Book Store

Analysis and Design Document

Student: Emiliana Popa

**Group: 30233**

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

Use Java/C# API to design and implement an application for the employees of a book store. The application should have two types of users (a regular user represented by the book store employee and an administrator user) which have to provide a username and a password in order to use the application.

# Functional Requirements

The regular user can perform the following operations:

* Search books by genre, title, author.
* Sell books.

The administrator can perform the following operations:

* CRUD on books (book information: title, author, genre, quantity, and price).
* CRUD on regular users’ information.
* Generate two types of reports files, one in pdf format and one in csv format, with the books out of stock.

Constraints:

* The information about users, books and selling will be stored in multiple XML files. Use the Model View Controller in designing the application. Use the Factory Method design pattern for generating the reports.
* All the inputs of the application will be validated against invalid data before submitting the data and saving it.

# Non-functional Requirements

The application is portable, being written in Java. It is easy to be maintained. The security could be improved by encrypting the user’s password.

2. Use-Case Model

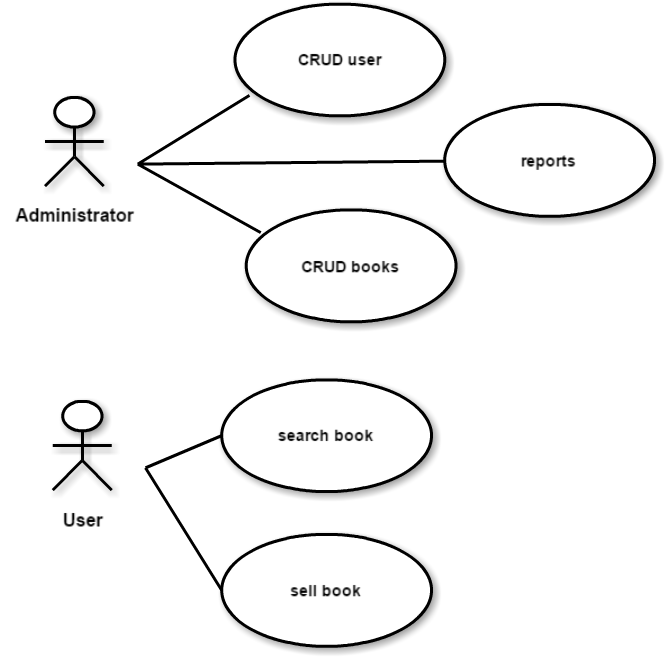
*Use case:* log in

*Level:* user-goal level

*Primary actor:* Regular user

*Main success scenario:* provide username and password and press the log in button in which case the actor will see a menu of the rest of actions he can perform

*Extensions:* the actor doesn’t have an account so he has to ask the administrator to create one



3. System Architectural Design

**3.1 Architectural Pattern Description**

Model View Controller or MVC as it is popularly called, is a software design pattern for developing web applications. A Model View Controller pattern is made up of the following three parts:

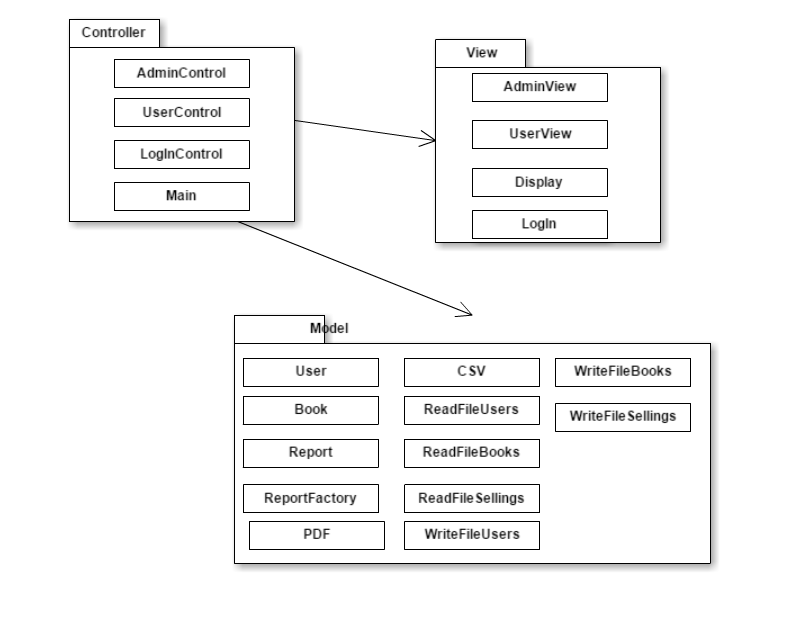
Model - The lowest level of the pattern which is responsible for maintaining data.

