Assignment1
Analysis and Design Document

Student: Albert Erika-Timea
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

[Application description]

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/view profile, manage personal information, enroll to classes and exams. The second type of user (teacher/administrator) can perform CRUD on students' profiles and generate reports on activities performed by the students. Both types of users need to sign in with username or password before being able to perform any other type of operation.

1.2 Functional Requirements

[Present the 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: new student user creates profile
3. Manage student profile: in case of student users they can manage their profile by viewing, updating and deleting it
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. CRUD: teachers can create, read, update, delete students' information
7. Generate report: teachers can generate reports on activities involving students regarding their performance

1.3 Non-functional Requirements

[Discuss the non-functional requirements for the system]

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

[Create the use-case diagrams and provide one use-case description (according to the format below).
Use-Case description format:
Use case: <use case goal>
Level: <one of: summary level, user-goal level, sub-function>
Primary actor: <a role name for the actor who initiates the use case>
Main success scenario: <the steps of the main success scenario from trigger to goal delivery>
Extensions: <alternate scenarios of success or failure>

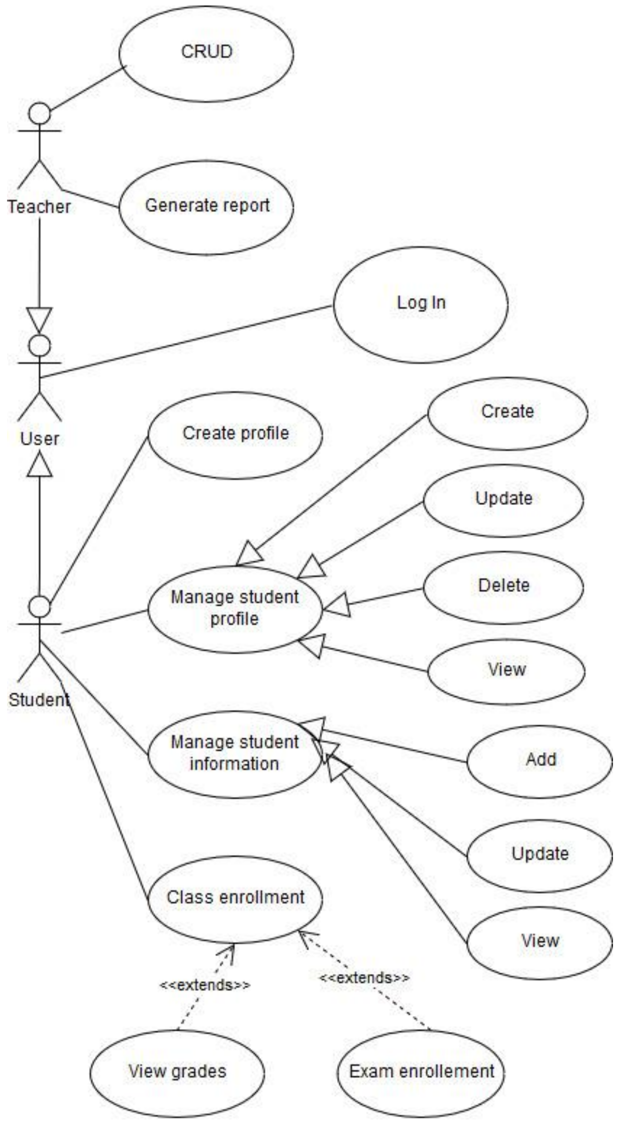
Use case: Log in
Level: user-goal level

Primary actor: Teacher
Main success scenario:

1. teacher enters password and username
2. teacher presses Log In button
3. teacher enters authorized area and reads the system displaying “Successfully logged in”

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.”



3. System Architectural Design

3.1 Architectural Pattern Description

[Describe briefly the used architectural patterns.]

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.

