ASSIGNMENT 1

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 a Java 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.

# 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

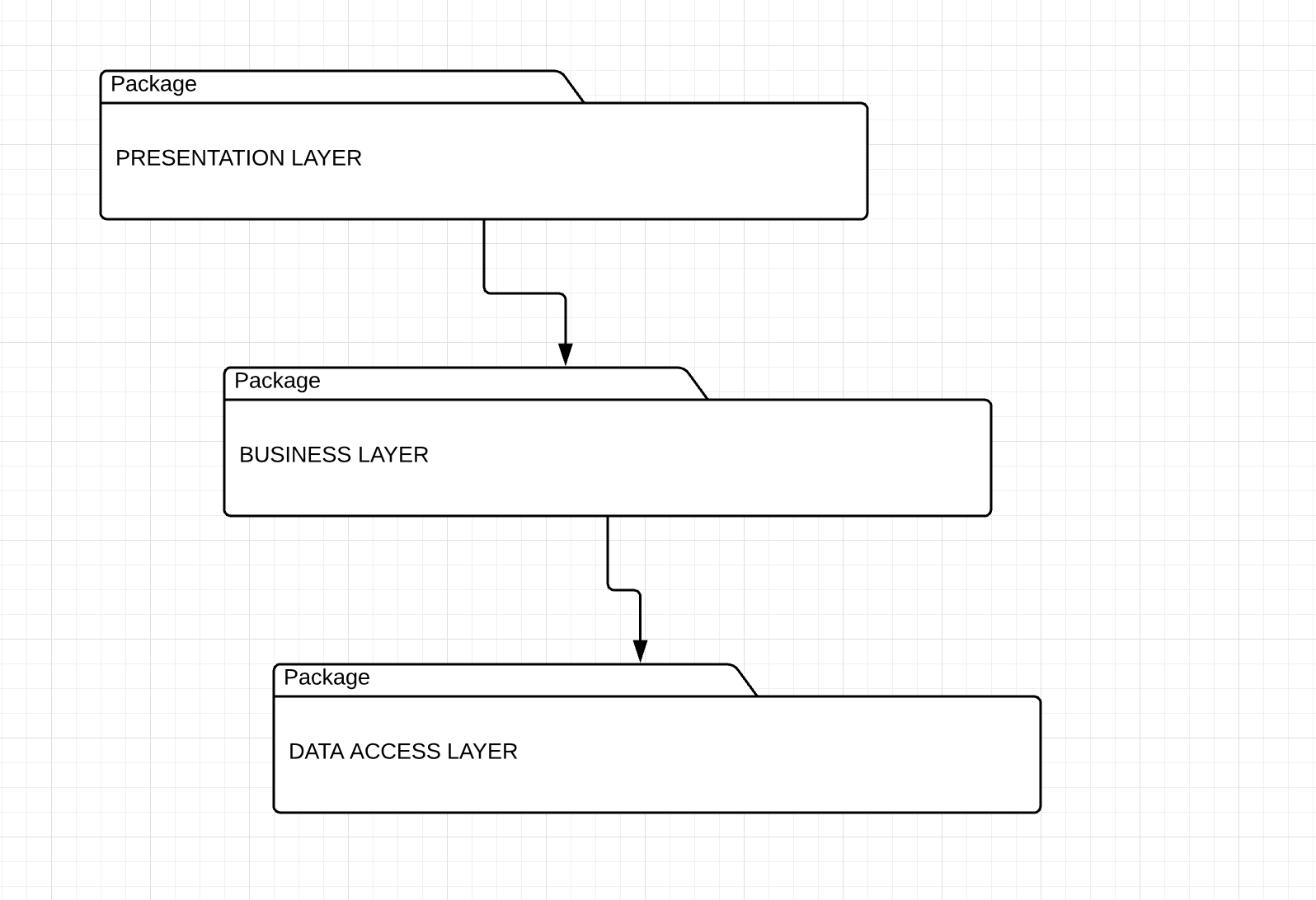
