Assignment 2
Analysis and Design Document

Student: Margin Razvan Cristian
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The assignment states that we need to implement an application for managing a department inside our University. The application should have two types of users (the regular user or the student and the administrator or the teacher). Each one of the user have to login with a username and a password in order to access the functionalities of the application.

The student should be able to add/update/delete/view his personal information (name, identity card number, personal numerical code, address, etc.). Furthermore, he can also add/update/delete/view his student profile, which contains information about his group, grades, or lecture enrollments.

The teacher should be able to create/read/update/view student information and generate a report for a certain student activities.

1.2 Functional Requirements

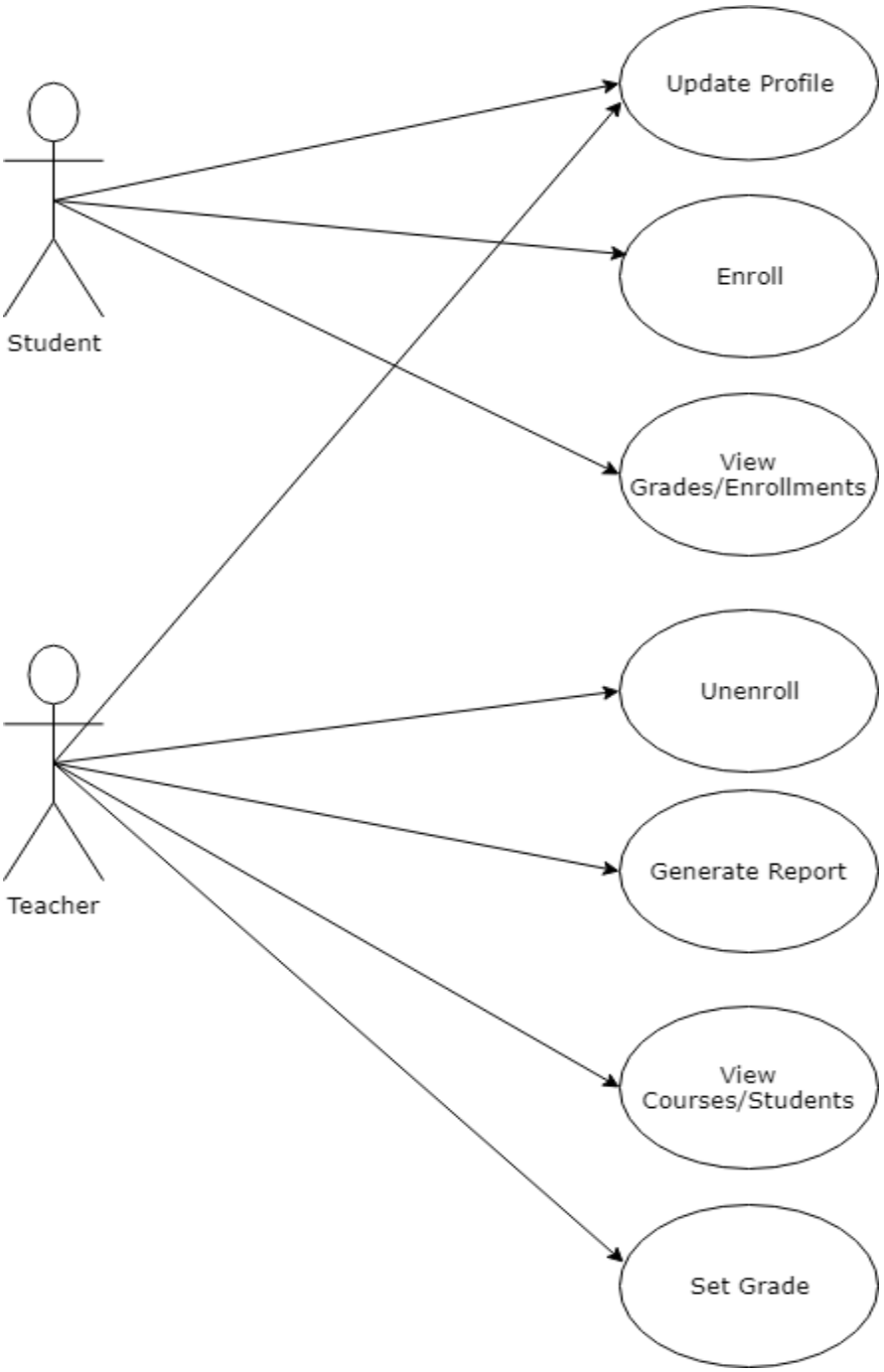
- Student can change its personal information, with results in the database storage
- Student can enroll to the courses available, with results in the database storage
- Student can view its personal information / grades / enrollments
- Teacher can change a student grades / enrollments
- Data is stored in a relational database
- Report is stored in NoSQL database

