ASSIGNMENT 2

Analysis and Design Document

Student: Andrei Branga

**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 assignment consists of implementing an MVC Web application for the management of students in the CS Department at TUCN.

# Functional Requirements

The application should have two types of users (student and teacher/administrator user) which have to provide a username and a password in order to use the application.

The regular user can perform the following operations:

- Add/update/view client information (name, identity card number, personal numerical

code, address, etc.).

- Create/update/delete/view student profile (account information: identification number,

group, enrolments, grades).

- Process class enrolment (enroll, exams, grades).

The administrator user can perform the following operations:

- CRUD on students information.

- Generate reports for a particular period containing the activities performed by a student.

In the development I will use the ASP .NET MVC framework together with Entity Framework.

Other requirements:

* Use MySql as well as using a NoSQL DB for storing the reports
* use the ORM One-to-Many / Many-to-Many / ... multiplicity mechanisms
* use 1 DP from FactoryMethod / Builder
* use Dependency Injection or create Integration Tests for 1 flow

# Non-functional Requirements

* Usability – the application should be easy to use for the end-user
* Testability – the application should be very well tested

2. Use-Case Model

Use case: Generate marks for the past Exam session

Level: user-goal level

Primary actor: Supervisor

Main success scenario:

1. The supervisor is successfully logged in
2. The supervisor selects an active year of study
3. The supervisor selects a group
4. The supervisor selects a student
5. The supervisor generates the needed report

