**Full Stack Development with MERN Project**

# 1. Introduction

* **Project Title:** BookNest – MERN Stack Online BookStore
* **Team Members:** LTVIP2025TMID53930
* Satwik Indupuri
* Palak Suryawanshi
* Palivela Sai Dhanush Raj
* Padala Divya Sri

# 2. Project Overview

 **Purpose**:  
BookNest is a full-fledged MERN Stack web application designed for an online book marketplace. It allows users to browse, buy books, manage orders, and provides separate dashboards for sellers and admins.

 **Features**:

* 📚 Book browsing by users.
* 🛒 Cart management and order placement.
* 🧑‍💼 Sellers can add/edit/delete their books.
* 👨‍💼 Admin can manage users, books, and orders.
* 🔐 JWT-based authentication for role-based access.
* 📈 Dashboards with charts and metrics.

# 3. Architecture

 **Frontend**:  
Built using **React.js** and **Vite**. Components are modularized. Routes managed using react-router-dom. State handled locally and with context where necessary.

 **Backend**:  
Developed with **Node.js + Express.js**, structured with separate route, controller, and middleware files.

 **Database**:  
MongoDB with **Mongoose** ODM. Collections:

* users: stores user, seller, admin credentials.
* books: stores book details with seller reference.
* orders: stores placed order data including books and users.

# 4. Setup Instructions

* **Prerequisites:**
* Node.js
* Express.js
* MongoDB
* React js

**Quick  Start**

* npm create vite@latest
* cd my-app
* npm install
* npm run dev

**Installation:**

git clone https://github.com/your-username/booknest.git

* cd BookNest

# Setup backend

* cd Backend
* npm install
* touch .env

# Add MONGO\_URI and PORT in .env

* npm start

# Setup frontend

* cd ../frontend
* npm install
* npm run dev

# 5. Folder Structure

* **Client:**

**Frontend (frontend/)**:

* src/public/
* components/ # Navbar, Layout
* pages/ # Login, Register, Admin Dashboard, Seller Dashboard, AddBook,

Books, Cart, EditBook, Landing, MyOrders, MyProducts.

* styles/ # Custom CSS
* services/ # Axios instance
* **Server:**

Backend/

* routes/ # userRoutes, bookRoutes, orderRoutes
* controllers/ # bookcontroller,ordercontroller,usercontroller
* models/ # Mongoose schemas
* middleware/ # Auth middleware

# 6. Running the Application

• Provide commands to start the frontend and backend servers locally.

o **Frontend:** cd frontend , npm run dev

O **Backend:** cd backend , npm start

# 7. API Documentation

# Users

# POST /api/users/register – Register a user

# POST /api/users/login – Login and receive JWT

# GET /api/users – (Admin) Get all users

# DELETE /api/users/:id – (Admin) Delete a user

# Books

# POST /api/books – (Seller) Add book

# GET /api/books – Get all books

# GET /api/books/seller/:sellerId – Get seller's books

# DELETE /api/books/:id – Delete book

# Orders

# POST /api/orders – Place order

# GET /api/orders/my-orders/:userId – Get user orders

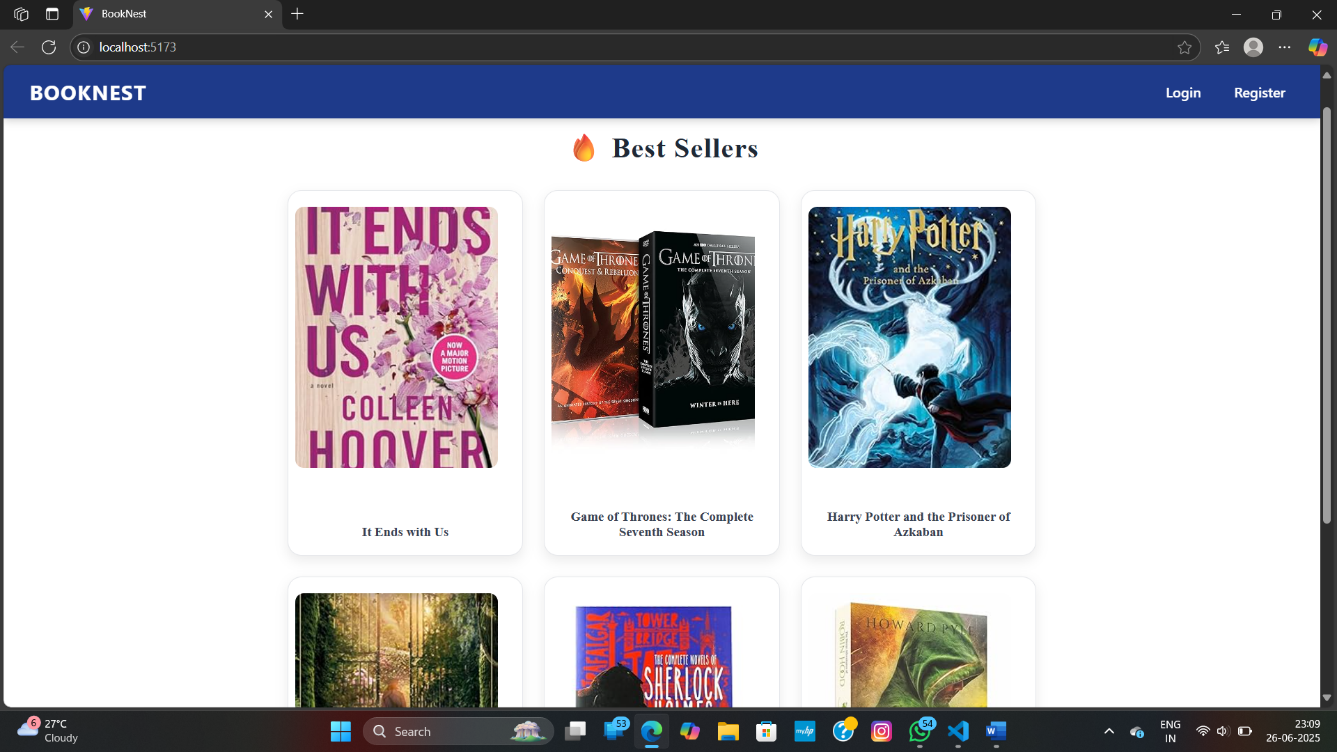
# GET /api/orders/all – (Admin) Get all orders

