<Assigment2-Students management>

Student:Lăzăreanu Sabina-Ștefana

**Group:30233/2**

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/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:

- 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.

**Requirements**

- Create the analysis and design document

- Implement and test the application.

* 1. **Functional Requirements**

Each user have his own account with username and password but teachers have a common account with :

Username: admin

Password: admin

Depending on the account with the user logged in, will appear a different interface for performing actions in the application. For student will appear an interface that allows him to make CRUD operation on his own data and choose the course that he wants to take part and also the grades that he obtained. For admin will appear a different interface, a complex one that allows him to make CRUD operations on students and put grades for each student.

\* The student (regular user) can perform the following operations:

- Create/update/delete/view on own 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. [1]

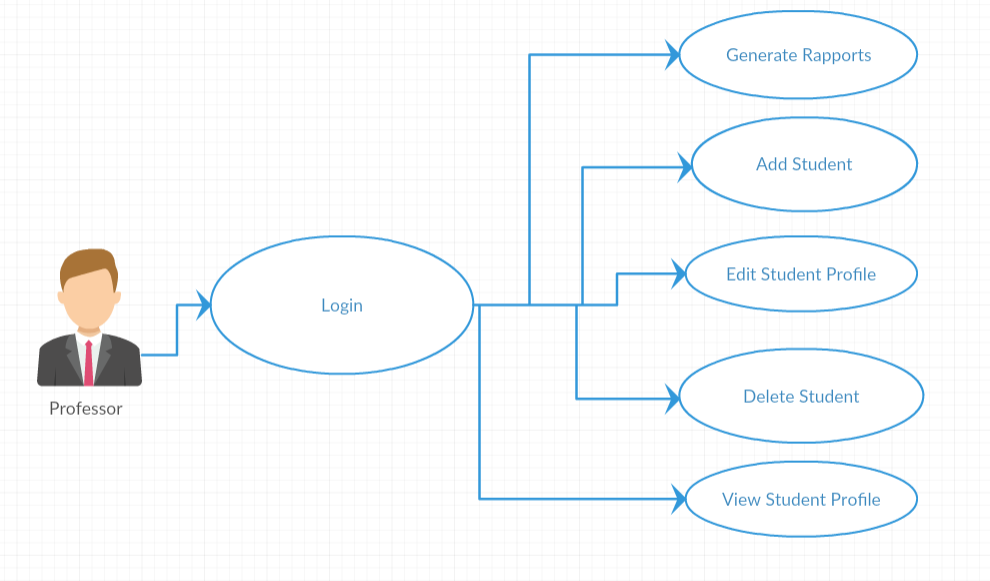
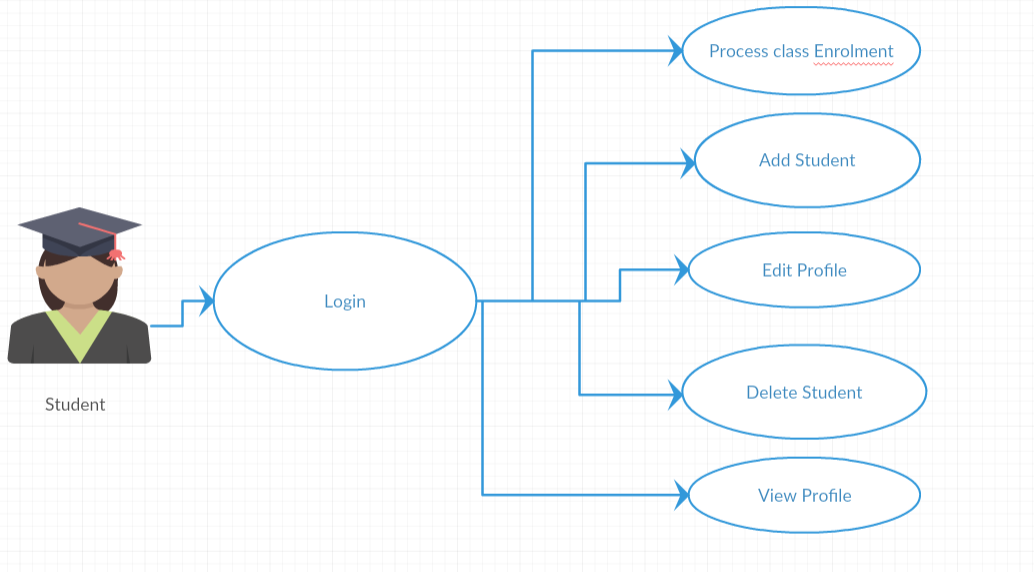
# Non-functional Requirements

The data about students, teachers and courses are stored in a relational database (MySql).

