Proiect Java Spring MVP - Online Bookstore

Adrian Cernat, Dragos Ciobanu

December 2025

1 Online Bookstore — MVP Specification

1.1 Business Requirements

1. **User Accounts & Roles**
   Customers can register and log in, while administrators manage the bookstore catalog.

2. **Catalog Browsing & Search**
   Users can browse and search books using filters such as title, author, category, price, and rating.

3. **Book Information**
   Each book displays details including title, ISBN, author(s), price, stock availability, and description.

4. **Shopping Cart**
   Users can add, update, or remove items from their cart, which persists across sessions.

5. **Checkout & Order Creation**
   Users can place orders by converting their cart into an order and providing shipping details.

6. **Payment Processing**
   Payments are handled through a mock payment gateway with success and failure handling.

7. **Inventory Control**
   Book stock is reduced when orders are placed and restored if orders are canceled.

8. **Order Tracking**
   Orders progress through the following states:
   `PLACED` → `PAID` → `SHIPPED` → `DELIVERED`

9. **Reviews & Ratings**
   Verified purchasers can leave reviews and rate books.

10. **Admin Catalog Management**
    Administrators can create, update, and delete books, authors, and categories.

1.2 MVP Features

- **User Registration & Authentication**
  Secure user registration and login using JWT-based authentication.
- **Catalog Browsing & Book Details**
  Public APIs to list books, search the catalog, and view detailed book information.
- **Shopping Cart Management**
  A persistent, user-linked cart with full CRUD support for cart items.
- **Checkout & Order Management**
  Transactional checkout process integrating payment handling and inventory updates.
- **Admin Catalog Operations**
  Admin-restricted APIs for managing books, authors, and categories.

1.3 Data Model Overview

Entities:
User, Address, Book, Author, Category, Cart, CartItem, Order, OrderItem, Payment

Relationships:

- User → Order (one-to-many)
- Order → OrderItem (one-to-many)
- Book ↔ Author (many-to-many)
- Book ↔ Category (many-to-many)
- User → Cart (one-to-one)

1.4 Cross-Cutting Concerns

- Centralized validation and exception handling using `@ControllerAdvice`
- Transaction management for checkout operations using `@Transactional`
- Aspect-Oriented Programming (AOP) for logging service execution times
- API documentation via Swagger/OpenAPI

1.5 Sample API Endpoints

- **Authentication**
  - `POST /api/v1/auth/register`
  - `POST /api/v1/auth/login`
- **Books & Catalog**
  - `GET /api/v1/books`
  - `GET /api/v1/books/{id}`
- **Shopping Cart**
  - `GET /api/v1/cart`
  - `POST /api/v1/cart/items`
- **Orders**
  - `POST /api/v1/orders/checkout`
  - `GET /api/v1/orders`
- **Admin Management**
  - `POST | PUT | DELETE /api/v1/admin/books/*`