Assignment1

Student: Albert Erika-Timea

**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

The software being designed is an application for students and teachers of the Technical University of Cluj-Napoca. The application has these mentioned two types of users, one (the student) is able to create/update/view profile, manage personal information, enroll to classes and exams, view grades. The second type of user (teacher/administrator) can view students’ profiles, generate reports on activities performed by the students and add new courses. Both types of users need to sign in with username or password before being able to perform any other type of operation.

# Functional Requirements

1. Create profile information: new student user enters new information on profile (for a new user no information is added until he/she doesn’t set it)
2. Manage profile information: users can manage their profile by adding, viewing, updating information
3. Manage student information: student users can add, update and view personal information
4. Class enrollment: a student can select a class and enroll to it; this may include enrolling in an exam an viewing grades obtained at a particular subject
5. Find student by Id: teachers can search students and view students’ information
6. Give grades: teachers can generate reports on activities involving students regarding their performance
7. Add new course: teachers can add new course

# Non-functional Requirements

1. Availability: the system needs to be available as much as possible since both students and teachers need to use it frequently, the percentage of time when the system is down should be as small as possible
2. Performance: the application needs to be performant as well; has to respond in time to the user request – there may be time periods when the system will run in overload mode, as too many requests are arriving (e.g. in exam period everyone wants to see their grades)
3. Security: in case of this system we need to distinguish between teachers and students; teachers are authorized for editing students’ grades, but students should no access this area. The application should be able to prevent student access to this section but should allow teachers to enter it.
4. Testability: the system is kind of complex and I need to design it in a way to be able to gradually test each component (by unit tests) but also the whole product when ready (user testing)
5. Usability: the whole application should be user friendly, not overcomplicated and should make it easy for the user to perform specific tasks. It should provide features that help avoiding errors when using it, efficient usage and t has to obtain user satisfaction.

2. Use-Case Model

Use case: Enroll to Exam

Level: user-goal level

Primary actor: Student

Main success scenario:

1. student opens drop-down list
2. student clicks on a desired course
3. student selects if he/she wants to take selected course or not
4. if student pressed Yes, course appears in the table of taken courses

Extensions:

* if student is already taking the course, the message “Course already taken” appears

*A picture containing text, map

Description generated with very high confidence*

3. System Architectural Design

**3.1 Architectural Pattern Description**

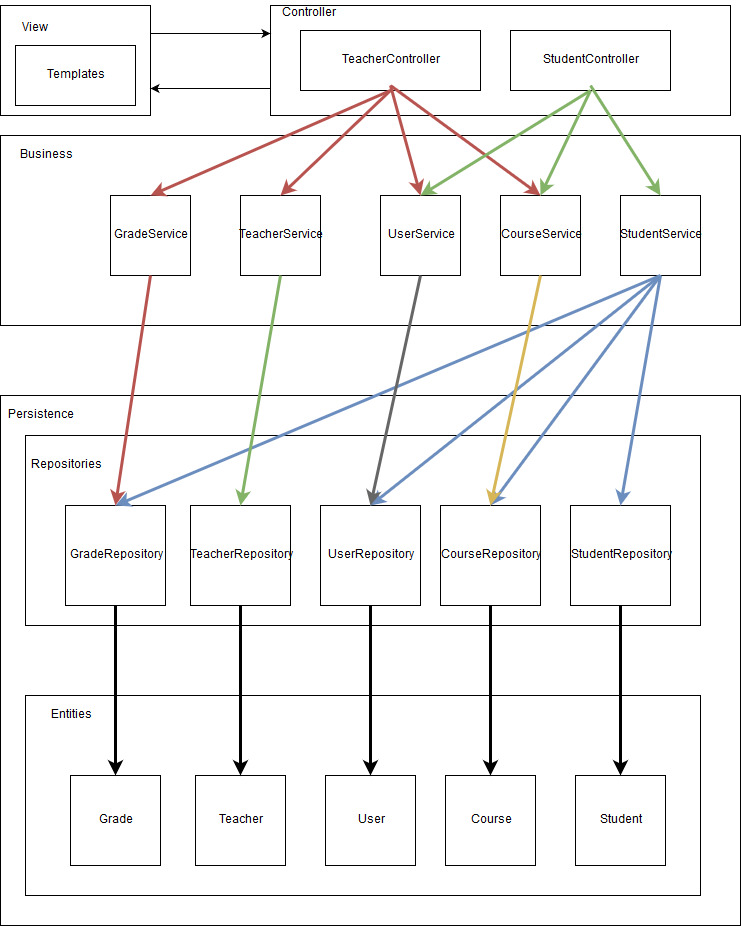
In this assignment we have to combine the already existing Layered Architecture with the MVC pattern. In case of the MVC it allows a loose coupling between input logic, UI logic and business logic. When we combine Layered architecture with MVC, we should get the following structure:

