5/10/2022

Software Requirements Specification

Faculty Management System

Manager **Mohamed Ahmed Sufi**

- Mohamed Ashraf Ibrahim (team leader)
- Mohamed Yasser Ali
- Yasmeen Saad Abdelfattah

Introduction

Purpose for SRS

The purpose of this document is to present a detailed description of the Faculty Management System. It will explain the purpose and features of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for the team members and will be proposed to Manager (Instructor) for his approval.

Objective of Project

The main objective of the project on Faculty Management System is to manage the details of faculty, Students, Doctors, and Courses. The purpose of the project is to build an application program to reduce the manual work for managing the faculty, Students. It tracks all the details about the Doctors, courses.

Glossary

Term	Definition
Activity Diagram	Diagram is often used in business process modeling. Illustrate the process flow in system.
Actor	User that uses system
Admin	Doctor who has management responsibilities
Class Diagram	Diagram shows how data are structured in Database.
Course	Subject
Entity	thing has properties or characteristics in Database, such as student , department.
Erd Diagram	Diagram shows the attributes of entity the relationships between entities
Database	Collection of all the information monitored by this system
Department	Department in faculty that contains some courses, students learn in it and Doctors teach in it
Doctor	Person who teaches in faculty
Form	User interface that user can use it to perform some actions
Framework	is a generic term commonly referring to an essential supporting structure which other things are built on top of.
Grades	The grades that Doctor assign them to student in particular Course. Composed from Mid-

	term, practical and final.
Processes	Means add, remove and delete operations
Programming Language	A programming language is a computer language that is used by programmers (developers) to communicate with computers.
RDMS	is a database that mainly stored data in a relational model by creating different relations between different data.
Student	Person who learns in faculty
Table	Container structure that contains the entities have related proprieties.

Other Resources

MICROSOFT PROJECT is used to mange working on the system between team. **MS** project will be proposed to Manager (Instructor) for his approval.

References

This document based on **IEEE STANDARD.**

Overall Description

System Environment

Edition

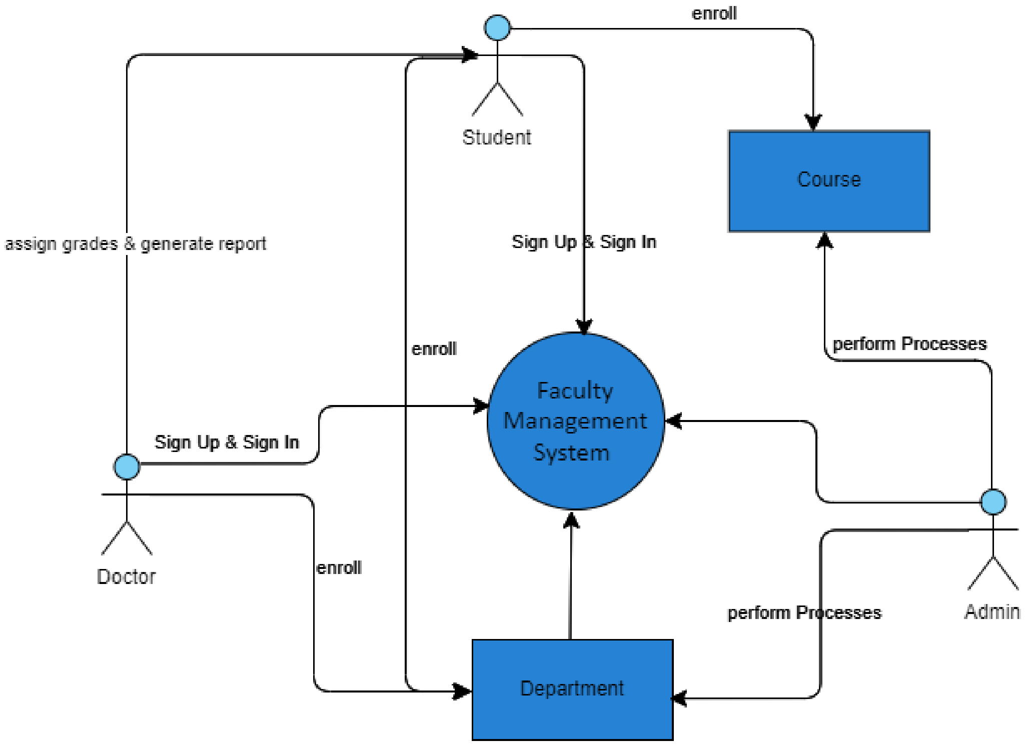


Figure 1: DFD level 0

The Faculty Management System has two actors (Admin is Doctor has more responsibilities.

