Your Books Everywhere!

Student: Alexandra Georgeana Czako

**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

You are tasked to build a 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

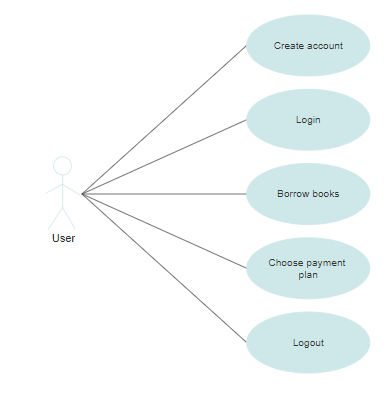
* Create account for user
* Choose payment plan for user
* Filter all the books by: author, title, genre; Show all the books available in the library
* Validate payments by staff

# Non-functional Requirements

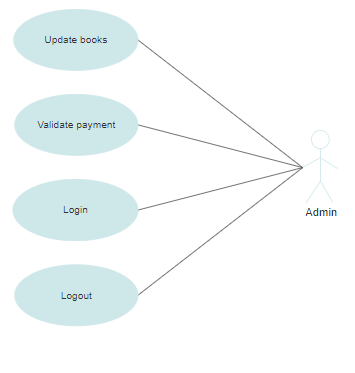
* Secured password
* Junit tests

2. Use-Case Model

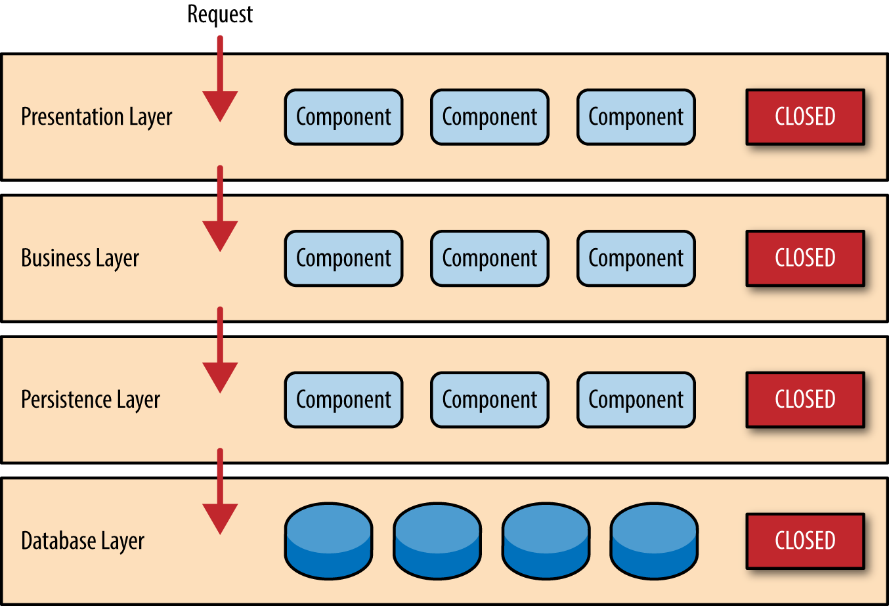
User’s use case model:



