Book Store Application

Student: Mureșan Marius-Sebastian

**Group: CSC I**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 4

1.3 Non-functional Requirements 4

2. Use-Case Model 5

3. System Architectural Design 5

4. UML Sequence Diagrams 7

5. Class Design 8

6. Data Model 10

7. System Testing 10

8. Bibliography 11

1. Requirements Analysis

# Assignment Specification

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.

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). –

The application should generate two types of report files, one in pdf format and one in csv format, with the books out of stock.

**1. Objective**

The main objective of this application is to have a user friendly environment that can help the employee to search the books in the database and the administrator to perform the CRUD easily.

**2. Application Description**

The application is designed with a login frame, where the user/employee or the admin has to login and after that judging by the status of the login another frame is opened, a user frame or an admin frame.

The admin frame is designed with a table that show all the information and 4 buttons:

* View all – which shows all the registered books
* Create – which adds a book to the Database
* Edit – edits a specific book determined by the Title and the Author
* Delete – deletes a specific book determined by the Title and the Author

The user/employee frame is designed with a table where the desired books are displayed. This frame also has a search textbox + a search button, which searches the database based on the check boxes on the right (title, author, genre).

**3. Application Constraints**

The application is based on a database which means that the only way the application runs is if the database created for this application is running in the background with the same credentials As when the application was developed and/or the database exists in a local server and the computer where the application runs. (the data base is provided, all that a user has to do is to import it)

**4. Requirements**

All the requirements for this application are: download all the dependencies/references for the project and install the database.

**5. Deliverables**

The application takes the input that the user gives , it processes this information and it tries to return the desired output by the user.

The application takes the input of the admin, it processes the information given and in doing so the application should update database instantly.

# Functional Requirements

The login table is divided in 2:

* Basic User – Cannot change the information of a book, can only sell books and search them
* Admin User – Can change information on books

When an employee sells a big quantity of a book and that book is out of stock, the program automatically creates a pdf file with the title and the author of that book in a folder called *Out of stock.* In the same time, the out of stock book is saved in a .csv file.

When an admin updated/edits the book that is out of stock and provides a new quantity for that book, the pdf file from *Out of stock* folder is deleted, and it is also removed from the .csv file.

# Non-functional Requirements

The whole system, being based on MVC design pattern it is structured in 3 packages:

* Model
* View
* Controller

Using the observer pattern too, the model, every time it is updated, it notifies the Observers (User and Admin view). When the observers are notified they update the Data Table, so that no change can take without somebody noticing.

2. Use-Case Model

A picture containing text, map

Description automatically generated

Fig 1. Use-Case Model

3. System Architectural Design

**3.1 Architectural Pattern Description**

this application uses the MVC architectural design. It is structured in 3 packages:

* Model
  + this package contains all the information that a book can have, and it has all the methods that affect a book (can gather the book info, update, edit and delete a book from the database)
* View
  + this package communicate with the Model as well as with the Controller Package.
  + This package contains the User Interface and it is only designed to show information to the user
  + It can tell the controller that the user wants to see something (ask information) and it can ask the Model to tell it’s status, so that the user can see the information live.
* Controller
  + This package contains the brains of the application. It is the one that does all the tasks judges if the user is a regular one or an admin and it procures the privileges one has.

**3.2 Diagrams**

To pattern used in this application is MVC architectural pattern as stated before.

This architectural pattern is divided into 3 packages.

The Model package has 2 classes:

* Book
* Login

These classes contain the information needed so that the application can run and perform the tasks it should.

The Book has as arguments: book\_id, title, author, genre, price and quantity.

The Login has as arguments: login\_id, username, password and status.

The Book has the methods that can get the information about a specific book from the database, tells if the book exists in the DB and update the info about the book in the Db.

The Controller Package has access to the model and the controller, it tells the view what to show the user and it tells the model what information it needs. The Controller provides two types of privileges (the user and the admin). It allows the user only to search and sell. The only modification it does to the db is that it changes the quantity.

The admin has the CRUD privileges, which are the main actions that can affect the db.

The View Package contains the User Interface classes:

* adminGUI
* userGUI
* loginGUI

The view can ask for information or for an action to take place. These actions are performed by the controller, but it can also ask directly the model to show it’s status.

