TUCN student management system

Student: Ács Dávid

**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 4

3. System Architectural Design 4

4. UML Sequence Diagrams 6

5. Class Design 8

6. Data Model 9

7. System Testing 10

8. Bibliography 11

1. Requirements Analysis

# Assignment Specification

A student management system, where information is stored and managed, related to students, courses, exams, enrollments. Teachers and administrators can view reports of student’s activities. Access to the system is only available if a correct username and password is provided.

# Functional Requirements

* + 1. **Login:** Users can log in if they provide the correct username, password combination.
    2. **User information:** Users can enter, modify, view information about themselves: such as name, ICN, PNC, address.
    3. **Student information:** Students can enter, modify, view, delete, view student profile: ID, group, enrolments, grades.
    4. **Process class enrollment:** Enroll students, grade students, view exams.
    5. **Administrators can manage student’s info:** administrators can create, read, update and delete all the information of students.
    6. **Reports for teachers:** Create a report, assessing a performance of a student for a specific period.

# Non-functional Requirements

* + 1. **Portability:** The system must be able to run on all major operating systems: Windows, Linux.
    2. **Response time:** The system must respond to user input in 1s in 99% of cases.
    3. **Maintainability:** Mean time to repair should be as low as 40 hours.
    4. **Readability:** The formatting of the code should reflect the logical structure of code.
    5. **Performance:** Performance (throughput) is not a priority.

2. Use-Case Model



Use-case diagram detailing the needed functionality of the system.

Use case: Enroll in course.

Level: user-goal level.

Primary actor: Student.

Main success scenario:

1. The student completes the login process (prerequisite).
2. The student selects the course, he/she wishes to attend and presses the enroll button.
3. If enrollment is possible, the student will be enrolled in the course.

Extensions: the student may not have permission to enroll in the course, so he/she will be presented with an error message.

3. System Architectural Design

**3.1 Architectural Pattern Description**

A layered architecture was choses, supplemented with **MVP.**

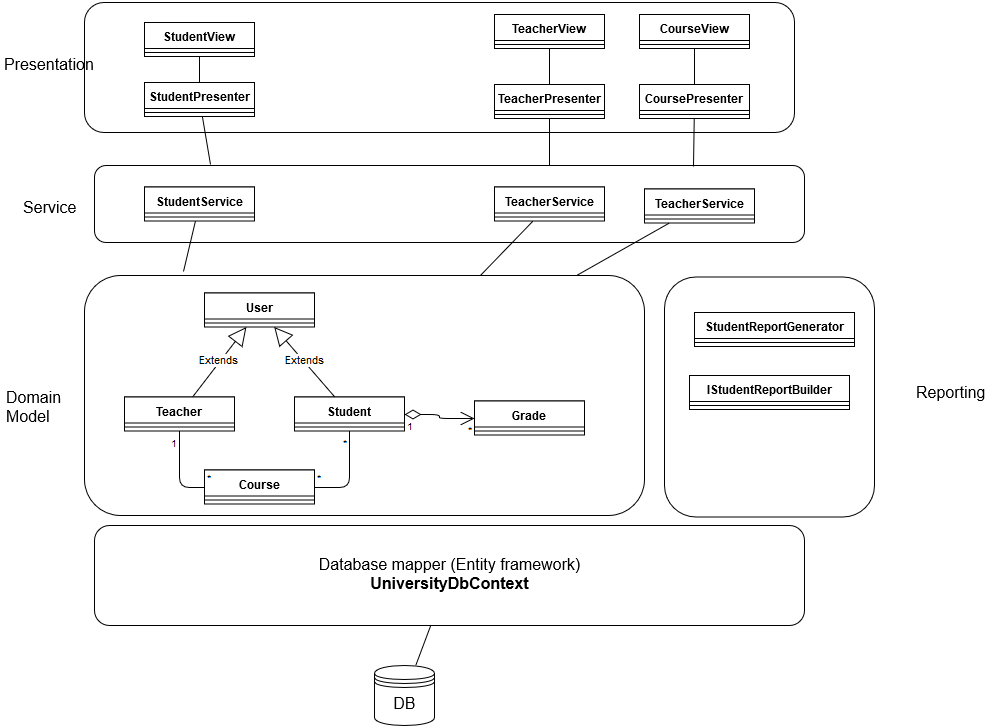
In Model-View-Presenter (MVP) the view is separate from the model since the presenter serves as a mediator between the two of these. Presenter handles the event coming from the view and acts on the model, if the model changes the presenter updates the view.

