ASSIGNMENT A1

Analysis and Design Document

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.

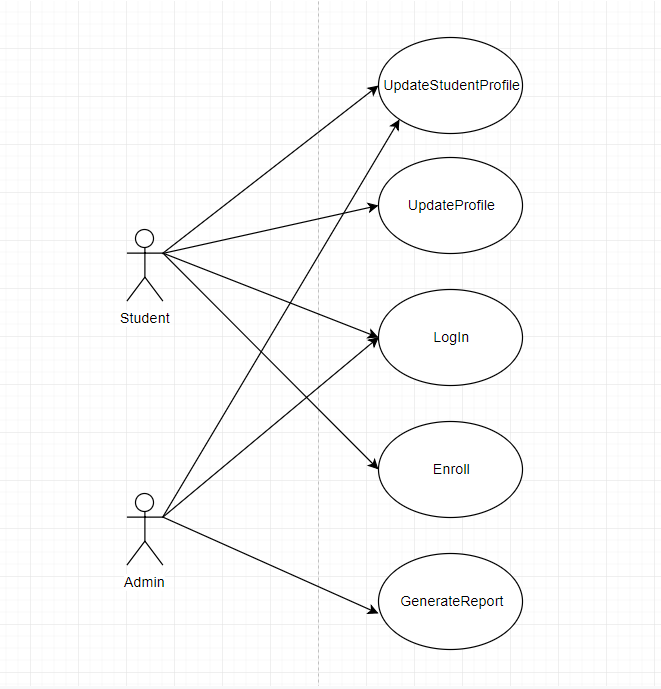
# Functional Requirements

* In functie de tipul de user si parola introdusa se creeaza o fereastra care sa afiseze informatii;
* User : adaugare, actualizare, afisare informatii;
* User : creare, actualizare, afisare profil student;
* Admin : CRUD pe profilul studentului;
* Admin : creare rapoarte pentru o anumita perioada care sa contina activitatile la care a participat un student.

# Non-functional Requirements

* Logarea si afisarea informatiilor sa nu necesite mult timp;
* Aplicatia sa isi continue functionarea, chiar daca unele date sunt gresite;
* Ar trebui generate erori la logare, daca este cazul;
* Sa gestioneze situatiile in cazul in care exista duplicate.

2. Use-Case Model



Use case : Logarea in sistemul de management al studentilor

Level : Sub-function

Primary actor : Student/Admin

Main success scenario : logarea a avut success si sunt afisate informatiile necesare

Extensions :

Datele de logare nu sunt corecte;

Se genereaza o eroare de logare;

End use case.

Use case : Modificarea datelor profilului studentului logat

Level : user-goal level

Primary actor : Student/Admin

Main success scenario : datele au fost modificate cu success

Extensions :

Alternative flow A:

Id-ul studentului cautat nu exista in baza de date;

Se genereaza o eroare;

End use case.

Use case : Inrolarea la un curs

Level : user-goal level

Primary actor : Student

Main success scenario : studentul s-a inrolat cu succes la un anumit curs, astfel isi poate vedea notele si examenele.

Extensions :

Studentul este deja inscris la un anumit curs;

Se afiseaza un mesaj de avertizare;

End use case.

Use case : Generarea unui raport pentru un anumit interval, care afiseaza cursurile la care a participat un student.

Level : user-goal level

Primary actor : Admin

Main success scenario : raportul este afisat si contine toate materiile la care studentul a participat in perioada respectiva.

Extensions :

Studentul cautat nu exista in baza de date;

Se afiseaza un mesaj de eroare;

End use case;

