ASSIGNMENT A2-MVC

Student: Saratean Timeea

**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

Design and implement a Java application for the management of students in the CS Department at TUCN. The application should have two types of users: student and teacher, 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:

* Add/ update/ view client information (name, identity card number, personal numerical code, address, etc.)
* Create/ update/ delete/ view student profile (account information: identification number, group, enrolments, grades)
* Process class enrolment (enroll, exams, grades)

The administrator user can perform the following operations:

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

Application constraints:

* The data will be stored in a relational database. Use the Layers architectural pattern to organize your application. Create the Data Access Layer using SQL statements in the way you find most suitable for the application
* All inputs of the application will be validated against invalid data before submitting the data and saving it in the database

# Non-functional Requirements

Performance

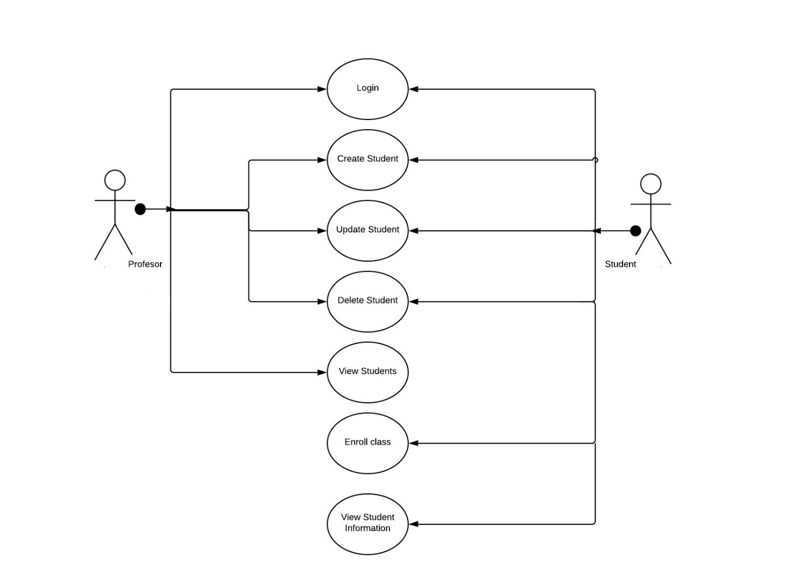
Scalability – the capacity to handle a growing amount of work

Maintainability – the probability of performing a successful repair action within given time

Data integrity – data is accurate and reliable

Capacity, Security, Usability.

2. Use-Case Model

****

Use case: Update student

Level: user-goal level

Primary actor: Teacher

Main success scenario:

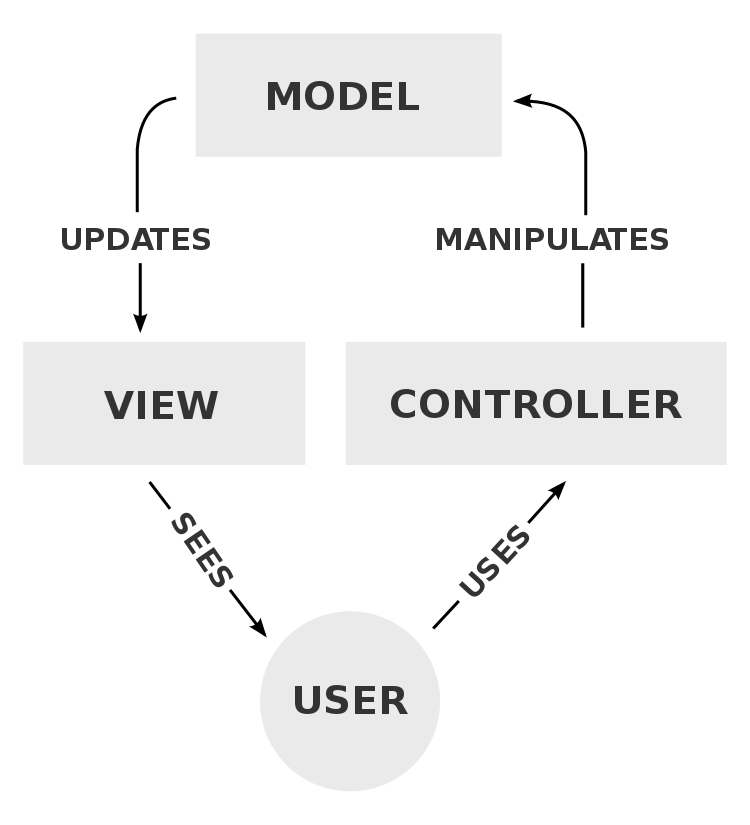
1. The teacher login within the application
2. The teacher selects a student
3. He updates the information he wants
4. In the database, the old student is deleted and a new student is created with all his new informations

Extensions: In case there is another student with the same information we want to update, the student will not be updated. Also, an error message will appear.

