Assignment 2
Analysis and Design Document

Student: George Cimpoies
Group: 30432

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

Design and implement a Java application for the management of students in the CS Department at UTCN.

1.2 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.

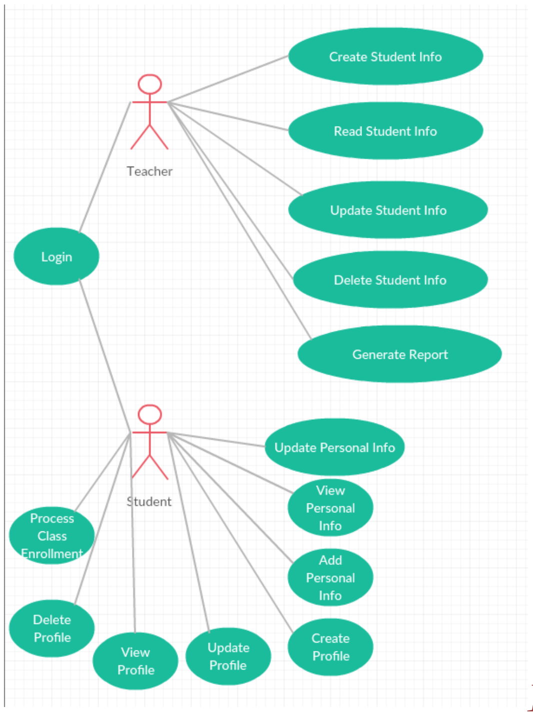
The regular user can perform the following operations:
- Add/update/view client information (name, identity card number, personal numerical code, address, etc.).
- Create/update/delete/view student profile (account information: identification number, group, enrolments, grades).
- Process class enrolment (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.

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: Teacher
Precondition: The teacher has to login and to select a student.
Main success scenario:
1. The teacher logins successfully.
2. The teacher selects a student.
3. The teacher selects a period.
4. The teacher generates a report based on the student's account information.

Use case: **Update Personal Information**
Level: User-goal level
Primary actor: Student
Precondition: The Student has to login
Main success scenario:
1. The student logins successfully.
2. The student chooses to update his personal information.
3. The student modifies the current information.
4. The updated information is saved.
Extensions: The user selects to cancel the operation.

3. System Architectural Design

3.1 Architectural Pattern Description

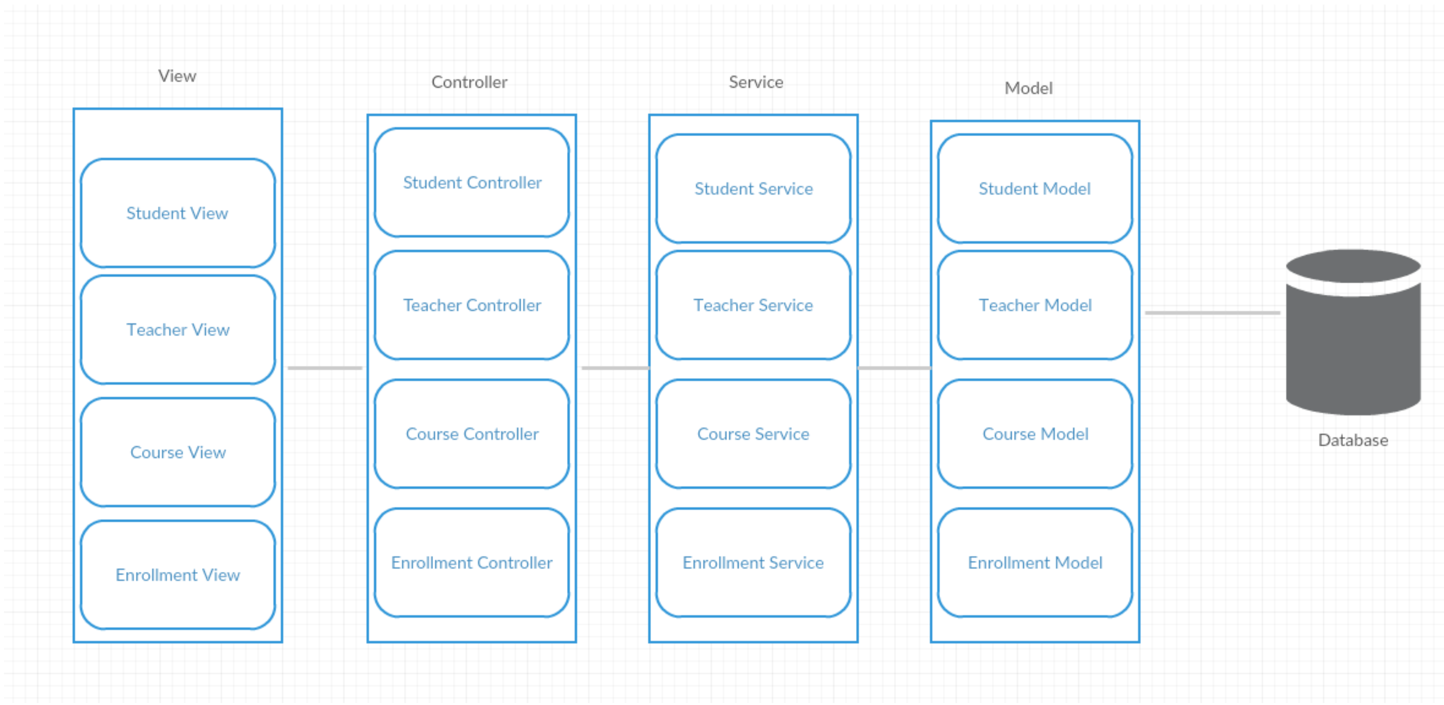
MVC Pattern (Model-View-Controller Pattern). We use this to separate the concerns within an application.