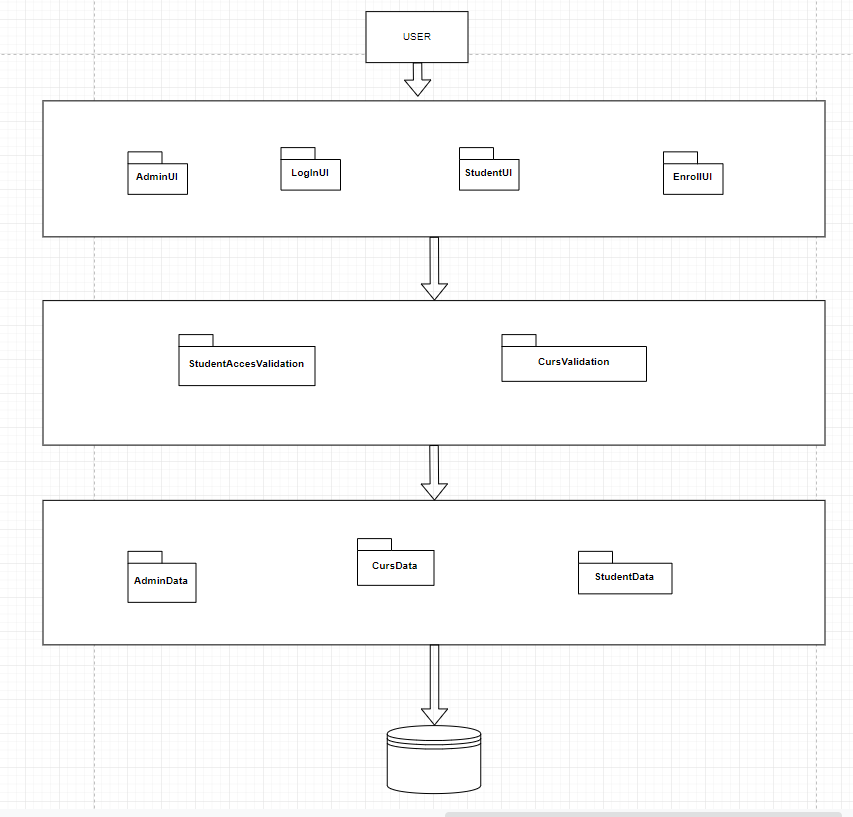
3. System Architectural Design

**3.1 Architectural Pattern Description**

Componentele din Layered architecture pattern sunt organizate în straturi orizontale, fiecare strat realizând un rol specific în cadrul aplicației (de exemplu, presentation logic sau business logic). Deși modelul de arhitectură în strat nu specifică numărul și tipurile de straturi care trebuie să existe în model, cele mai multe arhitecturi stratificate sunt alcatuite din patru straturi standard: Presentation layer, Business Layer, Persistence Layer si Database Layer.

Una dintre caracteristicile puternice ale modelului arhitectural stratificat este separarea preocupărilor între componente. Componentele dintr-un anumit strat se ocupă numai de logica care se referă la acel strat. De exemplu, componentele din stratul de prezentare se ocupă doar de logica de prezentare, în timp ce componentele care locuiesc în stratul de afacere se ocupă numai de logica de afaceri. Acest tip de clasificare a componentelor ușurează construirea de roluri eficiente și modele de responsabilitate în arhitectura.

**3.2 Diagrams**



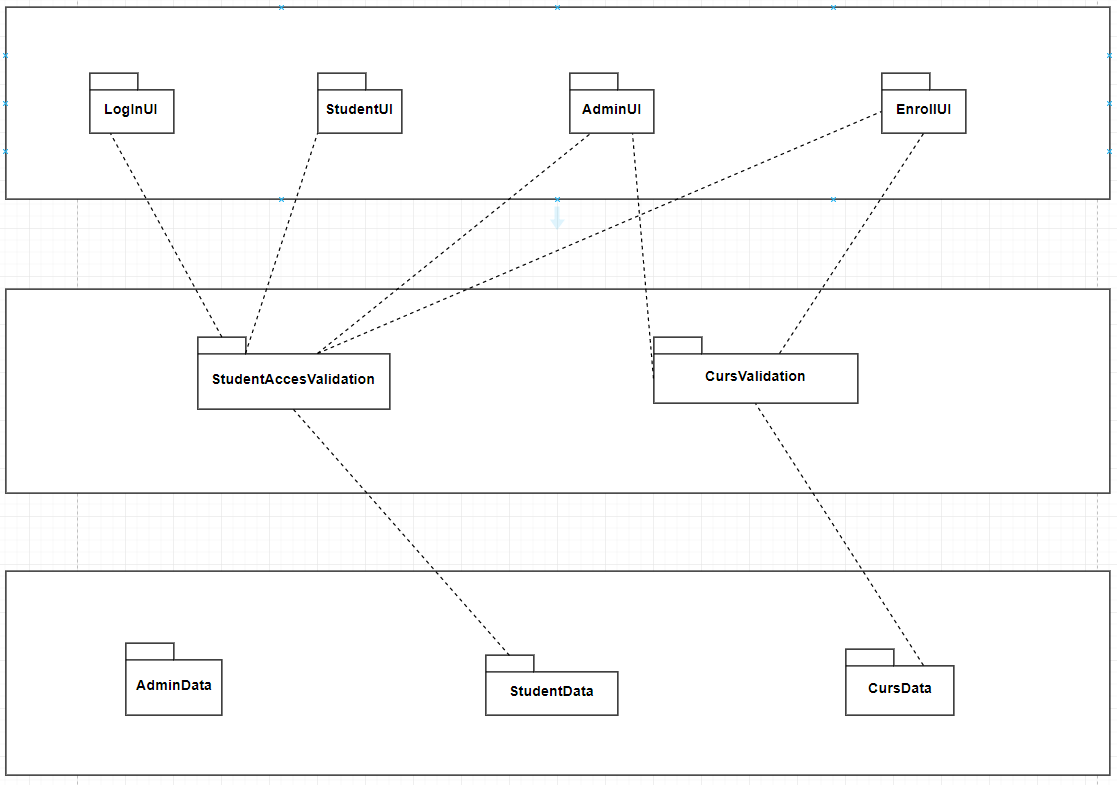
În acest scenariu, utilizatorii pot accesa aplicația prin intermediul stratului de prezentare, care comunică direct cu componentele din stratul de afaceri. Datele care se prelucreaza in stratul de afaceri provin dintr-o baza de date prin intermediul unor componente care acceseaza aceste date in stratul de date.