Doctors can sign up & in in the system, enroll in more than one department and teach many courses and get info about courses.

Admin can add, delete and update both courses and departments.

Student can sign up & in in the system, enroll in only one department and enroll into many courses and get info about courses.

Functional Requirements Specification

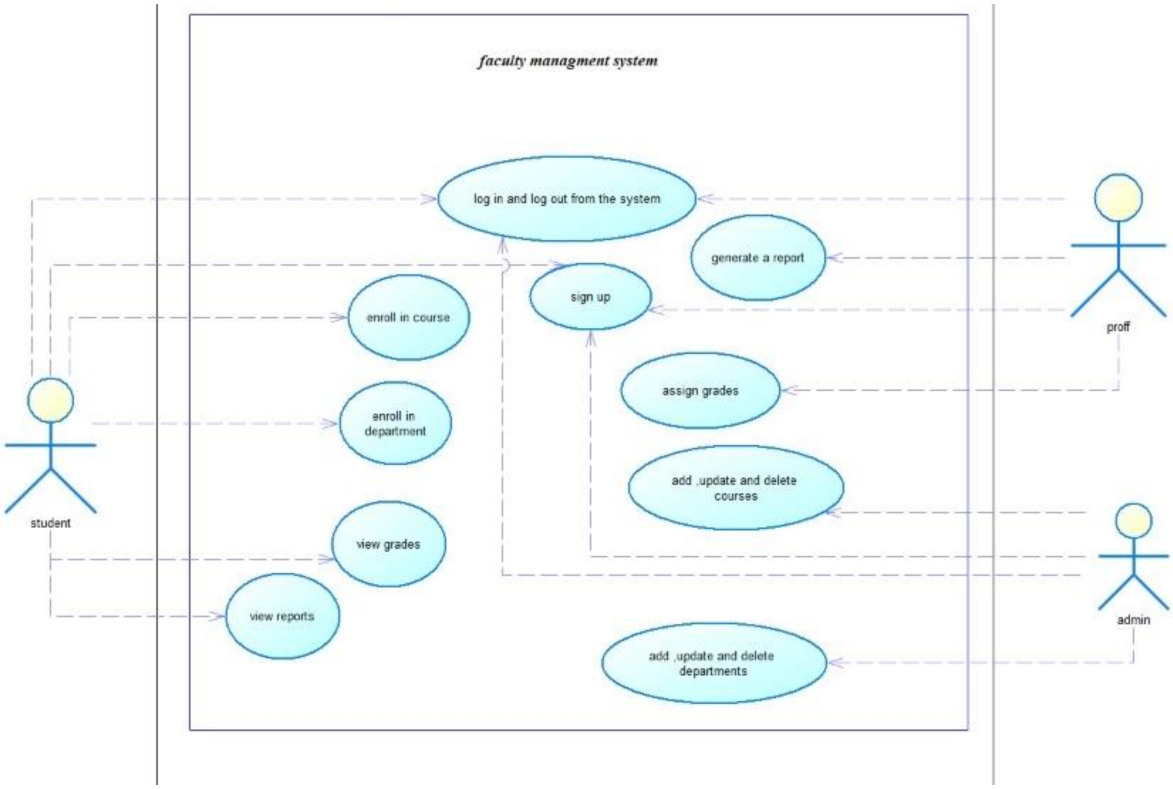


Figure 2: Use case Diagram

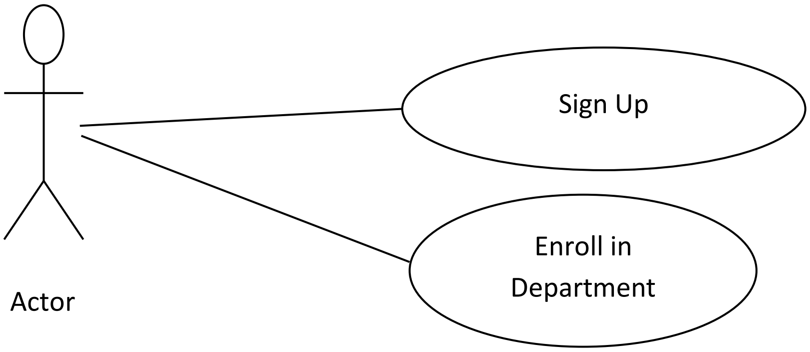
This section outlines the use cases for each actor.

