A1: Your Books Everywhere!

Analysis and Design Document

Student: Adrian Moldovan

**Group:** 30238

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 4

3. System Architectural Design 4

4. UML Sequence Diagrams 5

5. Class Design 5

6. Data Model 6

7. System Testing 8

8. Bibliography 8

1. Requirements Analysis

# Assignment Specification

Book management service:

A user should be able to create an account, choose a payment plan and login to search the book library.

Payments can be done via a cash only policy and need to be validated by library staff.

The library is managed by staff and can be filtered by release date, author, title, genre.

If a book is available a user can add it to your library. If not the user can join a waiting list. Once a book has been read by a user it can be returned via the online library return function. This assigns the book to the next user in the waiting list after validation of the return by library staff.

The service also provides users with dynamic recommendations based on latest trends (popular borrowed books) or user defined interests by genre or topic.

# Functional Requirements

-User Registration

-Payment system

-Library management

-Dynamic recommendations

-Store data

-Input data has to be validated

# Non-functional Requirements

-Payments:cash only polcy

-Waiting list for unavailable book

-Library is managed by staff.

-Transactions validated by staff

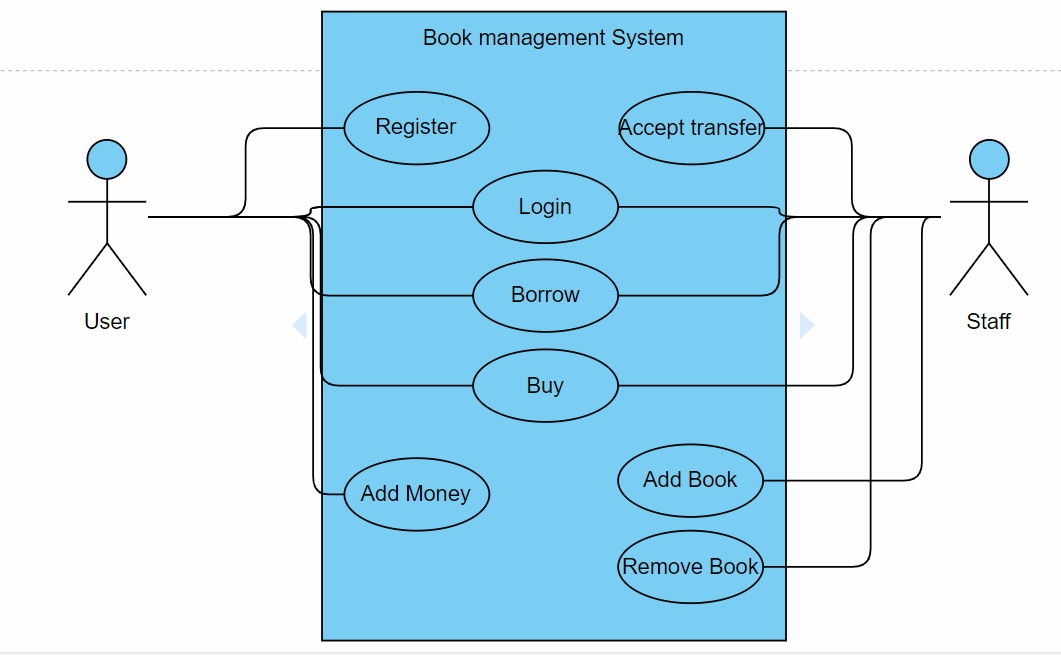
-Recommendations based on latest trends or user defined Interests by genre or topic

