Project: Digital Bookstore Management System

1. Introduction

This Low-Level Design (LLD) document describes the architecture, components, and functionalities of a **Digital Bookstore Management System**, which allows users to browse, purchase, and manage books online.

It supports both **Spring Boot (Java)** and **ASP.NET Core (.NET)** for backend development.

2. Module Overview

2.1 Book Catalog Management Module

- Manage books, authors, categories, and pricing.
- Allow bookstore admins to add, update, and remove books.

2.2 User Management Module

- Handles user registration, authentication, and profile management.
- Supports customer and admin roles.

2.3 Order Management Module

- Enables users to place, track, and manage orders.
- Includes payment integration for purchases.

2.4 Inventory Management Module

- Tracks available book stock and updates on purchase.
- Notifies admins of low stock levels.

2.5 Review & Rating Module

- Allows customers to submit book reviews and ratings.
- Helps improve customer engagement.

3. Architecture Overview

3.1 Architectural Style

- **Frontend:** Angular or React.

- **Backend:** REST API-based architecture.
- **Database:** Relational Database (MySQL/PostgreSQL/SQL Server).

3.2 Component Interaction

- Frontend interacts with the backend through REST APIs for all functionalities.
- Backend connects to the relational database for data persistence and retrieval.

4. Module-Wise Design

4.1 Book Catalog Management Module

4.1.1 Features

- View, search, and filter books by category or author.
- Admins can manage book listings.

4.1.2 Entities

- **Book**
    - BookID
    - Title
    - AuthorID
    - CategoryID
    - Price
    - StockQuantity

4.2 User Management Module

4.2.1 Features

- User registration, login, and profile management.
- Role-based access (Admin/Customer).

4.2.2 Entities

- **User**
    - UserID
    - Name
    - Email

    - Password
    - Role

4.3 Order Management Module

4.3.1 Features

- Users can add books to the cart and place orders.
- Order status tracking (Pending, Shipped, Delivered).

4.3.2 Entities

- **Order**
    - OrderID
    - UserID
    - OrderDate
    - TotalAmount
    - Status

4.4 Inventory Management Module

4.4.1 Features

- Tracks stock levels and prevents out-of-stock purchases.
- Sends alerts when stock is low.

4.4.2 Entities

- **Inventory**
    - InventoryID
    - BookID
    - Quantity

4.5 Review & Rating Module

4.5.1 Features

- Customers can leave reviews and rate books.
- Admins can moderate reviews.

4.5.2 Entities

- **Review**
    - ReviewID

    - UserID
    - BookID
    - Rating
    - Comment

5. Deployment Strategy

5.1 Local Deployment

- Frontend and backend deployed on developer machines for initial testing.

5.2 Testing Environments

- Use containerized setups for staging environments to ensure consistency with deployment.

6. Database Design

6.1 Tables and Relationships

- **Book**: Primary Key: BookID, Foreign Key: AuthorID, CategoryID.
- **User**: Primary Key: UserID.
- **Order**: Primary Key: OrderID, Foreign Key: UserID.
- **Inventory**: Primary Key: InventoryID, Foreign Key: BookID.

7. User Interface Design

7.1 Wireframes

- **Homepage**: Displays featured and recommended books.
- **Book Details Page**: Shows book information, price, and reviews.
- **Cart Page**: Allows users to review and proceed with their purchase.

8. Non-Functional Requirements

8.1 Performance

- Able to handle 500 concurrent users browsing books.

8.2 Usability

- Designed for ease of use with a clean UI.

8.3 Security

- Secure login and encrypted transactions.

8.4 Scalability

- Supports adding more books, users, and orders without performance degradation.

9. Assumptions and Constraints

9.1 Assumptions

- Users will have internet access to browse and purchase books.

9.2 Constraints

- Initially focused on a single bookstore, with multi-store support in future iterations.