Assignment A2
Analysis and Design Document

Student: Raul-Mihai Acu
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implementation of an application for the management of students in the Computer Science department of the Technical University of Cluj-Napoca. It will provide access to students, classes and teachers information and will be able to generate reports for a particular student`s activities performed over a given period of time.

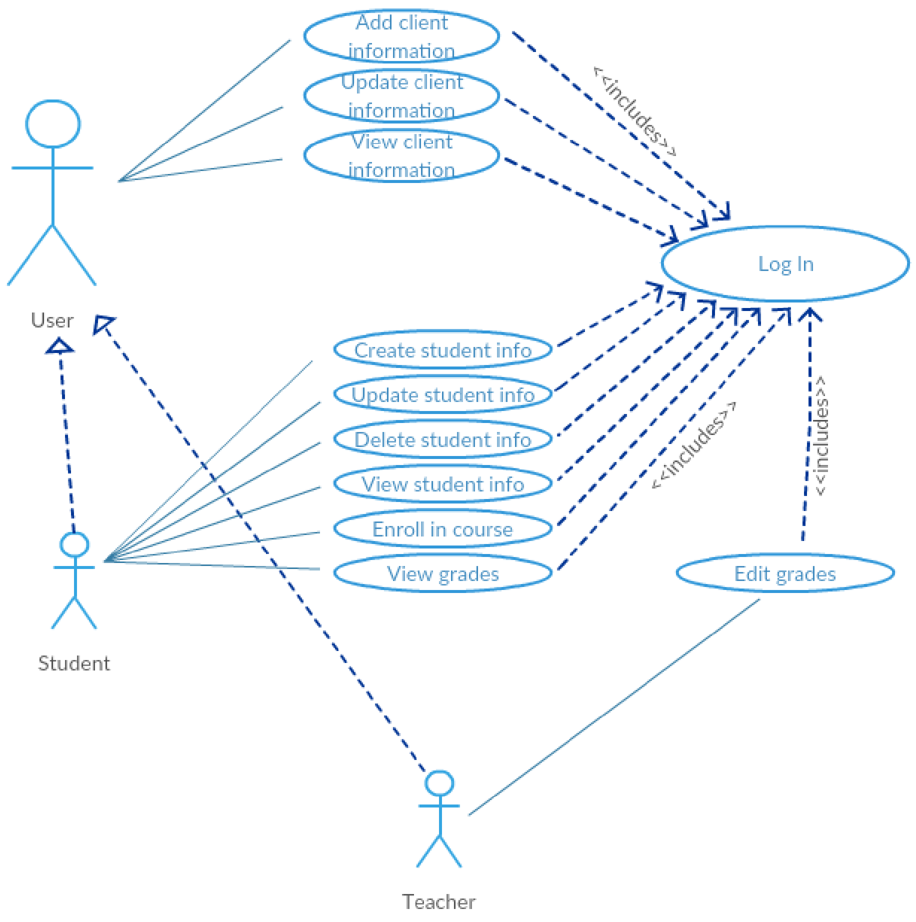
1.2 Functional Requirements

• The data will be stored in a relational database.
• The application should have a Layered architecture.
• All the inputs of the application have to be validated before submission.

1.3 Non-functional Requirements

• Accessibility and Usability – the application will have as targets all kinds of users.
• Readability – the code needs to be easily understood by other programmers, have a good naming convention.
• Security and Privacy – only the users will have access to the data.

2. Use-Case Model



- **Use case:** View Student Profile
- **Level:** User-goal level
- **Primary actor:** Student
- **Main success scenario:** The student logs in and chooses to view his profile. The info is displayed on the screen.
- **Extensions:** The student tries to log in, but provides wrong password and no access is granted. The student tries to log in, but does not have an account.