-Layered Architecture

-Factory method for building user recommendations

-Use Database for storage

2. Use-Case Model



3. System Architectural Design

**3.1 Architectural Pattern Description**

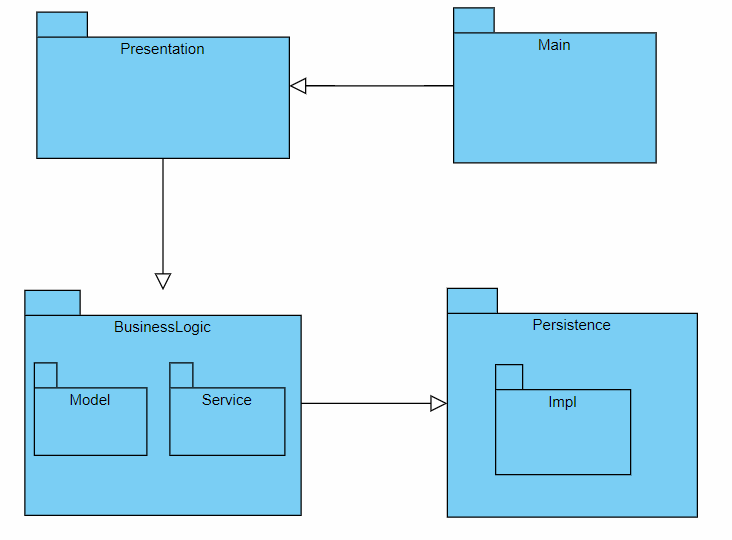
**Layered Architecture pattern:**

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Although the layered architecture pattern does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of four standard layers: presentation, business, persistence, and database

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request. For example, the presentation layer doesn’t need to know or worry about how to get customer data; it only needs to display that information on a screen in particular format. Similarly, the business layer doesn’t need to be concerned about how to format customer data for display on a screen or even where the customer data is coming from; it only needs to get the data from the persistence layer, perform business logic against the data (e.g., calculate values or aggregate data), and pass that information up to the presentation layer.