Stratul de prezentare conține componentele care implementează și afișează interfața utilizator și gestionează interacțiunea utilizatorului. Acest strat include comenzi pentru introducerea și afișarea de către utilizatori, pe lângă componentele care organizează interacțiunea utilizatorului.

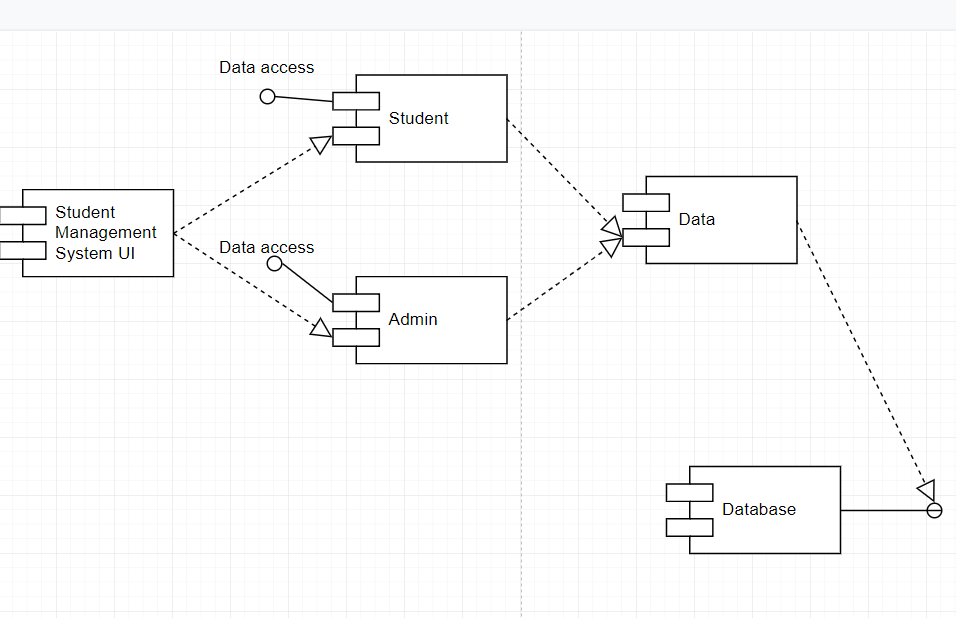
După ce componentele UI colectează datele solicitate de la utilizator și le transmit la nivelul de afaceri, aplicația poate folosi aceste date pentru a realiza un proces de afaceri. Logica de afaceri este definită ca orice logică de aplicație care se ocupă cu recuperarea, prelucrarea, transformarea și gestionarea datelor aplicației; aplicarea regulilor și politicilor comerciale; și asigurarea consecvenței și valabilității datelor.

In data layer avem Data Access Components care abordează logica necesară pentru a accesa stocurile de date care stau la baza acestora. Ele centralizează funcționalitatea comună de acces la date pentru a face aplicația mai ușor de configurat și menținut. Unele cadre de acces la date pot solicita dezvoltatorului să identifice și să implementeze o logică comună a accesului la date în componentele de ajutor auxiliare reutilizabile sau accesibile.