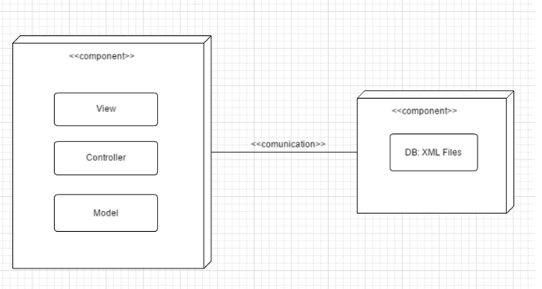
View - This is responsible for displaying all or a portion of the data to the user.

Controller - Software Code that controls the interactions between the Model and View.

MVC is popular as it isolates the application logic from the user interface layer and supports separation of concerns. Here the Controller receives all requests for the application and then works with the Model to prepare any data needed by the View. The View then uses the data prepared by the Controller to generate a final presentable response. The MVC abstraction can be graphically represented as follows.

**3.2 Diagrams**





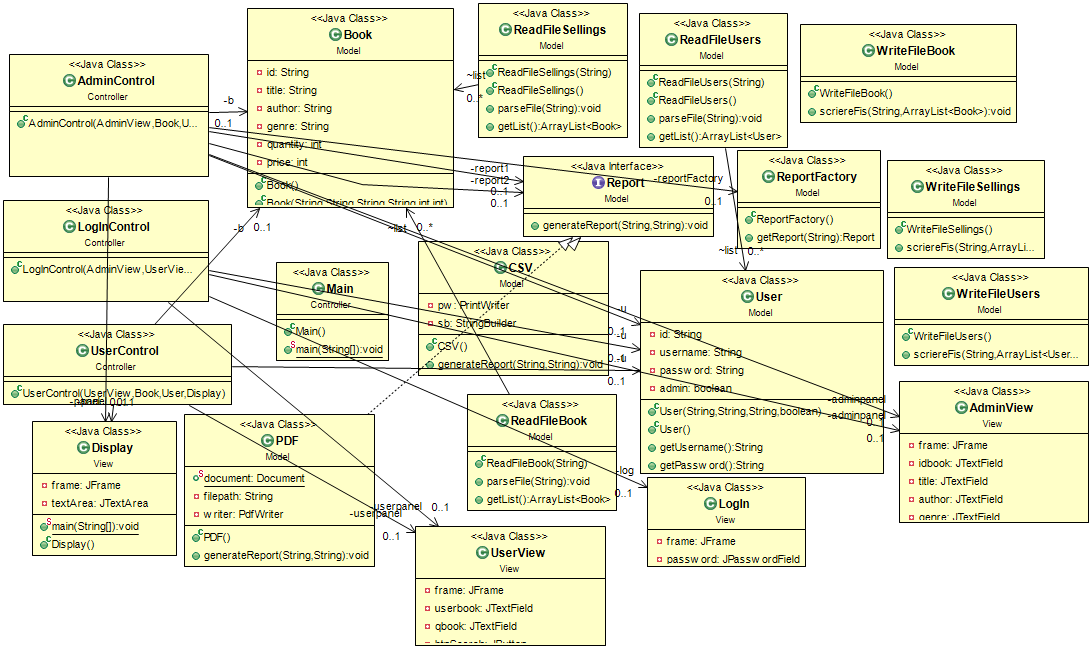
4. Class Design

**4.1 Design Patterns Description**

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.

**4.2 UML Class Diagram**



5. Data Model

I used three XML files for storing data about books, users and sellings and I used DOM to parse them. An XML file is based on a tree structure.

6. System Testing

I tested each method manually.

7. Bibliography

Martin Fowler et. al, Patterns of Enterprise Application Architecture