1.3 Non-functional Requirements

- Security
    - Student data should be protected against attackers so that no personal information is leaked

- Adaptability
    - This application could be used for managing other departments as well, so that it would work properly in other similar environments, with the adjustments made

2. Use-Case Model



Use-Case description format:

Use Case: Student enrollment
Level: user-goal level
Primary actor: Student
Main success scenario: Get Lectures => Pick Lecture => Student Enroll
Extensions: Failure if invalid data inserted or Student is already enrolled for that Lecture

3. System Architectural Design

3.1 Architectural Pattern Description

- Layering

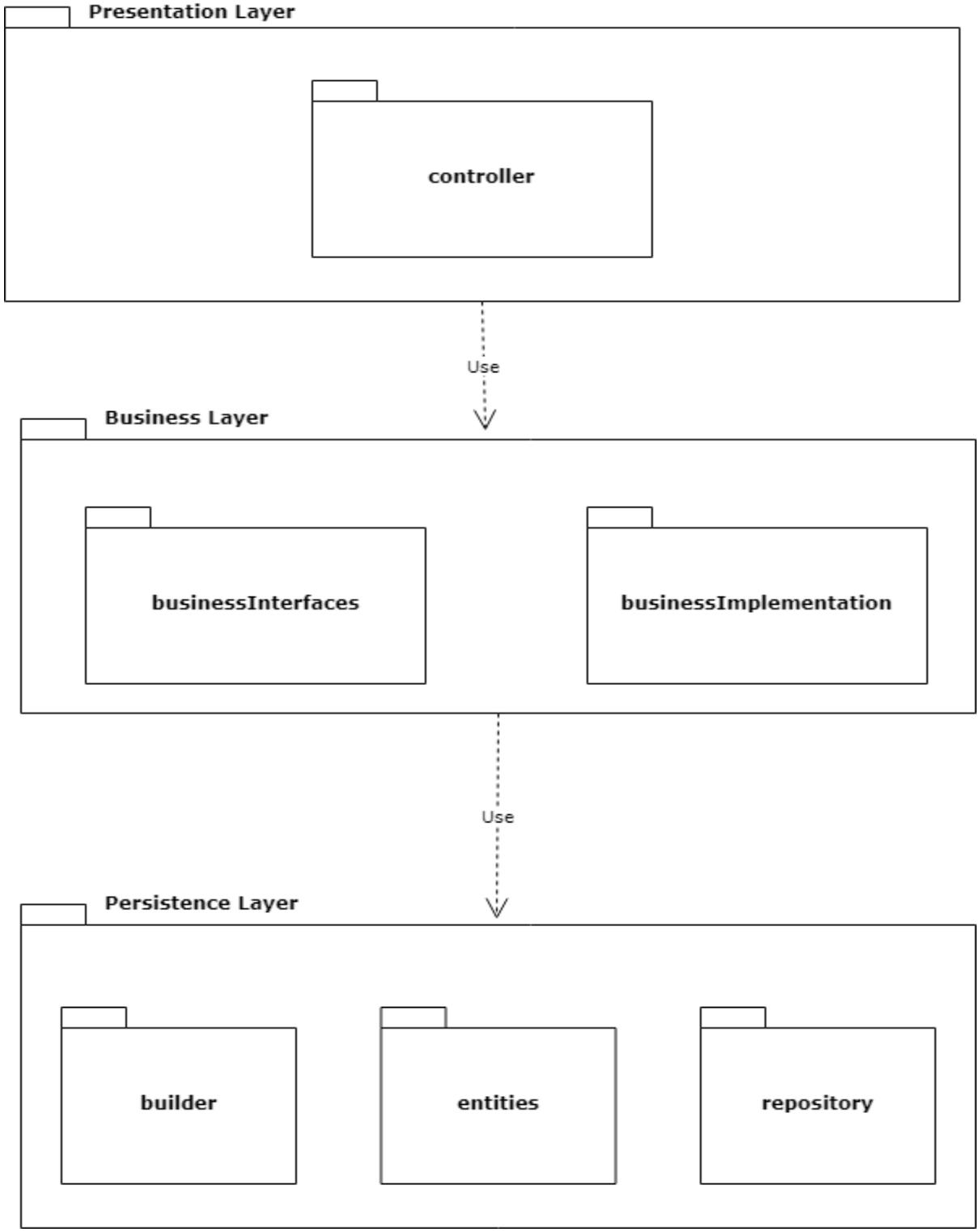
We use Layering architectural pattern in order to break up our software components into groups. Each of the layers are separated based on their primarily functional responsibilities. In our application we will have 3 layers. The Presentation layer will be responsible for displaying the data to the user, containing the user interface components. The Business layer will handle the logic behind the app. The Data layer will perform the communication between our application and the relational database used.

