Here's a high-level plan/design:

Design:

1. Requirements and Specifications
    - User Roles: Admin, Registered Users, and Guests.
    - Features:
        - User Authentication and Authorization.
        - Book Catalog with Search and Filter options.
        - Shopping Cart functionality.
        - Order Management.
        - Payment Integration.
        - Admin Dashboard for managing books, users, and orders.
2. System Architecture
    - Frontend: React.js or Angular for building the user interface.
    - Backend: Spring Boot for handling business logic and API endpoints.
    - Database: MySQL or PostgreSQL for storing data.
    - Authentication: JWT (JSON Web Tokens) for secure authentication.
    - Payment Gateway: Integration with a service like Stripe or PayPal.
3. Database Design
    - Users Table: Stores user information (id, username, password, email, role).
    - Books Table: Stores book details (id, title, author, genre, price, stock).
    - Orders Table: Stores order details (order_id, user_id, total_amount, order_status).
    - Order_Items Table: Stores items in each order (order_item_id, order_id, book_id, quantity).

4. **API Design**
    - User API:
        - POST /api/register: Register a new user.
        - POST /api/login: Authenticate and return a JWT.
        - GET /api/user: Get user profile.
    - Books API:
        - GET /api/books: Get a list of all books.
        - GET /api/books/:id: Get details of a specific book.
        - POST /api/books: Add a new book (Admin only).
        - PUT /api/books/:id: Update book details (Admin only).
        - DELETE /api/books/:id: Delete a book (Admin only).
    - Cart API:
        - POST /api/cart: Add a book to the cart.
        - GET /api/cart: Get the current user's cart.
        - DELETE /api/cart/:id: Remove a book from the cart.
    - Order API:
        - POST /api/orders: Place a new order.
        - GET /api/orders: Get the current user's orders.

    - GET /api/orders/:id: Get details of a specific order.

5. Frontend Design
    - Pages:
        - Home Page: Display featured books and categories.
        - Book List Page: Display the list of books with search and filter options.
        - Book Detail Page: Display detailed information about a specific book.
        - Cart Page: Display items in the user's cart.
        - Checkout Page: Handle order placement and payment.
        - Admin Dashboard: Manage books, users, and orders.
6. Implementation Steps
    1. Setup the Project:
        - Initialize a Spring Boot project for the backend.
        - Initialize a React.js or Angular project for the frontend.
    2. Implement User Authentication:
        - Setup user registration and login with JWT.
        - Protect API endpoints using JWT.
    3. Create Database Schema:
        - Design and create tables for users, books, orders, and order items.
    4. Develop Backend APIs:
        - Implement CRUD operations for books.
        - Implement cart functionality.
        - Implement order processing and payment integration.
    5. Develop Frontend Components:
        - Create components for displaying books, cart, and orders.
        - Implement forms for user registration, login, and checkout.
    6. Integrate Frontend with Backend:
        - Connect frontend components with backend APIs using HTTP requests.
    7. Testing and Deployment:
        - Write unit and integration tests for both backend and frontend.
        - Deploy the application to a cloud platform like AWS or Heroku.
7. Tools and Technologies
    - Frontend: React.js or Angular, Redux (for state management), Axios (for HTTP requests).
    - Backend: Spring Boot, Spring Security (for authentication), Hibernate (for ORM).
    - Database: MySQL or PostgreSQL.
    - Payment Gateway: Stripe or PayPal.
    - Deployment: Docker, AWS/Heroku.