3. System Architectural Design

**3.1 Architectural Pattern Description**

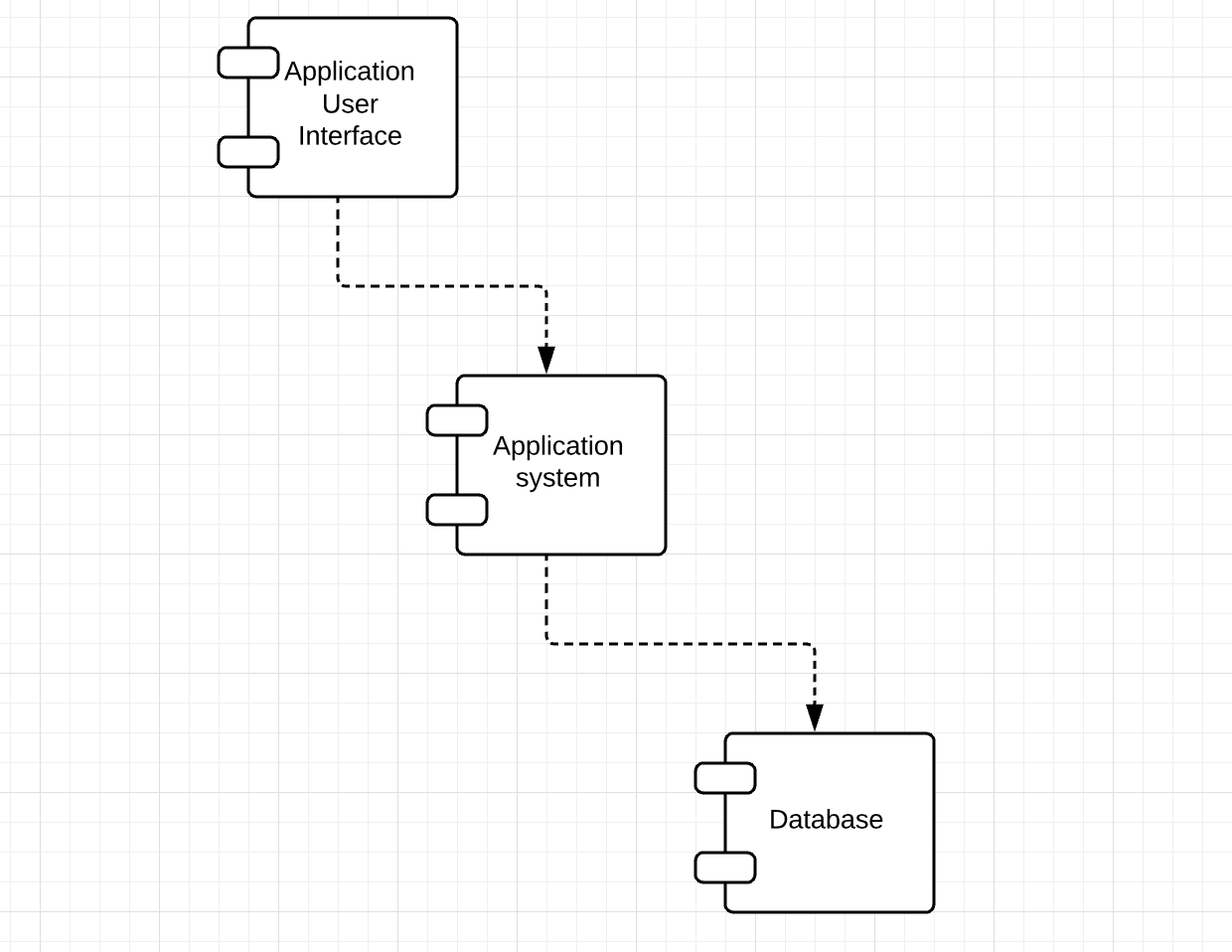
For implementing the application, I will use the Layered Architecture Pattern. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic).

