Assignment 1 – Ticket Selling App

Student: Cupsa Bogdan

**Group: 1**

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 4

3. System Architectural Design 5

4. UML Sequence Diagrams 6

5. Class Design 7

6. Data Model 8

7. System Testing 8

8. Bibliography 8

1. Requirements Analysis

# Assignment Specification

Design a ticket selling desktop application for a festival with 2 users: administrator and cashier.

# Functional Requirements

The functional requirements are the following:

* Administrators and Cashiers can register an account on the application.
* Administrators and Cashiers can login on to this application after they have a registered account.
* Administrators can see the table with the available shows.
* Administrators can see the cashiers that have registered.
* Administrators can delete shows.
* Administrators can delete cashiers.
* Administrators can add new shows.
* Administrators can export details about shows in a .csv file.
* Cashiers can sell tickets for each show.

# Non-functional Requirements

The non-functional requirements are the following:

* When a user registers the passwords should be encrypted. In this case the encryption algorithm is base64.
* The application should be scalable in case the database changes. In this case, the database is SQLite.

2. Use-Case Model

*Diagram

Description automatically generated*

*Diagram

Description automatically generated*

Use case: Export shows.

Primary actor: Administrator.

Main success scenario: A .csv file with details about the shows is created.

Extensions: Error message is displayed if this is not possible.

3. System Architectural Design

**3.1 Architectural Pattern Description**

The architectural pattern I used has 3 separates layers: Data Layer, Business Logic Layer and Presentation Layer.

* Data Layer handles the data models. Here we have the database tables.
* Business Logic Layer has the logic of the application. Here we have the abstractions for the database functions, so that the application is portable.
* Presentation Layer handles the user interface and is connected to the business logic layer.

**3.2 Diagrams**

Diagram

Description automatically generated

4. UML Sequence Diagrams

A picture containing text, indoor

Description automatically generated

5. Class Design

**5.1 Design Patterns Description**

* Singleton pattern is used to make sure that only one instance of a class is created during runtime: In our case I use UserRepository and ShowRepository.
* Observer pattern is used when a button is pressed and then the table with cashiers or shows is updated.
* Repository design pattern facilitates easier access to the database.

**5.2 UML Class Diagram**

A picture containing schematic

Description automatically generated

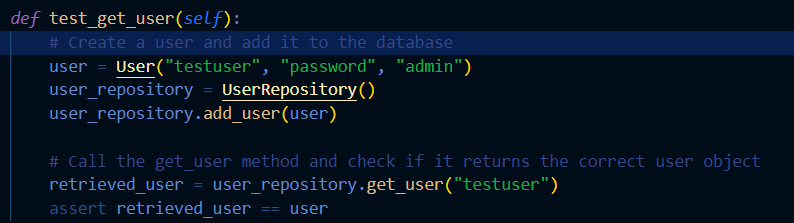
6. Data Model

The data model is formed of 2 classes: User and Show. I also have the UserRepository and ShowRepository so I apply the singleton design pattern and the Database class to help me with the queries.

* User: username, password, role
* Show: name, singer, date, number of tickets, genre.

7. System Testing

As for the testing, I have performed unit tests using the python unittest module where I assert if the test result fails or not in relation to the expected result.



8. Bibliography

* <https://docs.python.org/3/>
* <https://doc.qt.io/qtforpython/>
* <https://refactoring.guru/design-patterns>
* <https://www.youtube.com/playlist?list=PLzMcBGfZo4-lB8MZfHPLTEHO9zJDDLpYj>
* https://www.youtube.com/watch?v=sm1mokevMWk&ab\_channel=TechWithTim