ASSIGNMENT A1 - LAYERS

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

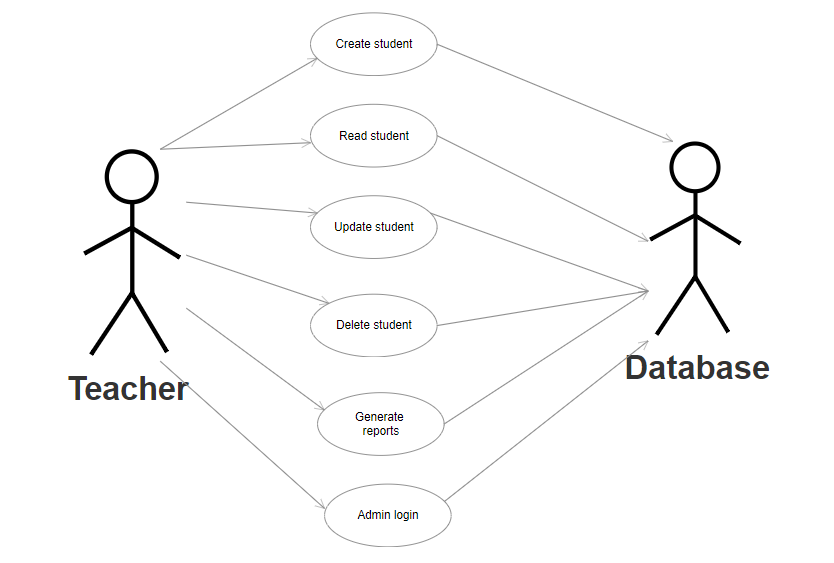
Maintainability – the probability of performing a successful repair action within given time

Data integrity – data is accurate and reliable

Capacity, Security, Usability.

2. Use-Case Model

**Use-Case diagram for Teacher:**



Use case: Update student

Level: user-goal level

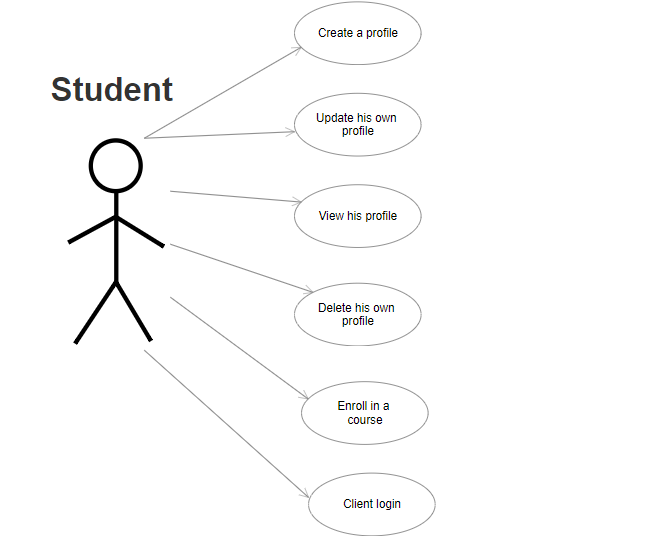
Primary actor: Teacher

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.

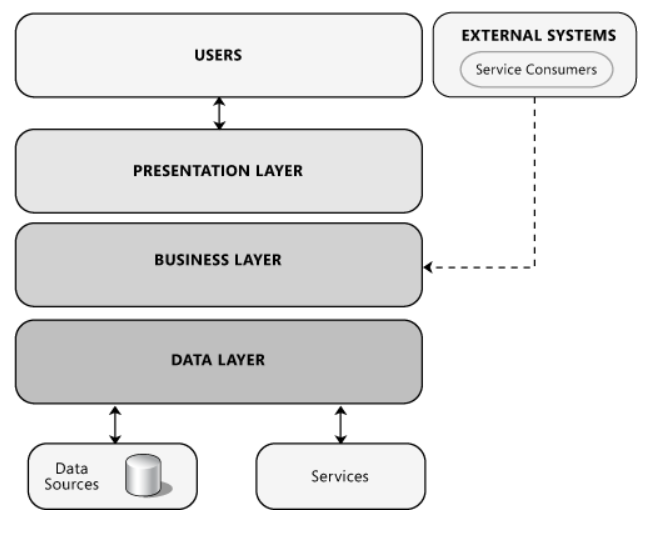
**Use-case diagram for student:**

****

3. System Architectural Design

**3.1 Architectural Pattern Description**

I made my application using Layers Design Pattern. With this architectural design pattern, you can decompose the design into logical groupings of software components. These logical groupings are called layers. Layers help to differentiate between the different kinds of tasks performed by the components, making it easier to create a design that supports reusability of components. Each logical layer contains a number of discrete component types grouped into sub layers, with each sub layer performing a specific type of task.