3. System Architectural Design

**3.1 Architectural Pattern Description**

**Model–View–Controller** (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

****

**Model**

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface.It directly manages the data, logic and rules of the application.

**View**

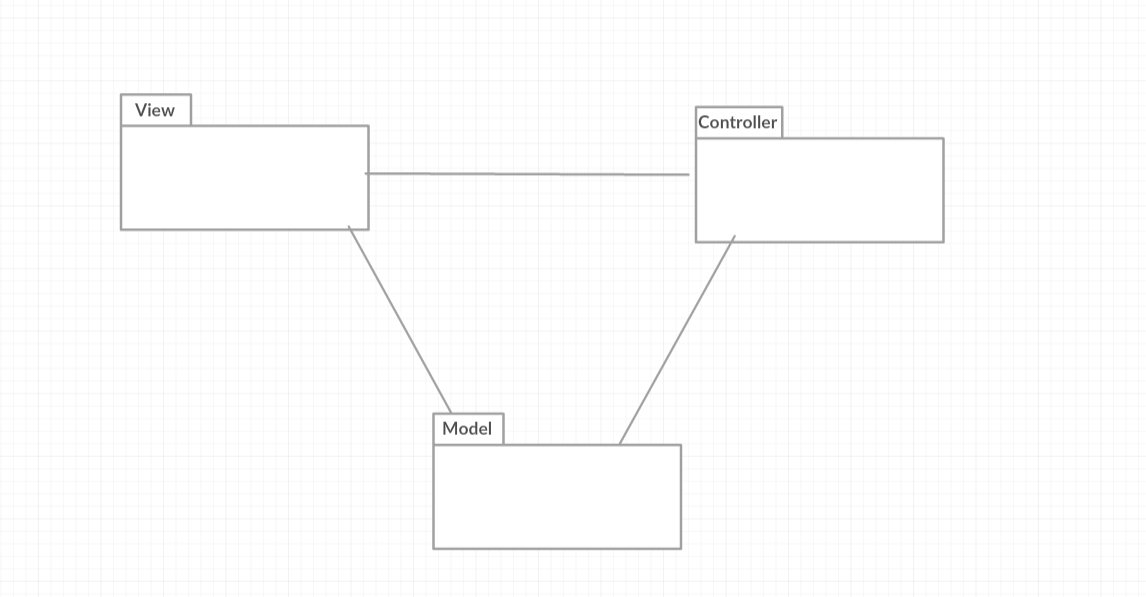
Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

**Controller**

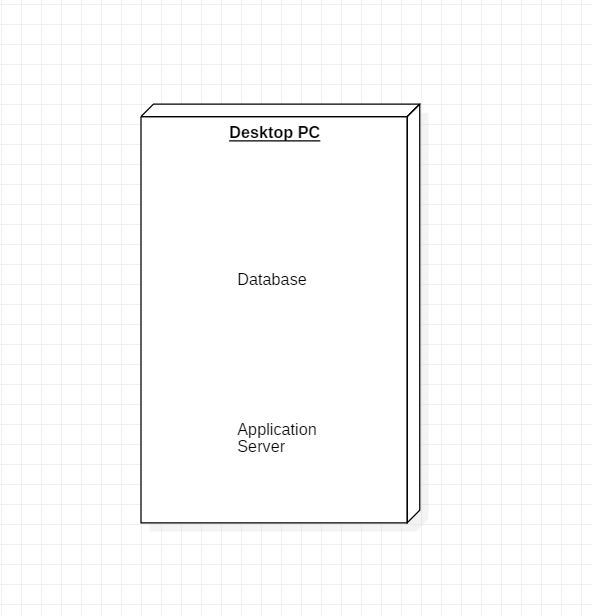
Accepts input and converts it to commands for the model or view.