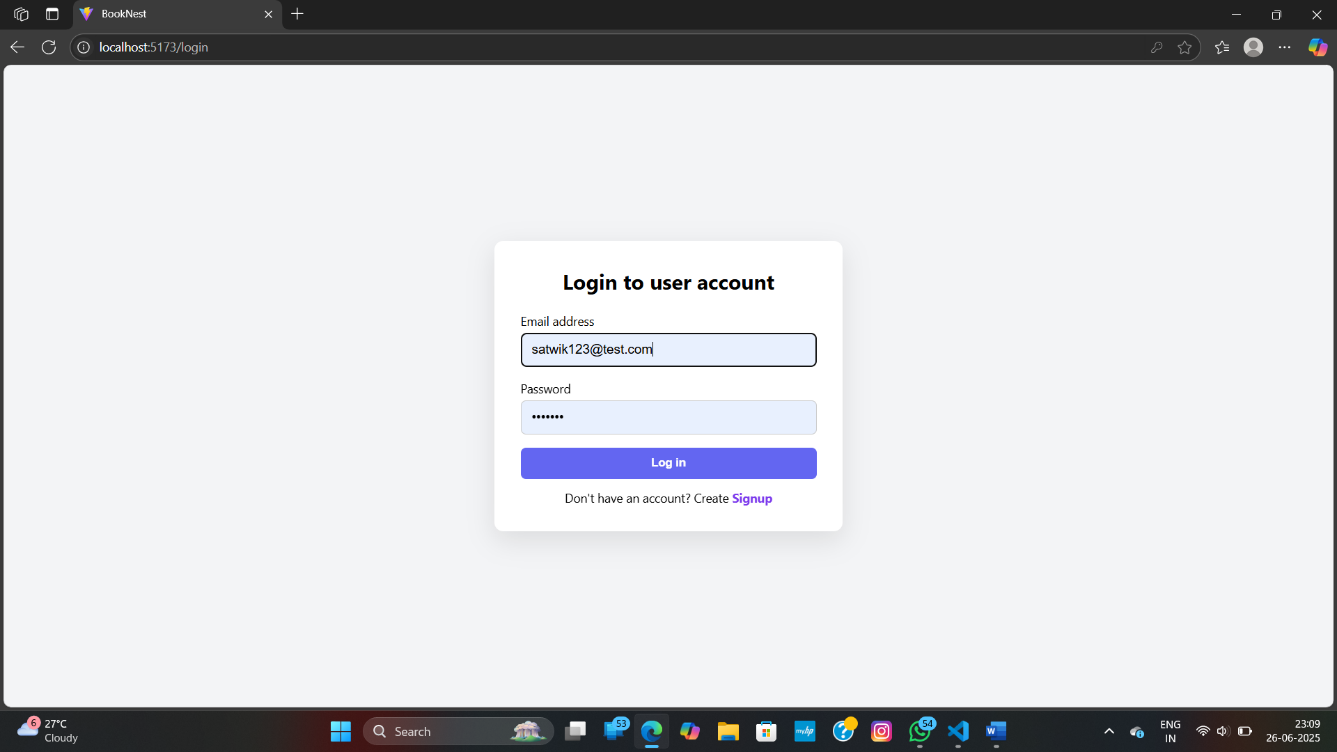
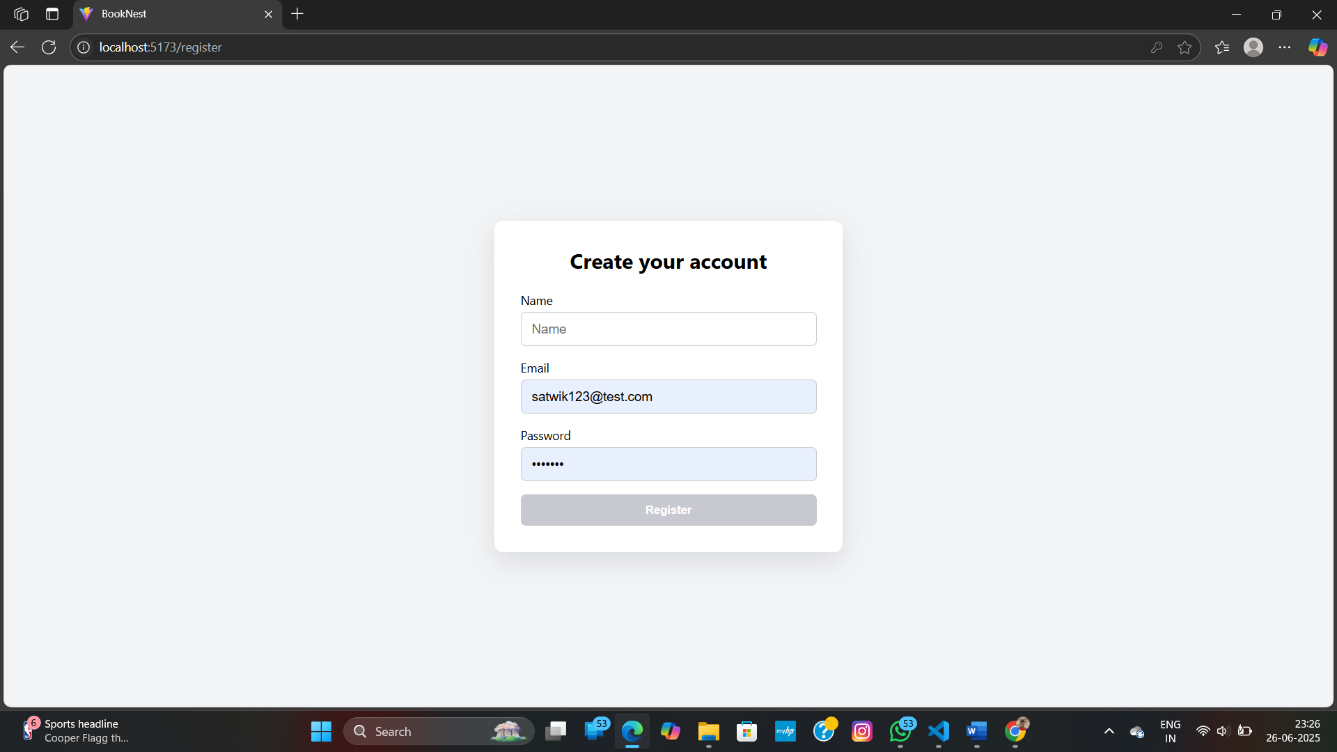
# PUT /api/orders/:id/status – Update order status