Use Actor to represent to student, doctor. to describe the common use cases (scenarios).

Actor Use Cases

Use case: Sign Up & Enroll in Department

Diagram



Brief Description

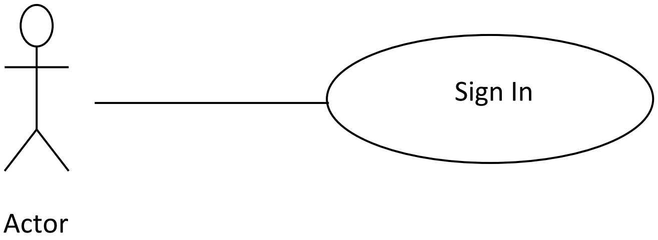
The Actor must first register into system

Initial Step-By-Step Description

1. The system displays the available departments for student to enroll.
2. The Student enters his main information (ID, Name, Phone, ….).
3. The Student selects the desired department.
4. The Student press on the sign up button.
5. The system adds this student to Database.

Use case: Sign In

Diagram



Brief Description

In order for an actor to enter the system, he must each time he logs in.

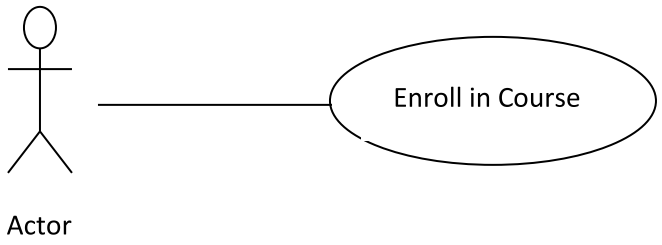
Initial Step-By-Step Description

Before this use case can be initiated, the actor has already accessed the Faculty Management System.

1. The Student enters his **ID** an **Password**.
2. The system verifies from **ID**.
3. If **ID** isn't found, the system display "**ID** isn't found" massage.
4. Else The system verifies from **Password**.
5. If **Password** isn't correct, the system display "**ID** isn't correct" massage.
6. Else the system switches into the Actor Form.

Use case: Enroll in Course

Diagram



Brief Description

For student can enroll in more than one course.

For doctor can teach more than one course.

Initial Step-By-Step Description

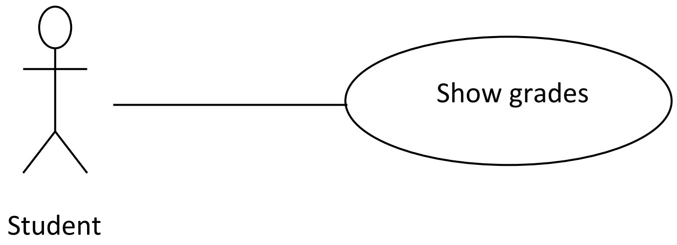
Before this use case can be initiated, the actor log in the Faculty Management System successfully and actor access the particular form of actor.

1. The system displays the available courses.
2. The Actor selects the desired course.
3. The Actor press on the enroll button.
4. The system should relate this actor to desired course in Database.

Student Use Cases

Use case: Show Grades

Diagram



Brief Description

Student only shows the grades of every course, He enrolled in these courses, and can't edit into these grades.

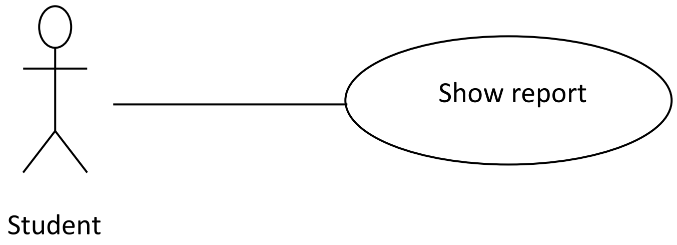
Initial Step-By-Step Description

Before this use case can be initiated, the student log in the Faculty Management System successfully and student access the student form. Must enrolled in at least one course.