After everything is in place, all 3 packages are codependent. For the Model to be able to edit something to the database, it needs permission from the controller. In order to give information, it needs to be asked to do so. For the view package to be able to give access to the user to make changes, it needs to ask permission from the controller. The only time it goes directly to the model is when it only need to read information (to update the table)

4. UML Sequence Diagrams

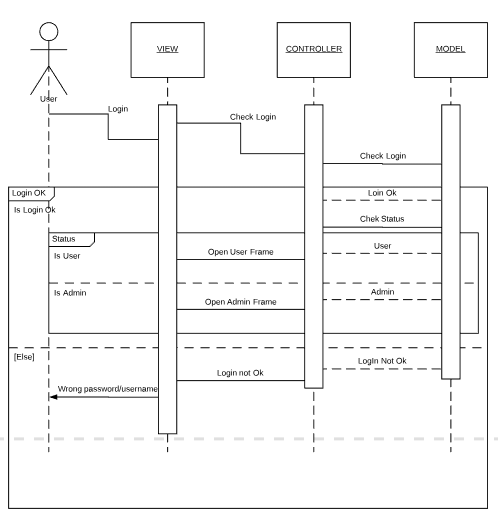


Fig 2. Login Sequence UML Diagram

5. Class Design

**5.1 Design Patterns Description**

**Structural Patterns**

These design patterns concern class and object composition. Concept of inheritance is used to compose interfaces and define ways to compose objects to obtain new functionalities.

**State Pattern**

In State pattern a class behavior changes based on its state. This type of design pattern comes under behavior pattern.

In State pattern, we create objects which represent various states and a context object whose behavior varies as its state object changes.

**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.

**MVC Pattern**

MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

View - View represents the visualization of the data that model contains.

Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.1

**5.2 UML Class Diagram**

A picture containing screenshot

Description automatically generated

Controller Package

A screenshot of a cell phone

Description automatically generated

Model Package

A picture containing screenshot

Description automatically generated

View Package

Fig 3. Class Diagram UML

7. System Testing

Login Test

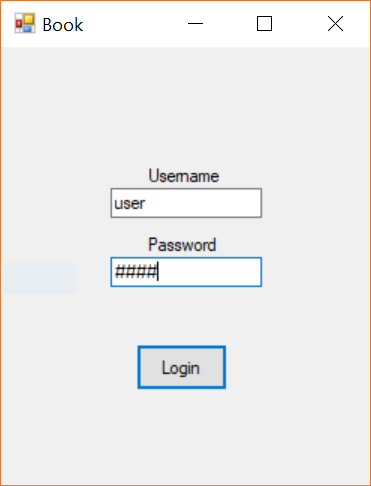


Fig 5. Login Window

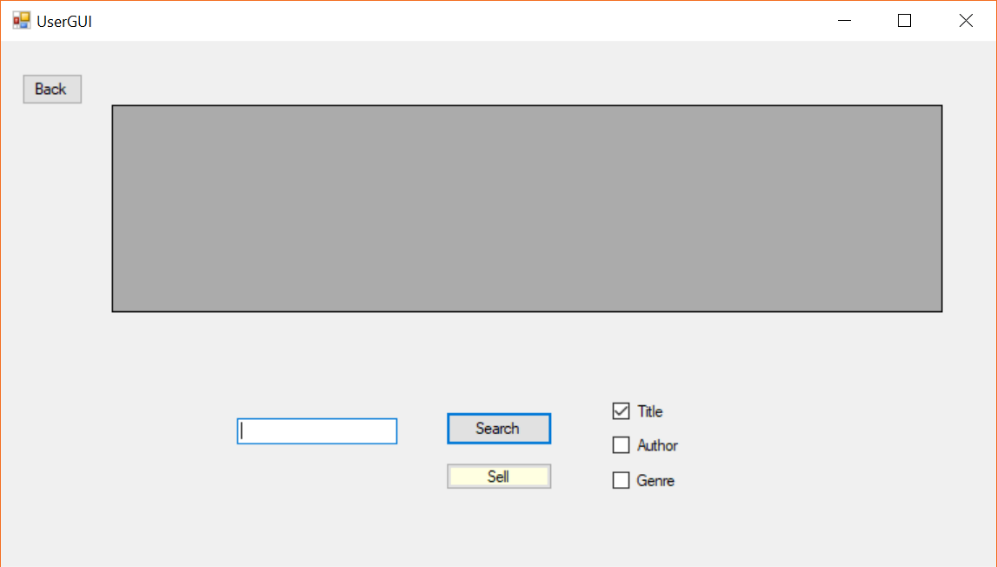


Fig 6. User Login Ok

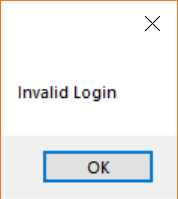


Fig 7. Wrong Login

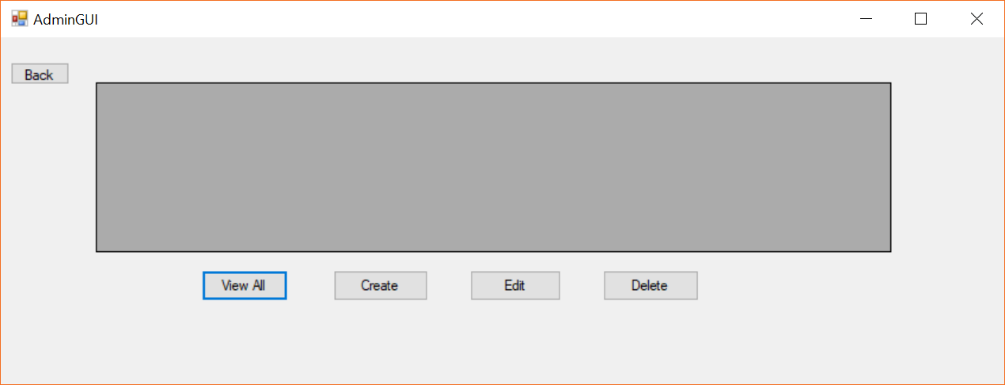


Fig 8. Admin Login Ok

Delete Book Test

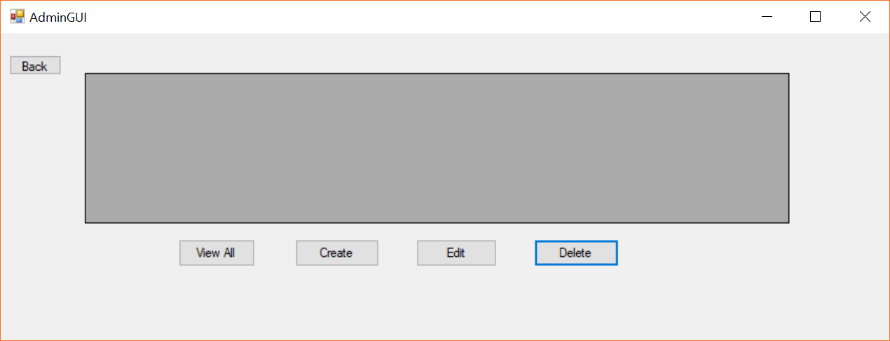


Fig 9. Delete Button

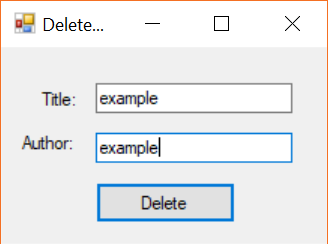


Fig 10. Book to Delete

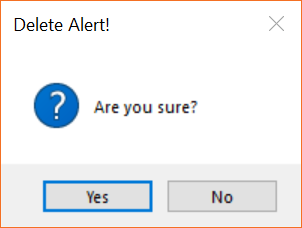


Fig 11. Confirmation

A screenshot of a cell phone

Description automatically generated

Fig. 12 Delete not ok

Create Test

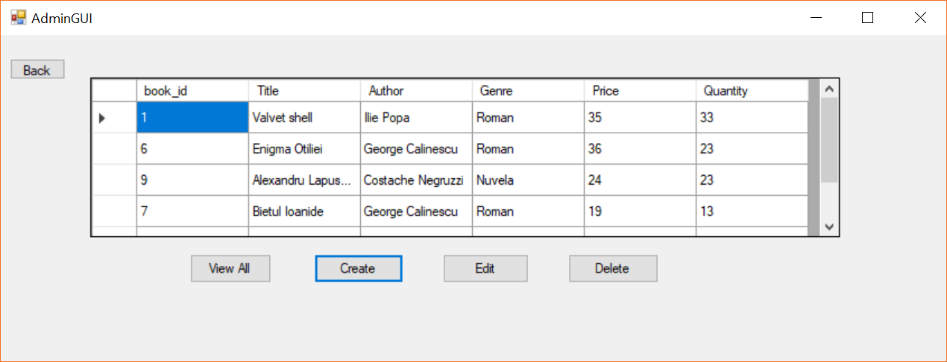


Fig 13. Create Button

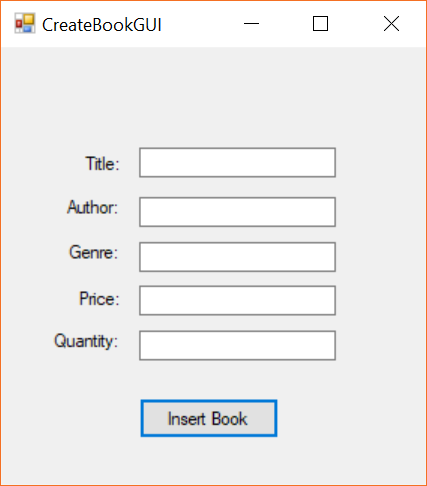


Fig 14. Create Window

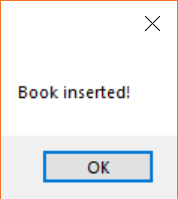


Fig 15. Create Ok

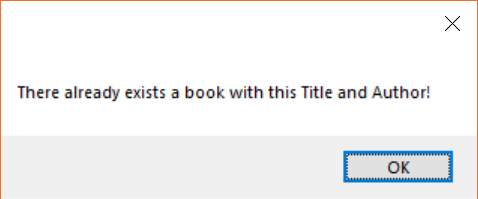


Fig 16. Create not OK

Bibliography

1 - https://www.tutorialspoint.com/design\_pattern/state\_pattern.htm