**3.2 Diagrams**

**Package Diagram:**

****

4. UML Sequence Diagrams

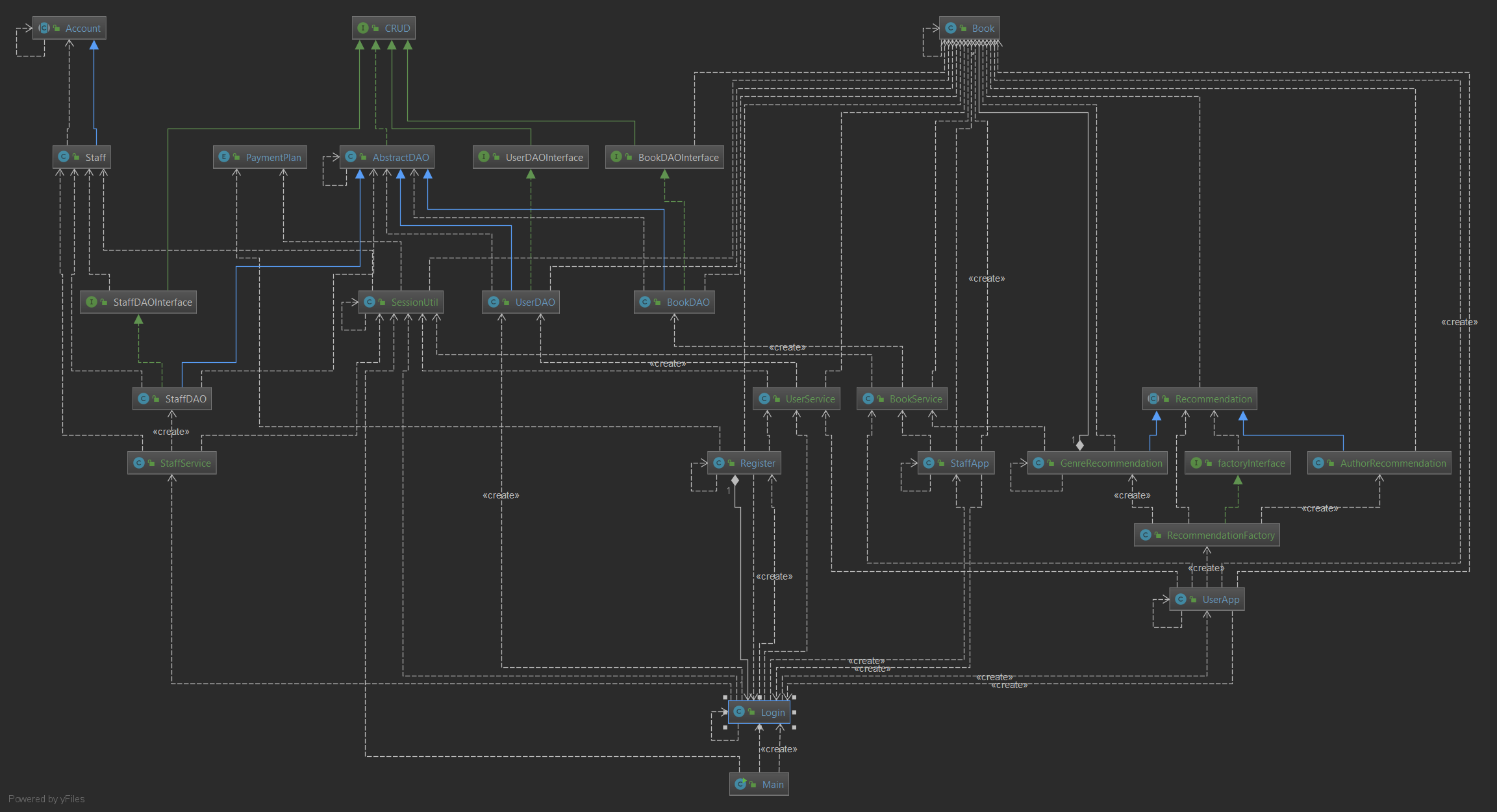
5. Class Design

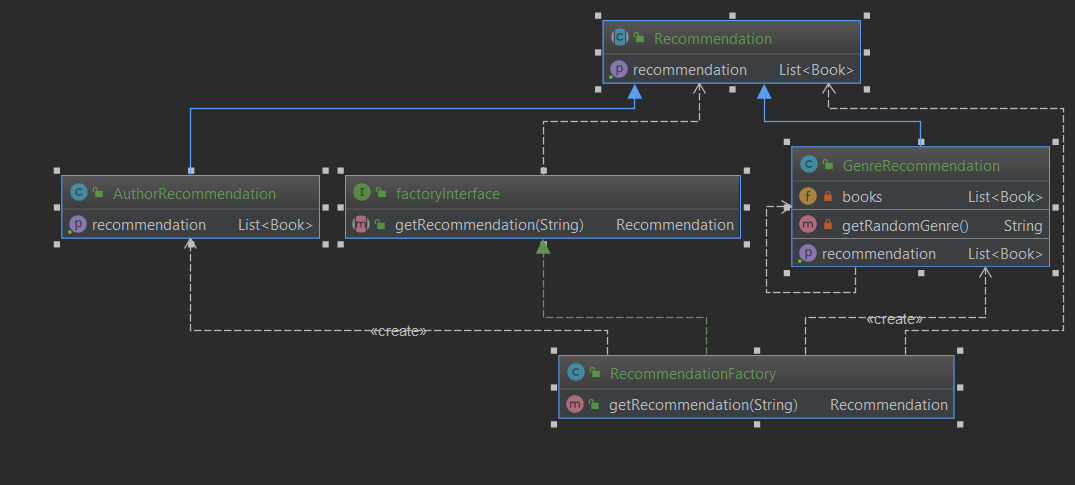
**5.1 Design Patterns Description**

**Factory method pattern:**

The **factory method pattern** is a [creational](https://en.wikipedia.org/wiki/Creational_pattern) pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created. This is done by creating objects by calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes rather than by calling a constructor.

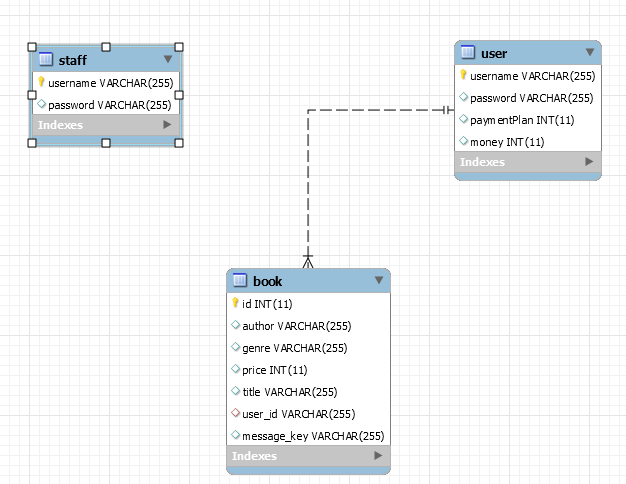
**5.2 UML Class Diagram**

**





6. Data Model



7. System Testing

8. Bibliography

<https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>

<https://www.youtube.com/watch?v=y2DD8tAjM0E>

<https://www.baeldung.com/simplifying-the-data-access-layer-with-spring-and-java-generics>

<https://en.wikipedia.org/wiki/Factory_method_pattern>