A screenshot of a cell phone

Description generated with very high confidence

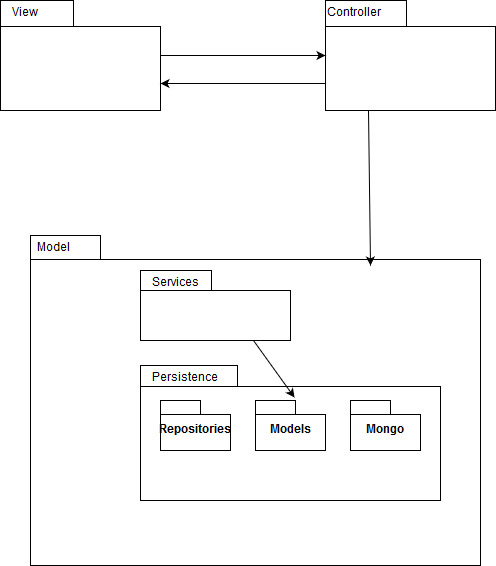
This means that the View would transmit and retrieve data from the Model only thru the Controller, there would be no interaction between the View and the Model. The Model itself combines all the business logic and data model; it holds all the logical operations that is performed on the data model, and the data model itself. The Controller does not hold any business logic.

From layered point of view, this will still remain a layered architecture because what was the Presentation layer until now still doesn’t violate the principle of it and the parts of the Model represent the Business Logic and Data Access Layer from the Layered Architecture.



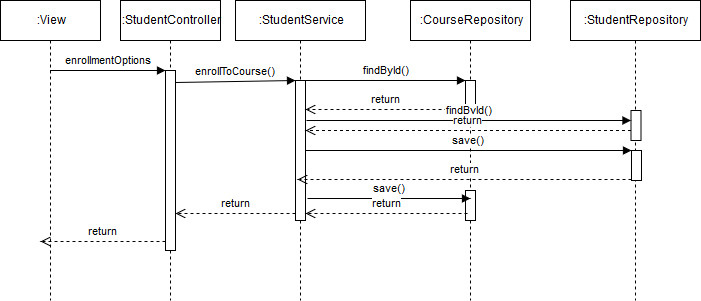
**3.2 Diagrams**

For this application I am using the MVC architectural pattern over the Layered Architecture. The MVC provides the decoupling of the View and the Model by communication going through the Controller. The whole architecture still keeps its Layered form as the different components of the Layered Architecture are clearly separable.



