Your Books Everywhere!

Analysis and Design Document

Student: Bianca Elena Dondas

**Group: 30238**

Contents

[1. Requirements Analysis 3](#_Toc4878772)

[1.1 Assignment Specification 3](#_Toc4878773)

[1.2 Functional Requirements 3](#_Toc4878774)

[1.3 Non-functional Requirement 3](#_Toc4878775)

[2. Use-Case Model 3](#_Toc4878776)

[3. System Architectural Design 4](#_Toc4878777)

[3.1 Architectural Pattern Description 4](#_Toc4878778)

[3.2 Diagrams 5](#_Toc4878779)

[4. UML Sequence Diagrams 5](#_Toc4878780)

[5. Class Design 6](#_Toc4878781)

[5.1 Design Patterns Description 6](#_Toc4878782)

[5.2 UML Class Diagram 7](#_Toc4878783)

[6. Data Model 8](#_Toc4878784)

[7. System Testing 8](#_Toc4878785)

[8. Bibliography 8](#_Toc4878786)

# 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.

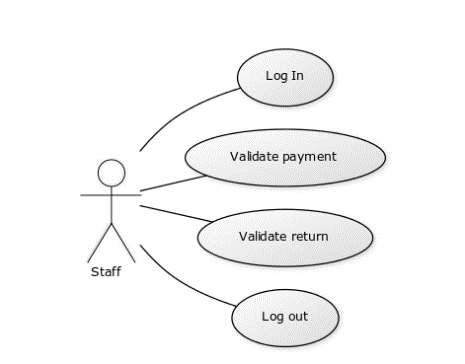
## Functional Requirements

* Create account
* Choose payment plan
* Filter by: author, title, genre
* Borrow & return service
* Validate payments and validate return book by staff

## Non-functional Requirement

* Password security: secured with md5 function

# 2. Use-Case Model

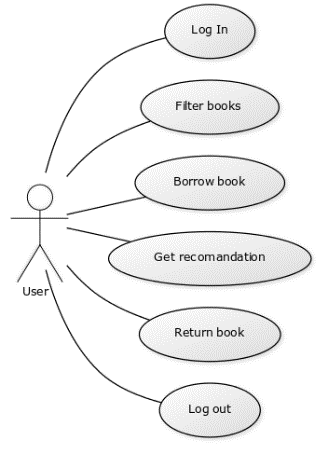


Use case: Validate return

Level: User- goal level

Primary actor: Staff

Main success scenario: The librarian connects to the app using the log in page. Then he check the return requests. He remove the person from return request table and the assign the book to the next person from the reading list.



Use case: Borrow book and return

Level: User- goal level

Primary actor: User

Main success scenario: The user connects to the app using the log in page. Then he can choose a book from the list. When he decides what he wants to read, he adds the book to his reading list. Once he finish the book he request the return of the book, which will be validated by staff.

# 3. System Architectural Design

## 3.1 Architectural Pattern Description

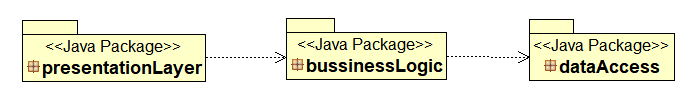
**Business logic** is the part of the program that encodes the real-world business rules that determine how data can be created, stored, and changed. It is contrasted with the remainder of the software that might be concerned with lower-level details of managing a database or displaying the user interface, system infrastructure, or generally connecting various parts of the program.

Most commonly this is accomplished using 3 layers:

* **User Interface Layer (UI)**: Where all presentation and user interaction takes place. Displays and receives data to and from the user.
* **Business Logic Layer (BLL)**: Application processing. Coordinates data between the UI and DAL.
* **Data Access Layer (DAL)**: Where data management occurs. Typically using a database or web service*.*

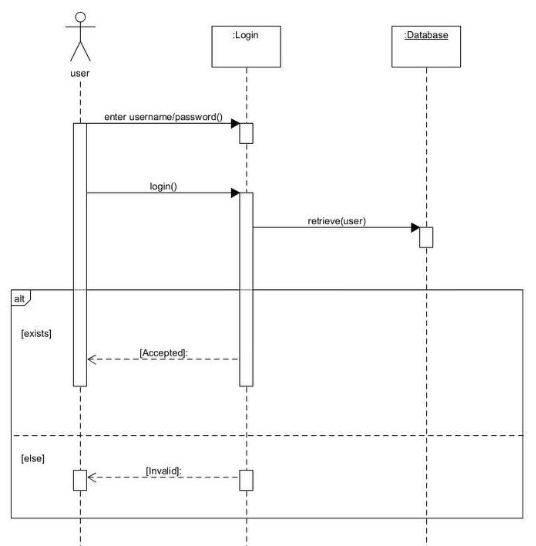
## 3.2 Diagrams

Package diagram:



# 4. UML Sequence Diagrams

Below is presented the sequence digram for log in action



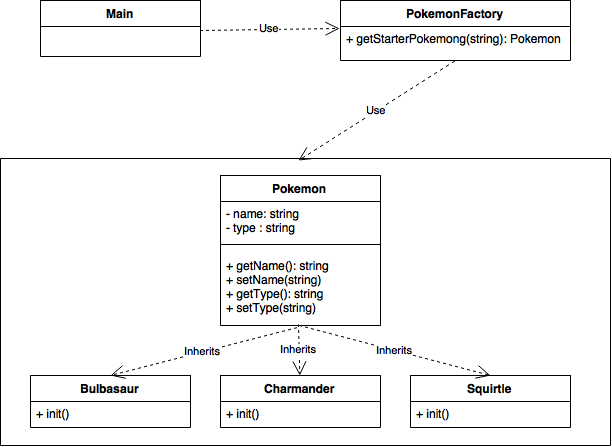
# 

# 5. Class Design

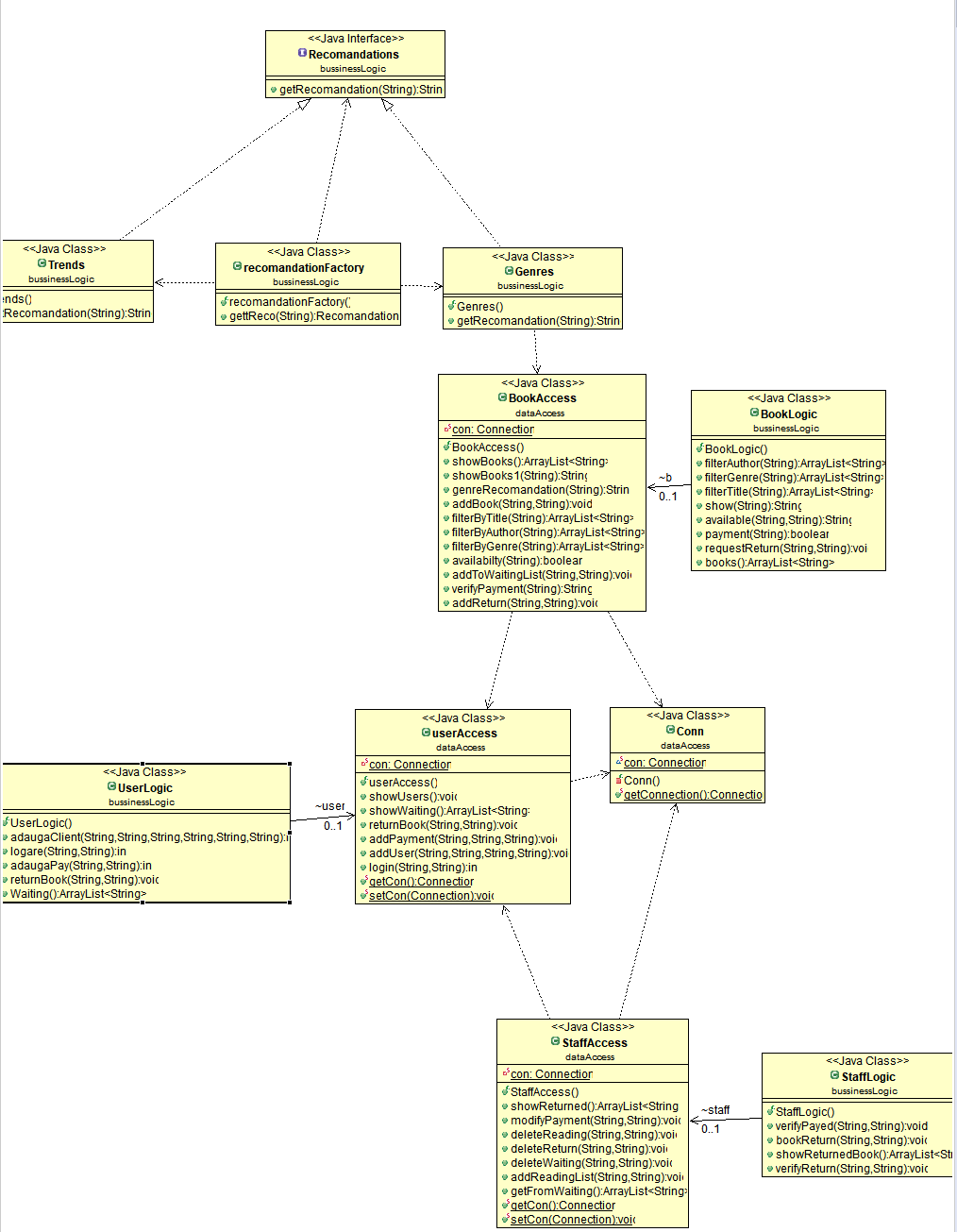
## 5.1 Design Patterns Description

Factory pattern:

Factory pattern is one of the most used design patterns in Java. 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.



## 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>