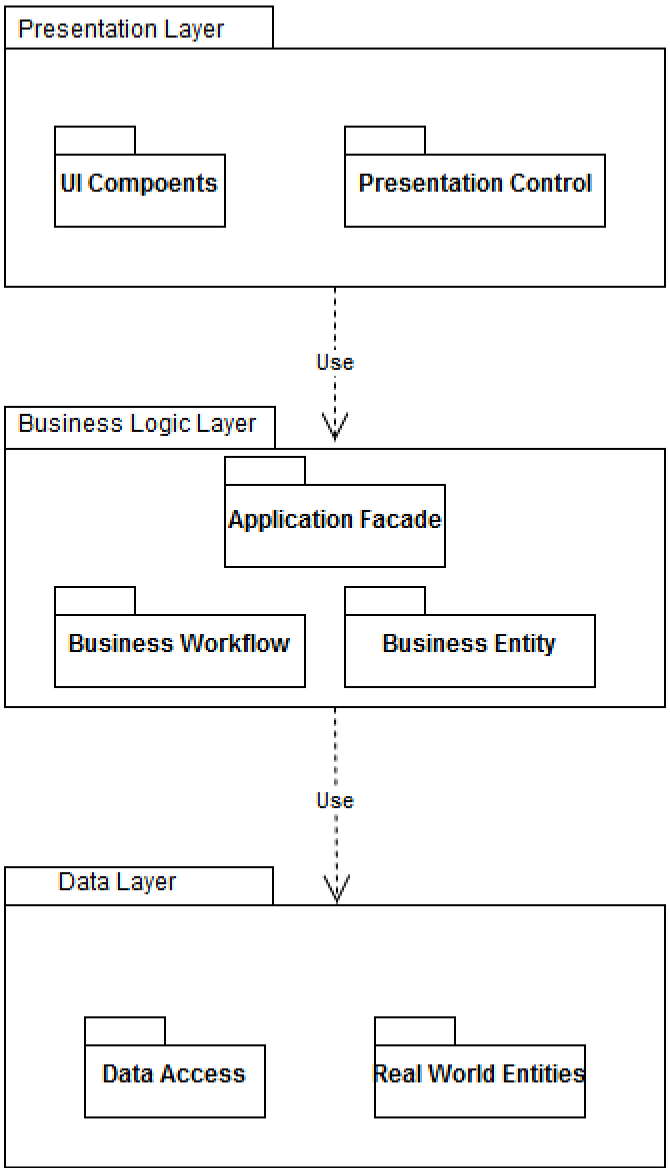
The Layered architectural pattern we partition the system in layers that communicate with each other, without introducing any cycles. These layers represent components that would be part the logical partitioning of the system.

Dividing our code in this way, helps maintainability and clarifies the whole design. These layers should represent different packages in our application.

3.2 Diagrams

[Create the system's conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]

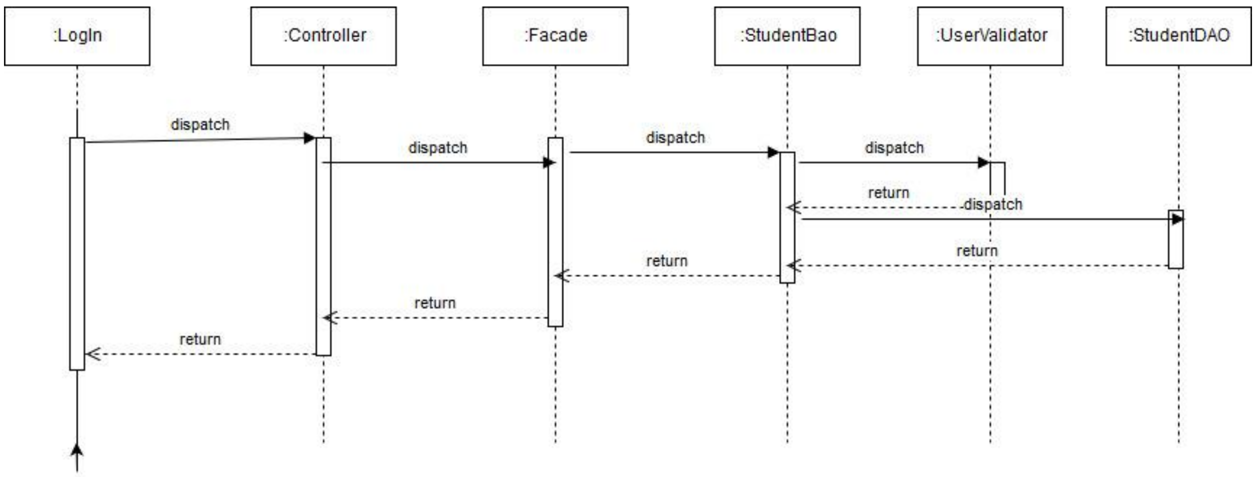
For this application I am using the Layered architecture pattern. On the lowest layer (Data Layer) there is a component for accessing the database (this one stores the DAO classes) and another one for the real-world entities (Student, Teacher, Course). On the layer above (Business Logic Layer) the components 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.



4. UML Sequence Diagrams

[Create a sequence diagram for a relevant scenario.]