Extensions: The student was expelled thus not found the usual way

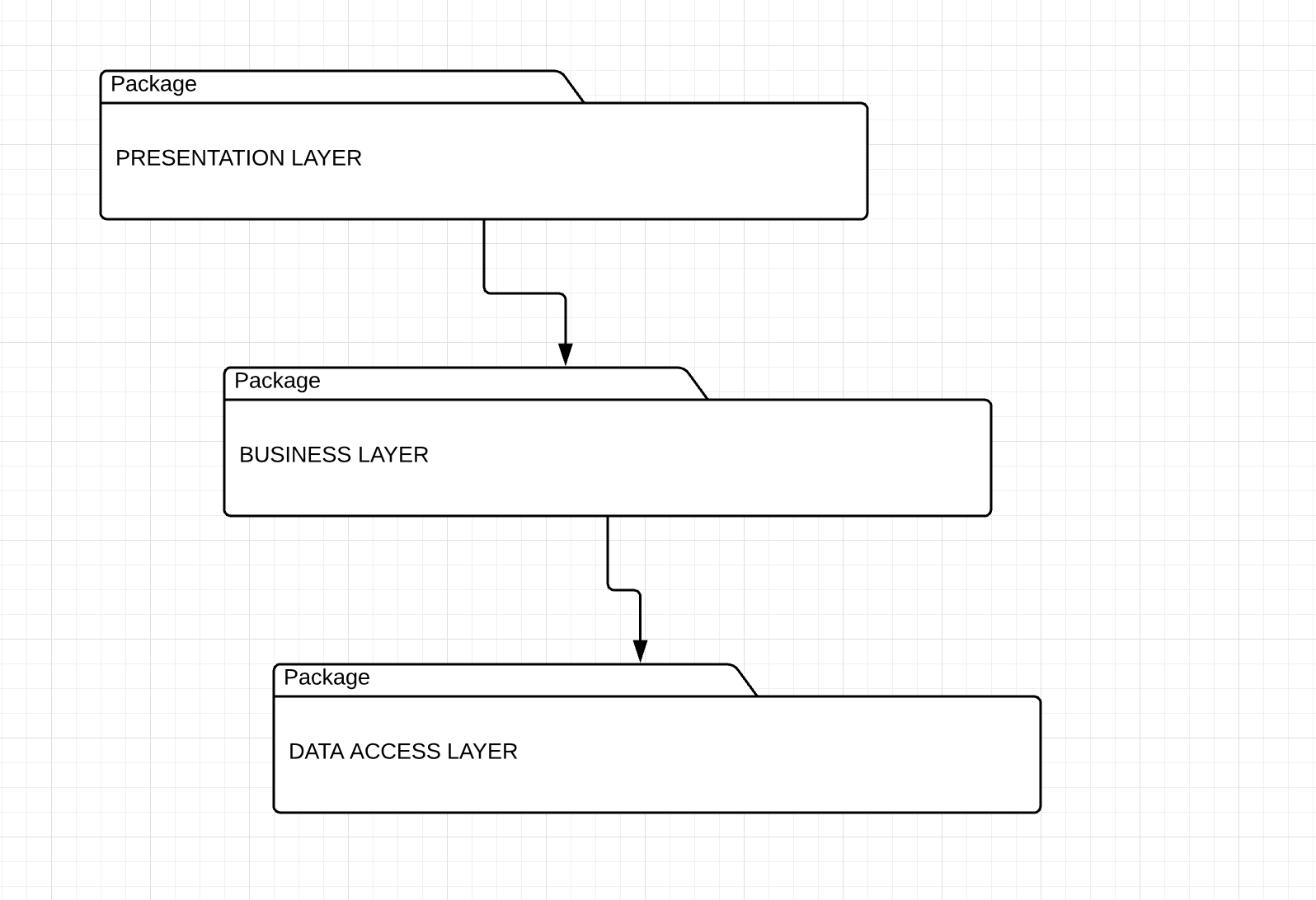
3. System Architectural Design

**3.1 Architectural Pattern Description**