1. The system displays the grid-view about student’s courses as Course ID , Course Name and the Grades.

Use case: Show Report

Diagram



Brief Description

Student only shows the report that Doctor assign to him.

Initial Step-By-Step Description

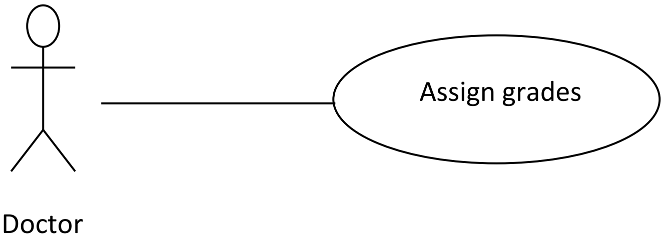
Before this use case can be initiated, the student log in the Faculty Management System successfully and student access the student form. Must enrolled in at least one course. Must Doctor assign the report first.

1. The system displays the Report.

Doctor Use Cases

Use case: Assign Grades

Diagram



Brief Description

Doctor will assign the grades to specific Student in The specific Course.

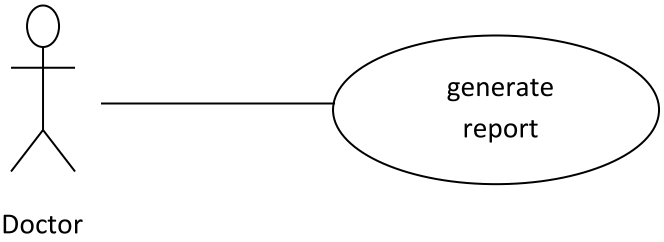
Initial Step-By-Step Description

Before this use case can be initiated, the doctor log in the Faculty Management System successfully and doctor access the doctor form. And the students are enrolled to the courses, that doctor teach them.

1. Doctor enters Student **ID**.
2. The system verifies from **ID.**
3. If **ID** isn't found, the system display "**ID** isn't found" massage.
4. Else The system loads the student data from Database.
5. Doctor enters the grades in the place assigned for them.
6. Doctor presses the confirm button.
7. The system will add these to Database.

Use case: Generate Report

Diagram



Brief Description

Doctor will assign the report to specific Student.

Initial Step-By-Step Description

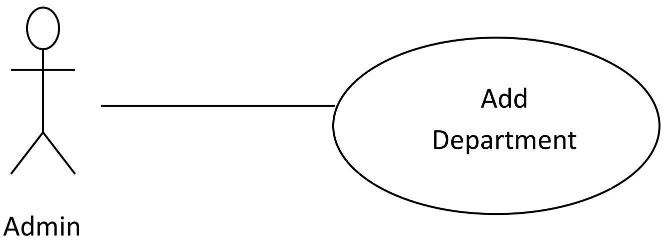
Before this use case can be initiated, the doctor log in the Faculty Management System successfully and doctor access the doctor form. And the students are enrolled to the courses, that doctor teach them.

1. Doctor enters Student **ID**.
2. The system verifies from **ID.**
3. If **ID** isn't found, the system display "**ID** isn't found" massage.
4. Else Doctor enters the report data.
5. Doctor presses the confirm button.
6. The system will add these to Database.

Admin Use Cases

Use case: Add Department

Diagram



Brief Description

Admin will add new department.

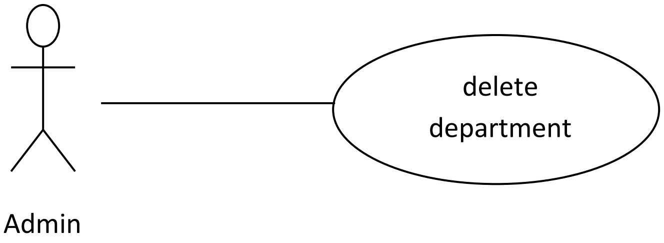
Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form.

1. The Admin enters Department data.
2. The Admin presses the add button.
3. The system will add the new department into Department table in Database.