Log In



5. Class Design

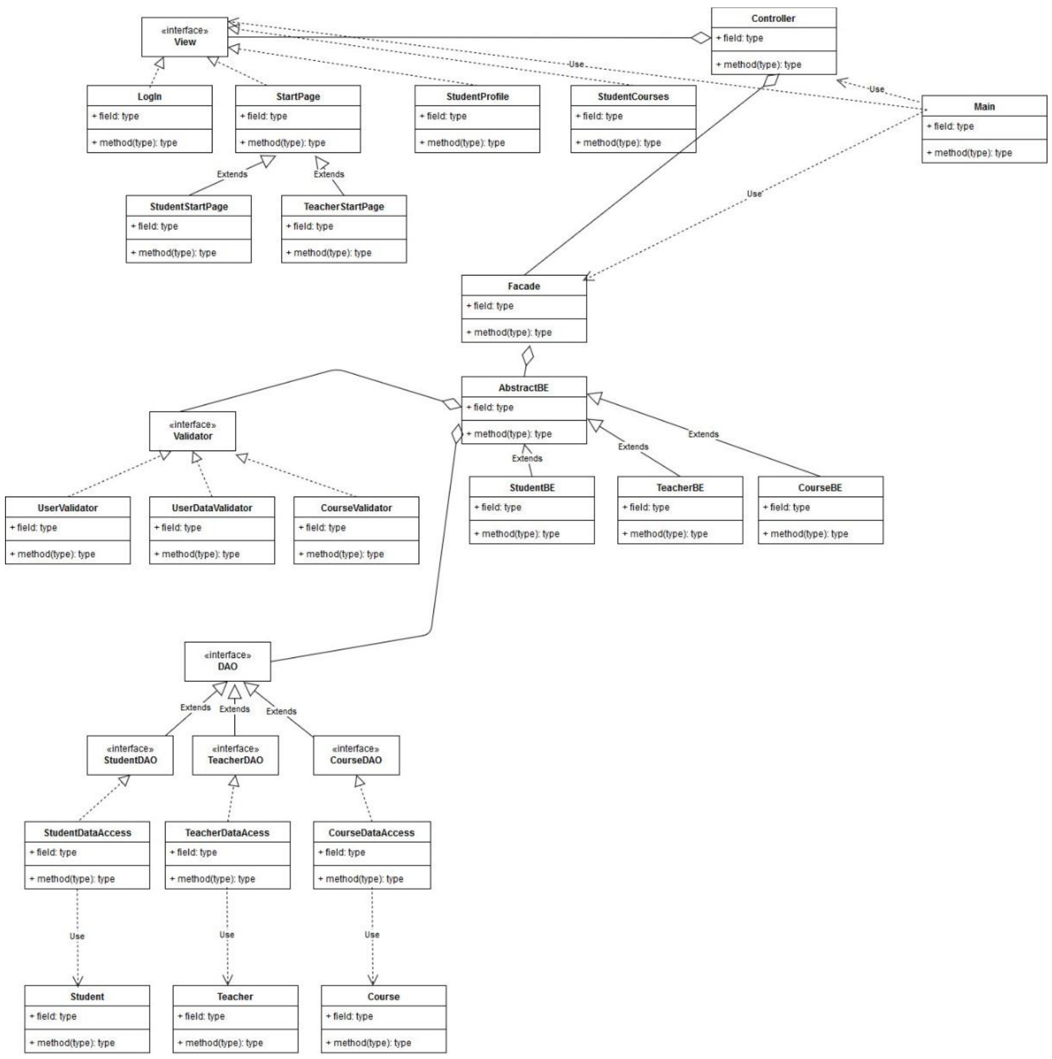
5.1 Design Patterns Description

[Describe briefly the used design patterns.]

- In the BLL we may use the Façade design pattern to create an abstraction of the below components for the level above (Presentation layer). This way we don’t need to access each component differently, we have an instance for each of them in the façade.
- In the Data Access Layer we need the Factory method design pattern to create the DAO objects. The data access changes in case of each object and for every type we are interested in different data. To create a maintainable design we create an interface for the DAO that is used by the upper layer.

5.2 UML Class Diagram

[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]



6. Data Model

[Present the data models used in the system's implementation.]

The application's main real-world components are the students, teachers and the courses; these are the entities that define the organization of data. They are all components of Data Access Objects that make a connection to a database and allow to access the actual data.

Since our application may get complicated when not following principles and predefined architectures, we need to make use of this. In our case, following the SOLID principle is essential, so that we can create a maintainable software with reusable components. Another solution that also contributes to the maintainability of the application is to follow the Layered architecture.

In case of the GUI, the DAO and the Business Entity I used generalization to help and contribute to the extensibility of the software. I used also several interfaces, for the same reason and also for those cases when something chages in the behavior of it (it is easier to handle changes).

7. System Testing

[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]

- Unit testing
- User testing

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