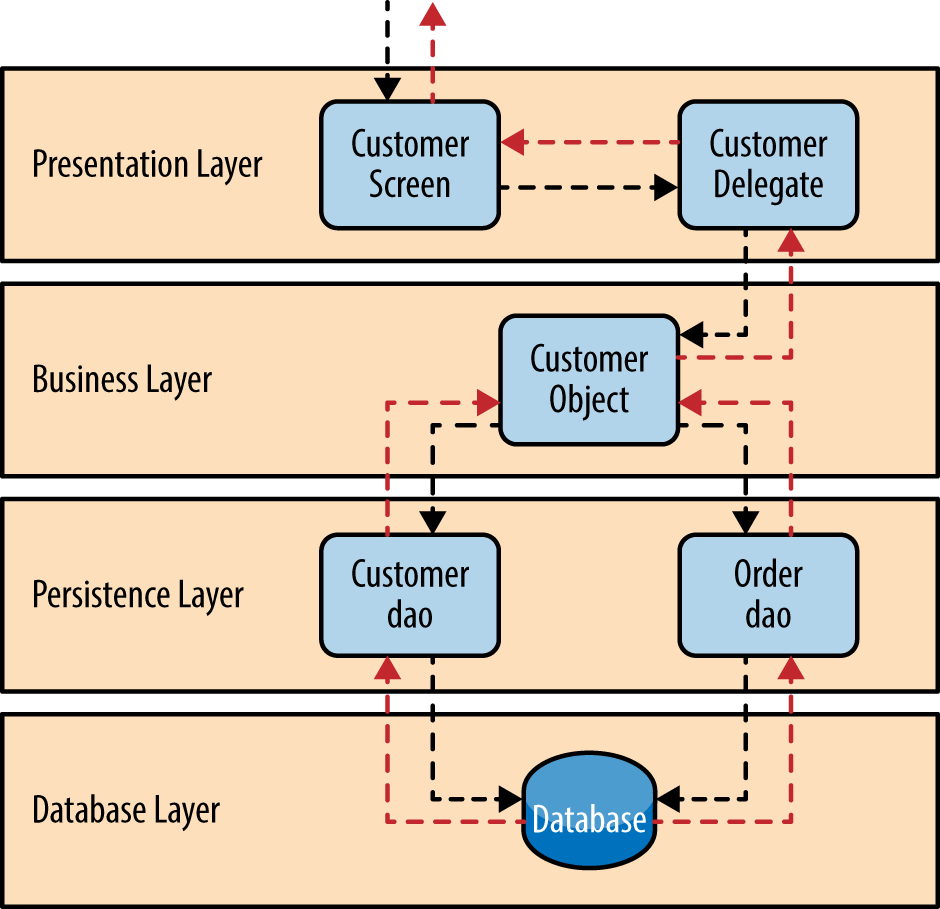
Admin’s use case model:



3. System Architectural Design

**3.1 Architectural Pattern Description**

The most common architecture pattern is the layered architecture pattern. The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application

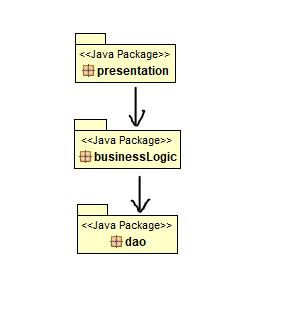
Each layer of the layered architecture pattern has a specific role and responsibility within the application:

* presentation layer would be responsible for handling all user interface and browser communication logic
* business layer would be responsible for executing specific business rules associated with the request
* persistence layer(data access layer) would be responsible for the data access fuctions
* database layer repesents the database

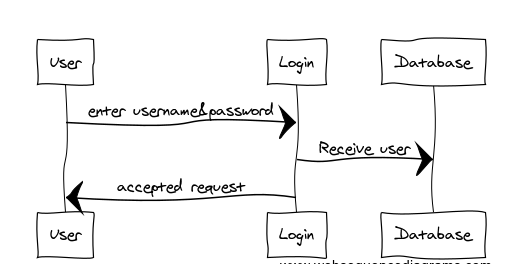
So why not allow the presentation layer direct access to either the persistence layer or database layer? After all, direct database access from the presentation layer is much faster than going through a bunch of unnecessary layers just to retrieve or save database information. The answer to this question lies in a key concept known as layers of isolation*.*