**3.2 Diagrams**

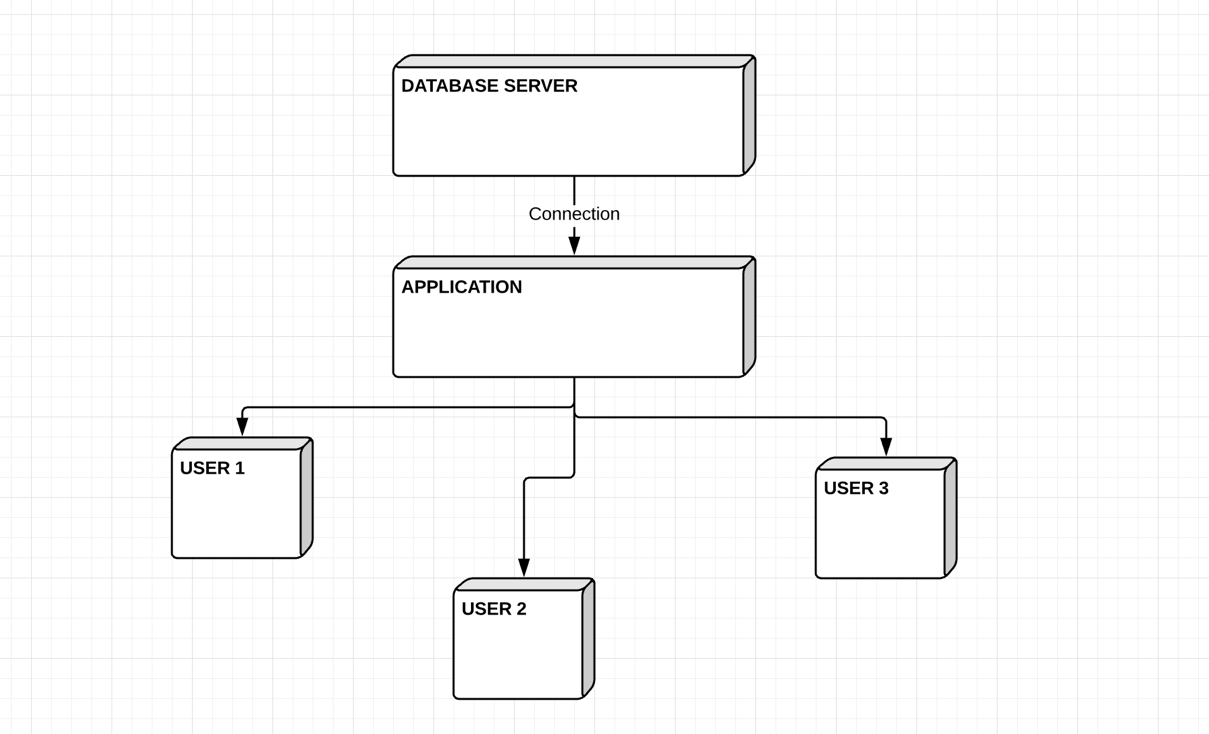
**Package Diagram:**

****

**Component Diagram:**