- Model-view-controller

We use the MVC architectural pattern in order to divide our application into 3 connected components. The Model will contain the data representation of our problem domain. The View is the presentation, i.e the User Interface of the application. The controller will be the core of our application, which will handle inputs and convert it into commands for the model or view.

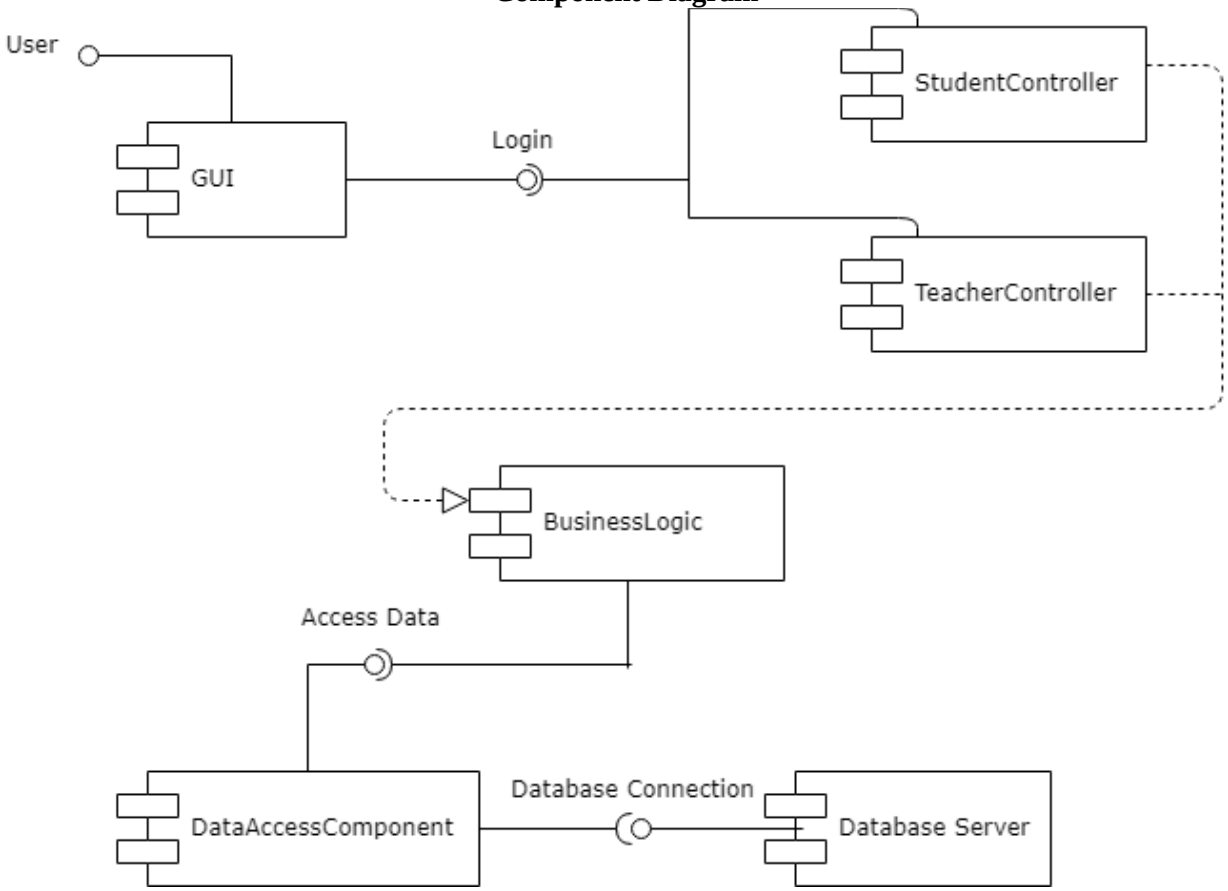
3.2 Diagrams

Package Diagram + Layer Pattern



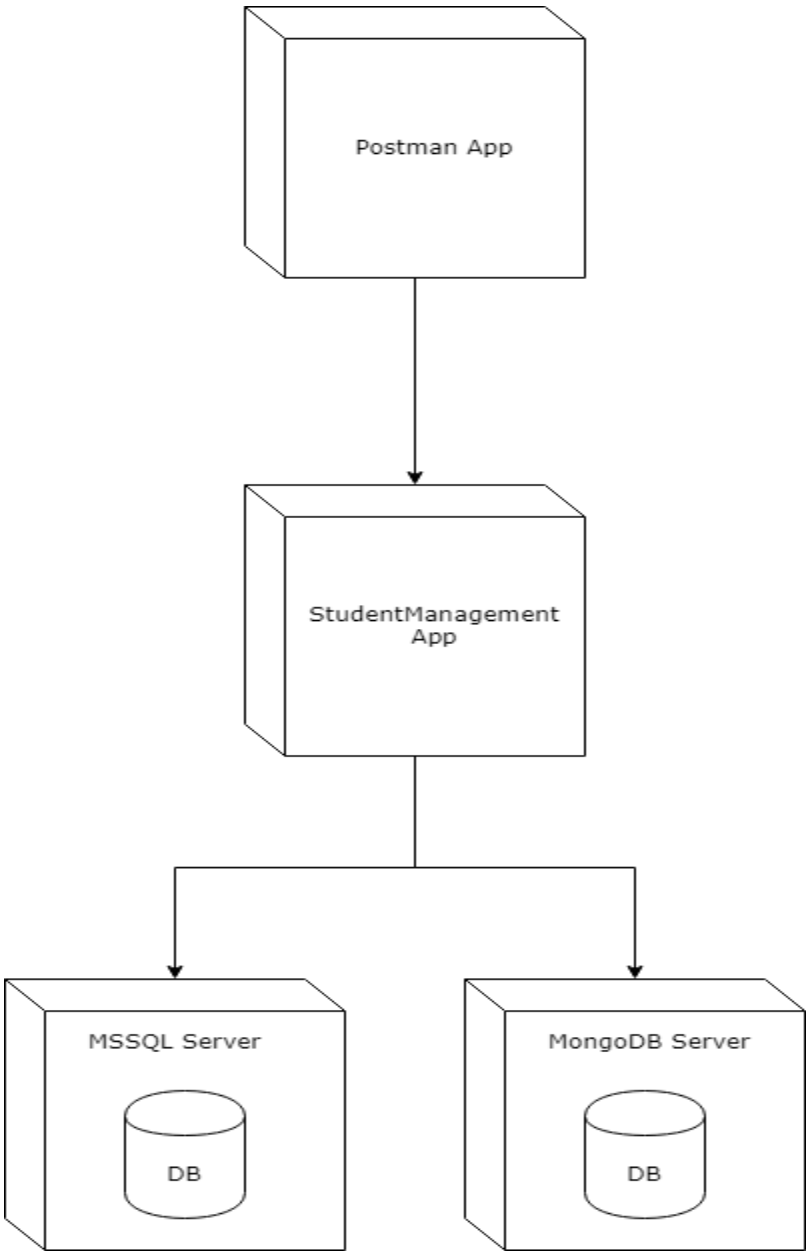
The package diagram shows the structure of our application through packages and dependencies between packages. One can observe that we use a 3 layered architectural pattern, each layer with its own responsibility. The responsibilities of the layers have been explained in the paragraph above.