Modified data in a database should be updated for all users accessing it within 2 seconds.

This is a scalable application that can be extended easily. The security of this application is good because we store the passwords in database using an encrypting method that is really popular and that is SHA-256. [1]

2. Use-Case Model



3. System Architectural Design

**3.1 Architectural Pattern Description**

This app design is based on *Layered Architecture and MVC Design Pattern*. It is the most common architecture pattern, otherwise known as the n-tier architecture pattern. This pattern is the de facto standard for most Java EE applications and therefore is widely known by most architects, designers, and developers. The layered architecture pattern closely matches the traditional IT communication and organizational structures found in most companies, making it a natural choice for most business application development efforts.

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). Although the layered architecture pattern does not specify the number and types of layers that must exist in the pattern, most layered architectures consist of four standard layers: presentation, business, persistence, and database.

Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request.

Has 3 layers on one tier. First layer is represented by the package named user.interface, than we have the BLL layer(Business Logic Layer) that is the “brain of application”, and we have DAL(Data Access Layer) that contains communication with database.

***Model–View–Controller*** (usually known as MVC) is an [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern) commonly used for developing [user interfaces](https://en.wikipedia.org/wiki/User_interface) 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](https://en.wikipedia.org/wiki/Code_reuse) and parallel development.

Traditionally used for desktop [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUIs), this architecture has become popular for designing [web applications](https://en.wikipedia.org/wiki/Web_application).

