# Online Book Store API

## Objective

Develop a RESTful API for an online book store using Spring Boot and PostgreSQL. The API should allow users to browse, search, and purchase books, as well as manage their user accounts and orders.

## Requirements

1. Database Design:
   - Design a PostgreSQL database schema to store information about books, authors, users, orders, and reviews.
   - The schema should include tables for books `(id, title, author, description, price, quantity)`, authors `(id, name)`, users `(id, username, email, password)`, orders `(id, user_id, order_date, total_amount)`, order_items `(id, order_id, book_id, quantity)`, and reviews `(id, user_id, book_id, rating, comment)`.
2. RESTful API Endpoints:
   - Implement the following API endpoints:
     - GET `/books`: Retrieve a paginated list of all books.
     - GET `/books/{id}`: Retrieve details of a specific book by ID.
     - GET `/books/search`: Search for books based on title, author, or description.
     - POST `/orders`: Place a new order for one or more books.
     - GET `/orders/{id}`: Retrieve details of a specific order by ID.
     - POST `/users`: Create a new user account.
     - PUT `/users/{id}`: Update user account information.
     - POST `/reviews`: Add a new review for a book.
     - GET `/books/{id}/reviews`: Retrieve all reviews for a specific book.
3. Data Validation and Error Handling:
   - Implement proper data validation for request payloads and handle errors gracefully.
   - Return appropriate HTTP status codes and error messages for invalid requests or resource constraints.
4. Authentication and Authorization:
   - Implement user authentication using JWT or Spring Security.
   - Secure the API endpoints that require authentication, such as placing orders and managing user accounts.
   - Implement role-based authorization to restrict access to certain endpoints based on user roles (e.g., admin, customer).
5. Database Transactions:
   - Ensure data consistency by using database transactions for critical operations like placing orders and updating book quantities.
6. Testing:
   - Write unit tests to cover the critical functionality of the API endpoints and service layer.
   - Implement integration tests to verify the interaction between the API, service layer, and database.
7. Documentation:

- Provide clear and concise API documentation using tools like Swagger or Spring REST Docs.
- Include examples of request and response payloads for each API endpoint.

## Evaluation Criteria

- Code quality, readability, and adherence to best practices.
- Proper use of Spring Boot features and annotations.
- Correct implementation of RESTful principles and HTTP methods.
- Efficient database design and query optimization.
- Proper handling of edge cases and error scenarios.
- Test coverage and the quality of unit and integration tests.
- API documentation clarity and completeness.