Component Diagram



The Component Diagram describes the components that are used to make the functionalities of our application. As we can see we have 3 main components: GUI, Controller (Student/Teacher), BusinessLogic and Database. The controllers require the information from the GUI in order to pass the functionalities to the business logic, depending on the logged in user (student or teacher).

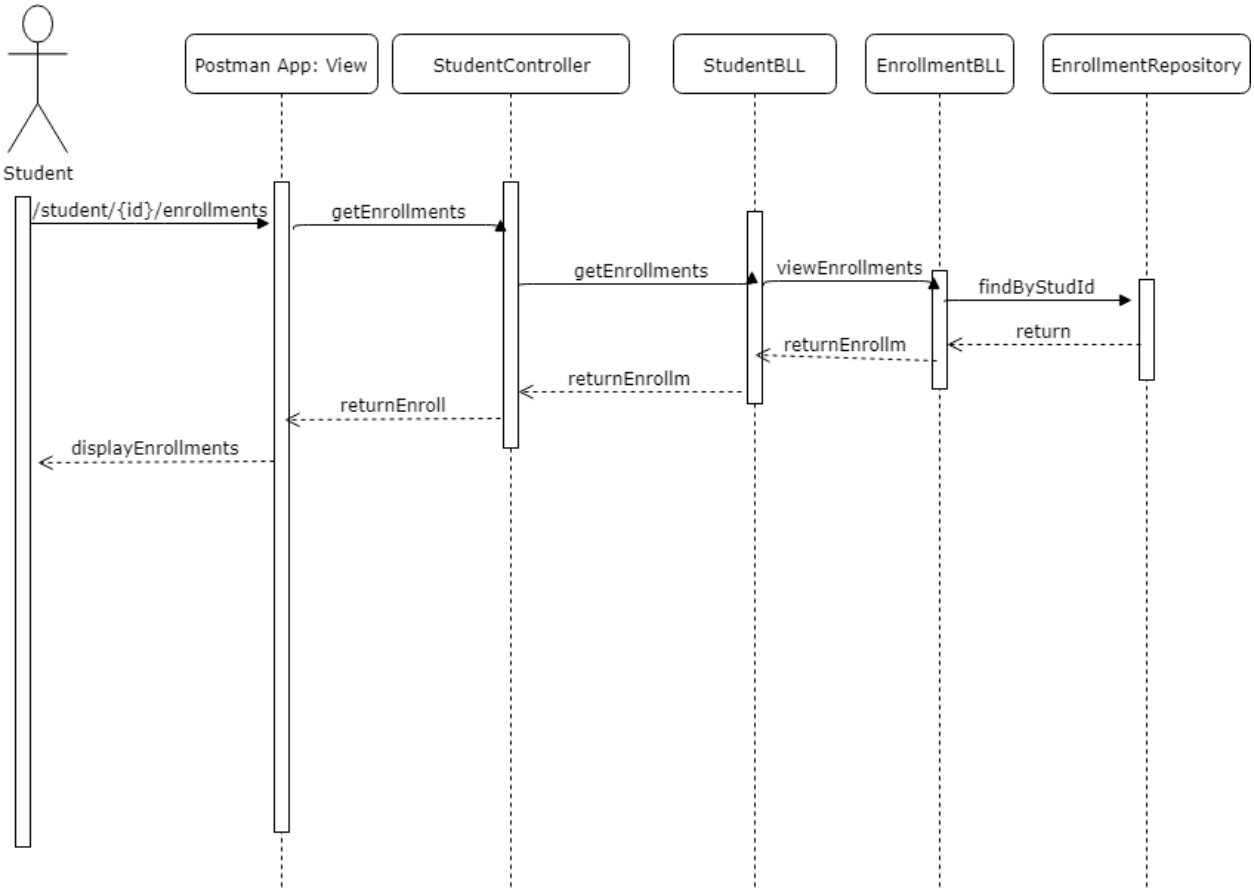
Deployment Diagram



The deployment diagram is a structure diagram through which we see the architecture of our application distributed through artifacts. The artifacts represent physical elements from the real world that are a result of a development process. We have the Postman artifact which works as a GUI, the StudentManagement artifact which controls the whole system, processing the data, and the 2 Database Servers, in which we store the data.