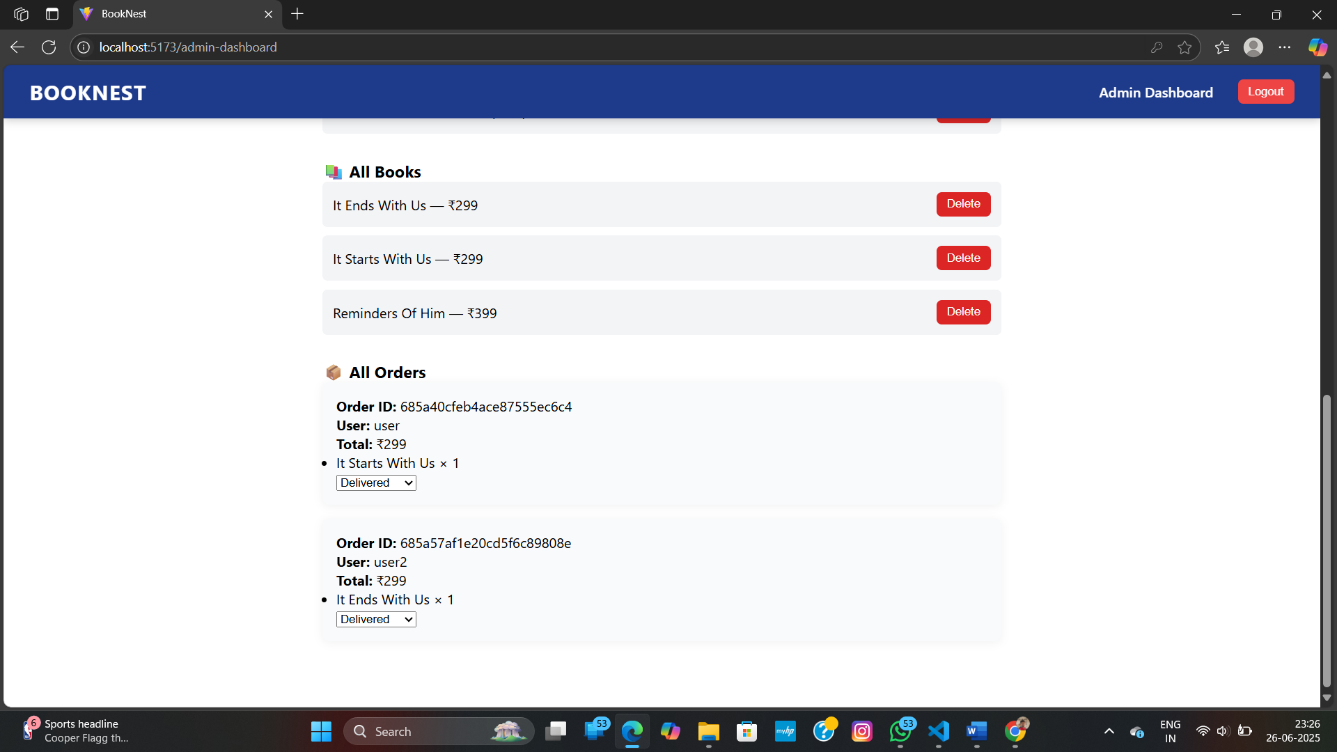
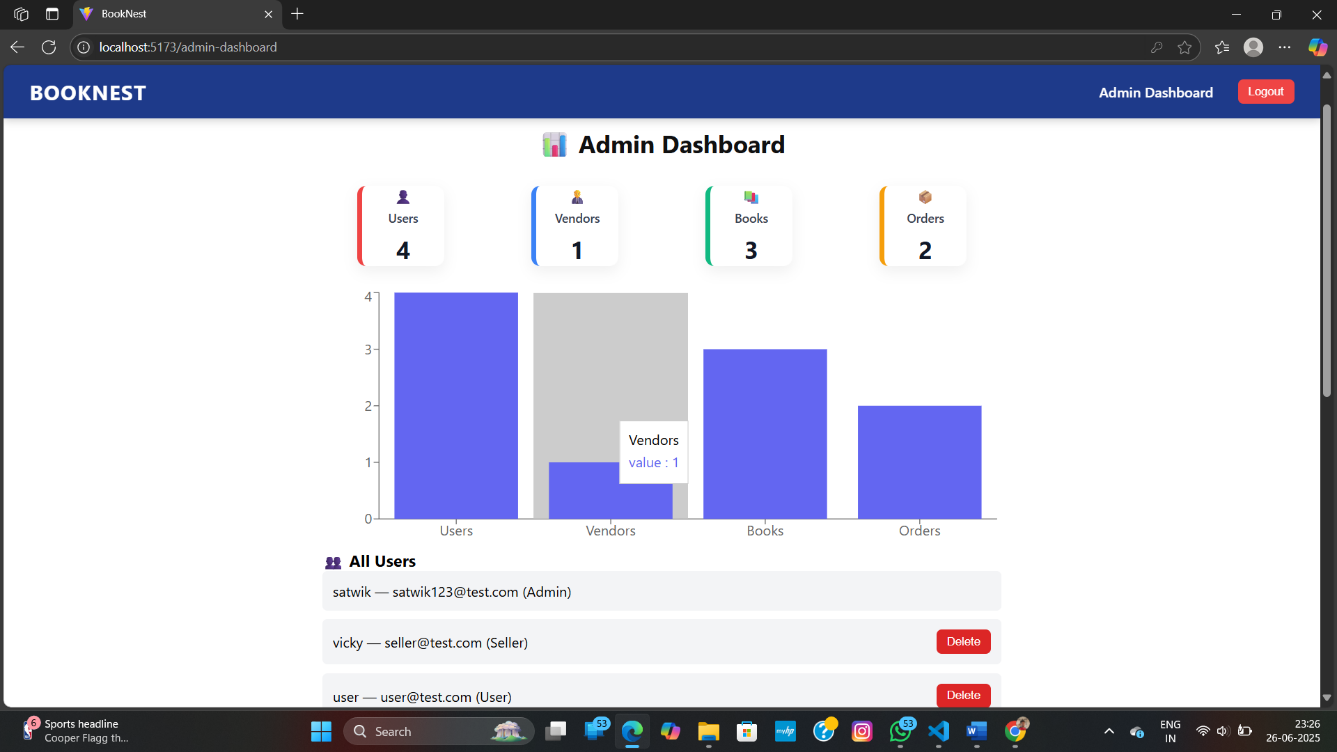
# 8. Authentication

