Student:Aionitoaie Mihai

**Group: 30233**

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

Design and 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) 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.

# Functional Requirements

- Depending on the type of user and password entered, a window is created to display information;

- User: add, update, display information;

- User: create, update, display student profile;

- Admin: CRUD on student profile;

- Admin: Creating reports for a specific period that contains the activities a student has attended.

# Non-functional Requirements

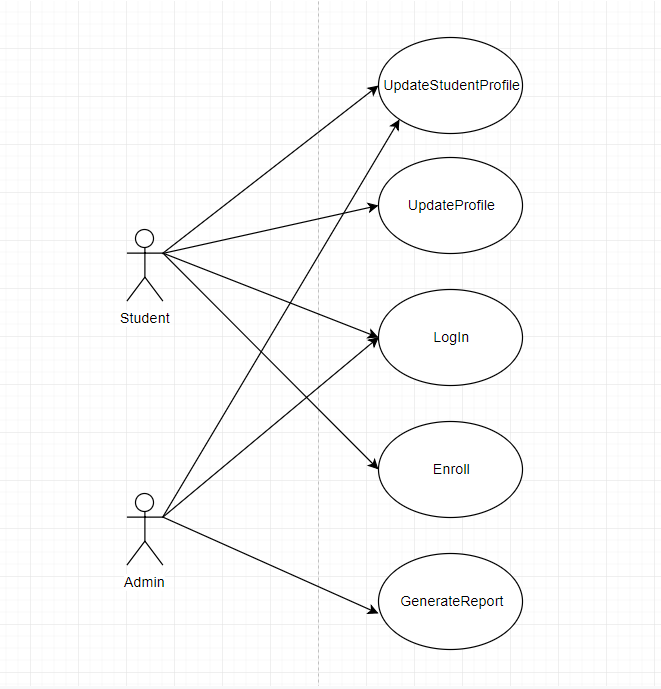
- Logging and displaying information does not take long;

- The application will continue to run, even if some data are wrong;

- Logging errors should be generated, if applicable;

- Manage situations where there are duplicates.

2. Use-Case Model



Use case: Log in to the student management system

Level: Sub-function

Primary actor: Student / Admin

Main success scenario: The login was successful and the required information is displayed

Extensions:

Login data is not correct;

A logon error is generated;

End use case.

Use case: Modify logged student profile data

Level: user-goal level

Primary actor: Student / Admin

Main success scenario: Data changed successfully

Extensions:

Alternative flow A:

The student ID you are looking for does not exist in the database;

An error is generated;

End use case.

Use case: To enroll in a course

Level: user-goal level

Primary actor: Student

Main success scenario: The student successfully enrolled in a particular course so he can see his grades and exams.

Extensions:

The student is already enrolled in a certain course;

A warning message is displayed.

End use case.

Use case: Generate a report for a specific interval that displays the courses attended by a student.

Level: user-goal level

Primary actor: Admin

Main success scenario: The report is displayed and contains all the subjects the student has attended during that period.

Extensions:

 The student searched does not exist in the database;

An error message is displayed;

End use case;