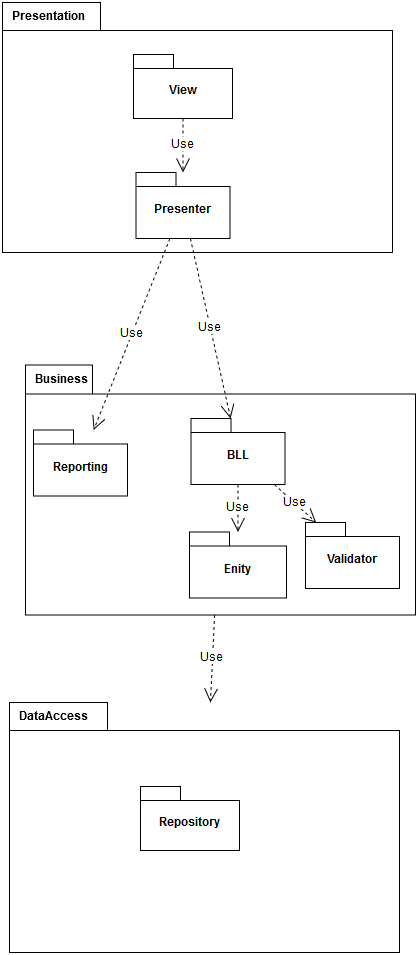
* 1. **Diagrams**

As you can see, the presentation layer contains the Views and Presenters, while the Domain Model is exposed to the Presenters through the Services.

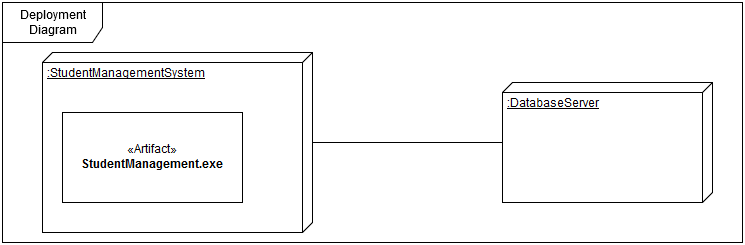
For the business logic the **Domain Model** was chosen.

For mapping the Domain Model to the database Entity Framework will be used, with the Code-First approach.



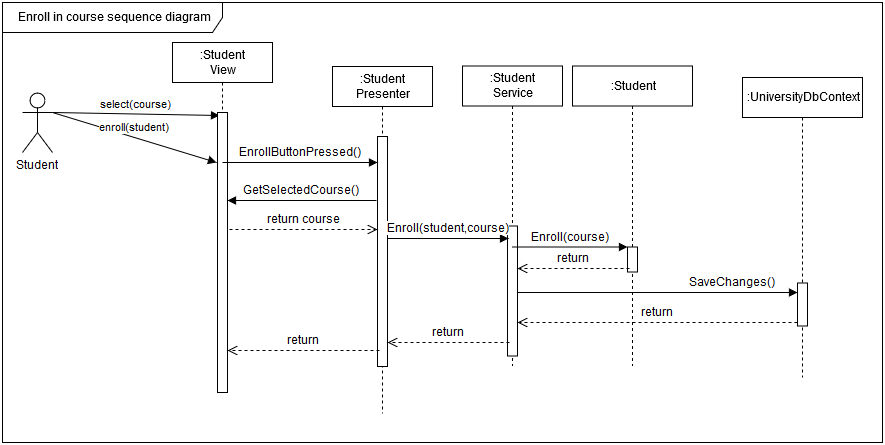


The component diagram is really similar to that of a layered system.



Since the application is run on a single tier, the deployment diagram consists of two servers: The main server where the application will run and the Database server where the database will run.