3. System Architectural Design

3.1 Architectural Pattern Description

Layered architecture

Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application. 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

Model–view–controller (MVC)

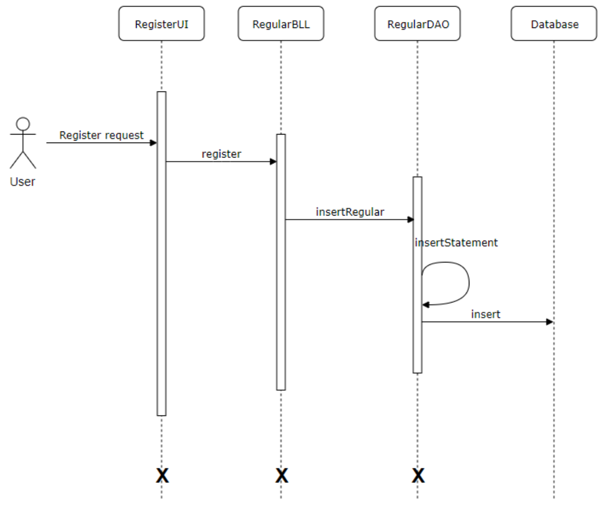
The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain,
independent of the user interface. It directly manages the data, logic and rules of the application.

A view can be any output representation of information, such as a chart or a diagram.
Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

The third part or section, the controller, accepts input and converts it to commands for the model or view.

- The model is responsible for managing the data of the application. It receives user input from the controller.
- The view means presentation of the model in a particular format.
- The controller is responsible for responding to the user input and perform interactions on the data model objects.
- The controller receives the input, optionally validates the input and then passes the input to the model.

4. UML Sequence Diagram



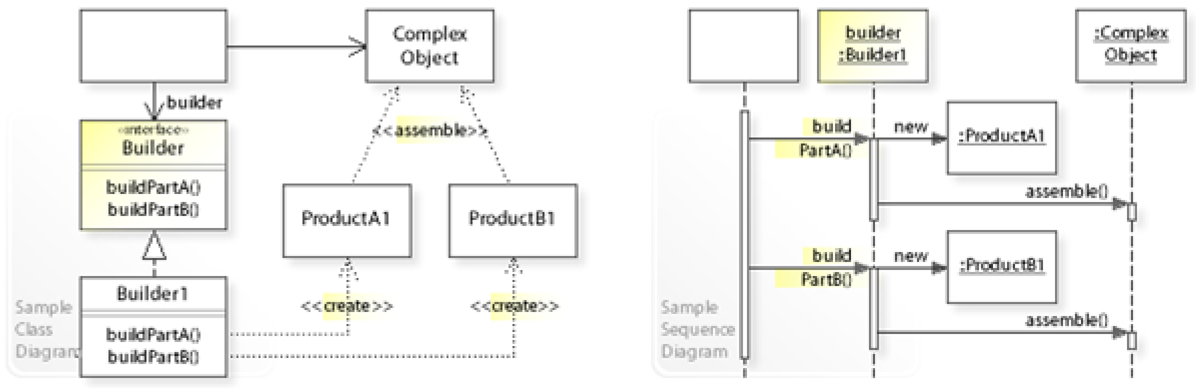
5. Class Design

5.1 Design Patterns Description

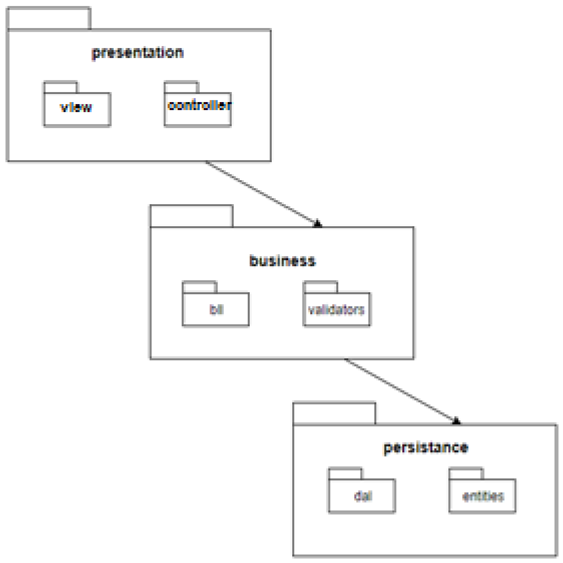
Factory pattern is one of the used patterns, specifically into the persistence layer. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object. In Factory pattern, we create object without exposing the creation logic to the client.

Creating and assembling the parts of a complex object directly within a class is inflexible. It commits the class to creating a particular representation of the complex object and makes it impossible to change the representation later independently from the class.

