Assignment A1

Student: Adrian Timis

**Group: 30432**

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

This application aims to develop a management system for students enrolled in the Computer Science department at the Technical University of Cluj-Napoca. The application will help students to better manage their courses and make it easier for teachers to grade and generate reports for the students.

# Functional Requirements

The application has two types of users: student and teacher/administrator. The functional requirements are described for each type of user.

The regular user can perform the following operations:

* Add/update/view client information
* Create/update/delete/view student profile
* Process class enrolment

The teacher/administrator can perform the following operations:

* CRUD on student’s information
* Generate reports for a particular period containing the activities performed by a student

# Non-functional Requirements

Non-functional requirements for this assignment include:

* Manageability
* Data integrity
* Reliability

Each of these represent a requirement for the software product in order for it to be viable in real life conditions. Manageability means that the system will not be too difficult to manage, as it is expected to have a large number of users, and a difficult managing process could take its toll on the user experience. Data integrity is vital as we are dealing with personal data, and data concerning grades which matter for such things like scholarships, Erasmus selection, etc. Reliability means that even in conditions which are not favorable, ex: many users using the service at once, the system can still behave as normal, despite unfriendly conditions.

2. Use-Case Model

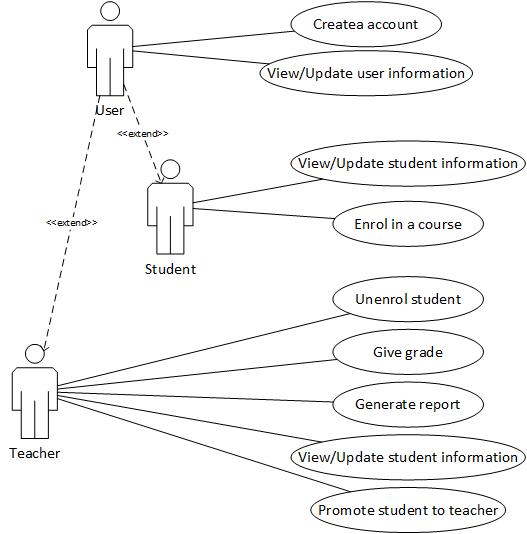
Use case: Application Uses

Level: Summary Level

Primary actor: Regular User

Main success scenario: Create Account

Extensions: view/update student’s information, generate reports, enroll in a course, update/view user information, update/delete/view student profile, promote student to teacher



