

# **Student Management System Analysis and Design Document**

**Student: Andreea Ionutas  
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	5
5. Class Design	7
6. Data Model	9
7. System Testing	9
8. Bibliography	9

# 1. Requirements Analysis

## 1.1 Assignment Specification

The application will serve as a management tool for the CS Department at UTCN. It can be used by both students and teacher/administrator. Having a user-friendly graphical interface and being connected to a database, the application can perform specific actions according to each type of user.

## 1.2 Functional Requirements

There are two types of users: student or teacher/administrator.

Operations that can be performed by a regular user:

- Add/update/view client information (name, identity card number, personal numerical code, address, etc.)
- Create/Update/Delete/View student profile (account information)
- Process class enrolment

Operations that can be performed by the administrator user:

- CRUD on students information
- Generate reports for a particular period containing the activities performed by a student.

## 1.3 Non-functional Requirements

- Adaptability: the system should be able to adapt itself fast and efficiently to any type of changes.
- Stability: most of the objects will be stable over time and will not need changes
- Reusability: the system can be used in various platforms/contexts.

# 2. Use-Case Model

**Use case:** Generate report

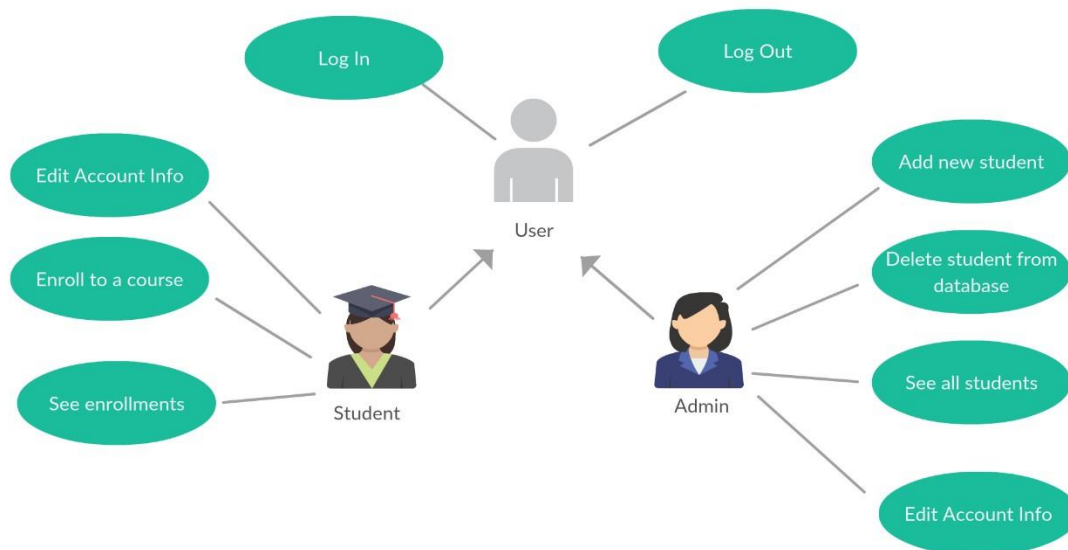
**Level:** User Goal Level

**Primary actor:** Administrator

**Main success scenario:**

- Log in into application
- Select a student
- Click on generate report
- Visualize a report containing the student's performances