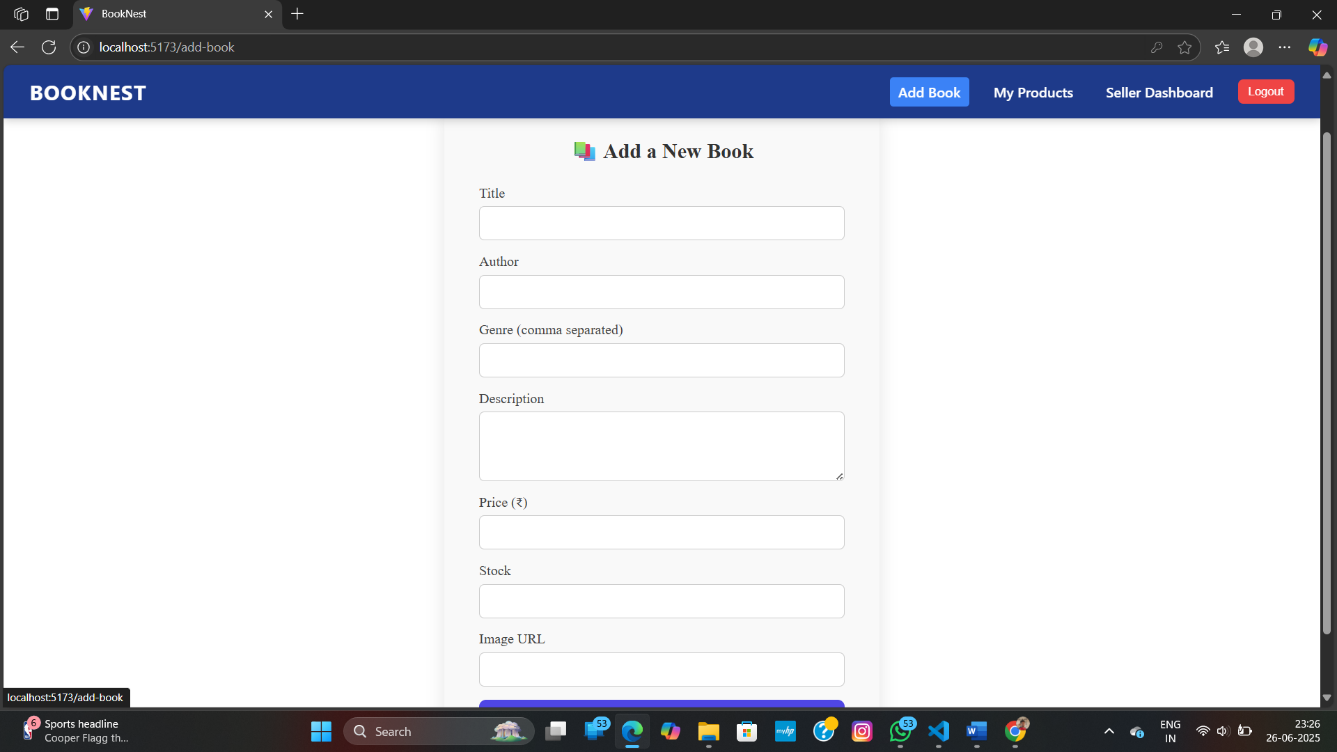
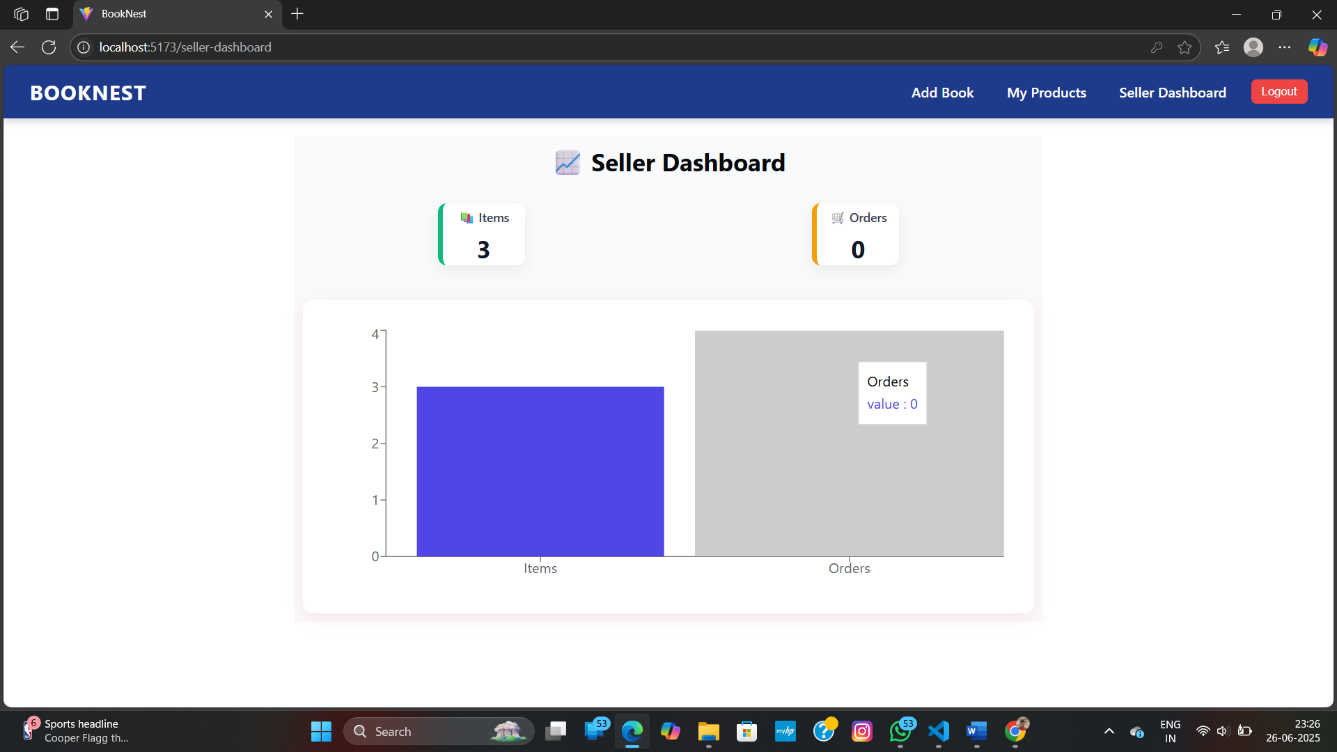
* Implemented using **JWT tokens**.
* On login/register, token is stored in localStorage.
* verifyToken middleware ensures protected routes.
* Admin/Seller roles determined from token payload.

