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. Log in: each of the two types of users need to log in before being able to perform other operations
2. 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)
3. Manage profile information: users can manage their profile by adding, viewing, updating information
4. Manage student information: student users can add, update and view personal information
5. 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
6. Find student by Id: teachers can search students and view students’ information
7. Give grades: teachers can generate reports on activities involving students regarding their performance
8. 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: Log in

Level: user-goal level

Primary actor: Teacher

Main success scenario:

1. teacher enters password and username
2. teacher presses Log In button
3. teacher enters authorized area

Extensions:

* incorrect username or password – system displays “No user with such username or password”
* system is down - system displays “System error. Try again in a few seconds.”

*A picture containing text, map

Description generated with very high confidence*

3. System Architectural Design

**3.1 Architectural Pattern Description**

The chosen architectural pattern is Layers. An architectural pattern itself basically provides a set of principles, that, when being followed, allow us to partition our software in a way that we create separate components that later help design reuse as well.

The Layered architectural pattern we partition the system in layers that communicate with each other, without introducing any cycles. These layers represent components that would be part the logical partitioning of the system. Dividing our code in this way, helps maintainability and clarifies the whole design. These layers should represent different packages in our application.

A close up of a map

Description generated with high confidence

**3.2 Diagrams**

For this application I am using the Layered architecture pattern. On the lowest layer (Data Layer) there is a component for accessing the database (this one stores the DAO classes) and another one for the real-world entities (User, Student, Teacher, Course, ). On the layer above (Business Logic Layer) the components handle the specific purpose of every operation; here we process the data received from the Presentation Layer and pass it to the data layer. Also, this is the place where if the user needs some data that is the result of some specific operations on the data, we perform this and “send” it to the upper layer. At the top there is the Presentation Layer that hold all the UI components (in our case we need several views for the different user types, authorized and unauthorized areas, for different facilities). This layer also need to hold the controller, the component, that connects the business logic and the user interface.

A screenshot of a cell phone

Description generated with high confidence

4. UML Sequence Diagrams

**Log In**

A screenshot of a social media post

Description generated with very high confidence

5. Class Design

**5.1 Design Patterns Description**

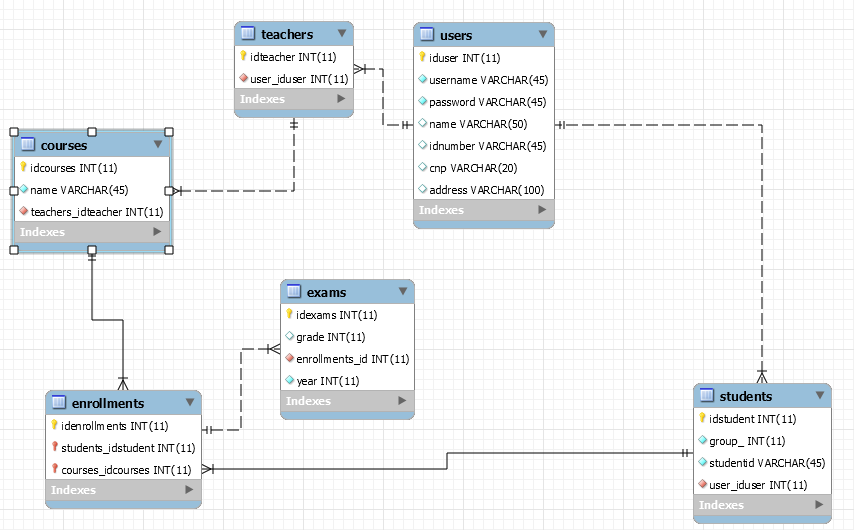
* In the Data Access Layer I used the Singleton Design Pattern to create one single instance of the ConnectionFactory. All the data access objects that are trying to reach the database are using the same connection.

**5.2 UML Class Diagram**

*A close up of a map

Description generated with high confidence*

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>