Project Title: Bookstore Management System – Full-Stack Web Application (MERN Stack / React + Node.js + Express + MongoDB)

Client Overview:

The client is a bookstore owner who wants a complete web-based solution to manage books, users (customers and admins), and orders. The system should support user interaction through a web interface and provide a seamless experience for browsing books, managing inventory, registering users, and processing orders.

Project Objective:

To design and develop a full-stack bookstore management system using **React** for the frontend and **Node.js with Express.js** for the backend. The application will support **user authentication**, **book browsing**, **order placement**, and **inventory management**, with data stored in **MongoDB**. Admins will have additional privileges to manage inventory and orders.

Frontend Requirements (React):

1. **User Interface:**
    - Responsive design using **React + CSS/SCSS or UI frameworks** (e.g., Bootstrap, Material-UI).
    - Separate views for:
        - Homepage with featured books
        - Book listings with search & filter (by genre, author, title)
        - Book detail page
        - Shopping cart & checkout
        - Admin dashboard (book & order management)
2. **Authentication & Authorization:**
    - Login and registration pages
    - Token-based authentication (JWT)
    - Role-based access control:
        - Customers: view & order books
        - Admins: access to book and order management views
3. **State Management:**
    - Use **Redux** or **React Context** to manage application state (auth, cart, user info, etc.).
4. **Routing:**
    - Use **React Router** for client-side routing (e.g., /books, /login, /orders, /admin).

5. **UX Features:**
    - Pagination and search for book listings
    - Toast notifications for success/errors (e.g., react-toastify)
    - Confirmation dialogs for deletions and critical actions

---

Backend Requirements (Node.js + Express):

1. **API Development:**
    - RESTful API to handle:
        - Book Management
        - User Authentication
        - Order Processing
2. **API Endpoints Overview:**

Books:

- GET /api/books
- GET /api/books/:id
- POST /api/books (Admin only)
- PUT /api/books/:id (Admin only)
- DELETE /api/books/:id (Admin only)

Users:

- POST /api/register
- POST /api/login

Orders:

- GET /api/orders (Admin only)
- GET /api/orders/:id
- POST /api/orders
- PUT /api/orders/:id/status (Admin only)

3. **Security:**
    - Use **JWT** for authentication
    - Middleware for **role-based access control**
    - Store hashed passwords using **bcrypt**
4. **Database Design (MongoDB):**

- Collections:
    - Users (name, email, hashed password, role)
    - Books (title, author, price, genre, stock, ISBN, description, image URL)
    - Orders (user info, list of books, status, total price, payment status)
- Mongoose schemas with validation

5. **Performance Enhancements:**
    - Add **pagination** and **search** in book and order endpoints
    - Optimize queries and use indexes where necessary
6. **Error Handling:**
    - Return proper HTTP status codes and messages
    - Catch unhandled errors using centralized error middleware
7. **API Documentation:**
    - Use **Swagger** or **Postman collections** to document and test endpoints

---

Project Timeline:

Week	Tasks
1	Set up project, create backend (Express) and frontend (React) structure, implement user registration/login
2	Build book management module (CRUD) and frontend views with search/pagination
3	Develop order processing, integrate cart & checkout, admin dashboard for orders
4	Error handling, testing, polish UI/UX, API documentation, deployment (e.g., Vercel + Render/Heroku)

Deliverables:

1. Full-stack web application with React frontend and Node.js backend
2. RESTful API with JWT authentication
3. MongoDB integration with proper schema design
4. Secure, role-based access controls
5. Responsive UI with admin and customer dashboards
6. API documentation (Swagger or Postman)
7. Testing for core features (e.g., Jest for backend, React Testing Library for frontend)

Evaluation Criteria:

- **Functionality:** All major features implemented (books, users, orders)
- **Security:** Proper authentication and authorization mechanisms
- **Code Quality:** Modular, reusable components and well-structured backend
- **UI/UX:** Clean, user-friendly interface with mobile responsiveness
- **Performance:** Fast, responsive UI and optimized API
- **Optional Features:** Extra points for features like reviews, Stripe payment, or real-time notifications

---

Technologies:

- **Frontend:** React, React Router, Redux / Context API, Axios
- **Backend:** Node.js, Express.js
- **Database:** MongoDB + Mongoose
- **Authentication:** JWT + bcrypt
- **Testing:** Jest, React Testing Library, Postman
- **Deployment:** Vercel (frontend), Render/Heroku (backend)

Optional Enhancements:

- **Payment Gateway Integration** (Stripe/PayPal)
- **Book Reviews and Ratings**
- **Email Notifications** (e.g., order confirmations)
- **Real-Time Admin Dashboard** (e.g., using Socket.IO)

This Bookstore Management System REST API project will allow you to focus on building robust and scalable RESTful services using Java and Spring Boot, while also giving you practical experience with user authentication, database management, and performance optimization.