📘 BookInventory Project Documentation

Author: M S Janani Priyadharshini

Technology Stack: Spring Boot 3.5.6, Java 21, MySQL, JPA, Swagger, JWT, Maven

# 🧩 Project Overview

BookInventory is a full-stack backend application designed to manage a comprehensive book inventory system. It supports operations such as managing books, authors, publishers, categories, reviews, shopping carts, and user roles. The system is built using Spring Boot and follows a layered architecture for scalability and maintainability.

# 🛠️ Technologies Used

|  |  |
| --- | --- |
| Technology | Purpose |
| Spring Boot | Core framework for building RESTful APIs |
| Spring Data JPA | ORM for database operations |
| Spring Security + JWT | Authentication and authorization |
| Lombok | Reduces boilerplate code |
| MySQL | Relational database |
| Swagger (Springdoc OpenAPI) | API documentation |
| Maven | Build and dependency management |
| Java 21 | Latest stable Java version for enhanced performance and features |

# 📁 Project Structure

src/  
└── main/  
 └── java/  
 └── com/example/bookInventory/  
 ├── config/ # Swagger & Security Config  
 ├── controller/ # REST Controllers  
 ├── entity/ # JPA Entities  
 ├── exception/ # Custom Exceptions  
 ├── repository/ # Spring Data Repositories  
 ├── service/ # Service Interfaces & Implementations  
 └── BookInventoryApplication.java  
 └── resources/  
 └── application.properties  
test/  
└── java/com/example/bookInventory/ # Unit & Integration Tests

# 🔐 Security

- JWT-based Authentication: Secure endpoints using JSON Web Tokens.  
- Role-based Access Control: Managed via Permrole entity and Spring Security configuration.

# 📄 API Documentation

Swagger UI is enabled via springdoc-openapi-starter-webmvc-ui.  
Accessible at: http://localhost:8080/swagger-ui.html (default)

# 📚 Learning Outcomes

* Spring Boot Mastery: Deep understanding of building RESTful services.
* Security Implementation: JWT integration and role-based access.
* Database Design: Entity relationships like BookAuthor, PurchaseLog, etc.
* Exception Handling: Centralized error management using GlobalExceptionHandler.
* API Documentation: Swagger integration for interactive API exploration.
* Clean Architecture: Separation of concerns across layers.

# ✅ Improvements & Suggestions

## 🔧 Technical Enhancements

* Add Unit & Integration Tests: Use JUnit and Mockito for service and controller layers.
* Dockerize the App: Create a Dockerfile and docker-compose.yml for containerization.
* CI/CD Integration: Use GitHub Actions or Jenkins for automated builds and deployments.
* Caching: Integrate Redis for caching frequently accessed data.
* Rate Limiting: Protect APIs using tools like Bucket4j or Spring Cloud Gateway.

## 📈 Feature Enhancements

* User Registration & Login: Add endpoints for user onboarding.
* Email Notifications: Send confirmation emails for purchases.
* Admin Dashboard: Create a frontend (React/Angular) for managing inventory.

# 📦 Build & Run

Prerequisites:  
- Java 21  
- Maven  
- MySQL

Commands:  
  
mvn clean install  
mvn spring-boot:run

# 📌 Final Notes

This project showcases a robust backend system suitable for real-world applications. It demonstrates best practices in Spring Boot development, secure API design, and modular architecture. Gokul Anand Vijayakumar has built a scalable and extensible foundation for any book-related e-commerce or inventory system.