Your Books Everywhere!

Student: Ciuca Daniel Claudiu

**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 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The task is to build a book management service. The service helps users to borrow or return a book without having to go to a library. It’s very useful for the people that don’t have time to go to a library or for example if it’s raining out.

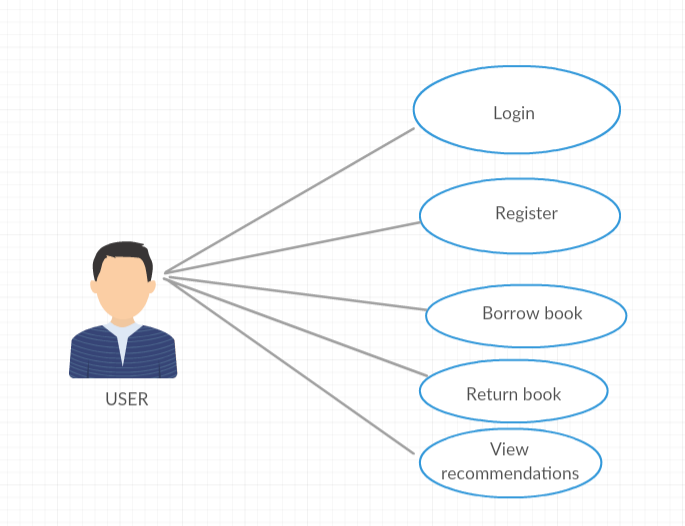
# Functional Requirements

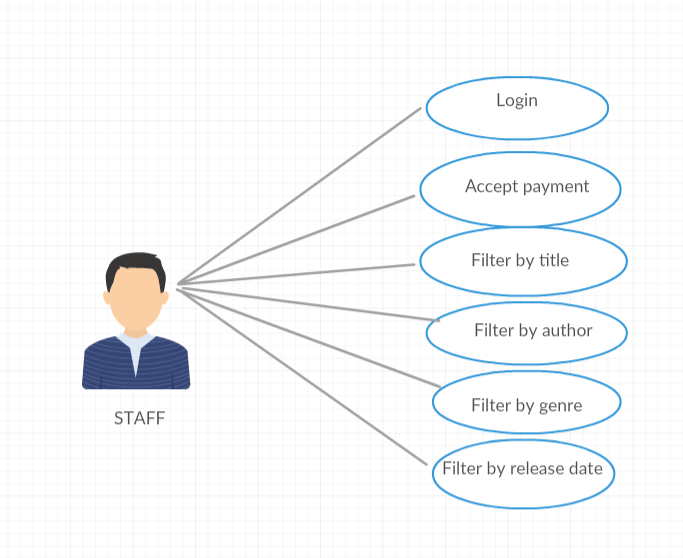
* Account creation: a user can create an account and have to choose username, password and payment method (Day, Week, Month, Year).
* Staff management: the staff can to accept a user (validate payment), filter books by title, author, genre and release date.
* Borrow a book: if a book is available a user can add it to your library. If not the user can join a waiting list.
* Return a book: 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.
* Recommendation service: The app provides recommendations to user by trend, user interest by genre.

# Non-functional Requirements

* Implement the application and test it
* Use an OOP language (Java)
* Use layered architecture
* Use a factory method for building user recommendations by genres.

2. Use-Case Model





Use case: Borrow a book

Level: user-goal level

Primary actor: User

Main success scenario:

* User clicks Register for the menu
* User introduces username, password and payment plan
* Staff accept user
* User click Back to login page
* User introduces username and password
* User clicks Login
* User introduces book’s title
* User clicks Borrow

Extensions:

* Staff don’t accept user payment plan

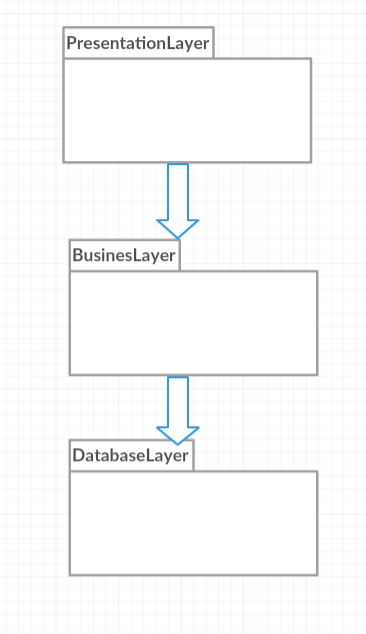
3. System Architectural Design

**3.1 Architectural Pattern Description**

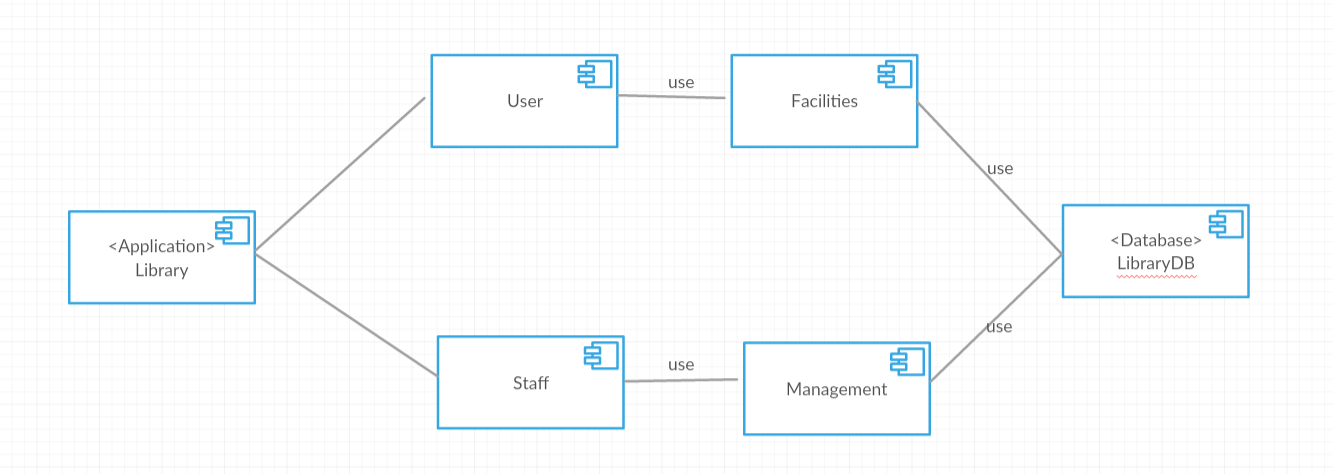
The layered architectural pattern can be used to structure programs that can be decomposed into groups of subtasks. Each layer provides services to the next higher level*.*