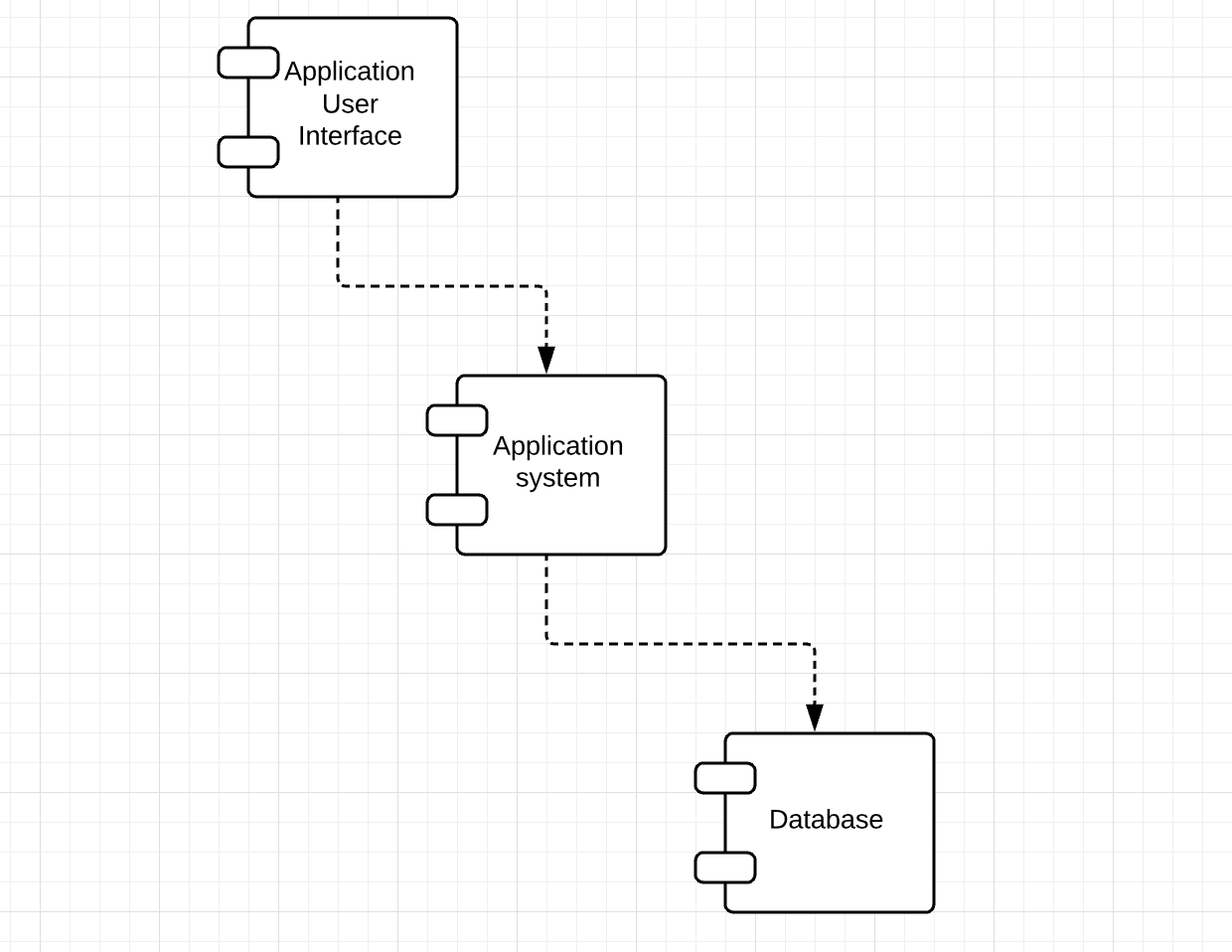
For implementing the application, I will use the Layered Architecture Pattern. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic).