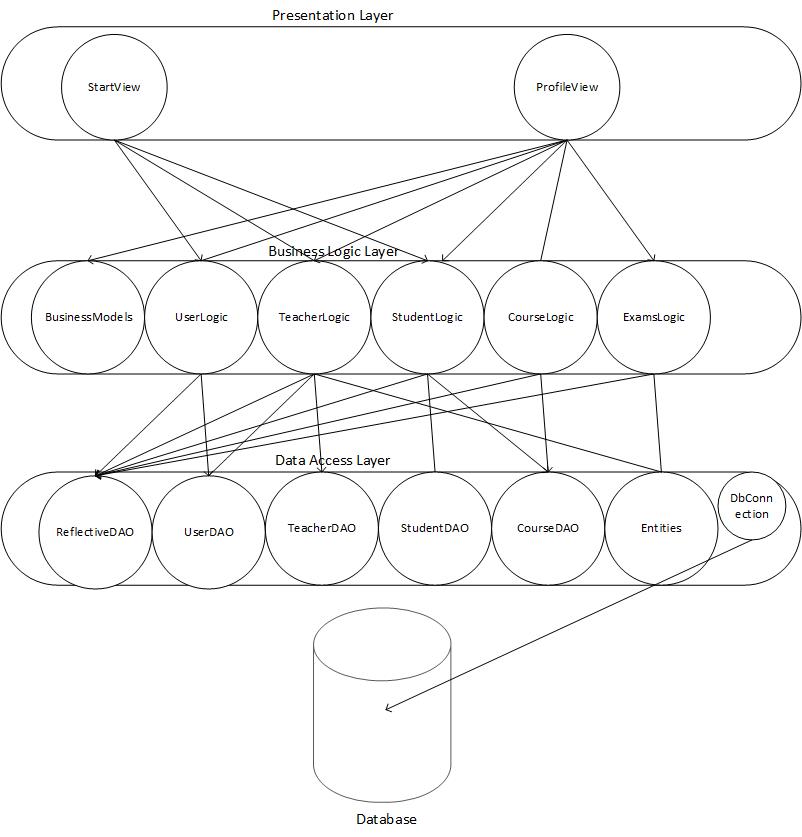
3. System Architectural Design

**3.1 Architectural Pattern Description**

The architectural pattern used in developing this application is the Layered Architecture. This architectural pattern splits the application into logical layers, by doing so the developers can easily differentiate between the functionalities provided by the software program, making it easier to create a reusable and maintainable computer program.

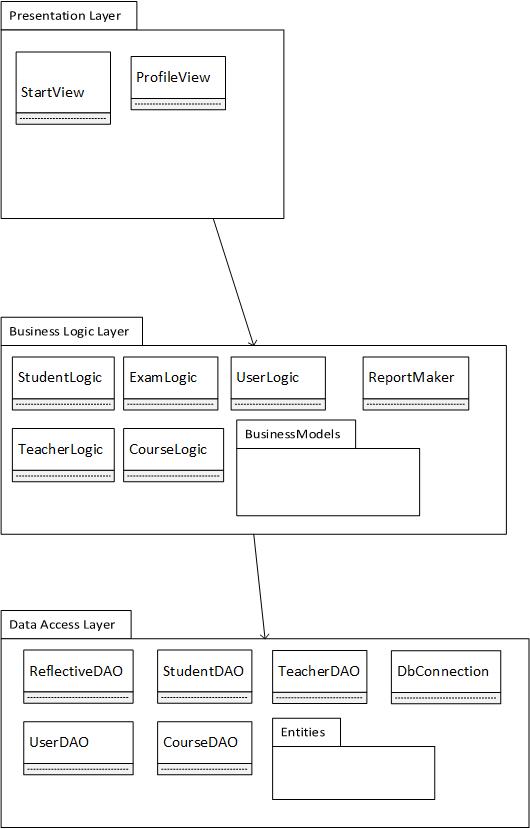
In the case of this application, the layers are: Presentation Layer, Business Layer, and Data Layer. Some other application may include a service layer as well, however, in this case there is no need for this layer.

**3.2 Diagrams**

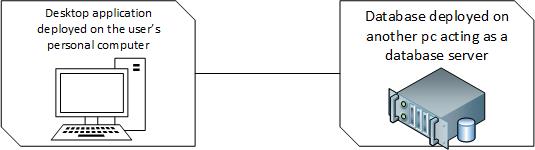


Above is a basic conceptual architectural design for this application. It can be clearly seen how software components which logically handle the same type of tasks are brought together in the same layer. As previously discussed, the application has 3 layers. The data layer, which handles the connection to the database server, in this case a postrgreSQL server, and a class which performs basic crud operations on the tables in the database. The business layer is where the actual functionality specific to this application is implemented. Here, the business classes and business logic use the functionalities provided by the data layer to manipulate data in the way that the application is meant to. Sitting on top of all these layers is the presentation layer, which is the graphical user interface with which the user interacts, it will be implemented using the UI library in java, Swing.