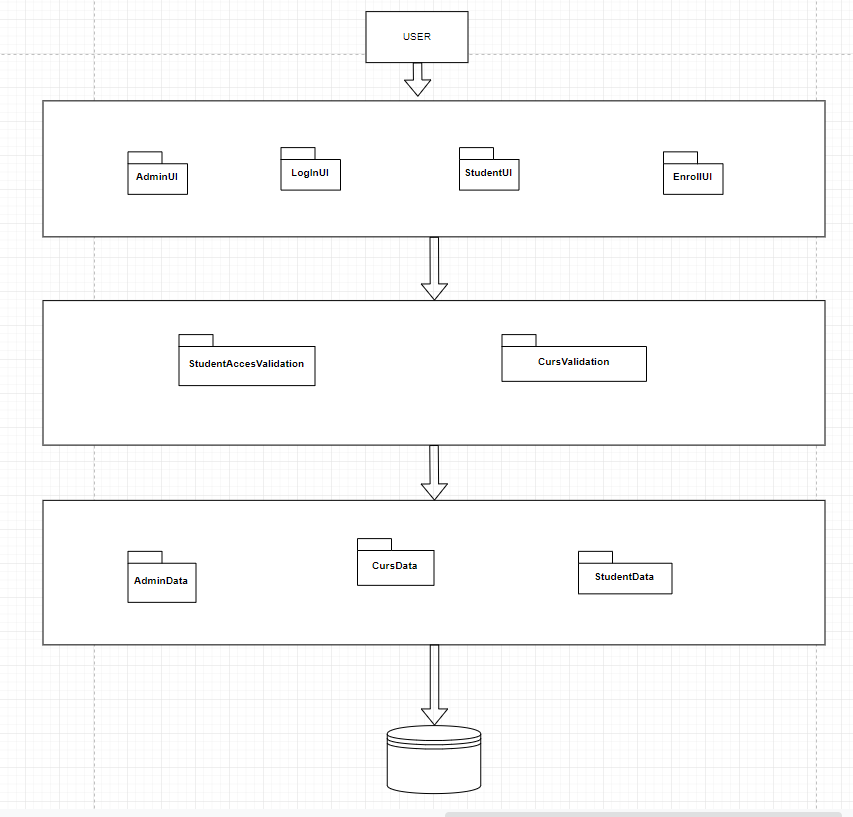
3. System Architectural Design

**3.1 Architectural Pattern Description**

Layered architecture pattern components are organized in horizontal layers, each layer performing a specific role within the application (for example, presentation logic or business logic). Although the Layer Architecture model does not specify the number and types of layers that need to be in the pattern, most layered architectures are made up of four standard layers: Presentation layer, Business Layer, Persistence Layer, and Database Layer.

One of the strong features of the layered architectural model is the separation of concerns between the components. Components in a particular layer only deal with the logic that refers to that layer. For example, the components in the presentation layer only deal with presentation logic, while the components that live in the business layer only deal with business logic. This type of component classification makes it easy to build efficient roles and models of responsibility in architecture.

**3.2 Diagrams**



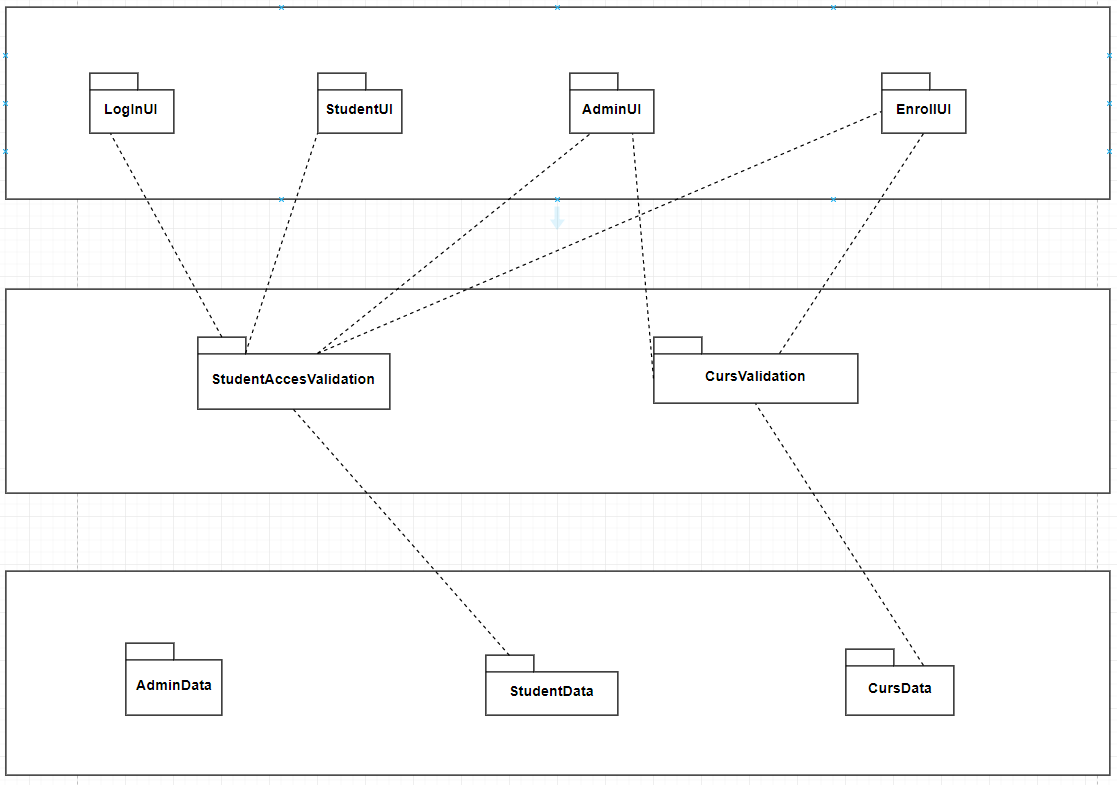
In this scenario, users can access the application through the presentation layer, which communicates directly with components in the business layer. The data that is processed in the business layer comes from a database through components that access these data in the data layer.