**3.2 Diagrams**

**Package diagram**



4. UML Sequence Diagrams

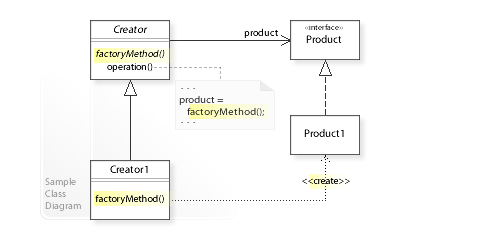


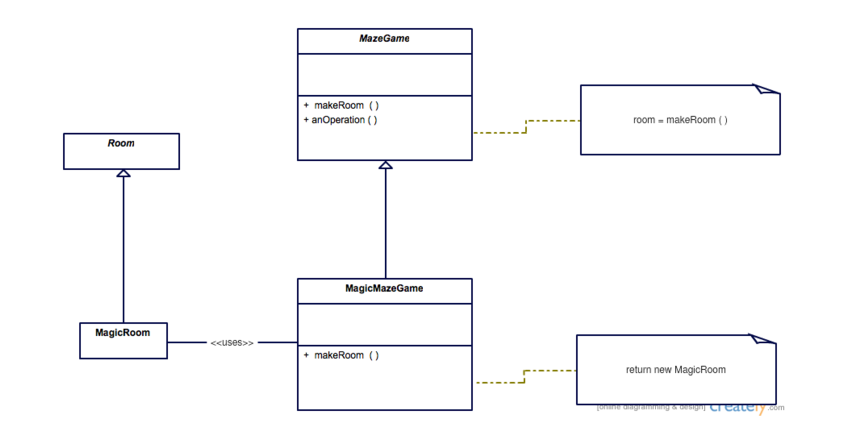
5. Class Design

**5.1 Design Patterns Description**

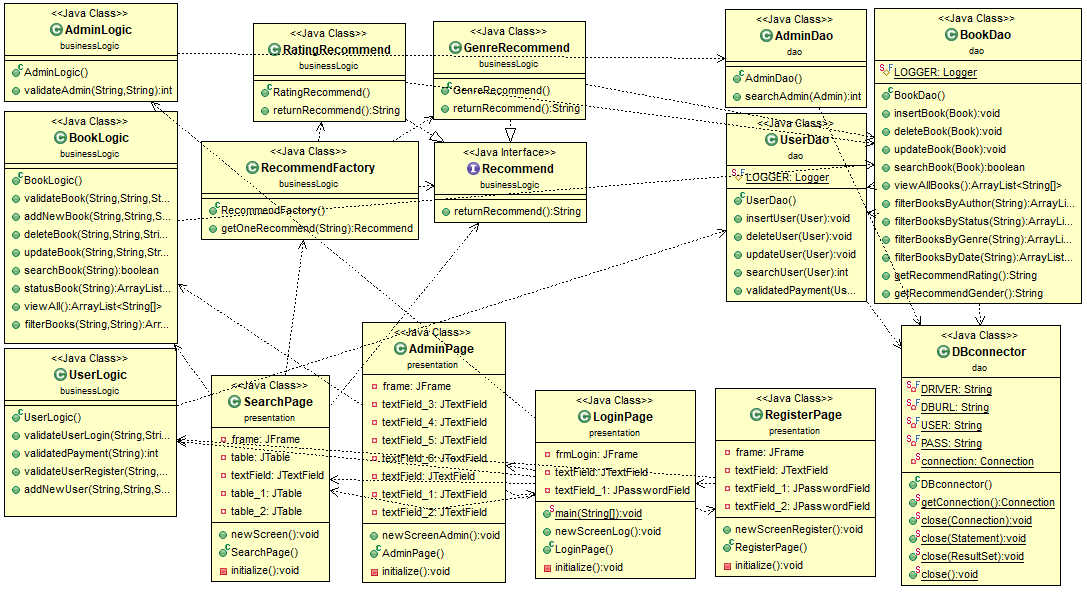
Factory method is a [creational design pattern](https://www.geeksforgeeks.org/design-patterns-set-1-introduction/), i.e., related to object creation. In Factory pattern, we create object without exposing the creation logic to client and the client use the same common interface to create new type of object.  
The idea is to use a static member-function (static factory method) which creates & returns instances, hiding the details of class modules from user.

Examples:





**5.2 UML Class Diagram**



6. Data Model

*[Present the data models used in the system’s implementation.]*

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography

* <https://www.sciencedirect.com/topics/computer-science/business-logic-layer>
* <https://objcsharp.wordpress.com/2013/07/22/what-is-a-business-logic-layer-anyway/>
* <https://www.tutorialspoint.com/design_pattern/factory_pattern.htm>