Package Diagram:

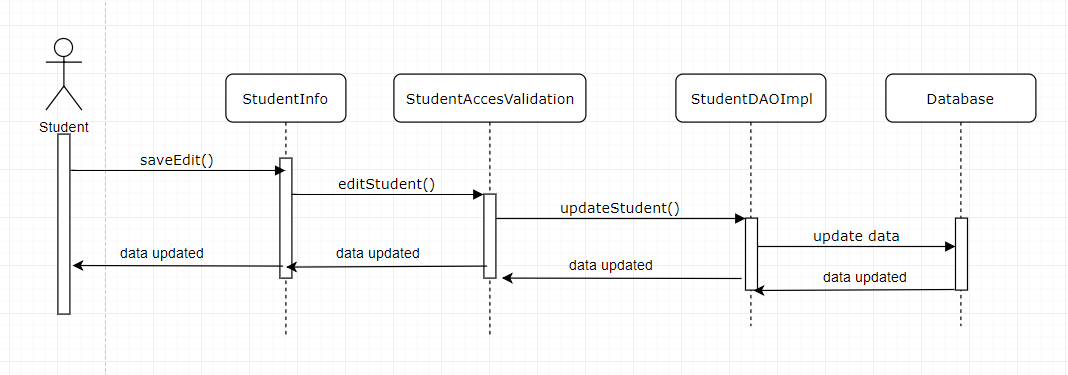


Component Diagram:



4. UML Sequence Diagrams

Scenario : edit student data

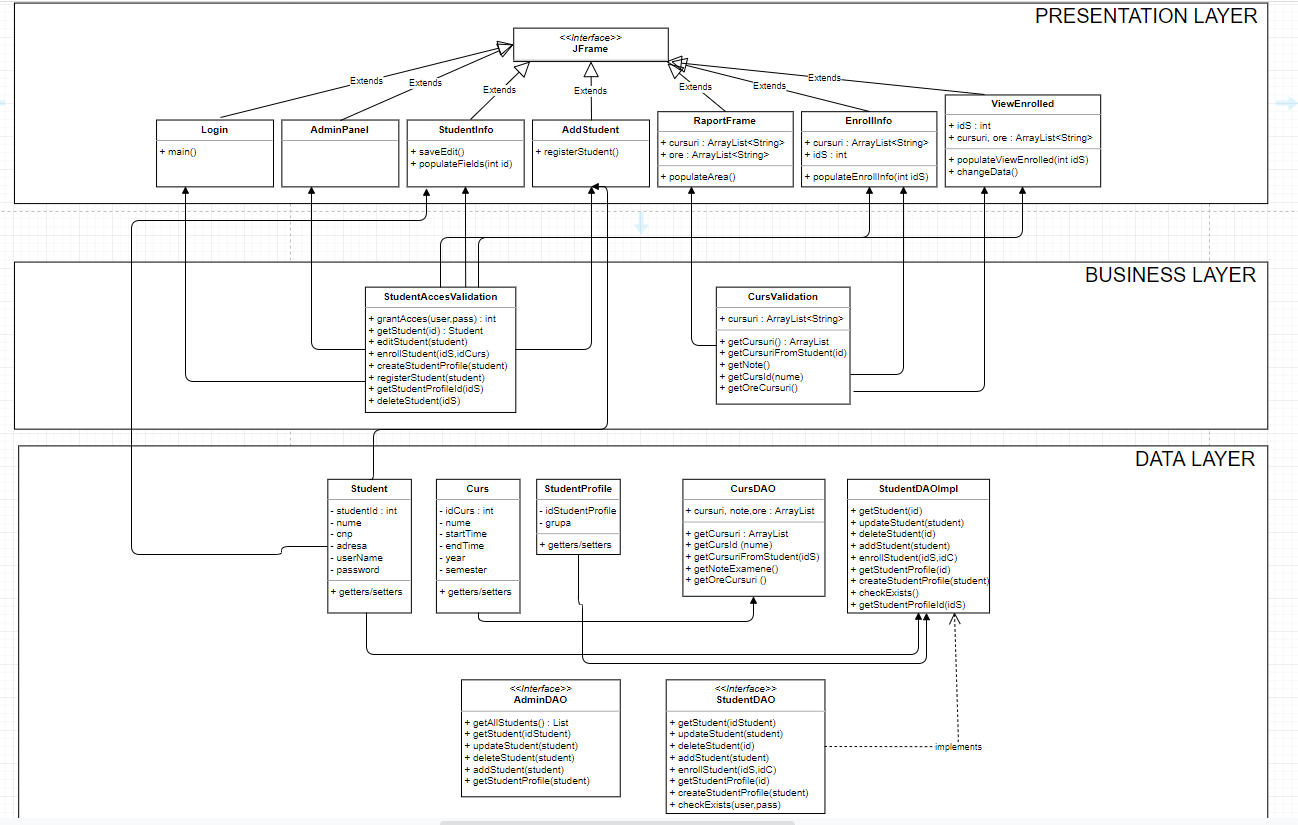


5. Class Design

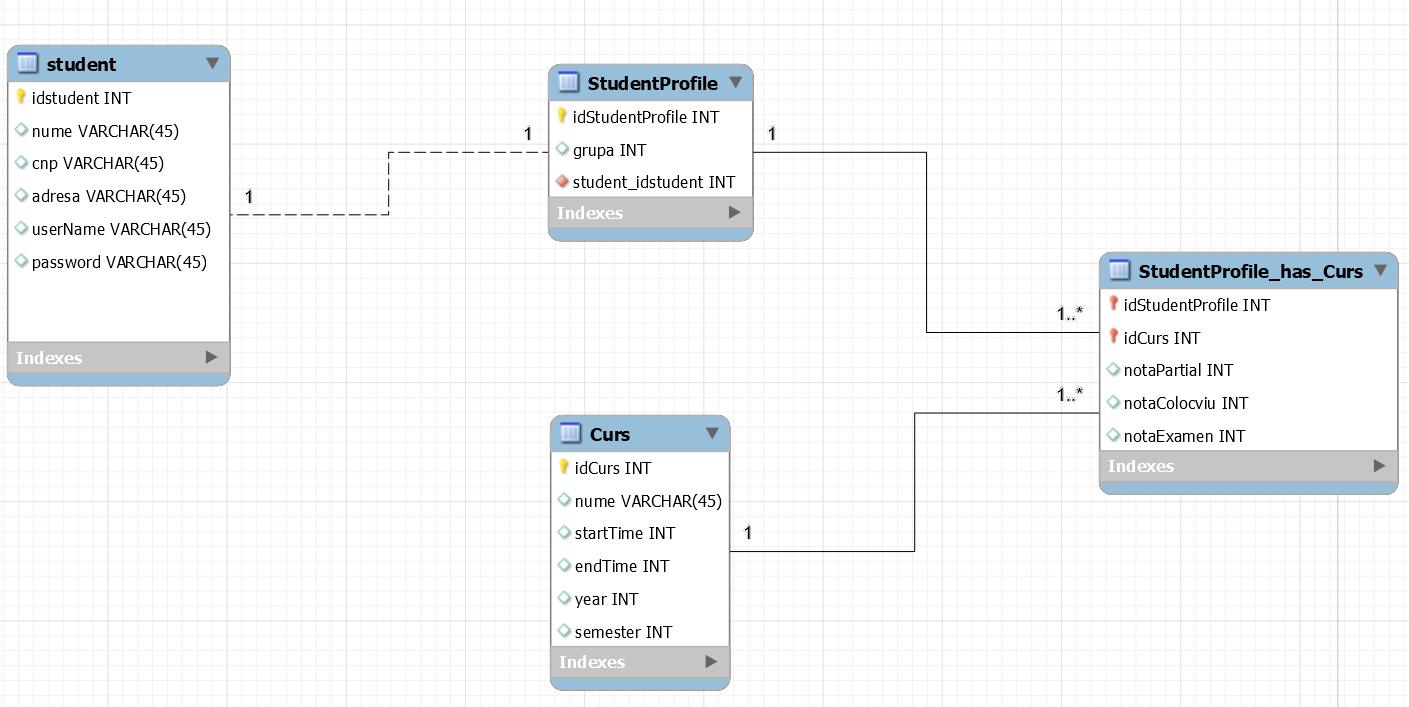
**5.1 Design Patterns Description**

None used.

**5.2 UML Class Diagram**



6. Data Model



7. System Testing

* Testarea manuala

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>