4. UML Sequence Diagrams

**Enrollment**



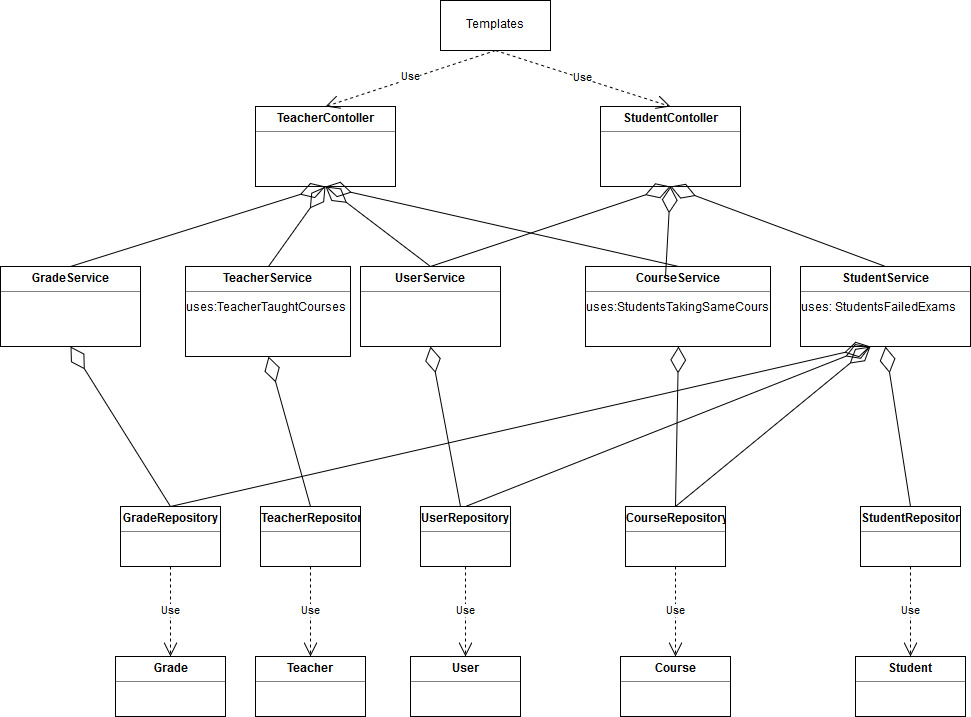
5. Class Design

**5.1 Design Patterns Description**

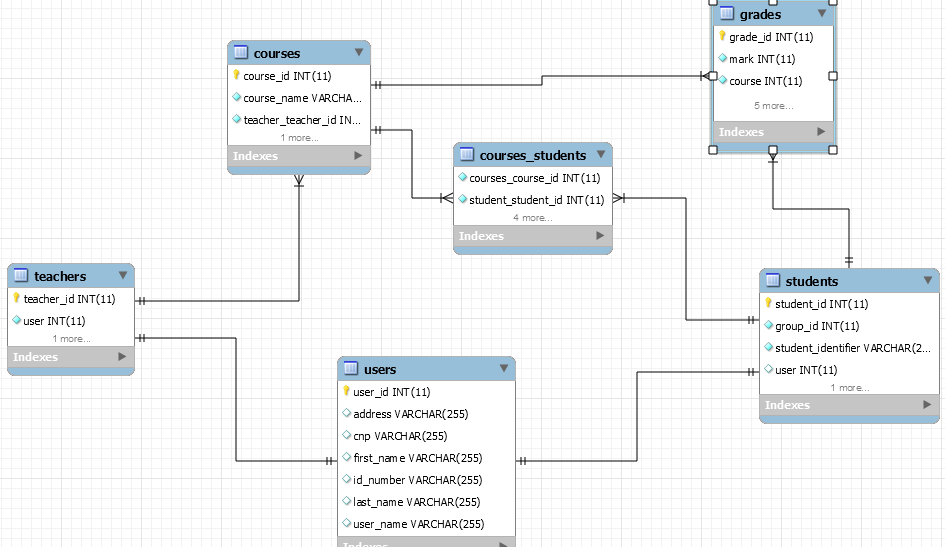
* Factory Method Design Pattern:

For this application a couple of different kinds of reports can be created; since we don’t know, which one is created by different services, I created an interface, that each report class implements. This way it is the user program that decides which report to generate, all having the same type.

**5.2 UML Class Diagram**

**

6. Data Model



7. System Testing

The way I tested the developed application was User Acceptance testing. Basically, I gradually developed every feature. Once I reached the UI level with it, I connected it to the business logic and I tested it the way a normal user would do it. I was putting the application in real world situations, trying to generate an error in the software. When I met an error (exception) I was tracing it back right until I found what caused it and I corrected it.

I was following this above mentioned procedure until I reached the final solution.

8.Bibliography

* <https://en.wikipedia.org>
* <https://stackoverflow.com>
* <https://msdn.microsoft.com/en-us/library/ee658117.aspx>
* <https://msdn.microsoft.com/en-us/library/ee658103.aspx>
* <https://msdn.microsoft.com/en-us/library/ee658081.aspx>
* <https://msdn.microsoft.com/en-us/library/ee658109.aspx>