**3.2 Diagrams**

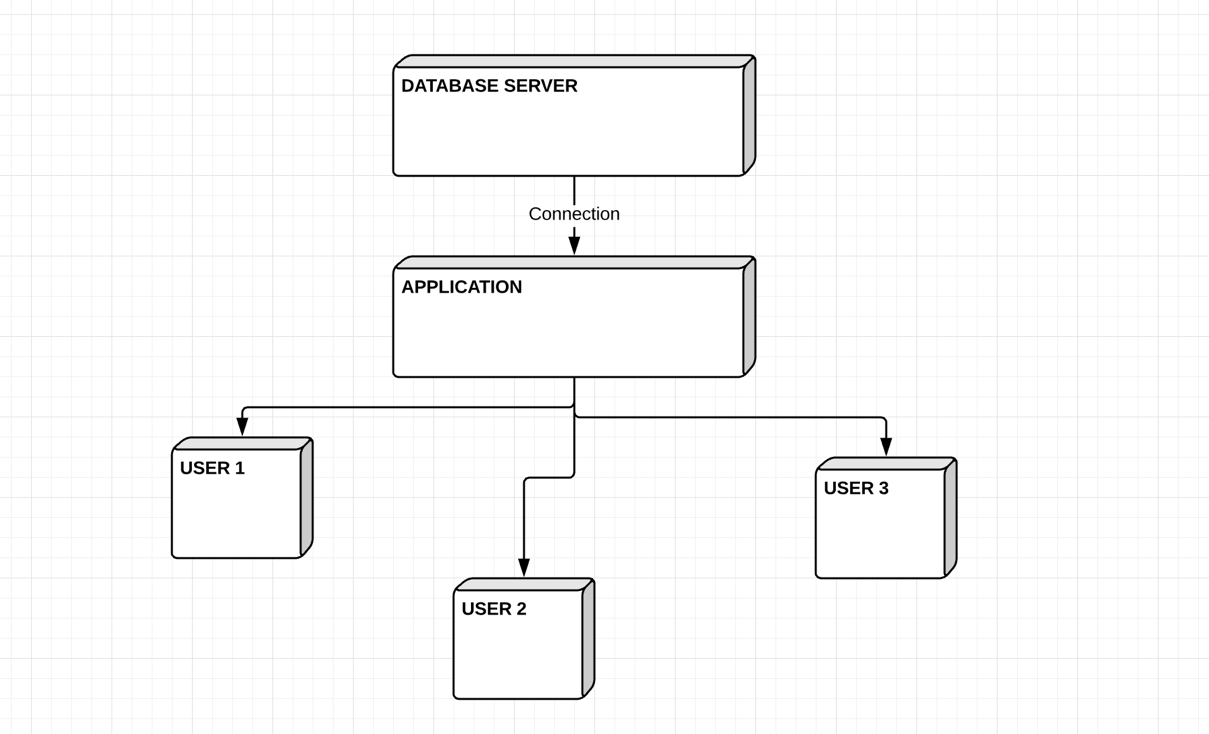
**Package Diagram:**

****

**Component Diagram:**

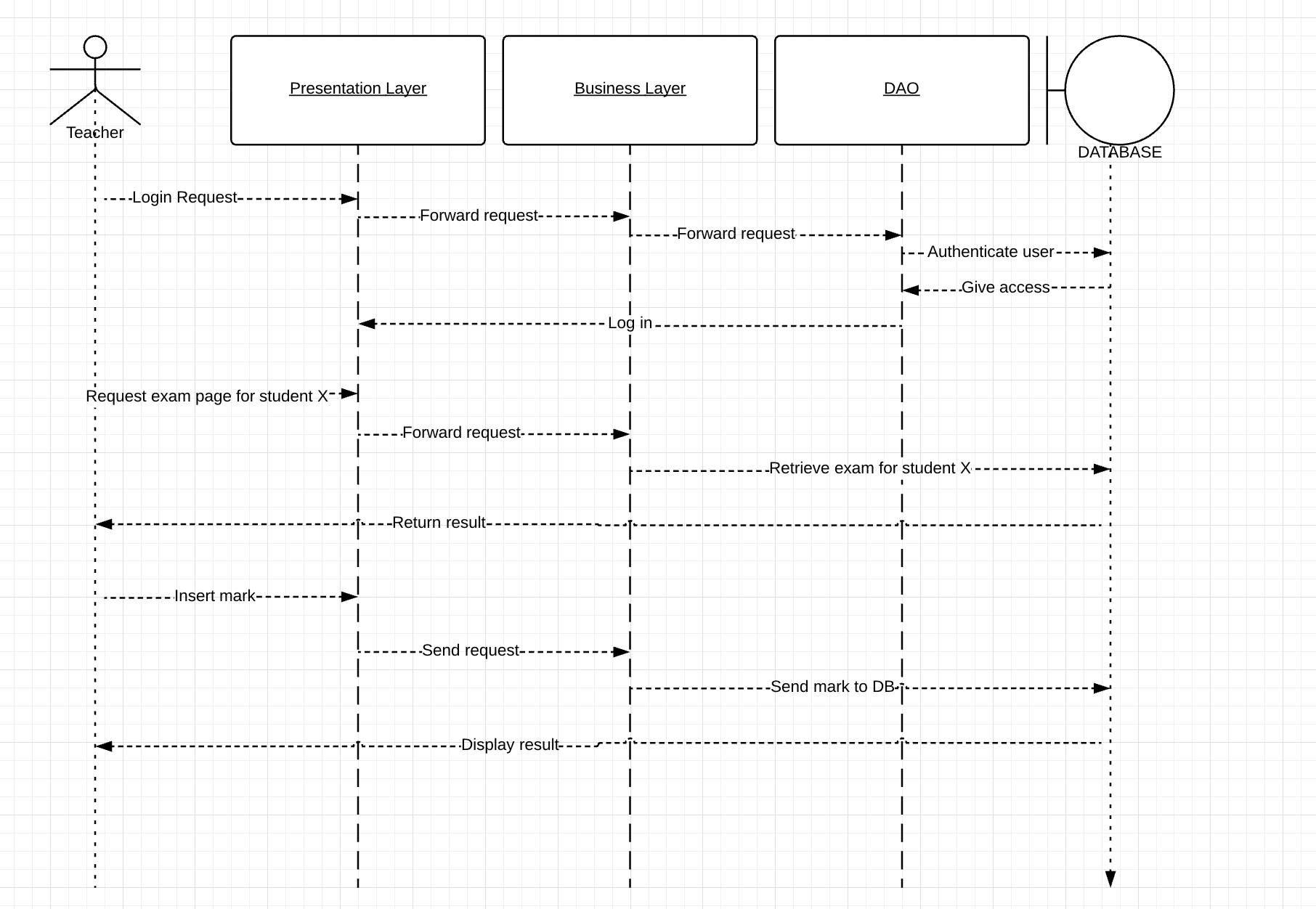
****

**Deployment diagram:**

****

4. UML Sequence Diagrams

Teacher adds a mark for a student:



5. Class Design

**5.1 Design Patterns Description**

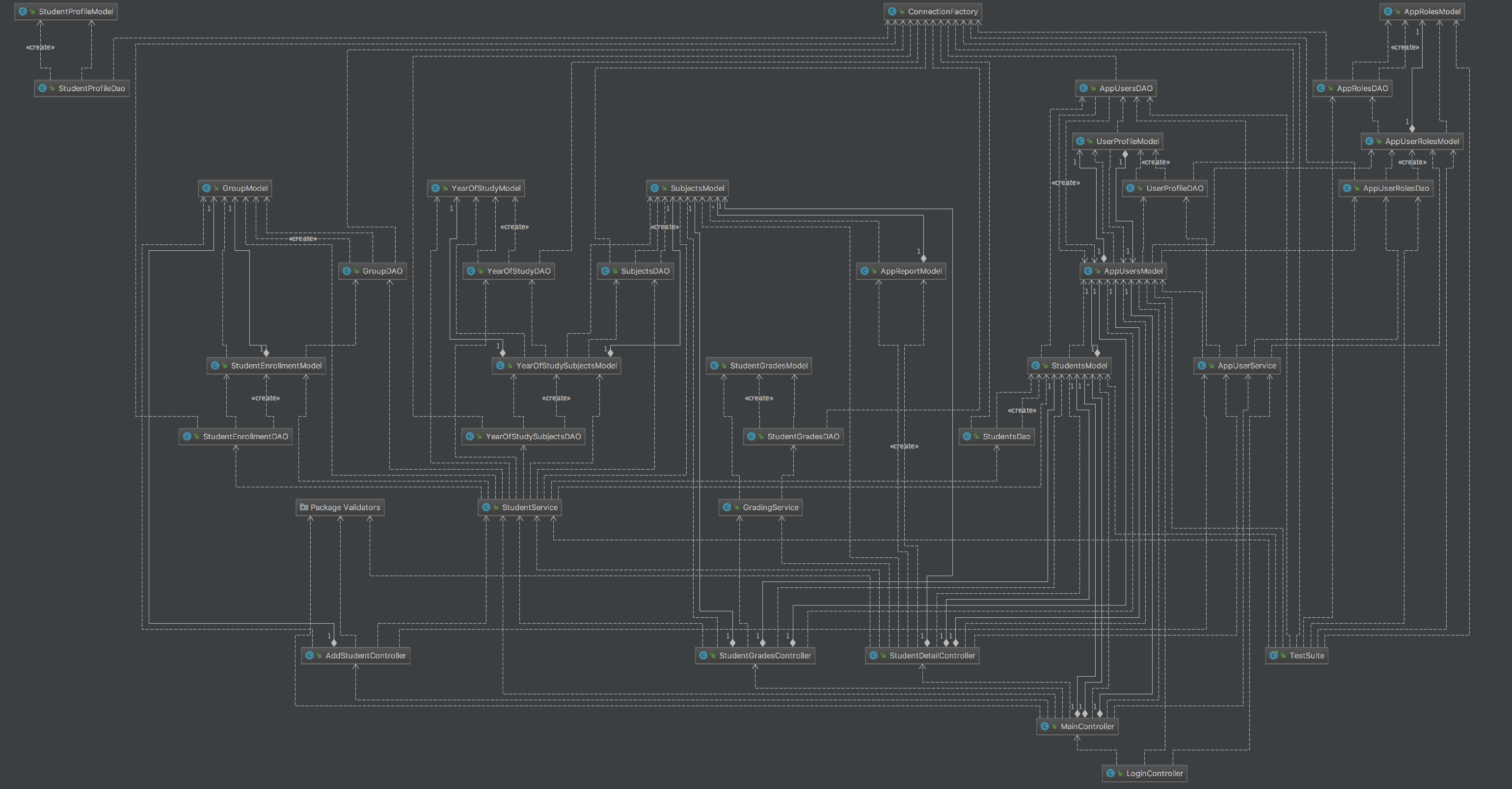
Each layer of the layered architecture pattern has a specific role and responsibility within the application. For example, a presentation layer would be responsible for handling all user interface and browser communication logic, whereas a business layer would be responsible for executing specific business rules associated with the request. Each layer in the architecture forms an abstraction around the work that needs to be done to satisfy a particular business request.

**MVC DESIGN PATTERN:**

View renders the data from the Model in response to the request made to the model by controlled events made by user interaction.

Model View Controller is a design approach to separate the application object model from GUI, originally invented around 80s. Then later on it has become a widely accepted common design pattern. The main objective behind this pattern is to decouple the view of the data (presentation layer) from the actual data processing so that the same model can be used for various views. This is achieved by using three different types of objects that interact with each other in loosely coupled manner with their discreet set of tasks.

**5.2 UML Class Diagram**



6. Data Model

An ORM class model is part of a data access layer first and maps the data object’s properties to a class(entity). If any conflict of interests arises between DAL and BLL, we will make changes to the BLL entities.

I will use the Entity Framework to model the data access layer:

Entity Framework (EF) is an object-relational mapper that enables .NET developers to work with relational data using domain-specific objects. It eliminates the need for most of the data-access code that developers usually need to write.

Entity Framework allows you to create a model by writing code or using boxes and lines in the EF Designer. Both of these approaches can be used to target an existing database or create a new database. This short video explains the differences and how to find the one that is right for you.

7. System Testing

The application will be covered by UnitTests.

Check that your code is working as expected by creating and running unit tests. It’s called unit testing because you break down the functionality of your program into discrete testable behaviors that you can test as individual units. Visual Studio Test Explorer provides a flexible and efficient way to run your unit tests and view their results in Visual Studio. Visual Studio installs the Microsoft unit testing frameworks for managed and native code. Use a unit testing framework to create unit tests, run them, and report the results of these tests. Rerun unit tests when you make changes to test that your code is still working correctly. When you use Visual Studio Enterprise, you can run tests automatically after every build.

Unit testing has the greatest effect on the quality of your code when it’s an integral part of your software development workflow. As soon as you write a function or other block of application code, create unit tests that verify the behavior of the code in response to standard, boundary, and incorrect cases of input data, and that check any explicit or implicit assumptions made by the code. With test driven development, you create the unit tests before you write the code, so you use the unit tests as both design documentation and functional specifications.

8. Bibliography

https://msdn.microsoft.com/en-us/library/hh694602.aspx