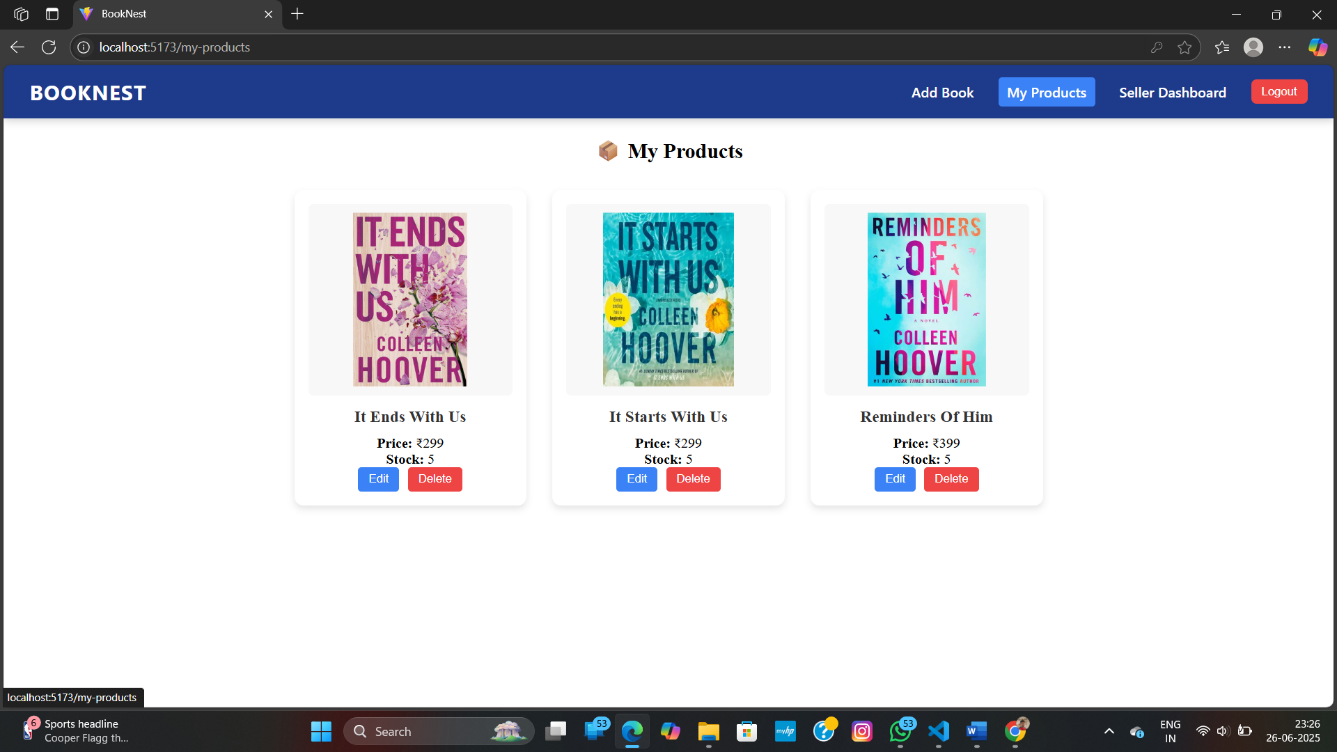
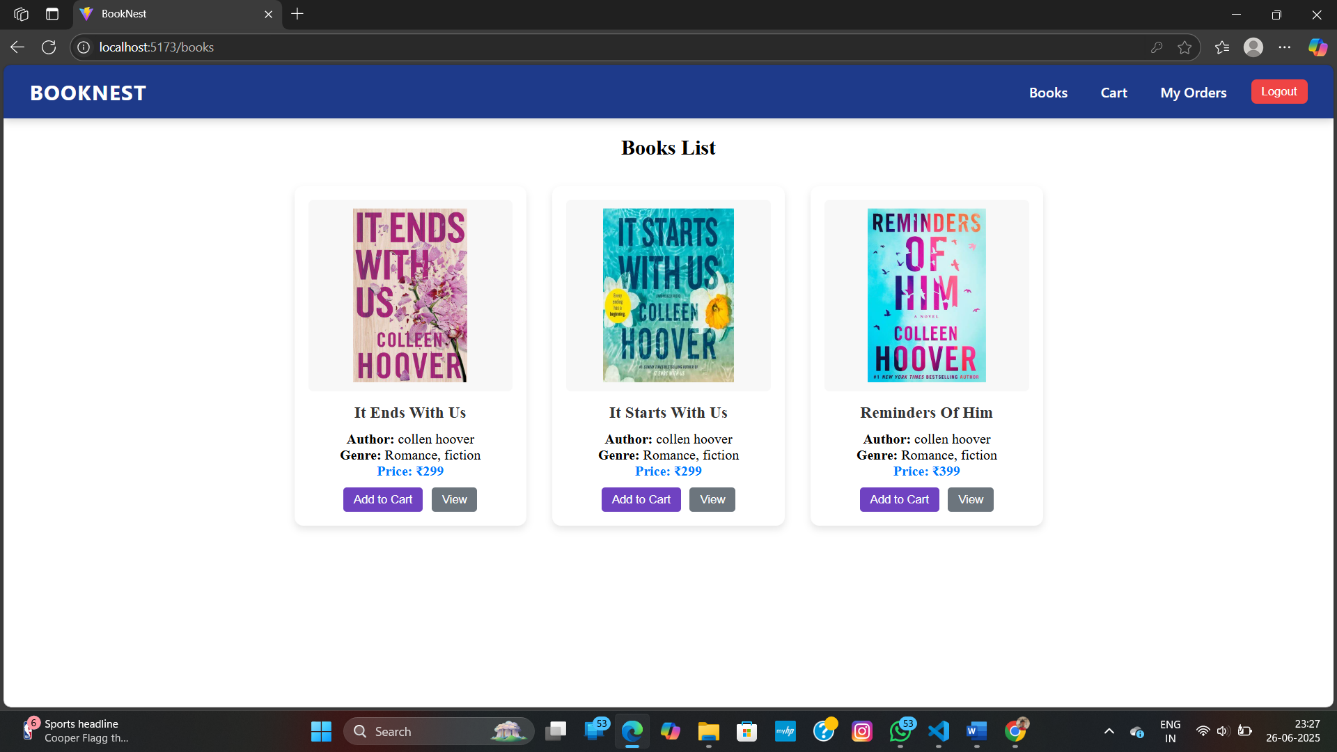
1. **User Interface** 
   * Provide screenshots or GIFs showcasing different UI features.
2. **Testing** 
   * Manual testing with real-time data.
   * Thunder Client used to test backend APIs.
   * MongoDB Compass used to inspect database state.
3. **Screenshots or Demo** 
   * Provide screenshots or a link to a demo to showcase the application.