**3.2 Diagrams**

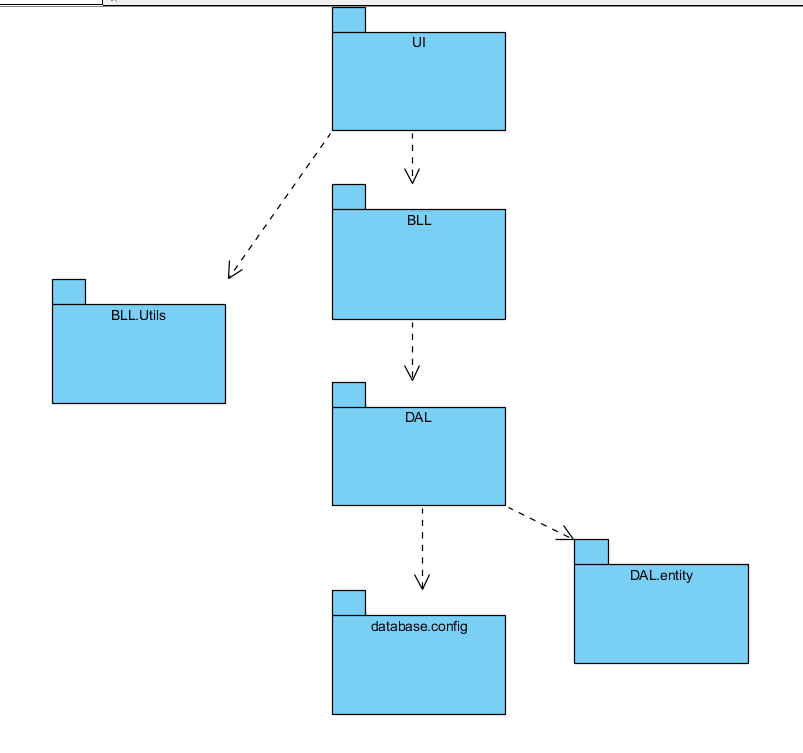
Presentation

BLL

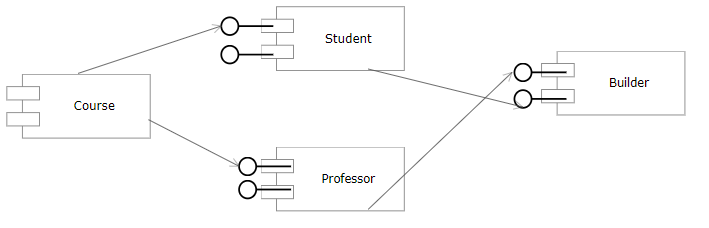
DAL

DataBase

****



Package Diagram



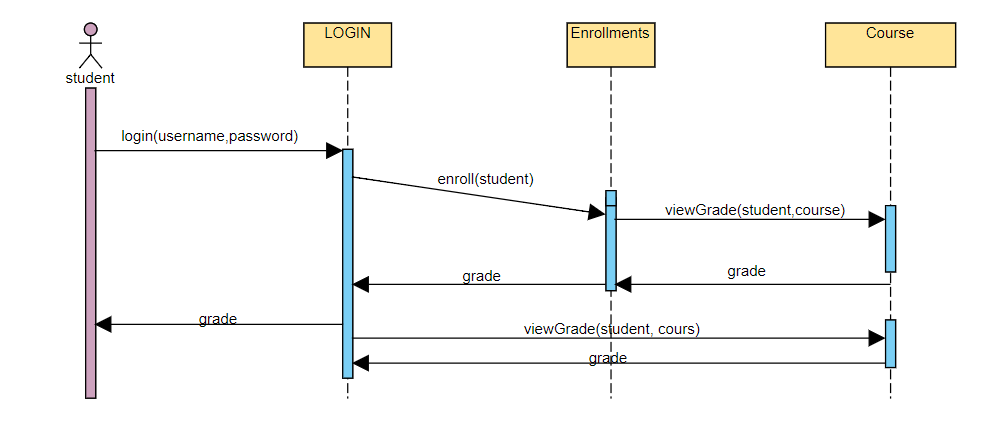
Component Diagram

**The presentation layer**or user interface layer is responsible for interactions with user – information gathering, and making operation with information, as well as displaying it in a legible and user-friendly

**The “business“ logic** **layer** plays the role of transferring information between the user interface and the data layer, including integration of the required decision logic or transformation of transferred data (calculations, aggregation of information from more data sources and the like).r