Use case: Delete Department

Diagram



Brief Description

Admin will remove the department and all students who belong to it.

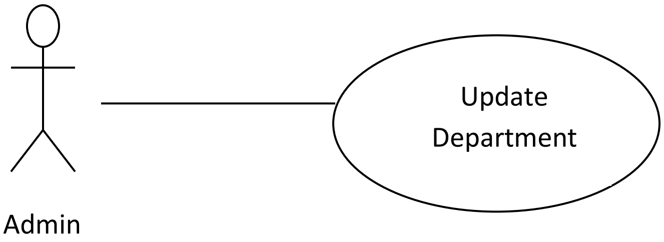
Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form.

1. The Admin selects the department to remove.
2. The system display confirmation massage.
3. The Admin confirms removing.
4. The system deletes the department from departments table.
5. The system deletes the students from students table are associated to the department.
6. The system deletes the department from doctor table and course table.

Use case: Update Department

Diagram



Brief Description

Admin will update department data.

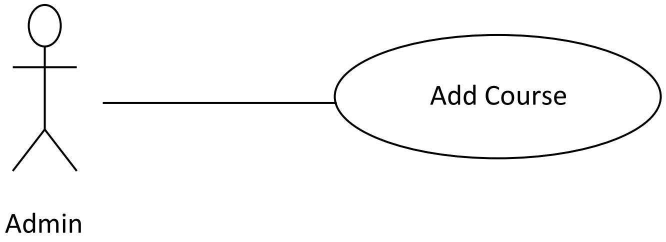
Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form, and at least one department is available.

1. The Admin selects desired Department.
2. The system shows the desired Department data.
3. The Admin change the data that should change
4. The Admin press the change button.
5. The system will change data of the department in Database.
6. The system must change these data also in students and courses associated to it.

Use case: Add Course

Diagram



Brief Description

Admin will add new course.

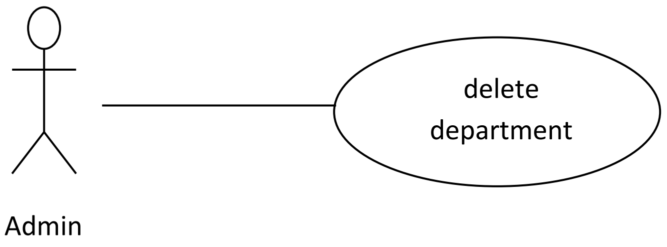
Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form, and at least one department is available.

1. The Admin enters Course data.
2. The Admin presses the add button.
3. The system will add the new course into Course table in Database.

Use case: Delete Course

Diagram



Brief Description

Admin will remove the course and all data that belong to it.

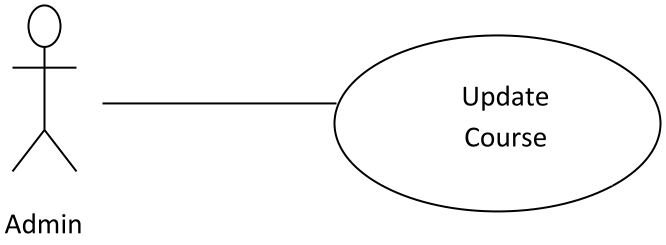
Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form, and at least one course is available.

1. The Admin selects the course to remove.
2. The system display confirmation massage.
3. The Admin confirms removing.
4. The system deletes the course from courses table.
5. The system deletes the course from doctor table, course table and Department table.

Use case: Update Course

Diagram



Brief Description

Admin will update course data.

Initial Step-By-Step Description

Before this use case can be initiated, the admin log in the Faculty Management System successfully and admin access the admin form, and at least one course is available.

1. The Admin selects desired Course.
2. The system shows the desired Course data.
3. The Admin change the data that should change
4. The Admin press the change button.
5. The system will change data of the course in Database.
6. The system must change these data also in students, courses and Department associated to it.