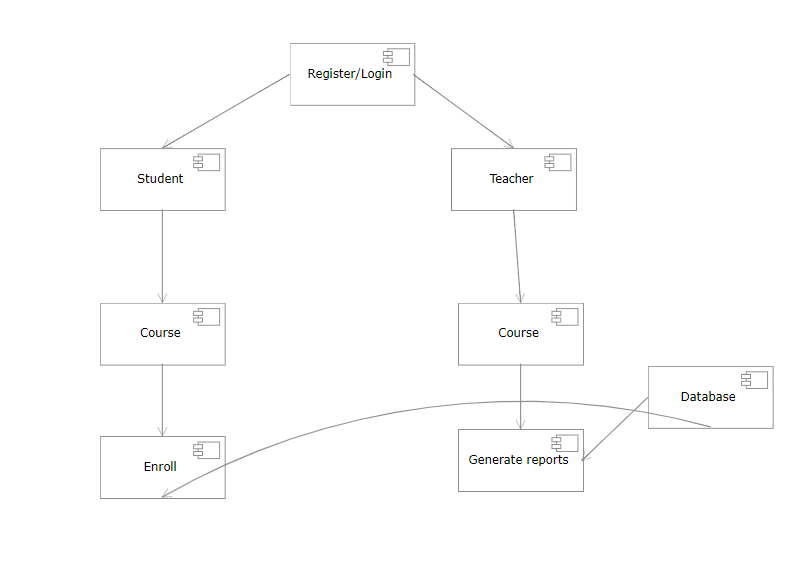
**3.2 Diagrams**

Package diagram:



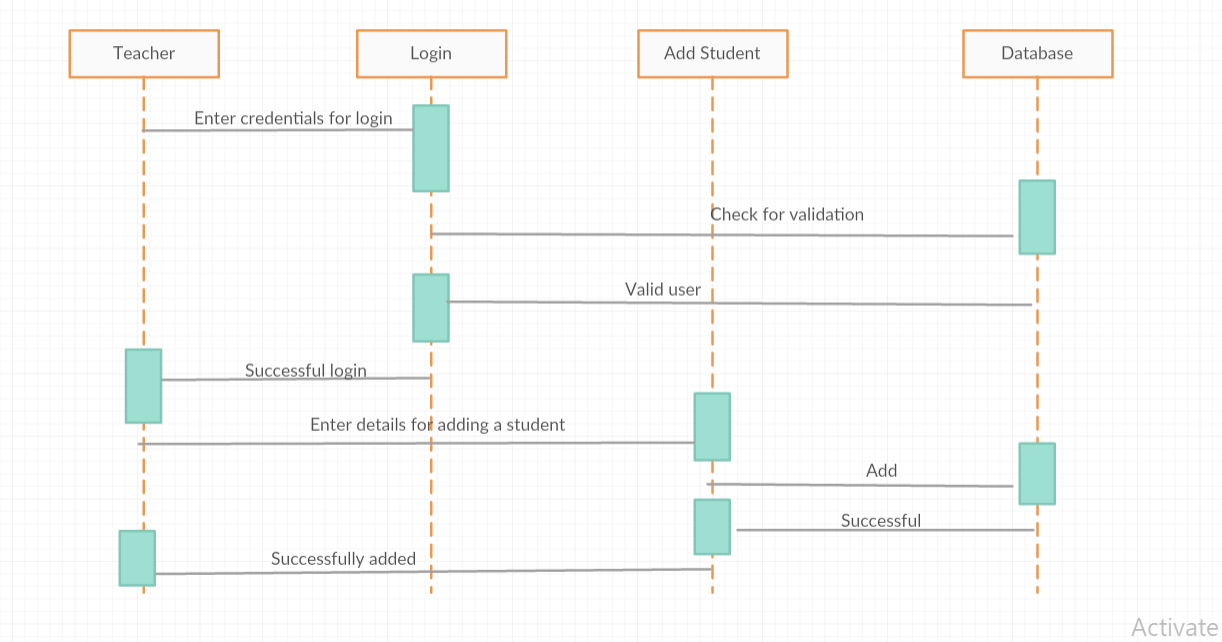
Deployment diagram:



Component diagram:

1. UML Sequence Diagrams

UML Sequence Diagram for adding a new Student :