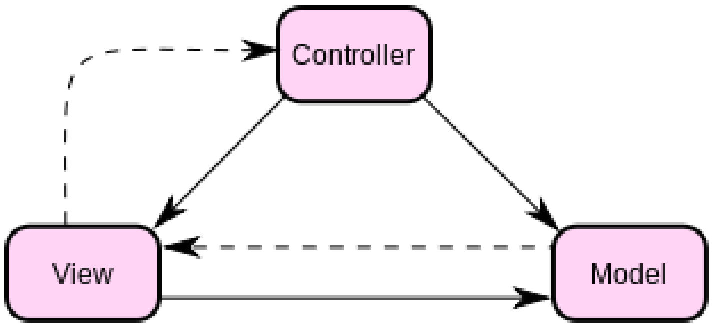
• **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.

• **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.

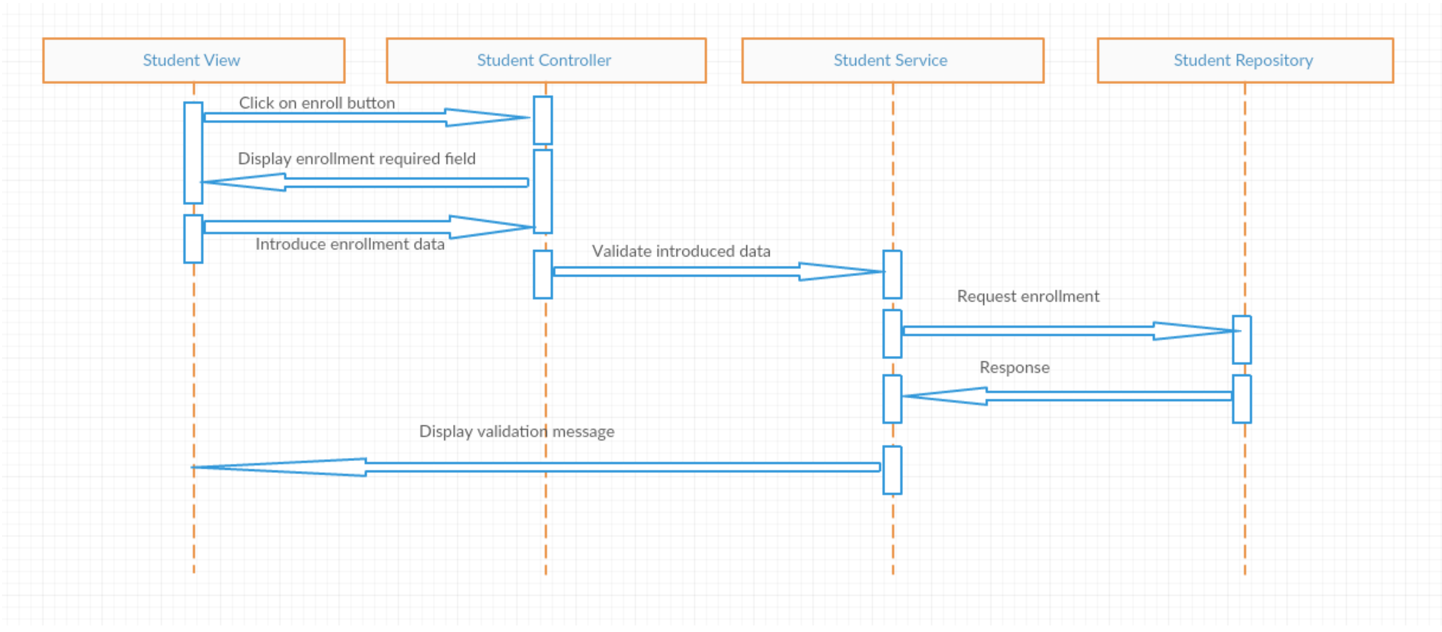
3.2 Diagrams


View
Controller
Service
Model
Student View
Teacher View
Course View
Enrollment View
Student Controller
Teacher Controller
Course Controller
Enrollment Controller
Student Service
Teacher Service
Course Service
Enrollment Service
Student Model
Teacher Model
Course Model
Enrollment Model
Database