Process Flow

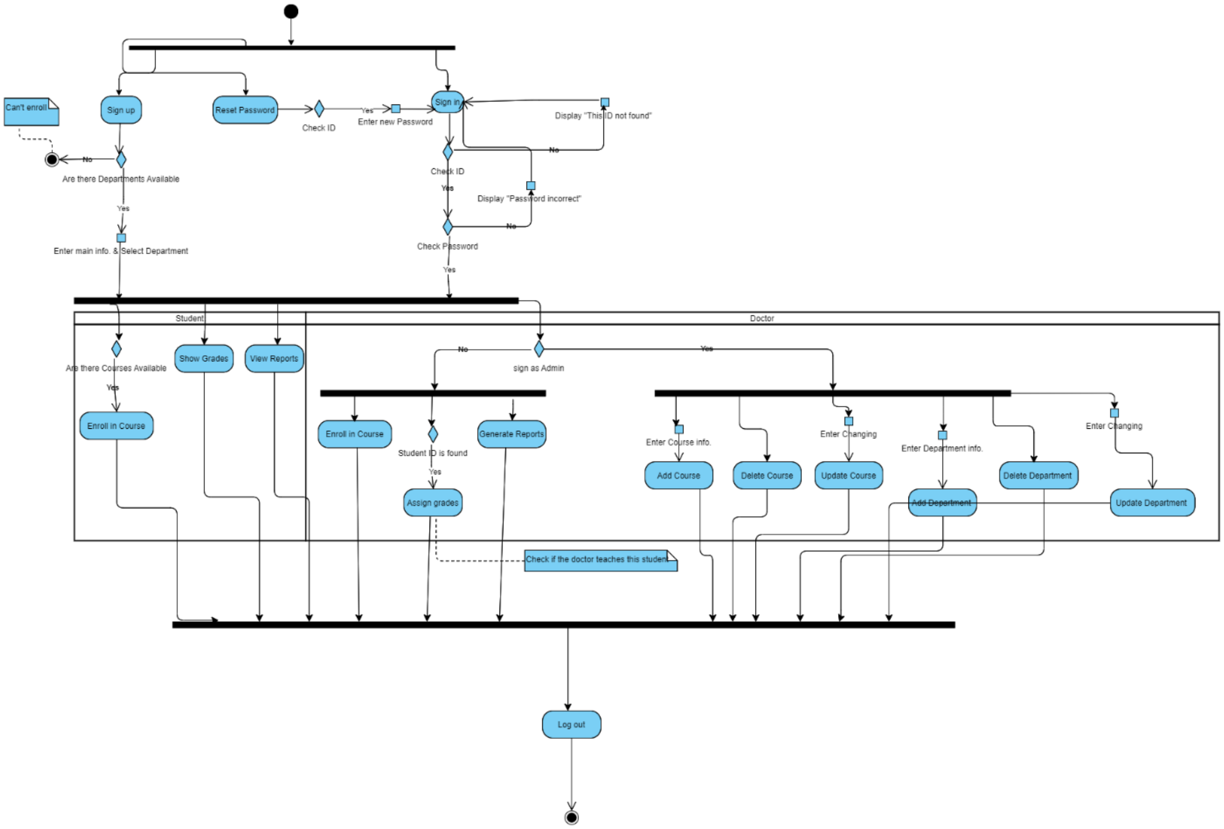


Figure 3: Activity Diagram

Forms Description

When the user opens the Application, a sign-in form appears, where user chooses his role (student, doctor) and logs in. Or he goes to the sign-up form, where he creates a new account and applies to a department.

Then the system moves to the form of the user role.

A form appears for the student with a set of tabs (enroll in a course, show courses grades, show the report)

The doctor appears in a form to choose a course (admin or regular doctor)

The doctor shows him a form with a set of tabs (Generate Report, Assign Grades)

The admin has a form showing a set of tabs (add a department, update a department, remove a department, add a course, update a course, remove a course).

