Assigment1

Student: Crisan Catalin

**Group: 30641**

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

Implement a Java application for the management of students in the CS Department at TUCN. The application should have two types of users (student and teacher/administrator user)

# Functional Requirements

Student functionality:

* CRUD on client information(name, identity card number, personal number, group, enrolments, grades
* CRUD on student profile(identification number, group, enrolments, grades
* Class Enrolment (enroll, exam, grades)

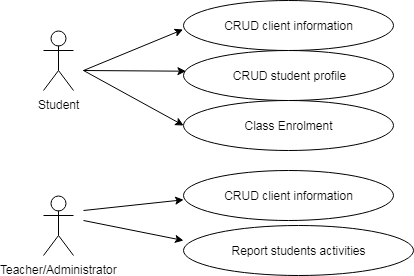
Administrator functionality:

* CRUD on student information
* Report for a particular period of student activities

# Non-functional Requirements

* Application environment is Java
* Dates are saved using into a database from MySql

2. Use-Case Model



*Use case:* functionality for student and administrator

*Level:* user-goal level

*Primary actor:* student and administrator

*Main success scenario:* an user can login, do CRUD operation and data are saved into database

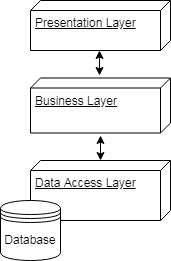
*Extensions:* Failure scenario is you can access the administrator page with student credentials

3. System Architectural Design

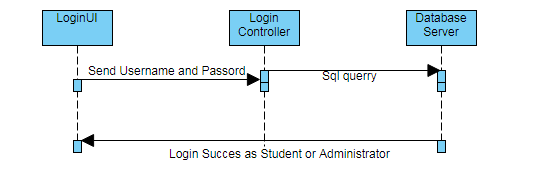
**3.1 Architectural Pattern Description**

*For this project I used architectural layers pattern*

**3.2 Diagrams**

**

4. UML Sequence Diagrams

**

5. Class Design

**5.1 Design Patterns Description**

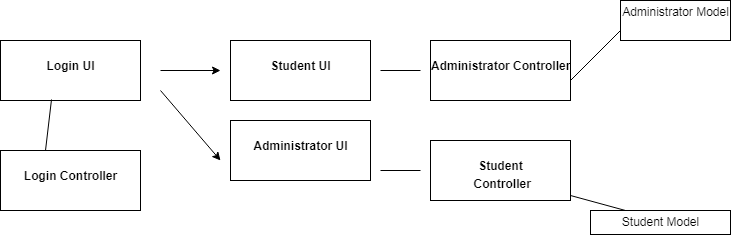
The model-view-controller (MVC) design pattern specifies that an application consist of a data model, presentation information, and control information.

The model contains only the pure application data; it contains no logic describing how to present the data to a user.

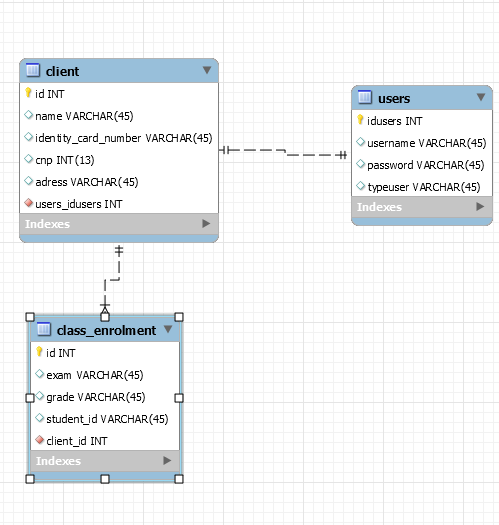
The view presents the model's data to the user. The view knows how to access the model's data, but it does not know what this data means or what the user can do to manipulate it.

The controller exists between the view and the model. It listens to events triggered by the view (or another external source) and executes the appropriate reaction to these events. In most cases, the reaction is to call a method on the model.

**5.2 UML Class Diagram**

**

6. Data Model

**

7. System Testing

- create test for correct/incorect login

- create test for admistrator modify student information

8. Bibliography