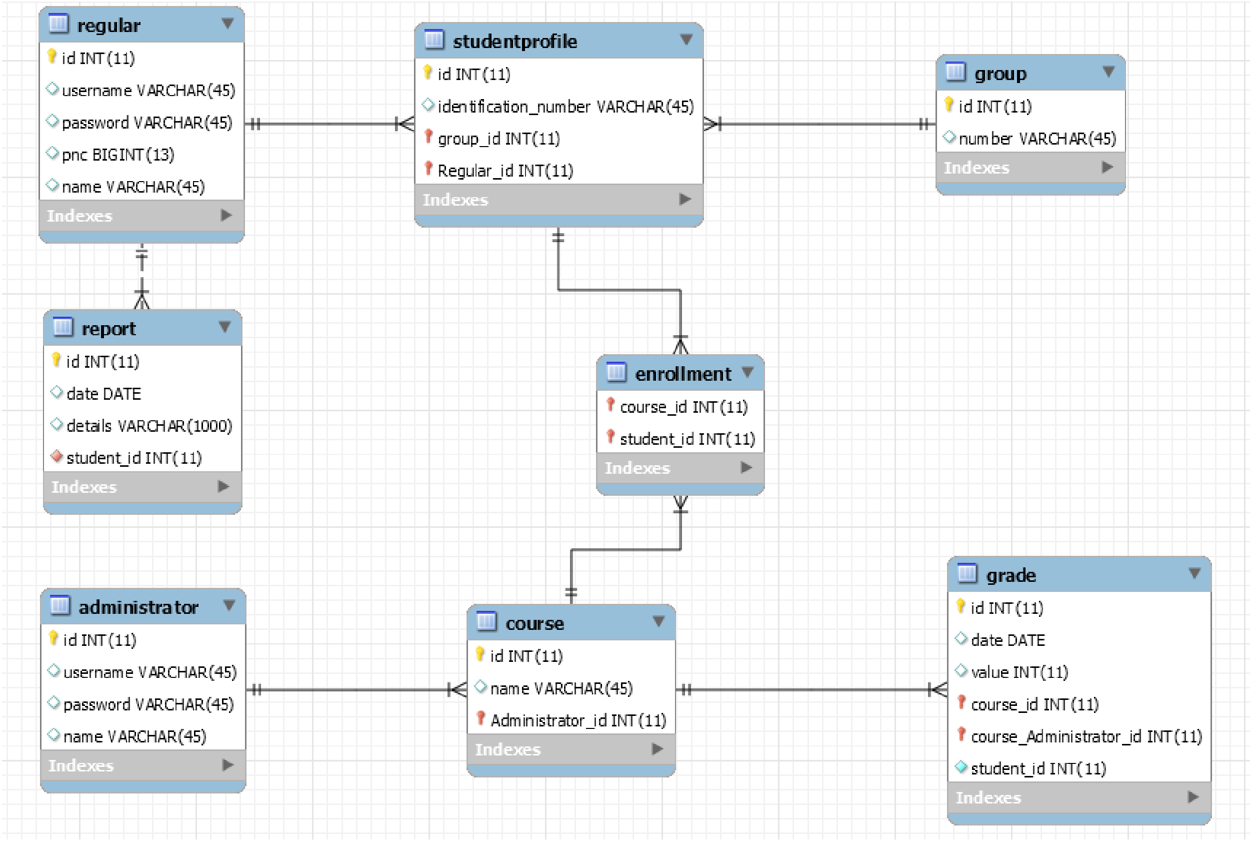
The **Builder design** pattern describes how to solve such problems:
• Encapsulate creating and assembling the parts of a complex object in a separate Builder object.
• A class delegates object creation to a Builder object instead of creating the objects directly.
• A class (the same construction process) can delegate to different Builder objects to create different representations of a complex object.



5.2 Packages Diagram



6. Data Model



7. System Testing

• **Unit testing** is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use.
• **Graphical user interface testing** is the process of testing a product's graphical user interface to ensure it meets its specifications. This is normally done through the use of a variety of test cases. To generate a set of test cases, test designers attempt to cover all the functionality of the system and fully exercise the GUI itself.
• **Usability testing** is a technique used in user-centered interaction design to evaluate a product by testing it on users. Setting up a usability test involves carefully creating a scenario, or realistic situation, wherein the person performs a list of tasks using the product being tested while observers watch and take notes (dynamic verification).

8. Bibliography

https://msdn.microsoft.com/en-us/library/ee658109.aspx
https://www.oreilly.com/ideas/software-architecture-patterns/page/2/layered-architecture
https://en.wikipedia.org/wiki/System_testing