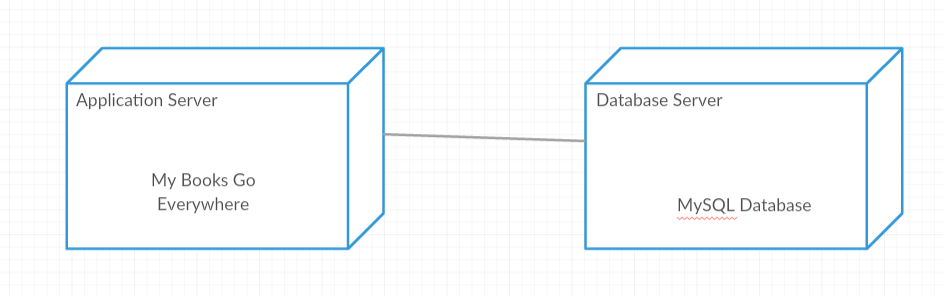
**3.2 Diagrams**



Package Diagram

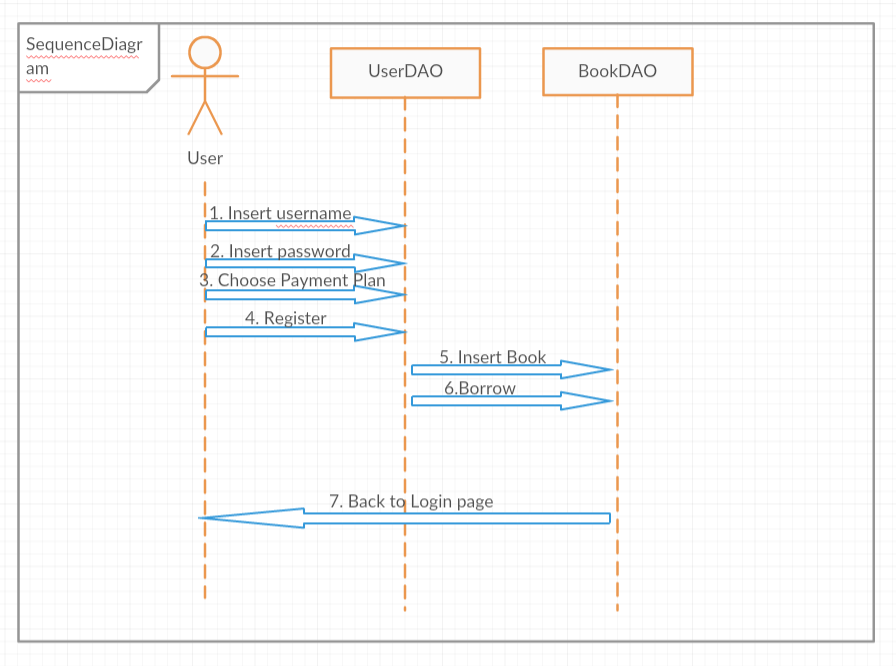


Component diagram



Deployment diagram

4. UML Sequence Diagrams



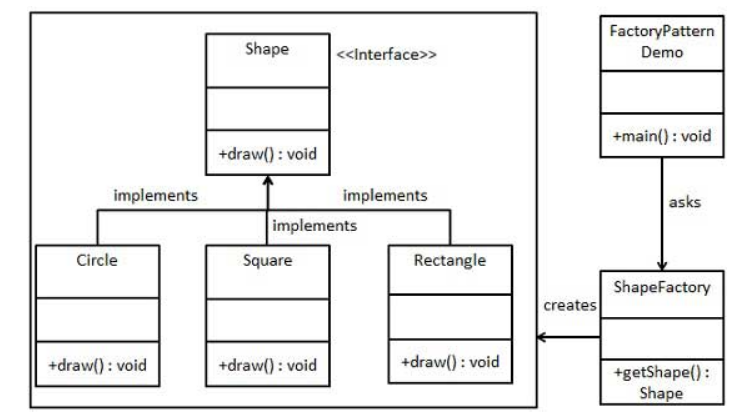
5. Class Design

**5.1 Design Patterns Description**

FactoryPattern: This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

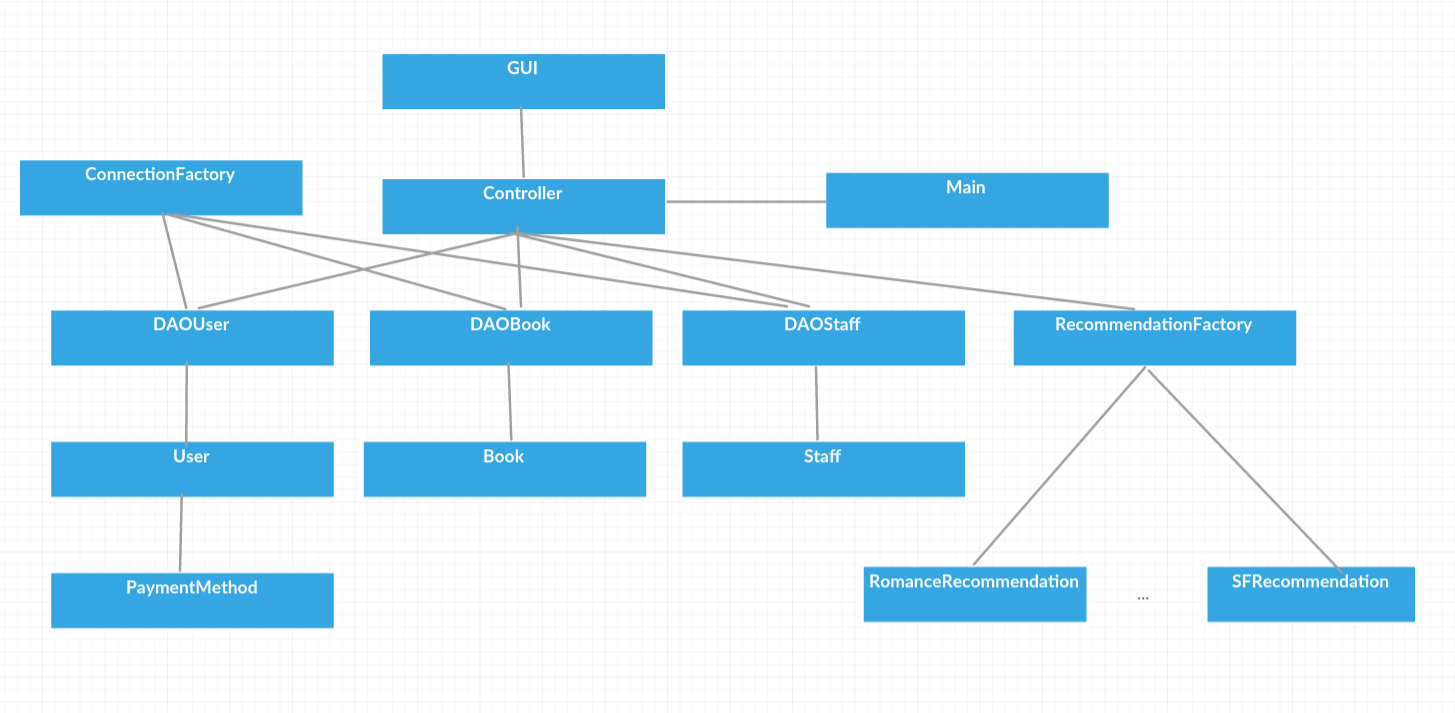
In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

Example:



Layered architecture is represented by the split of the application in layers: database -> service -> businessLogic -> presentation, and each one uses method only from one under it.

**5.2 UML Class Diagram**

6. Data Model

Is represented by entity subpackage of databaseLayer package, it consists of classes User, Staff, Book and PaymentMethod where each attribute of class corresponds with a column in the table.

7. System Testing

To test my application, I create a JUnit class that do 3 tests. In first, it verifies the registration of a user. In second, it verifies if the number of borrowed books is bigger after a user borrow a book. And in the last test, it verifies if the number of borrowed books is smaller after a user return a book. For this class, I needed to import JUnit library.

8. Bibliography

FactoryPattern:

<https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>

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

LayeredArchitecture:

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

Sequence diagram:

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