Non-Functional Requirements Specification

Logical Structure of the Data

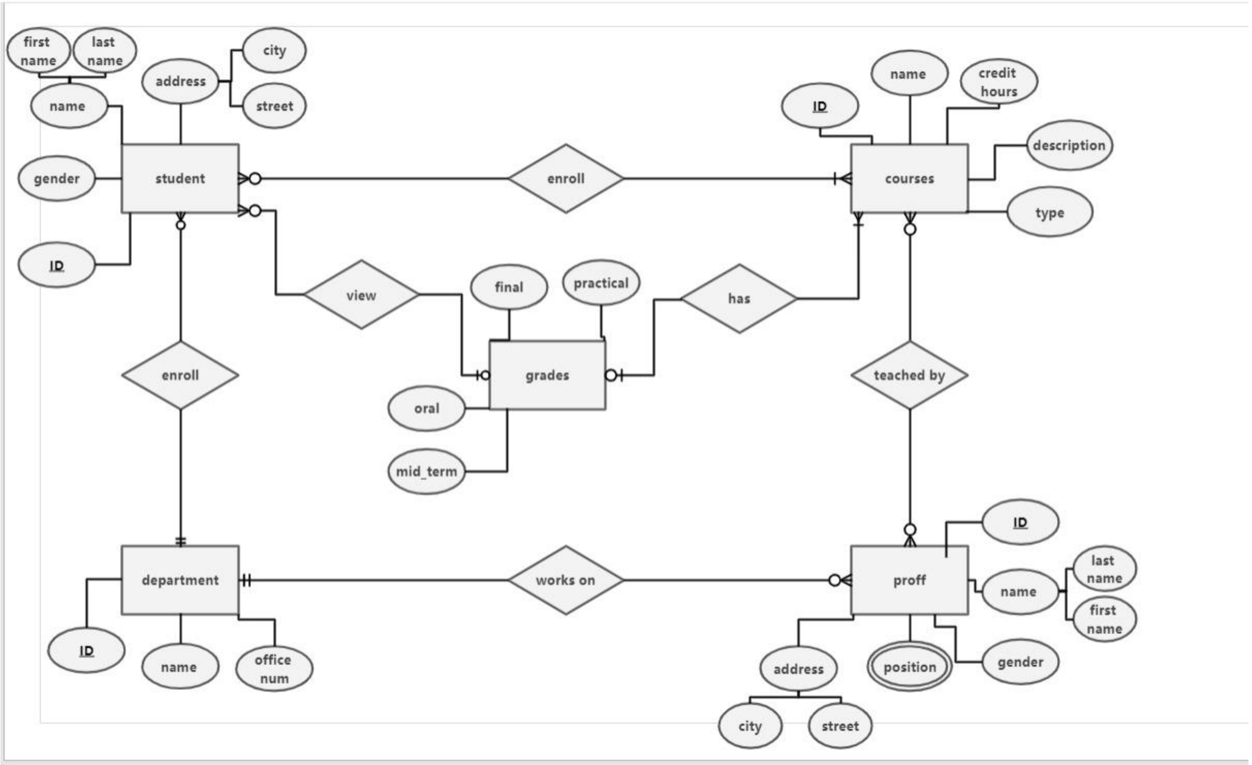


Figure 4: Erd Diagram

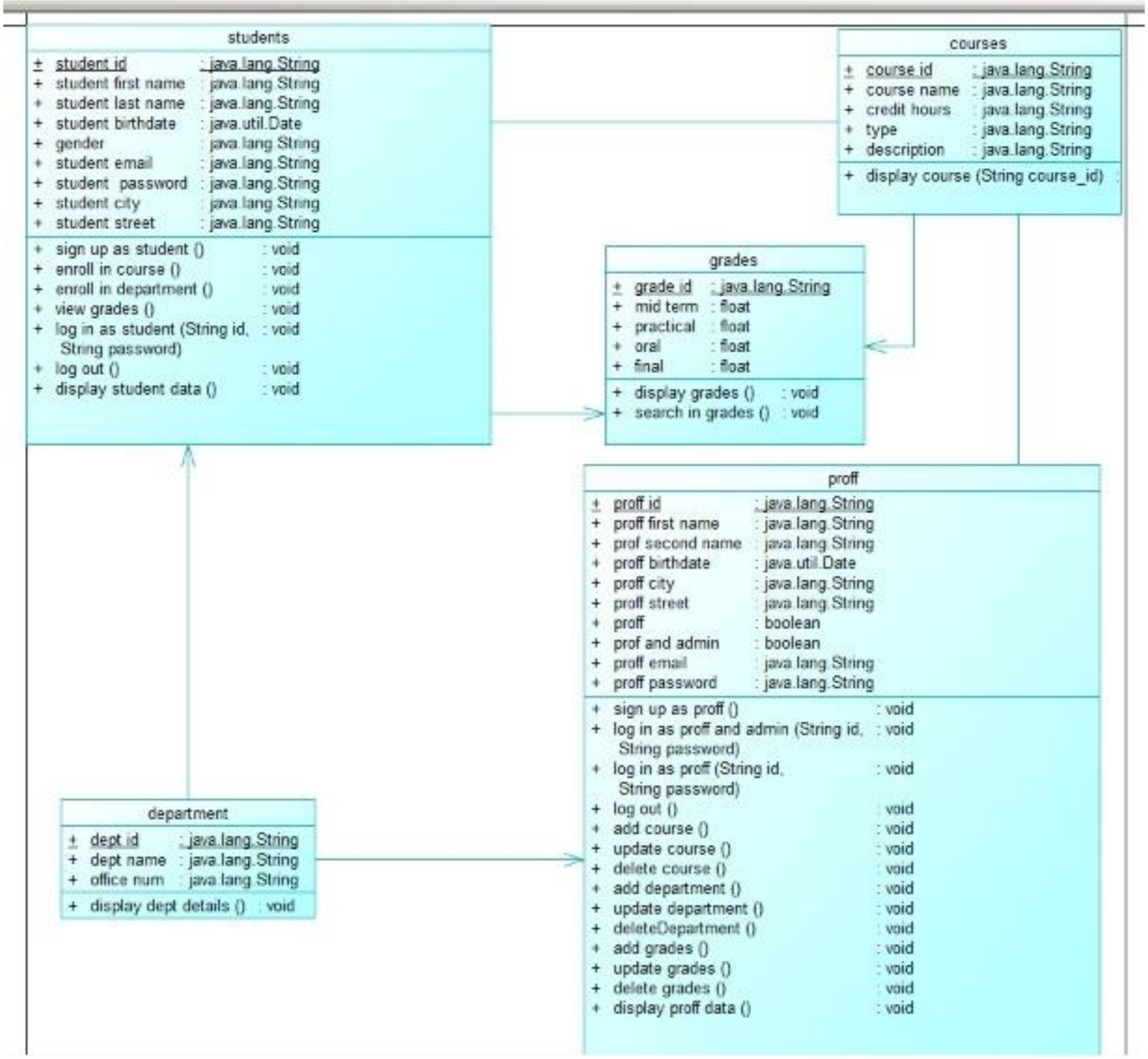

students
+ student id : java.lang.String
+ student first name : java.lang.String
+ student last name : java.lang.String
+ student birthdate : java.util.Date
+ gender : java.lang.String
+ student email : java.lang.String
+ student password : java.lang.String
+ student city : java.lang.String
+ student street : java.lang.String
+ sign up as student () : void
+ enroll in course () : void
+ enroll in department () : void
+ view grades () : void
+ log in as student (String id, String password) : void
+ log out () : void
+ display student data () : void
courses
+ course id : java.lang.String
+ course name : java.lang.String
+ credit hours : java.lang.String
+ type : java.lang.String
+ description : java.lang.String
+ display course (String course_id) :
grades
+ grade id : java.lang.String
+ mid term : float
+ practical : float
+ oral : float
+ final : float
+ display grades () : void
+ search in grades () : void
proff
+ proff id : java.lang.String
+ proff first name : java.lang.String
+ prof second name : java.lang.String
+ proff birthdate : java.util.Date
+ proff city : java.lang.String
+ proff street : java.lang.String
+ proff : boolean
+ prof and admin : boolean
+ proff email : java.lang.String
+ proff password : java.lang.String
+ sign up as proff () : void
+ log in as proff and admin (String id, String password) : void
+ log in as proff (String id, String password) : void
+ log out () : void
+ add course () : void
+ update course () : void
+ delete course () : void
+ add department () : void
+ update department () : void
+ deleteDepartment () : void
+ add grades () : void
+ update grades () : void
+ delete grades () : void
+ display proff data () : void
department
+ dept id : java.lang.String
+ dept name : java.lang.String
+ office num : java.lang.String
+ display dept details () : void

Figure 5: Class Diagram

Other

Programming language: **JAVA**

Framework: **Swing**

RDMS: **MySQL**