Book Store System Assignment

Documented by Harsh Mange

Table of Contents

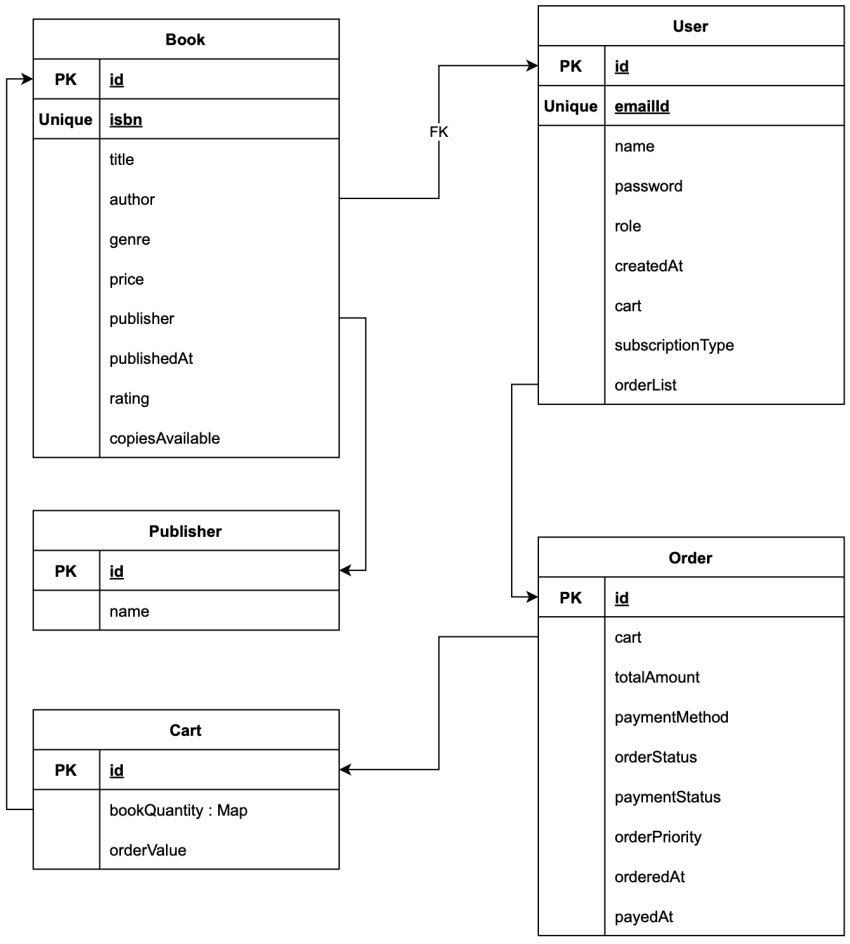
Introduction

Book Store app aims to provide basic functionality for managing books to placing an order. It is developed as an assignment, part of the recruitment process at HYCU.