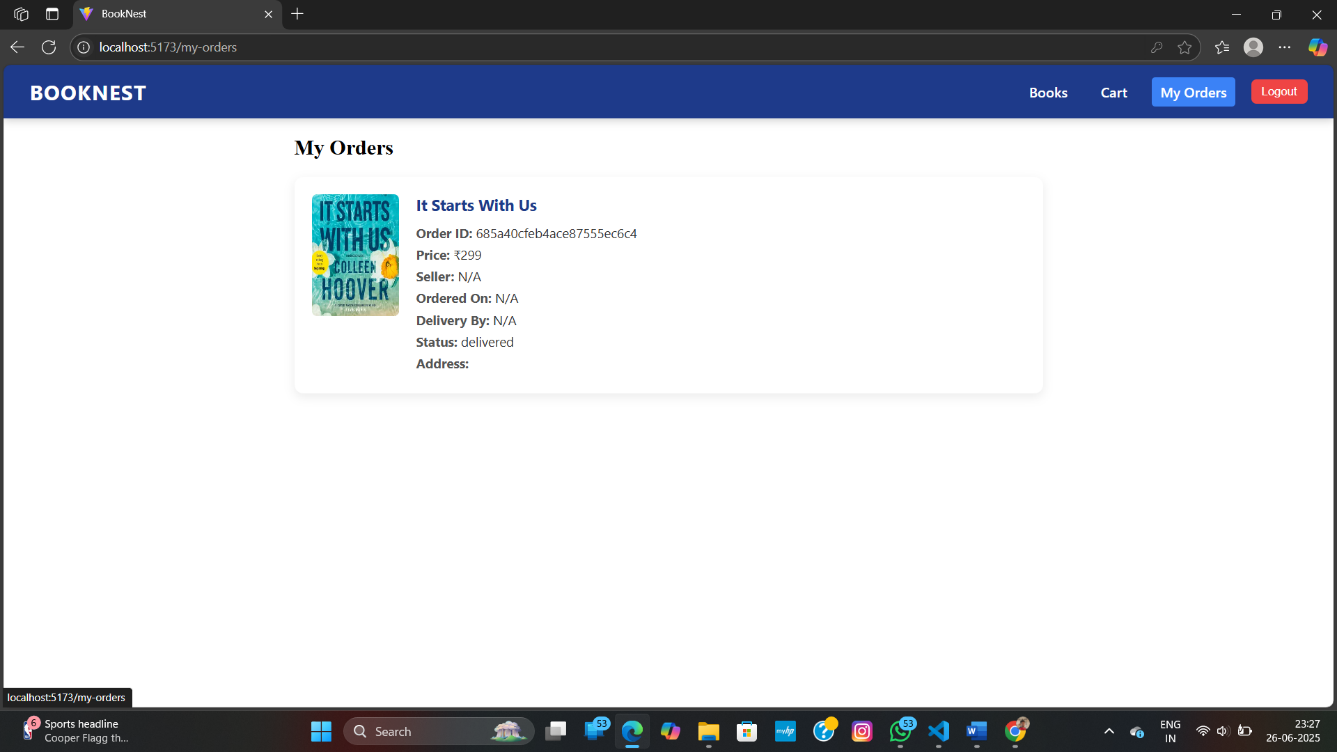
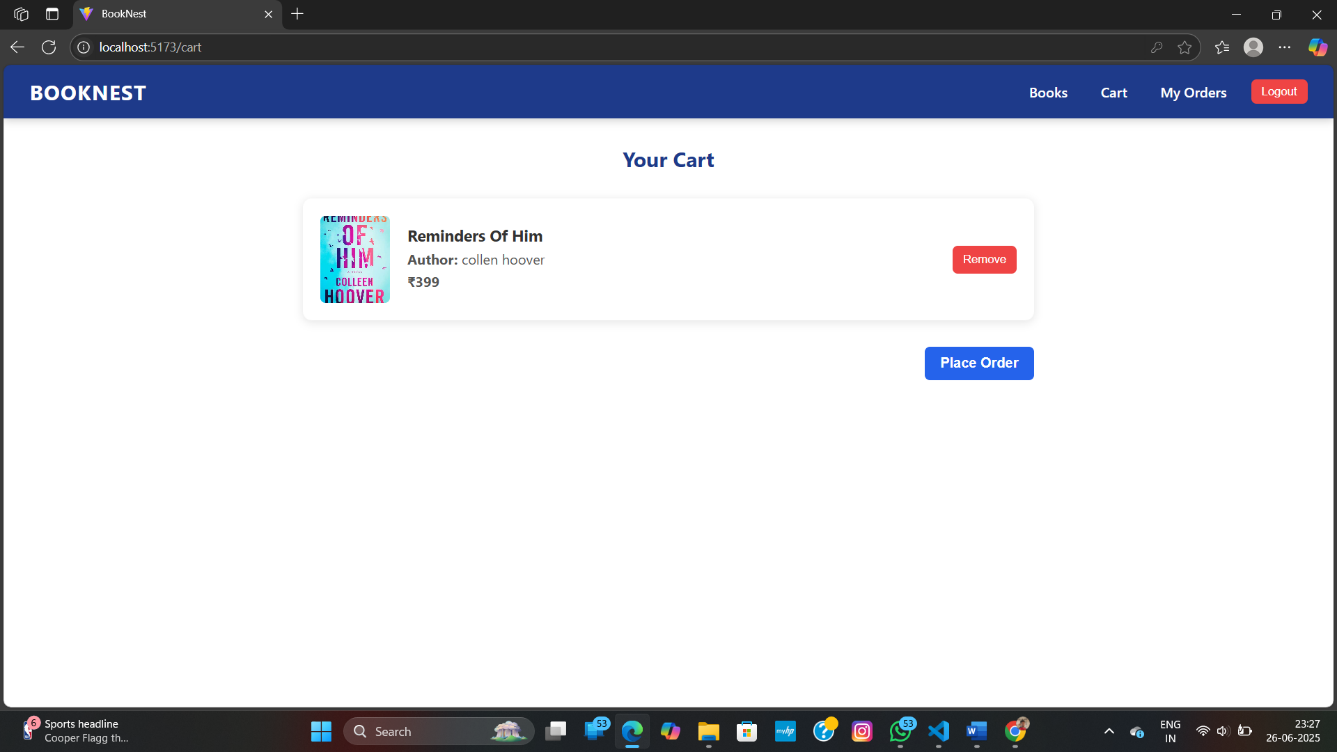












1. **Known Issues**

 Order status logic may need further scalability.

 Image upload currently works via URL only.

# 13. Future Enhancements

* Image upload via file (Cloudinary/Storage).
* Pagination and search for books.
* Role management UI (make user → seller).
* Email notifications on order placement.