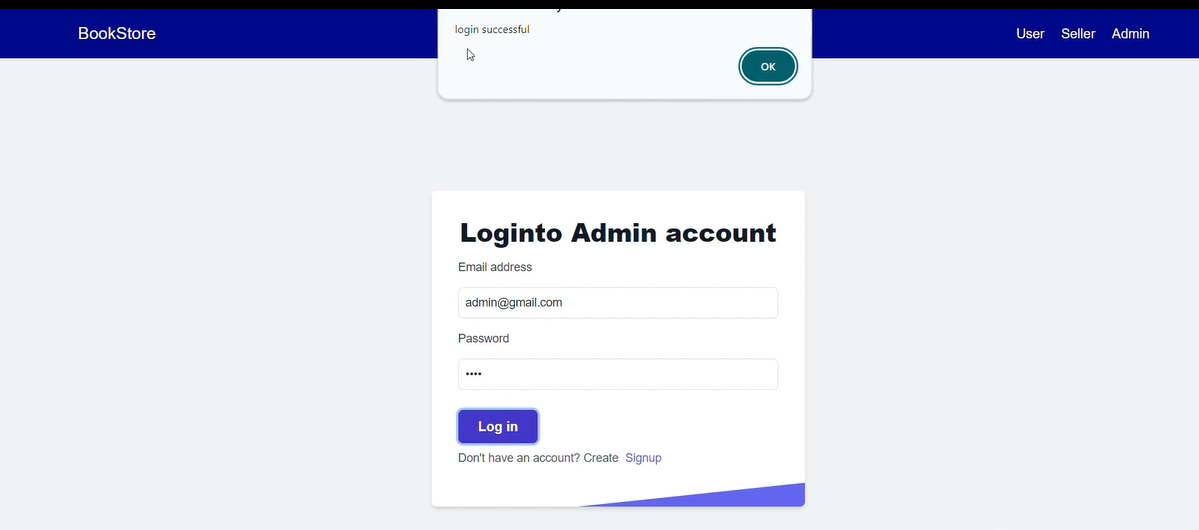












# **10. Testing**

* Manual testing of CRUD operations using **Postman**.
* UI testing by simulating user workflows (registration, adding to cart, checkout).
* Future: Integration of **Jest** and **React Testing Library** for frontend unit testing.

# **11. Screenshots or Demo**

<https://drive.google.com/file/d/1OL6IGM780tJvR3mrjfzFnvqC2KJweDdk/view?usp=drivesdk>

# **12.Known Issues**

* No integrated payment gateway (currently mock checkout).
* No advanced error handling on frontend for network failures.
* Admin panel for seller/book management is under development.

# **13. Future Enhancements**

* Payment gateway integration (Stripe/Razorpay).
* Email notifications upon order placement.
* Admin/Seller dashboard with book and order management.
* Reviews and rating system per book.

Unit and integration test coverage with CI/CD setup