Package diagram:

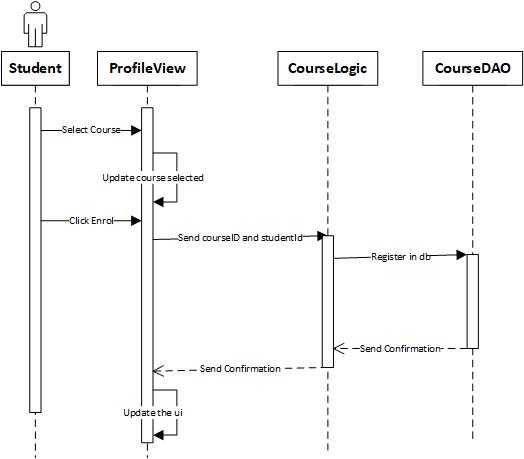


Deployment diagram:



4. UML Sequence Diagrams

Sequence diagram for enrolling in a course:

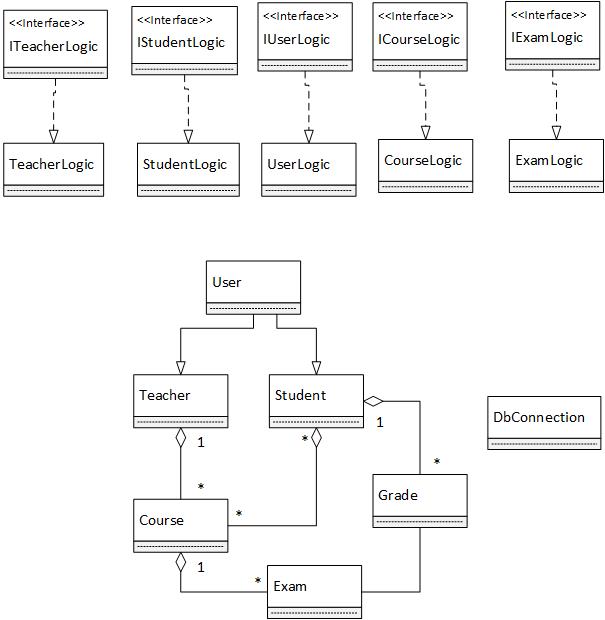


5. Class Design

**5.1 Design Patterns Description**

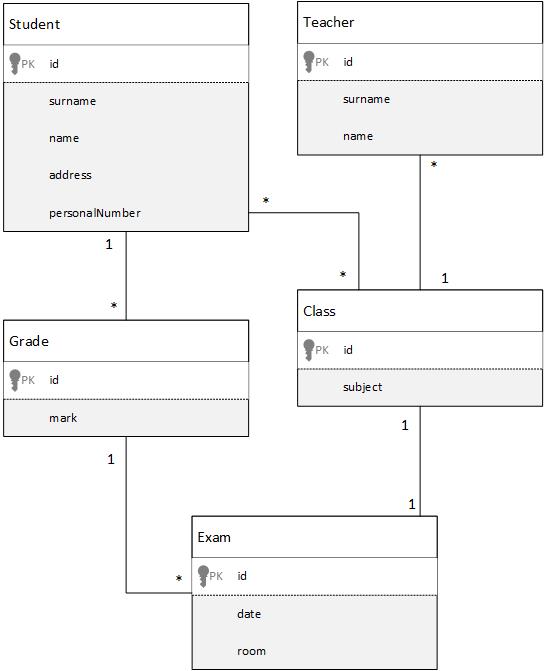
A design pattern used in this application is the singleton design pattern. It is used in the class that handles the db connection, because it is imperative that there is just one object which communicates directly with the database. The reason why there should be only one instantiated object communicating with the database is the fact that many database drivers are not thread safe, therefore more connections handled at the same time can cause problems in inconsistency of data, or even data loss.

**5.2 UML Class Diagram**

**

Above is a simple example of the class diagram of the application. The singleton design pattern is implemented in the DbConnection class. The design pattern is used by making the constructors of that class private, and holding an instance of itself as a private variable, and then providing a public method which return the instance, and if there is no instance created at that time, it creates one on the spot.

6. Data Model

**

7. System Testing

The system testing will be done mainly using unit testing and validation testing. As for methods, data-flow will be the preferred method.

8. Bibliography

* <https://msdn.microsoft.com/en-us/library/ee658109.aspx>
* <https://www.tutorialspoint.com/uml>
* https://herbertograca.com/2017/08/03/layered-architecture/