**The data storage layer** implements persistent data storage, with a relational database (MySql).[5]

4. UML Sequence Diagrams



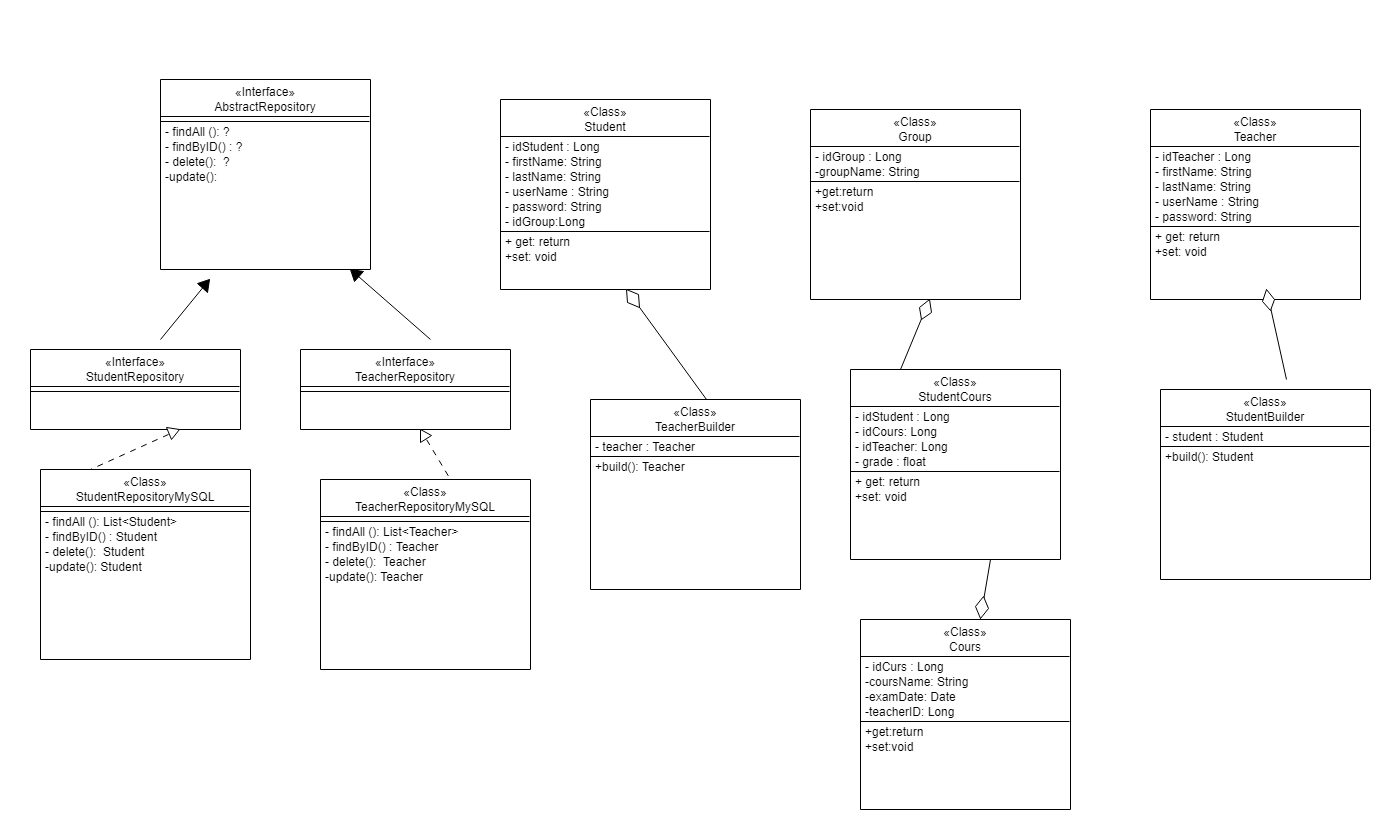
5. Class Design

**5.1 Design Patterns Description**

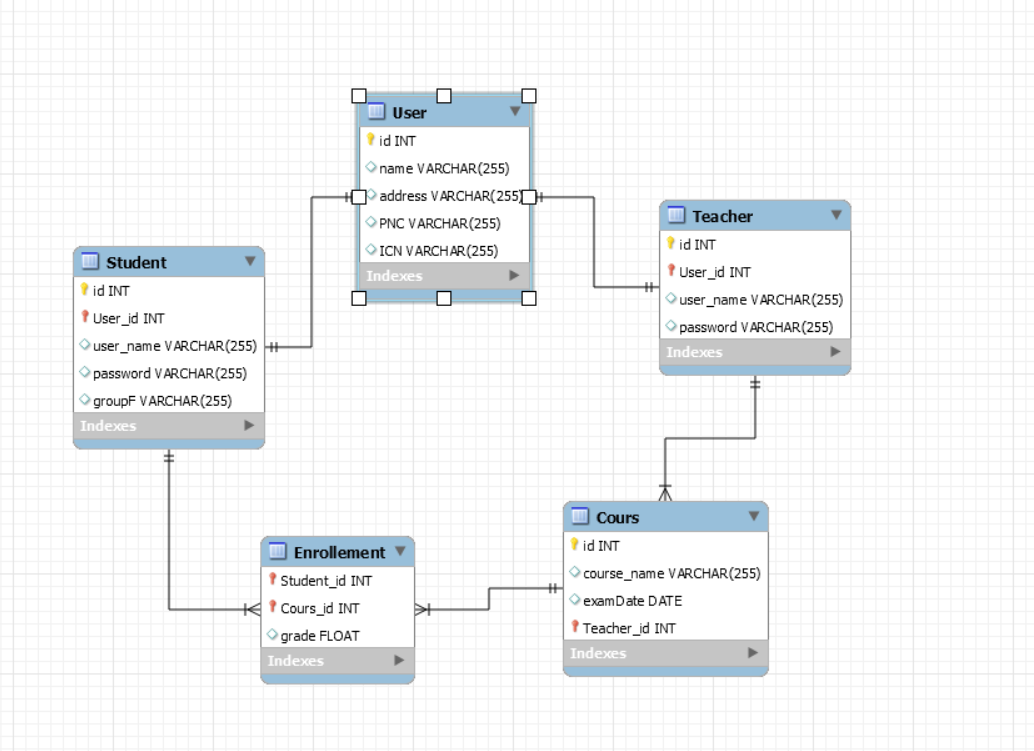
Design patterns that are used in this application are:

* Factory Design Pattern- This pattern is used for deciding at the compile time, database that we are using.
* Singleton Design Pattern- This pattern is used on database connection because connection with database has to be a single instance otherwise will appear conflicts and error.

**5.2 UML Class Diagram**



6. Data Model

**

Data Model contains four main components(Student, Teacher, Course, Group) and a secondary component that binds Course and Student named CourseStudent.[3]

7. System Testing

A unit test should test functionality in isolation. Side effects from other classes or the system should be eliminated for a unit test, if possible. That is why a choose to use MOKITO for making may JUnit Tests.

I create mock objects manually (via code) and use a mock framework to simulate these classes. Mock frameworks allow you to create mock objects at runtime and define their behavior. [6]

I also choose to test my application manually and trying to take care of each use case.

8. Bibliography

1. <https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>
2. <https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>
3. <https://www.cs.kent.ac.uk/people/staff/saf/dbdc/portfolios/jim/instructionaldesign/COMU114_lab_4.pdf>
4. <https://www.javatpoint.com/design-patterns-in-java>
5. <http://www.web-integration.info/en/blog/how-to-apply-the-principles-of-three-tier-architecture-in-a-web-integration-project/>
6. <https://www.vogella.com/tutorials/Mockito/article.html>