4. UML Sequence Diagrams

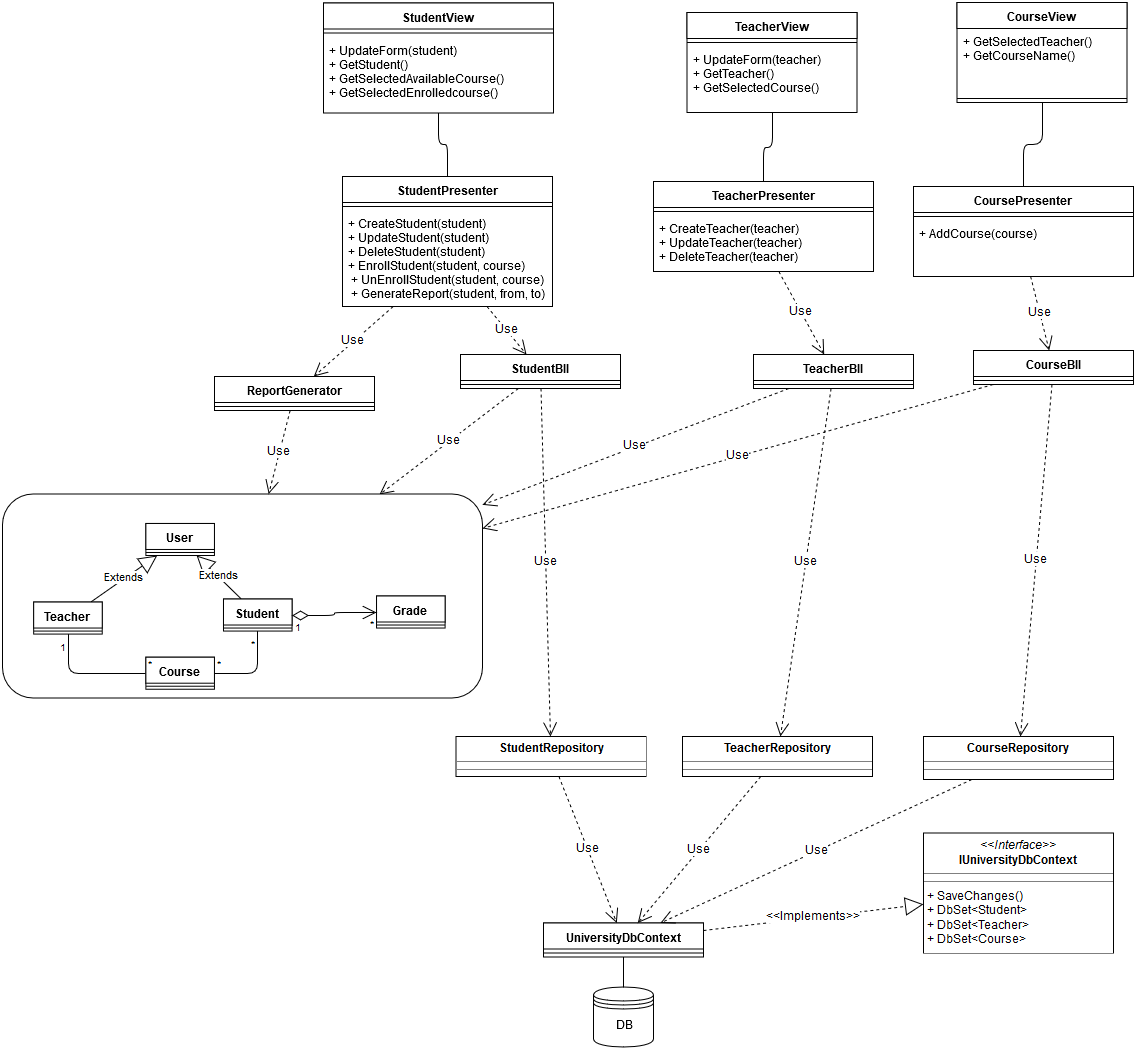


5. Class Design

**5.1 Design Patterns Description**

Repository design pattern for the database, Factory method for creating the required report generator. Builder used inside the report generator for creating different representations of the same report.

**5.2 UML Class Diagram**



6. Data Model



The data model (entity relationship model) is presented above. Please note that this database design was not designed by me, it was created by entity framework, using the code first approach.

7. System Testing

Unit testing and system testing will be performed as the system is implemented and once the system has been completely built. The unit testing method chosen is mocking the objects.

8. Bibliography

Inversion of Control with Microsoft Unity:

<https://msdn.microsoft.com/en-us/library/dn178470(v=pandp.30).aspx>