Structure of Layers Design Pattern 1

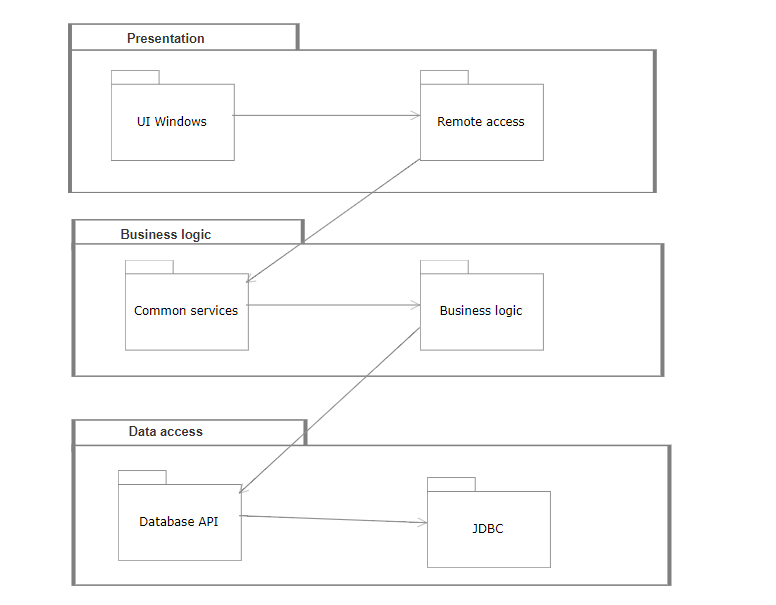
 Presentation layer contains the user oriented functionality responsible for managing user interaction with the system, and generally consists of components that provide a common bridge into the core business logic encapsulated in the business layer.

Business layer implements the core functionality of the system, and encapsulates the relevant business logic. It generally consists of components, some of which may expose service interfaces that other callers can use.

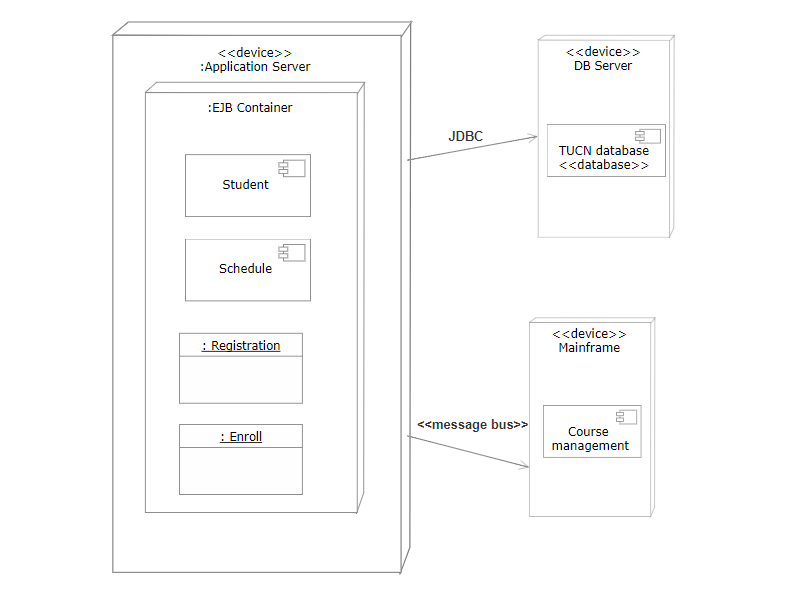
The data layer provides access to data hosted within the boundaries of the system, and data exposed by other networked systems; perhaps accessed through services. The data layer exposes generic interfaces that the components in the business layer can consume.