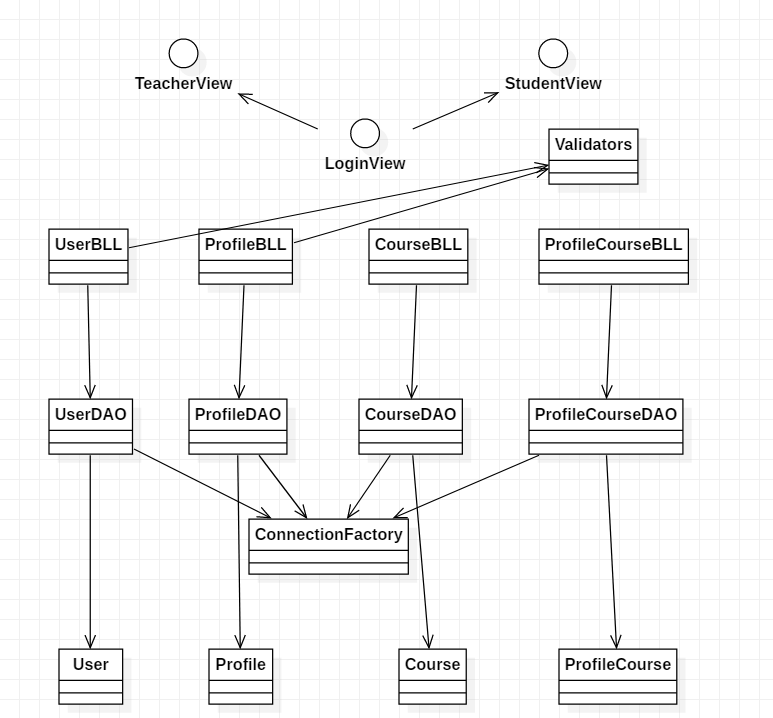


5. Class Design

**5.1 Design Patterns Description**

Singleton – for the database connection. A database connection class can be created by extending the Java Persistence API (JPA) or by creating a simple class with a method that returns the connection statement. Those who use JPA need not be concerned with this part of the article, because with JPA the persistence is based on the Singleton pattern.

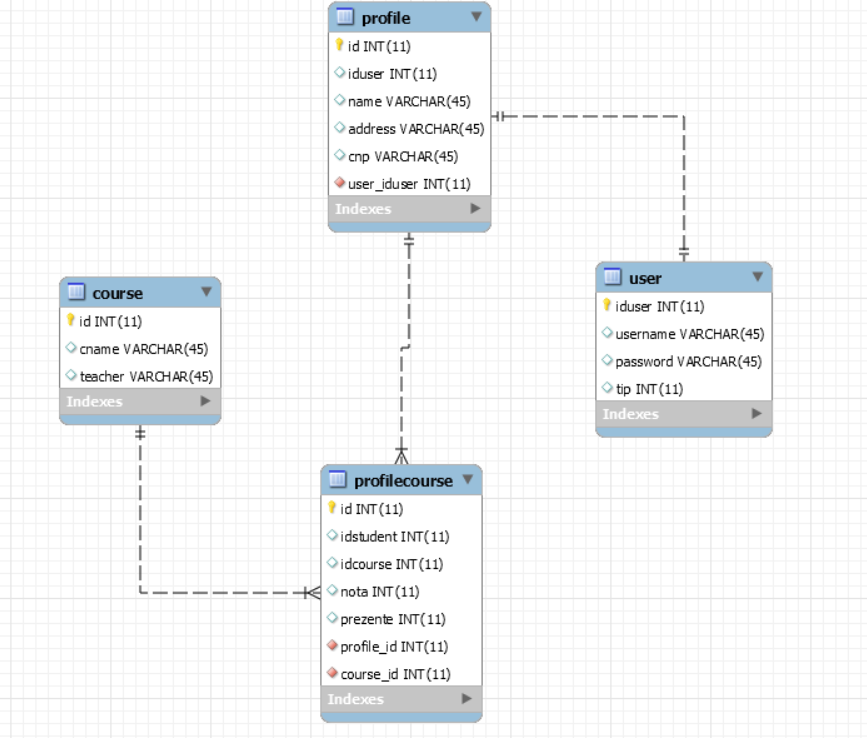
**5.2 UML Class Diagram**

****

1. Data Model

My application is made based on 4 tables:

* User : for class User. If the attribute tip is 0 then the user is a teacher and if the attribute is 1 then the user is a student
* Profile: for class Profile. Contains additional info for the user
* Course: for class Course
* ProfileCourse: this class is the link between the Profile table and the Course table because there is a many to many relationship between them



7. System Testing

For testing we use the package validators. Here, we verify if the username is an email, if the personal numeric code has 13 characters and if the name of a student doesn’t contain special characters.

8. Bibliography

<https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109%28v%3dpandp.10%29>

<https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

https://creately.com/diagram/example/ik3jjl1m1/Student%20management%20system