Assignment 2

Analysis and Design Document

Student: Radu Petrisel

**Group: 30432**

Table of Contents

[1.1 Assignment Specification 3](#_Toc511919887)

[1.2 Functional Requirements 3](#_Toc511919888)

[1.3 Non-functional Requirements 3](#_Toc511919889)

[2. Use-Case Model 4](#_Toc511919890)

[3. System Architectural Design 5](#_Toc511919891)

[3.1 Architectural Pattern Description 5](#_Toc511919892)

[3.2 Diagrams 6](#_Toc511919893)

[4. UML Sequence Diagram 8](#_Toc511919894)

[5. Class Design 9](#_Toc511919895)

[5.1 Design Patterns Description 9](#_Toc511919896)

[5.2.1 UML Class Diagram – server 10](#_Toc511919897)

[5.2.2 UML Class Diagram – client 11](#_Toc511919898)

[6. Data Model 12](#_Toc511919899)

1. Requirements Analysis

## 1.1 Assignment Specification

The application is used for the management of students in the Computer Science Department of Technical University of Cluj Napoca. The application has two users (student and teacher/administrator) which must provide a username and a password in order to use the application.

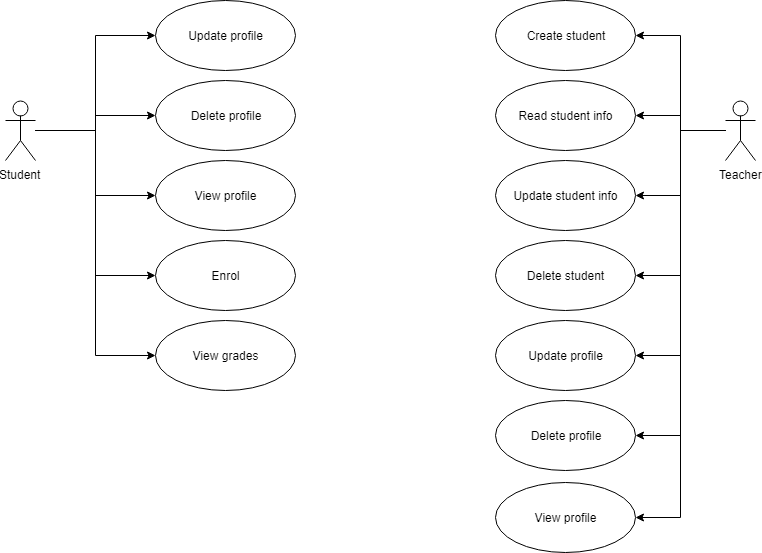
## 1.2 Functional Requirements

* create/read/update/delete students’ information (for teacher/administrator)
* generate reports (teacher)
* add/update/view info (all users)
* create/update /view profile (all users)
* process class enrolment (all users)

## 1.3 Non-functional Requirements

* reliability
* store data in a SQL Database
* store student reports in a NoSQL Database

# 2. Use-Case Model



**Use case:** create student

**Level:** user-goal level

**Primary actor:** teacher

**Main success scenario:** insert student info → system validate info → student created

**Extensions:** failure → invalid data or student already exists

# 3. System Architectural Design

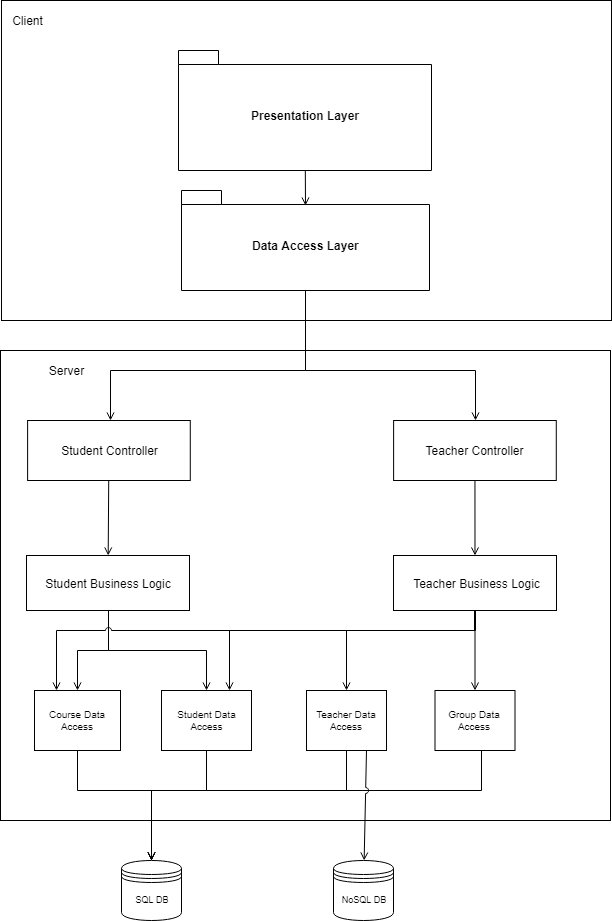
## 3.1 Architectural Pattern Description

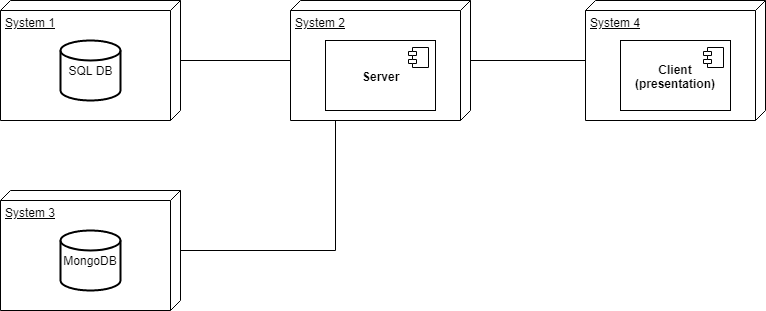
This system a Client-Server architecture and MVC (MC – server, VC – client).