Controller
View
Model

4. UML Sequence Diagrams

Request enrollment



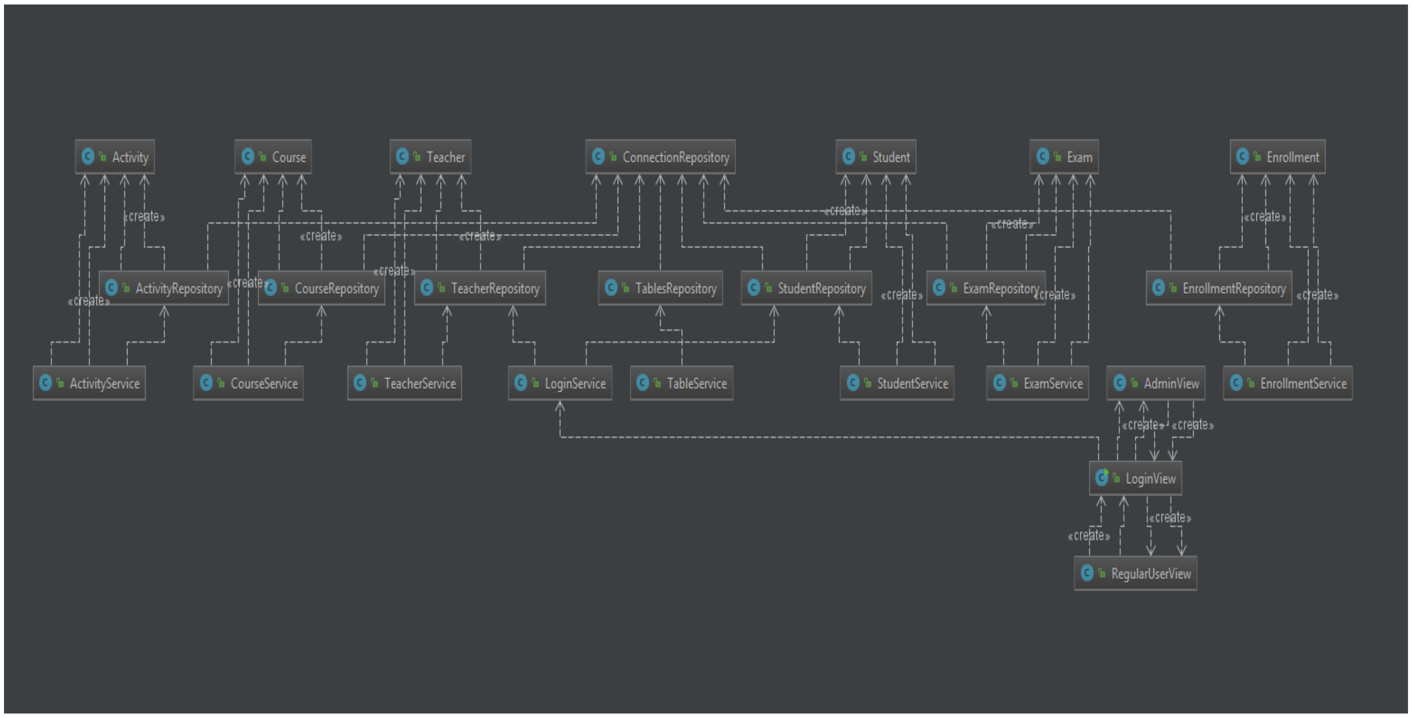
5. Class Design

5.1 Design Patterns Description

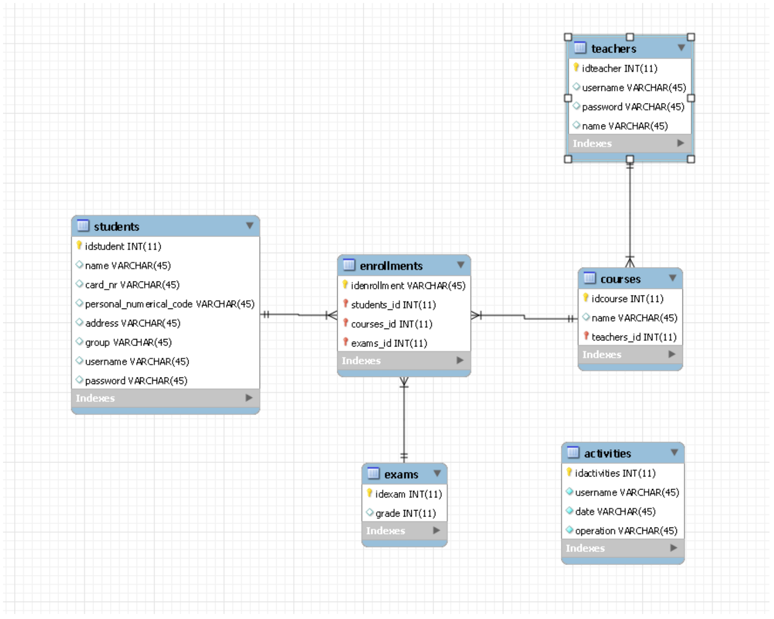
MVC – Model-View-Controller pattern – one of the most used design patterns to organize an application into three logical parts:
Models - data entities used throughout the application
View(s) – refers to the GUI side of the application
Controller – the "heart and brain" of the application

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

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://dzone.com/articles/junit-tutorial-beginners
https://www.umsl.edu/~sauterv/analysis/Fall2010Papers/varuni/