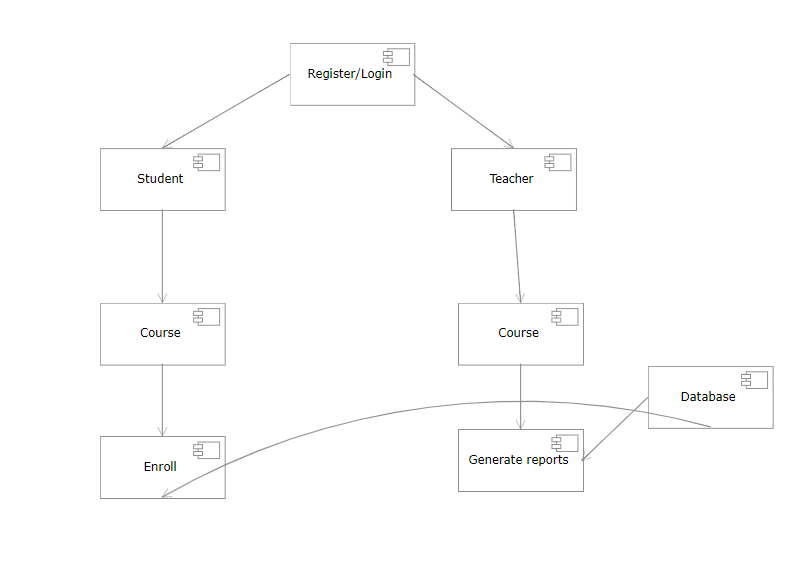
**3.2 Diagrams**

Package diagram:



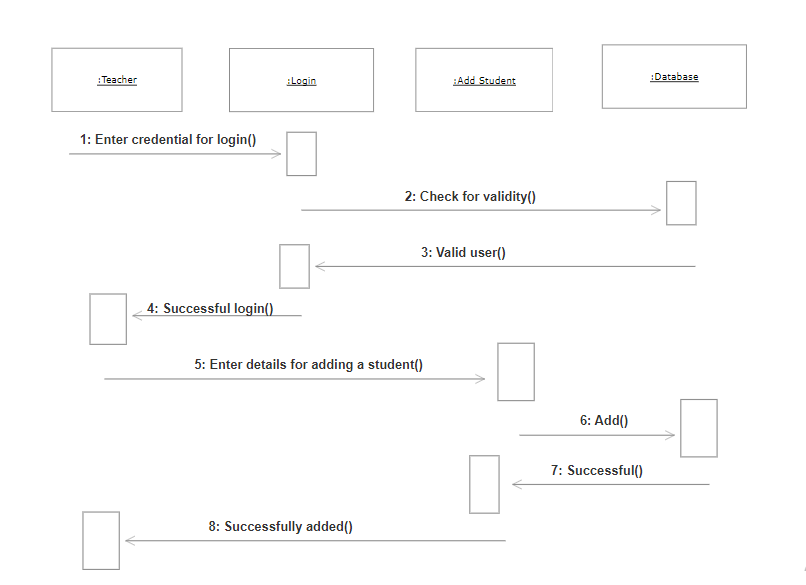
Deployment diagram:



Component diagram:

1. UML Sequence Diagrams

UML Sequence Diagram for adding a new Student :

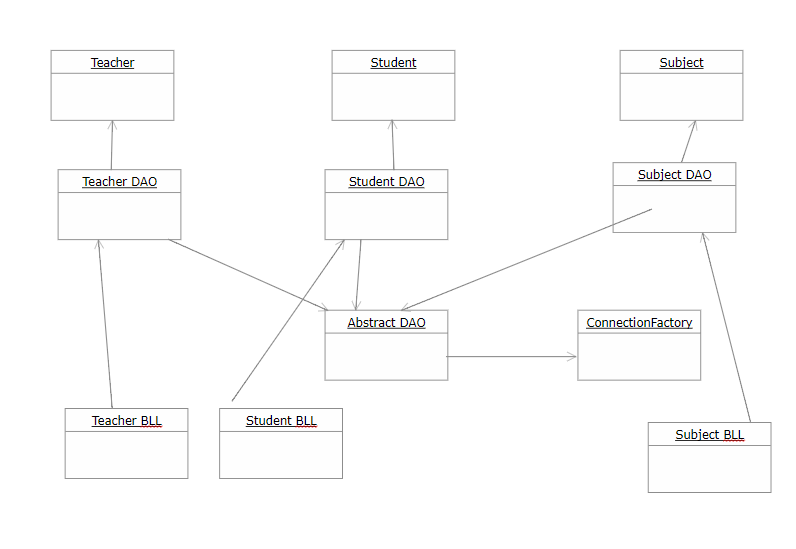


5. Class Design

**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**

****

6. Data Model

My application has a class Teacher that represents the Teacher table from the database. Every teacher has a name, id, email, password and a subject that he is teaching.

The class Students represents the Student table from the database. Every student has his own id, a name, email and a password. Also, a student has a list of every subject that he is taken.

And the class Subject is a class that represent the Subject table.

7. System Testing

Testing cases will be created to verify if a student can enroll to a course or if a teacher can create or add or update a student.

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