4. UML Sequence Diagrams

UML Sequence Diagram for viewing enrollments



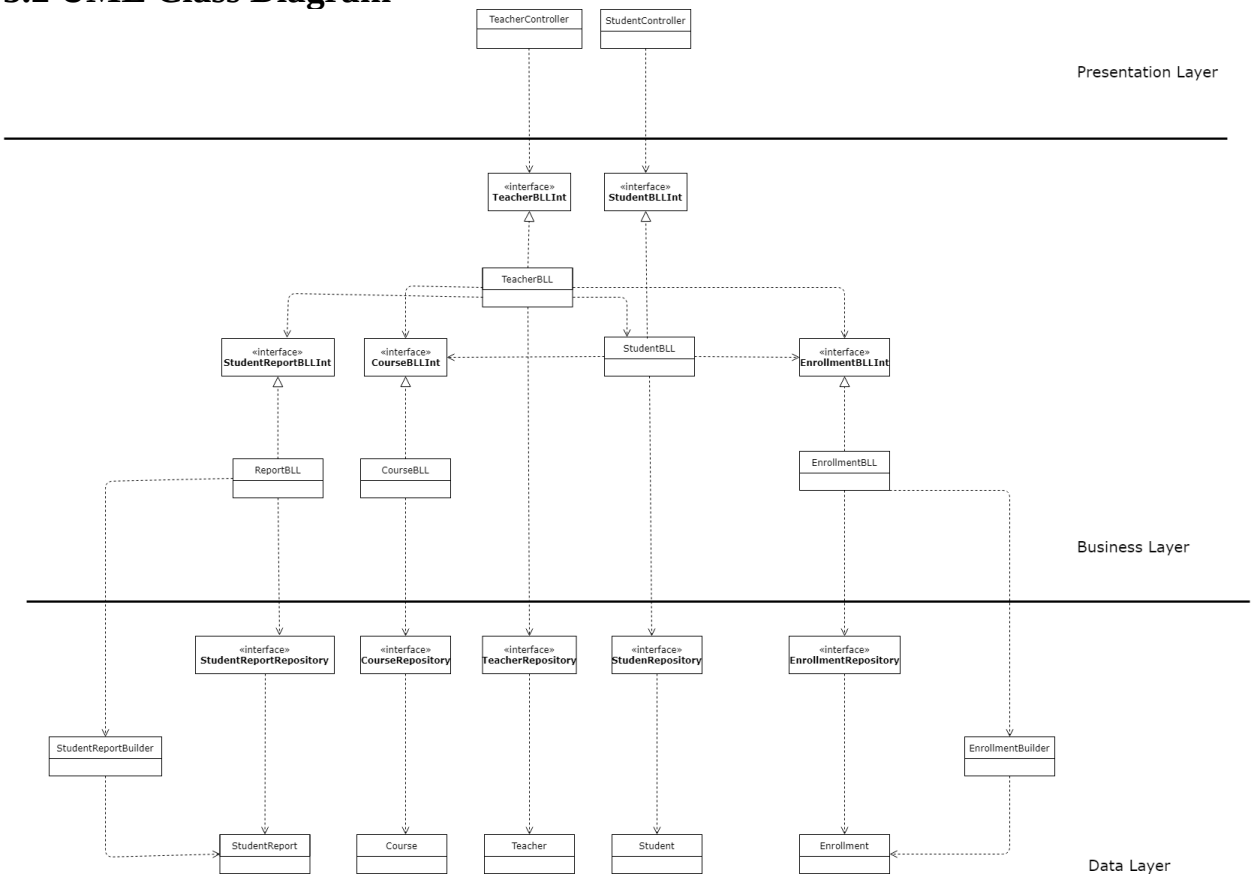
5. Class Design

5.1 Design Patterns Description

Repository Pattern is going to be used in this assignment in order to separate the model and the business logic that acts on the model.

Builder Pattern is going to be used in order to separate the data acquisition logic from the creation of the object itself.

5.2 UML Class Diagram

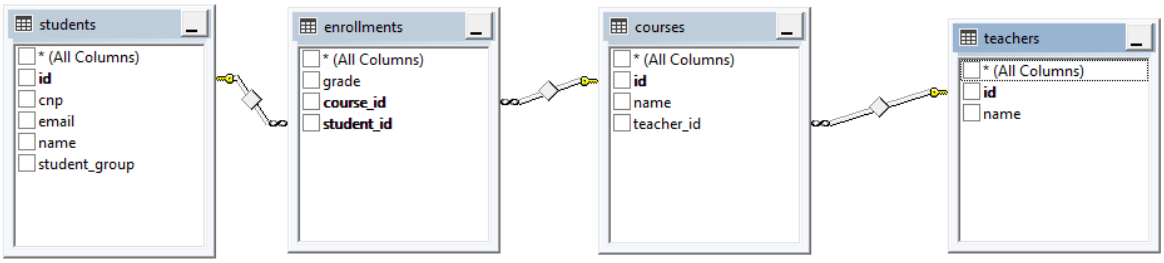


6. Data Model

The data model of this application consists in the main entities that we have also represented in the Class Diagram. We have the Student, the Teacher, the Course, and the Report. Between these models there are relationships that define the behavior of the application

- A student can enroll on several courses
- A teacher can have several courses

Therefore, we can see the data model and the relationships between them in the table below. We must take into account the fact that we have separated the Student and Course tables through an auxiliary table, in order to maintain the relationship between them (many to many).



Finally, we will need also a Report table in which a teacher can store a report regarding a student. This is done through a NoSQL database: MongoDB. In this database we will store the information regarding one particular student.

7. System Testing

In our application we will use Junit Tests in order to test the functionalities of our system.

8. Bibliography

https://www.techrepublic.com/article/commonly-used-architectural-patterns-in-java-applications/
https://msdn.microsoft.com/en-us/library/ee658109.aspx
https://www.lucidchart.com/pages/uml-component-diagram
https://www.lucidchart.com/pages/uml-deployment-diagram
https://www.thoughts-on-java.org/ultimate-guide-association-mappings-jpa-hibernate/
https://www.tutorialspoint.com/mongodb/index.htm