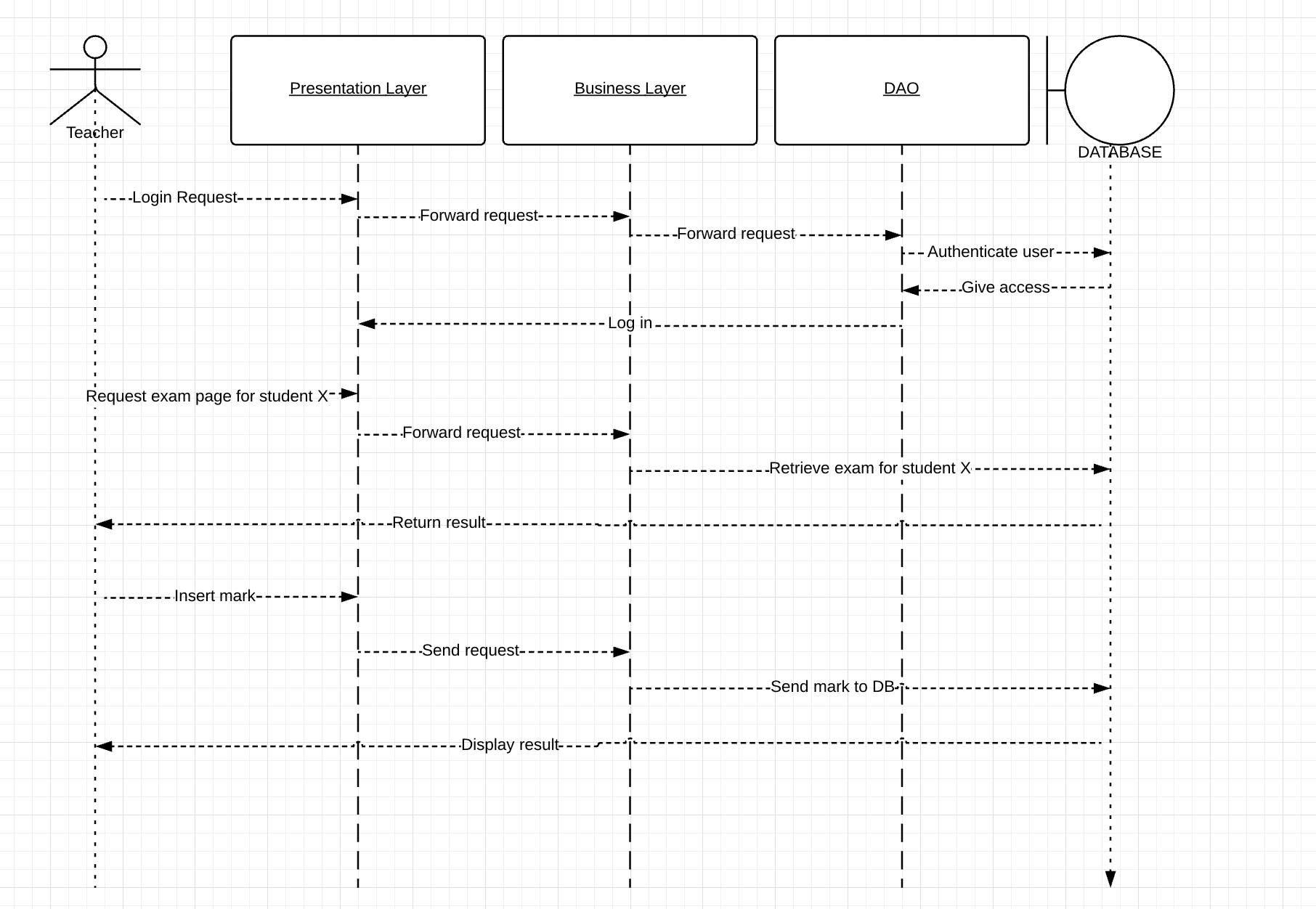
****

**Deployment diagram:**

****

4. UML Sequence Diagrams

Teacher adds a mark for a student:

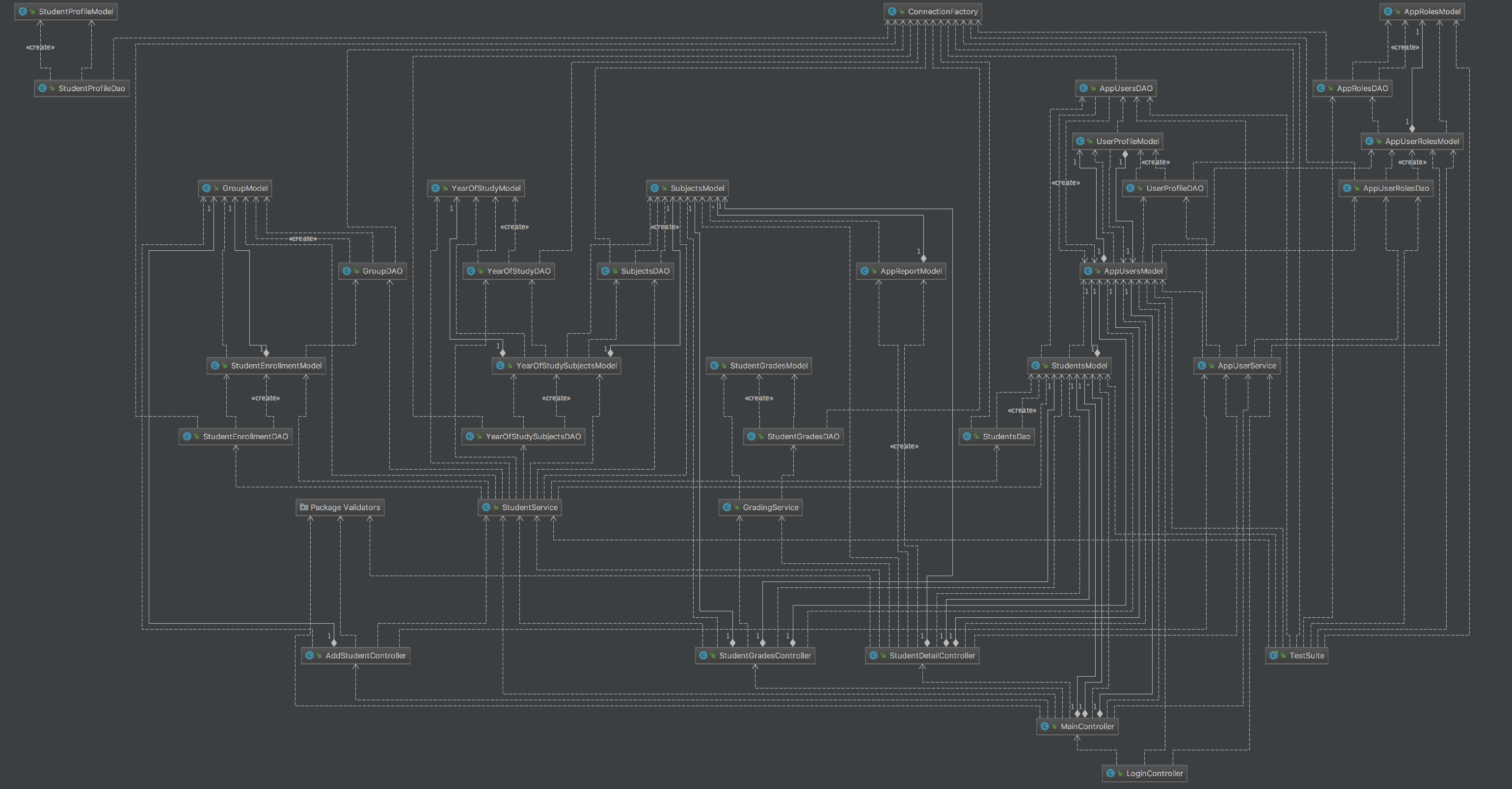


5. Class Design

**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.

**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.

7. System Testing

The application will be covered by Junit Tests.

A JUnit test is a method contained in a class which is only used for testing. This is called a Test class. To define that a certain method is a test method, annotate it with the @Test annotation.

This method executes the code under test. You use an assert method, provided by JUnit or another assert framework, to check an expected result versus the actual result. These method calls are typically called asserts or assert statements.

8. Bibliography

<http://www.vogella.com/tutorials/JUnit/article.html>

https://www.safaribooksonline.com/library/view/software-architecture-patterns/9781491971437/ch01.html