**Extensions:** Fail to log in into application (the login id or password were introduced incorrectly)



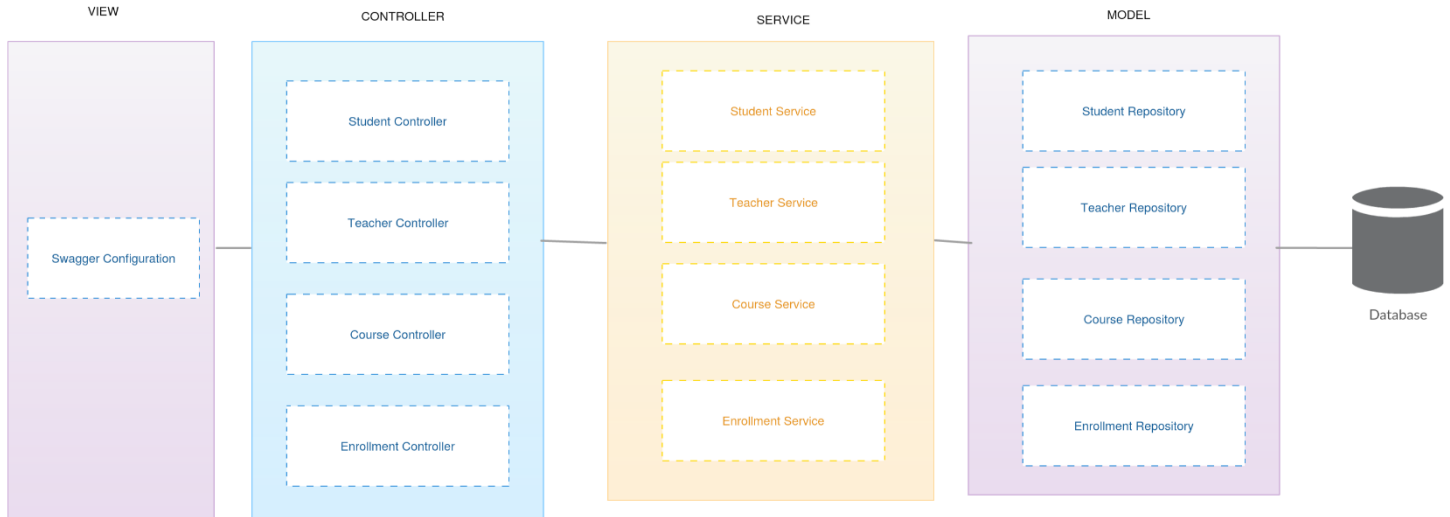
## 3. System Architectural Design

### 3.1 Architectural Pattern Description

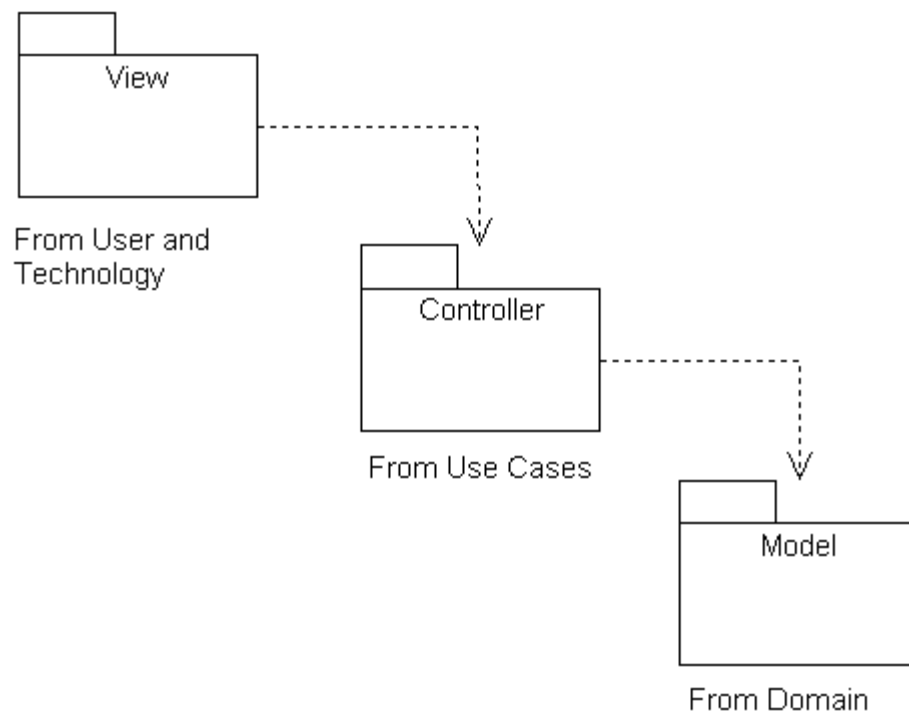
MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