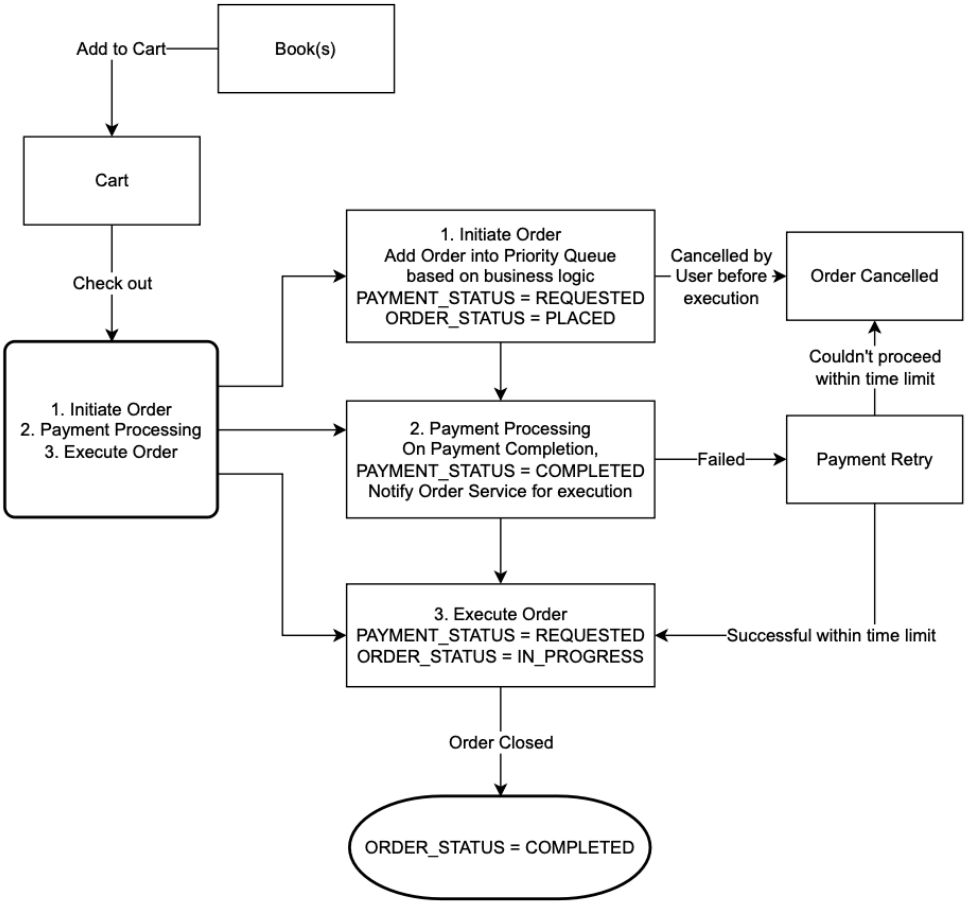
Entity Diagram

Entities

- User
- Book
- Publisher
- Cart
- Order



Flow (Core)


Add to Cart
Book(s)
Cart
Check out
1. Initiate Order
2. Payment Processing
3. Execute Order
1. Initiate Order
Add Order into Priority Queue
based on business logic
PAYMENT_STATUS = REQUESTED
ORDER_STATUS = PLACED
Cancelled by
User before
execution
Order Cancelled
Couldn't proceed
within time limit
2. Payment Processing
On Payment Completion,
PAYMENT_STATUS = COMPLETED
Notify Order Service for execution
Failed
Payment Retry
3. Execute Order
PAYMENT_STATUS = REQUESTED
ORDER_STATUS = IN_PROGRESS
Successful within time limit
Order Closed
ORDER_STATUS = COMPLETED

Architecture & Design Patterns (OOP)

- Implemented OOP concepts to keep things simple & extendable.
- Interface based implementation for services - extendability, base for multiple impl
- Singleton pattern to avoid service object duplicate creation - better memory mgmt
- Strategy design pattern - single feature can have multiple impl (i.e. price, rate based fetching)
- Field level validations in Model - avoid common input errors from user, improved security
- DAO to store data in-memory
- Exceptions for error handling
- Utils for validations & common functionality
- Enums for data consistency & extendability
- Priority queue to manage received orders, orders won't be executed directly. Priority given based on the business rule - i.e. user subscription, time stamp at which order was placed
- Logs - Used Slf4j
- Tests - JUnit

Scalability, Limitations & Solutions

Multithreaded Env - Order Execution

While adding books to cart, placing an order, the book quantity needs to be updated. In the case of a large user scale & multithreaded environment, transactions should be accurate. Locks, Synchronization, Concurrent Hashmap can solve this issue.

Inter Service Communication/MQ - Payment Processing

On order initiation, order service should wait for payment completion before the order execution. In this case Payment Service should notify Order Service about the payment status through message queues

Security - User exposed action points/APIs

Basic security practice like User input sanitization, data validations, user auth, access control, pagination, use of DTO