The presentation layer contains the components that implement and display the user interface and manage the user interaction. This layer includes user input and display commands, in addition to components that organize user interaction.

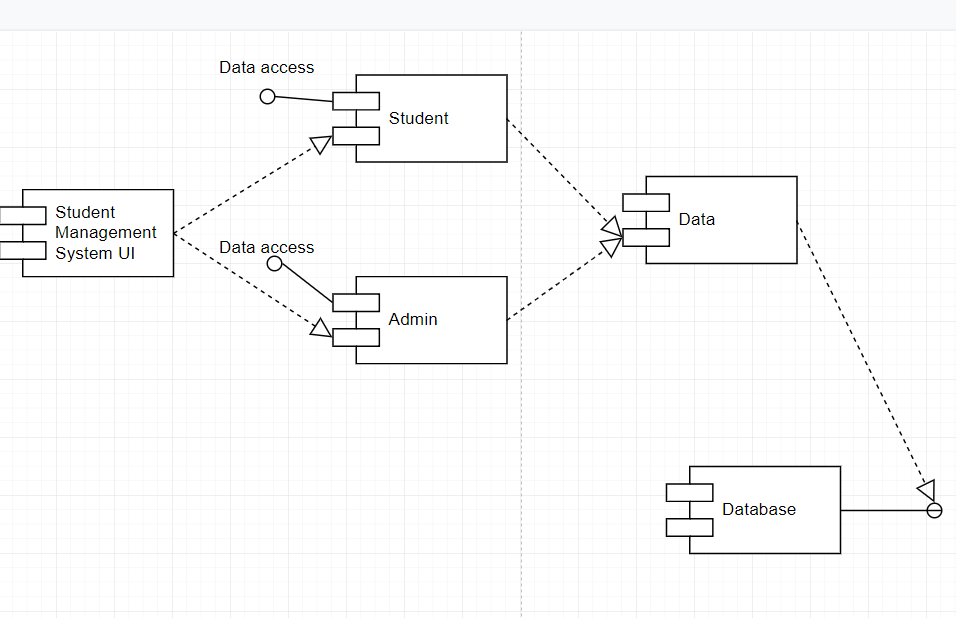
After the UI components collect the requested user data and pass it on to the business level, the application can use this data to conduct a business process. Business logic is defined as any application logic dealing with the recovery, processing, transformation, and management of application data; applying commercial rules and policies; and ensuring the consistency and validity of the data.

In the data layer we have Data Access Components that address the logic needed to access the underlying data stocks. They centralize common data access functionality to make the app easier to configure and maintain. Some data access frameworks may require the developer to identify and implement a common data access logic in the reusable or accessible auxiliary help components.

Package Diagram:

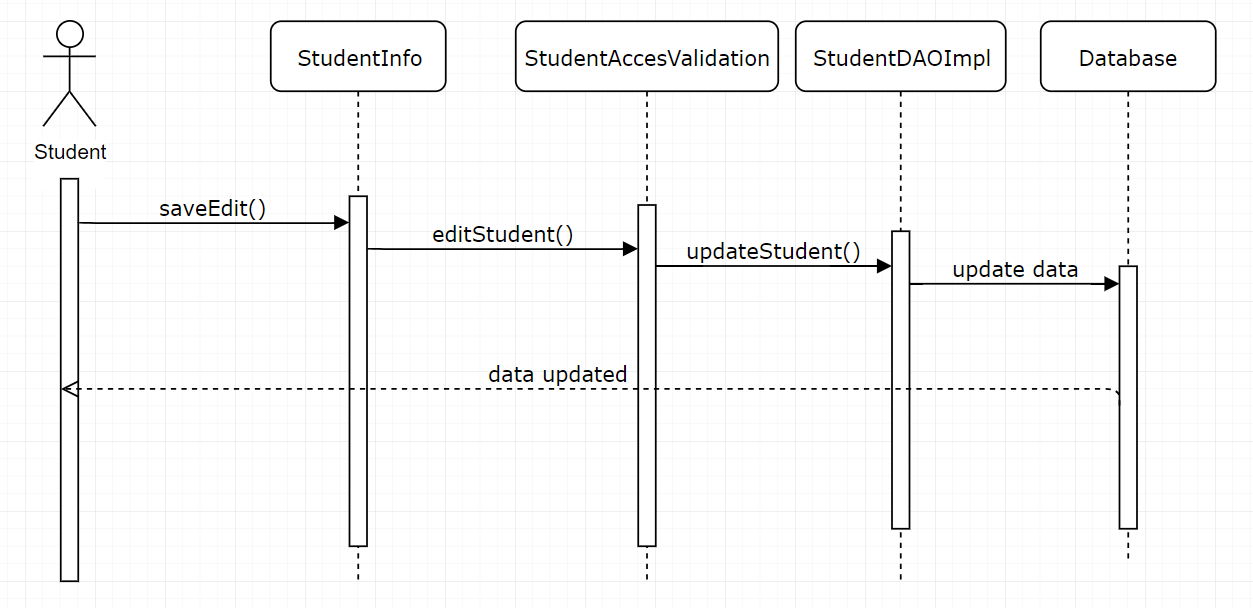


Component Diagram:



4. UML Sequence Diagrams

Scenario : edit student data

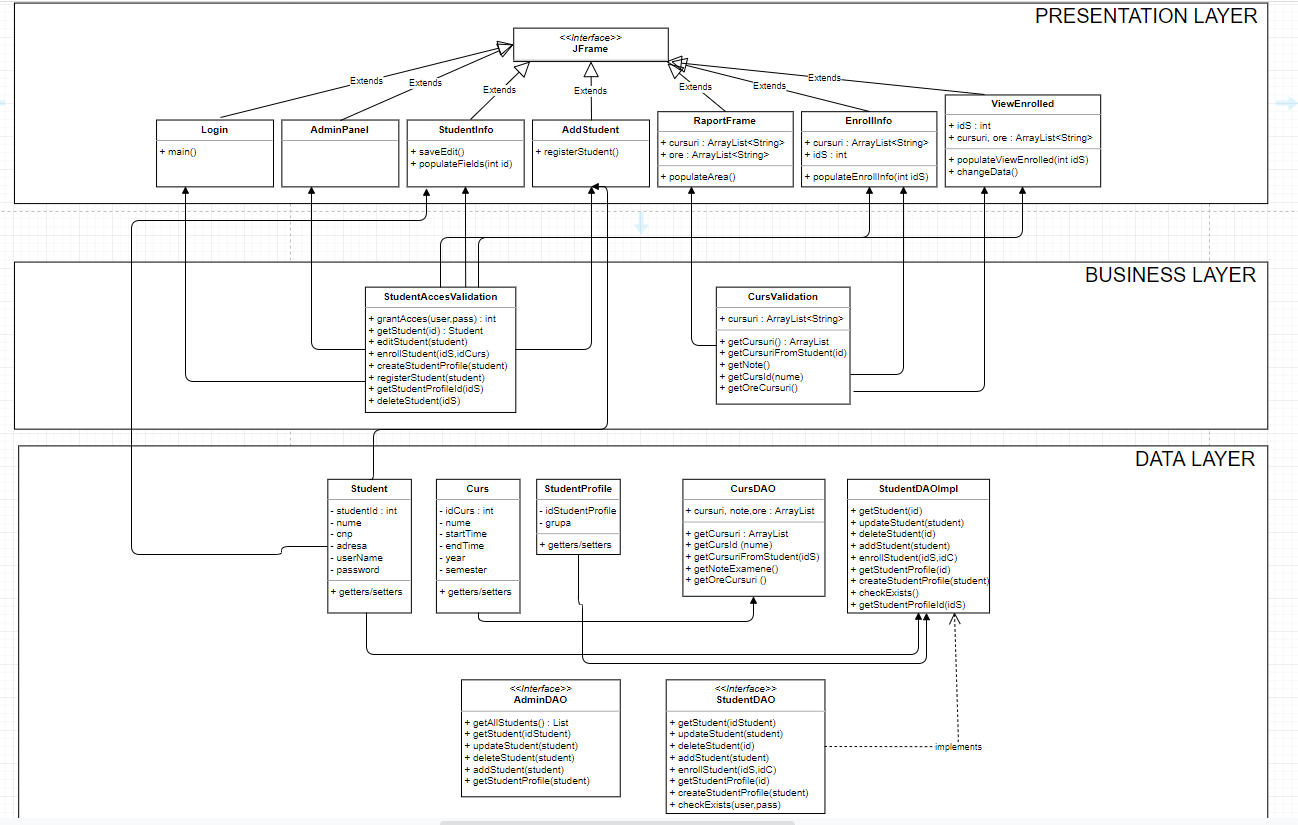


5. Class Design

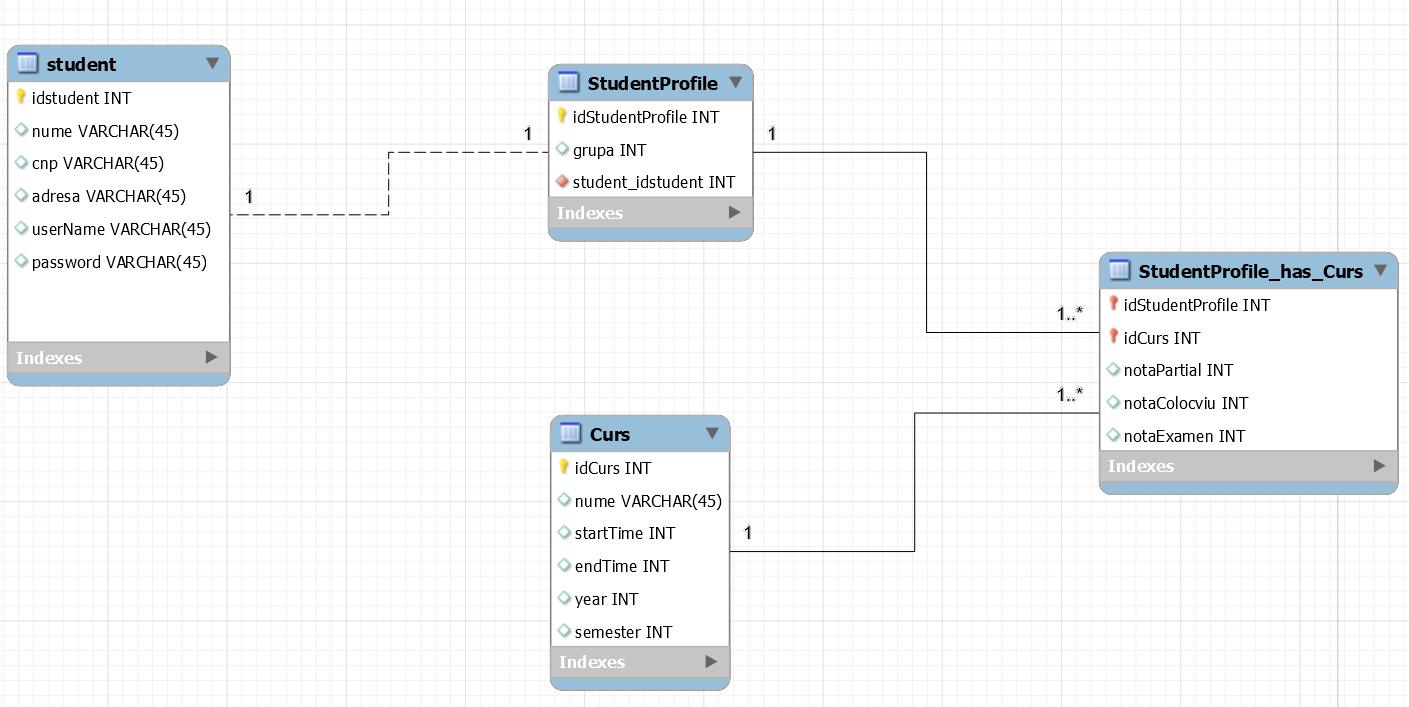
**5.1 Design Patterns Description**

None used.

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

**UNIT TESTING** is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

8. Bibliography

<https://opentextbc.ca/dbdesign01/chapter/chapter-5-data-modelling/>

<https://web.csulb.edu/~mopkins/cecs493/sdsfinaledit-140325142607-phpapp01.pdf>

<https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658116(v%3dpandp.10)>

<http://csis.pace.edu/~marchese/CS389/L9/Use%20Case%20Diagrams.pdf>

<https://creately.com/blog/diagrams/use-case-diagram-tutorial/>

<https://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/>

<https://www.edrawsoft.com/template-uml-deployment-diagram.php>