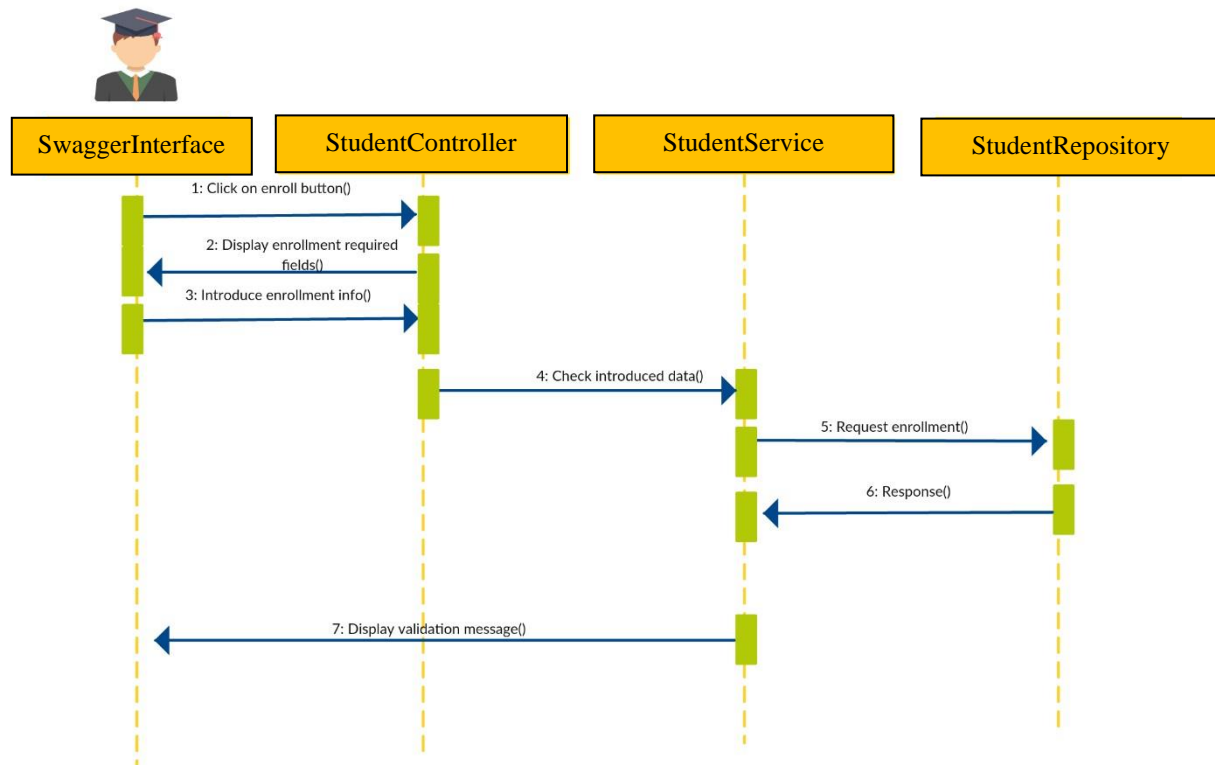
- **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