Both the client and the server have a layered architecture. The layered architecture is the most common architectural pattern nowadays. The pattern consists of logically diving the application in layers, each with its own part to play. The server has 3 layers: data access (DAL), business logic (BLL) and presentation layer controllers (PL). The client has two layers – services and presentation layer (views & small controllers – for making requests to the server).

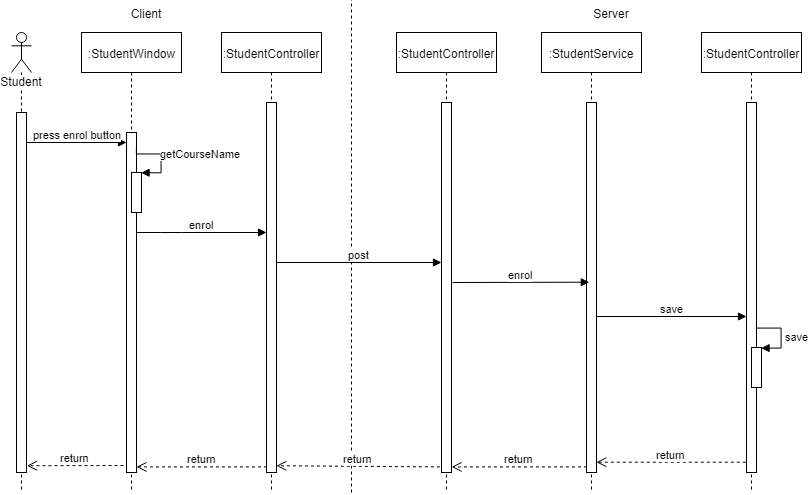
## 3.2 Diagrams

**Package Diagram**

****

**Deployment Diagram**

# 4. UML Sequence Diagram



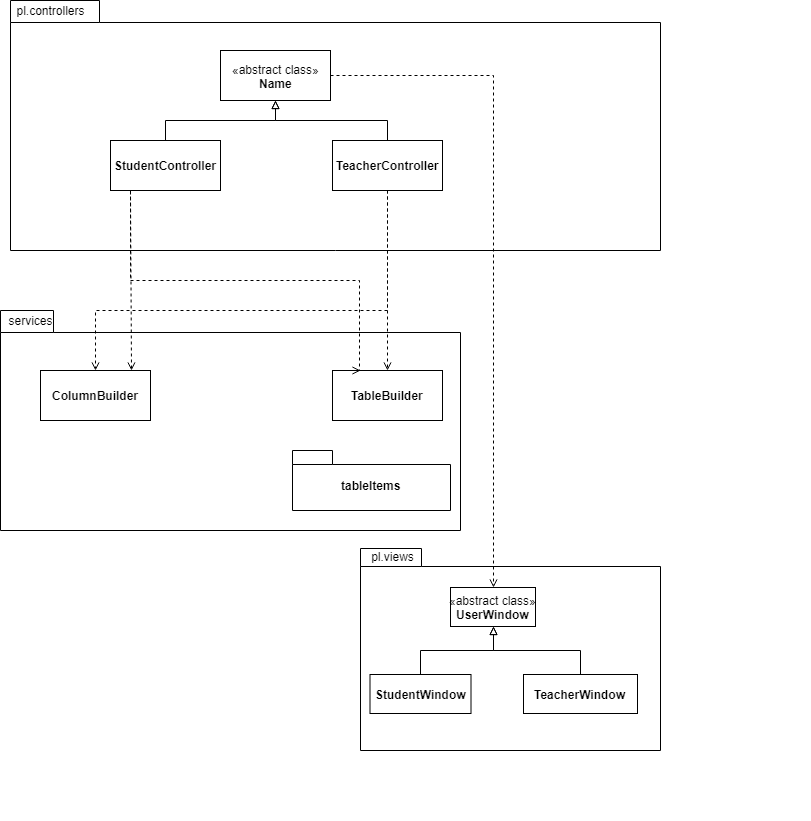
# 5. Class Design

## 5.1 Design Patterns Description

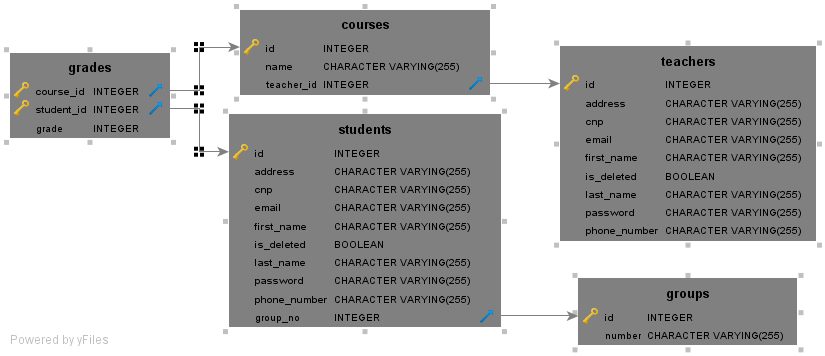
* Singleton – database connection has only one instance
* Design by contract – database access methods validate data before writing it to the database
* Builder pattern – students, tables and table columns have builders associated
* Dependency Injection – repositories and services use Spring injection mechanism

### 5.2.1 UML Class Diagram – server

### 5.2.2 UML Class Diagram – client



# 6. Data Model



8. Bibliography

[MySQL jdbc guide](https://dev.mysql.com/doc/connector-j/5.1/en/connector-j-usagenotes-connect-drivermanager.html)

[JavaFX Docs](https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm)

[Stackoverflow](http://www.stackoverflow.com)