## 3.2 Diagrams



### *MVC Pattern*





Sequence Diagram for Student Enrollment Process

## 5. Class Design

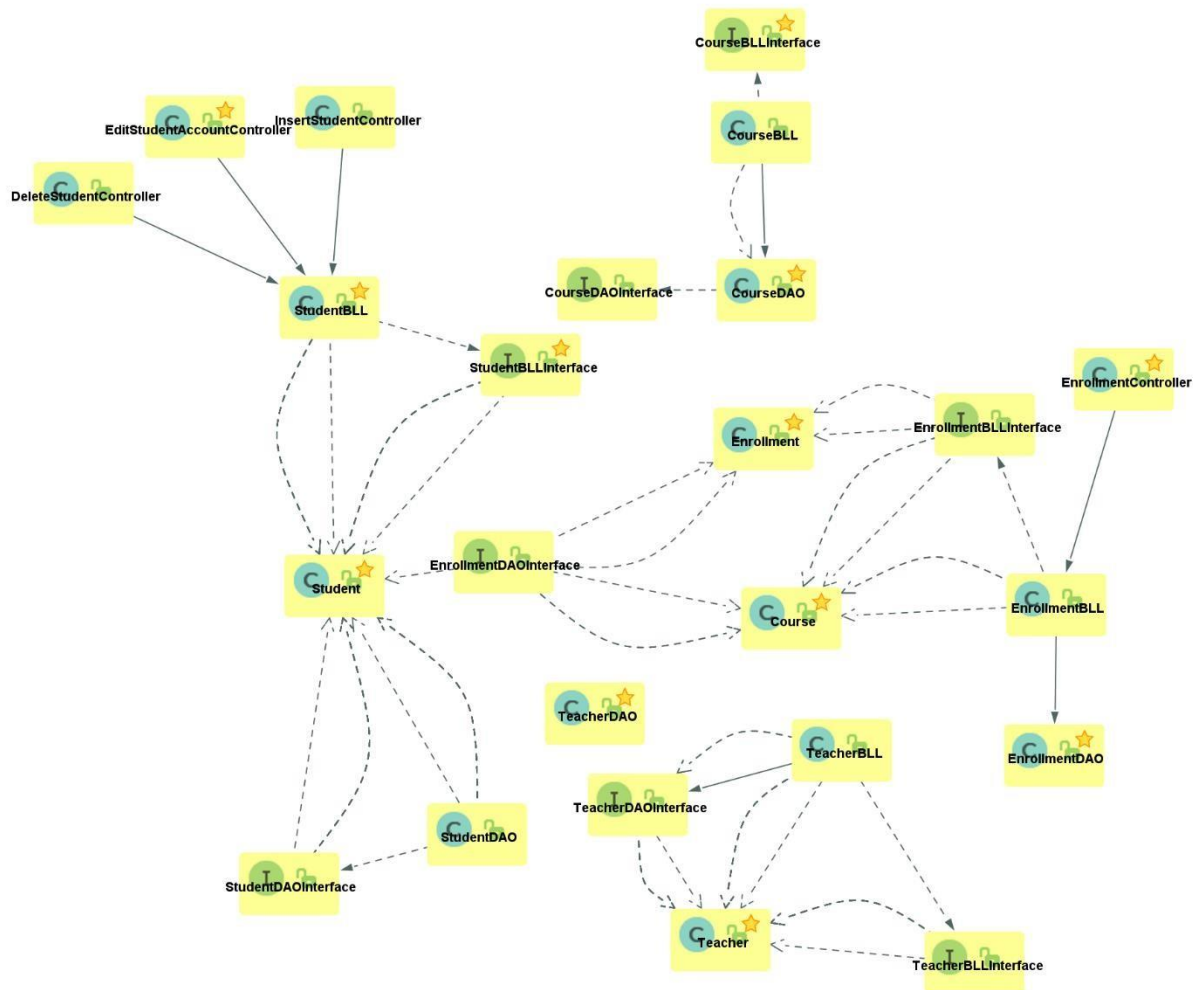
### 5.1 Design Patterns Description

Builder pattern builds a complex object using simple objects and using a step by step approach. This type of design pattern comes under creational pattern as this pattern provides one of the best ways to create an object.

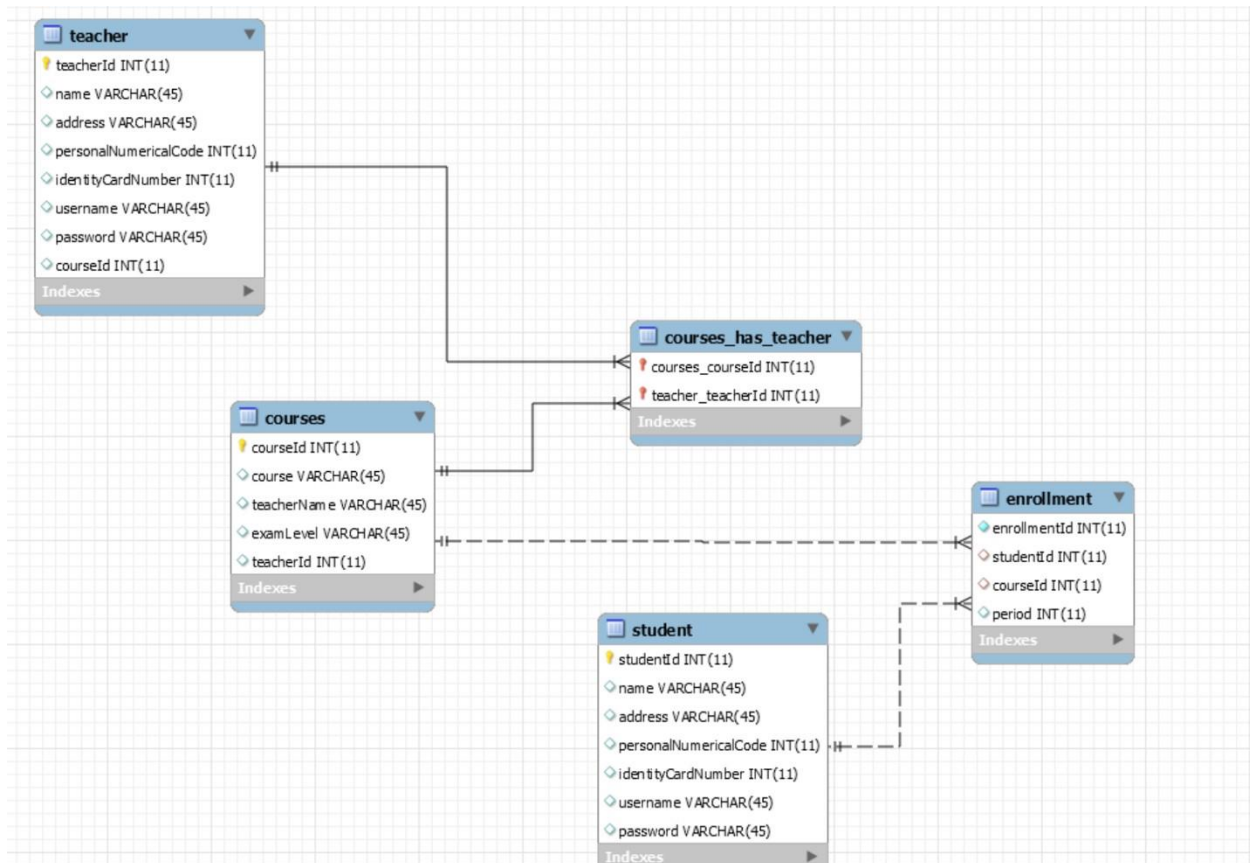
A Builder class builds the final object step by step. This builder is independent of other objects.

In our application, the builder design pattern will be used by creating a repository (an interface) for each entity, and make it extend the JpaRepository which will do all the CRUD operations on the entity.

## 5.2 UML Class Diagram



## 6. Data Model



## 7. System Testing

The system will be tested mainly with unit tests. 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. For the unit testing, we will use Junit4 together with Mockito.

## 8. Bibliography

[https://en.wikipedia.org/wiki/Singleton\\_pattern](https://en.wikipedia.org/wiki/Singleton_pattern)

<https://www.tutorialspoint.com/uml/index.htm>