Organizing College Studies

*Project Report*

Carl Habib, Sameeh Moghrabi, and Amer Taleb Al Ajouz

**Abstract**—Every university is in massive need of a system that can manage all the information of all students and employees. Our project consists of developing a system that can be used by universities in order to handle all issues regarding students, staff and faculty. The system is composed of a Database that stores all the information, in addition to a user interface which allows the user to interact with the Database. The main goal of this project is to improve and optimize the way the information is handled and stored.

**Index Terms** – University, Students, Staff, Faculty.

—————————— ◆ ——————————

# contents

[1 Introduction 1](#_Toc468574044)

[2 Organization 1](#_Toc468574045)

[3 Database 2](#_Toc468574046)

[4 User Interface 5](#_Toc468574047)

[5 Experimental Evaluation 7](#_Toc468574049)

[6 Extra features 8](#_Toc468574050)

[7 Conclusion 8](#_Toc468574051)

# 1 Introduction

when a student wants to add, drop or check a course, what should be actually accessed is a database containing all the information regarding all the courses available at the university. The database of this project basically contains all the information regarding the students enrolled in the University, the courses offered, information about faculty, programs offered in addition to information regarding the different schools. To create the database, we used the knowledge we learned in our database systems course with the help of MySQL workbench. The user interface was developed using JAVA.

# 2 Organization

The first step in our project was to decide on the functionalities the database was going to provide for the student, staff, faculty and admin.

A student can:

1. View the sections he’s currently registered in
2. Enroll in classes during registration period
3. Drop classes
4. View the grades of courses taken previously
5. View his financial aid details
6. Change his account password

A faculty member can:

1. View his details
2. View the courses he is teaching
3. View the sections he is teaching
4. View the students registered in the sections
5. View his advisees
6. Change his account password

A staff member can:

1. View his details
2. View the students he supervises in case he has any
3. View the information of a student he selects
4. Change his password

An admin can do everything the previous users can do and can view everything. In addition to that he can add and delete students and courses from the database.

# 3 Database

## 3.1 CDM

The next step was to establish a solid CDM that would describe specifically what we wanted our project to contain disregarding any implementation details. We came up at first with many CDMs but we were finally able to choose the best one that would reflect our view of the most suitable University System.

The conceptual schema contains 9 Entities which are : Student, staff, professor, school, department, program, course, course section and classroom. Each of these entities has its specific attributes that accurately describes it.

A **student** has attributes studentID, first name, last name, cumulative GPA, his major, his minor (if any), his major GPA, email and password. To uniquely identify between each student, we use the studentID since each student has a unique ID.

A **staff** has attributes staffID, first name , last name, salary, department, email, password and type (full time or part time). We identify different staff by the staffID.

A **professor** has attributes facultyID, first name , last name, salary, department, email, password and type (full time or part time) and office (if any). We identify different staff by the facultyID.

**Course** has attributes crsID, subject, number, level and title. It is uniquely identified using the crsID attribute.

**CourseSection** has attributes sectionID, term, schedule and type and we uniquely identify it using the sectionID attribute.

**ClassRoom** has attributes roomID, building, roomNumber, capacity and type, and is identified using roomID.

**School** has attributes schoolID, name, doc, accreditation, dean and assDean. We identify the school by its ID.

**Department** has attributes departmentID, doc, name. We identify it by its ID

**Program** has attributes programID, programName, programLevel, creditsRequired and accreditation. We identify the program by its attribute programID.

After defining all the entities and their attributes we define the relationships between them in addition to the cardinality constraints.

A **Student** has relationships with many entities. First of all, when a student starts university, he choses the program he wants to study, and he can only choose 1 and only 1 program. A student can register 1 or many courses, but attends a section of each course and not the whole course, that’s why we have to distinguish between attending and registering a course in this case. In addition, each student has an advisor and may have managers in case he has financial aid or is employed at the university.

A **Staff** can manage many students and can also have no students to manage. A staff is also employed by the department he works in at the University.

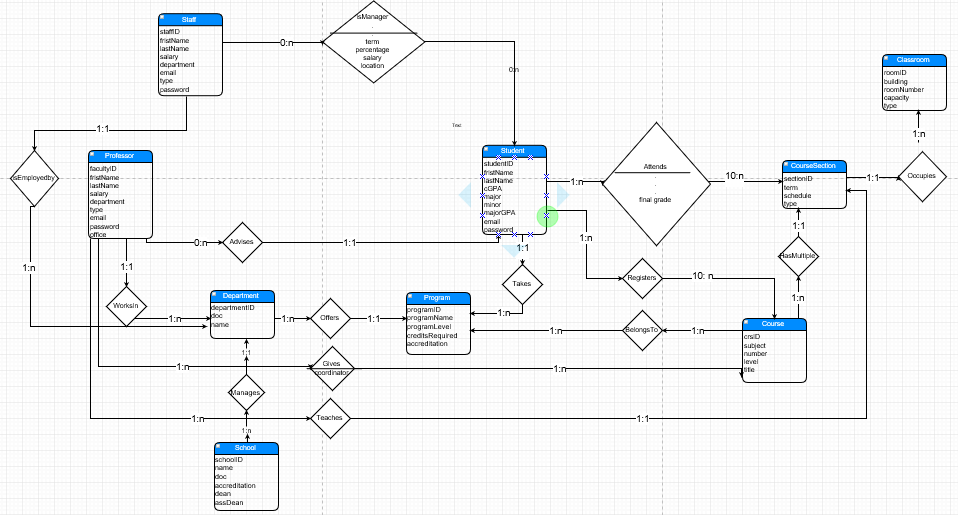
A **Professor/Faculty** can advise many students and can also not advise any student. He/She can work in in 1 and only 1 department, give many courses and teach many course sections.

A **Department** can have many instructors and staff working in it. It also offers multiple programs and is managed by 1 and only 1 school. On the other hand, a **School** can manage many departments.

A **Program** belongs to one and only one department, offers many courses and can be taken by many students.

A **Course** can be taught by one or many instructors, can belong to one or many programs, can be registered by many students, but should have a minimum of 10 students. A course can also have multiple course sections.

A **Course Section** belongs to one and only one course, is taught by only one instructor, should have at least 10 students registered in it and occupies one and only one classroom. A **class room** on the other hand can host more than one lecture during a university day.



## 3.2 LDM

After creating and finalizing the conceptual schema, we moved to the logical schema, which is the main reference when creating the database itself. Our LDM consists of a total of 14 Relations. Relations Student, Staff, Faculty, Department, School, Program, Course, Course Section and Class Room are relation based on the entities we had in our CDM. Looking at the cardinality constraints we have in our project, we have to create some extra relations.

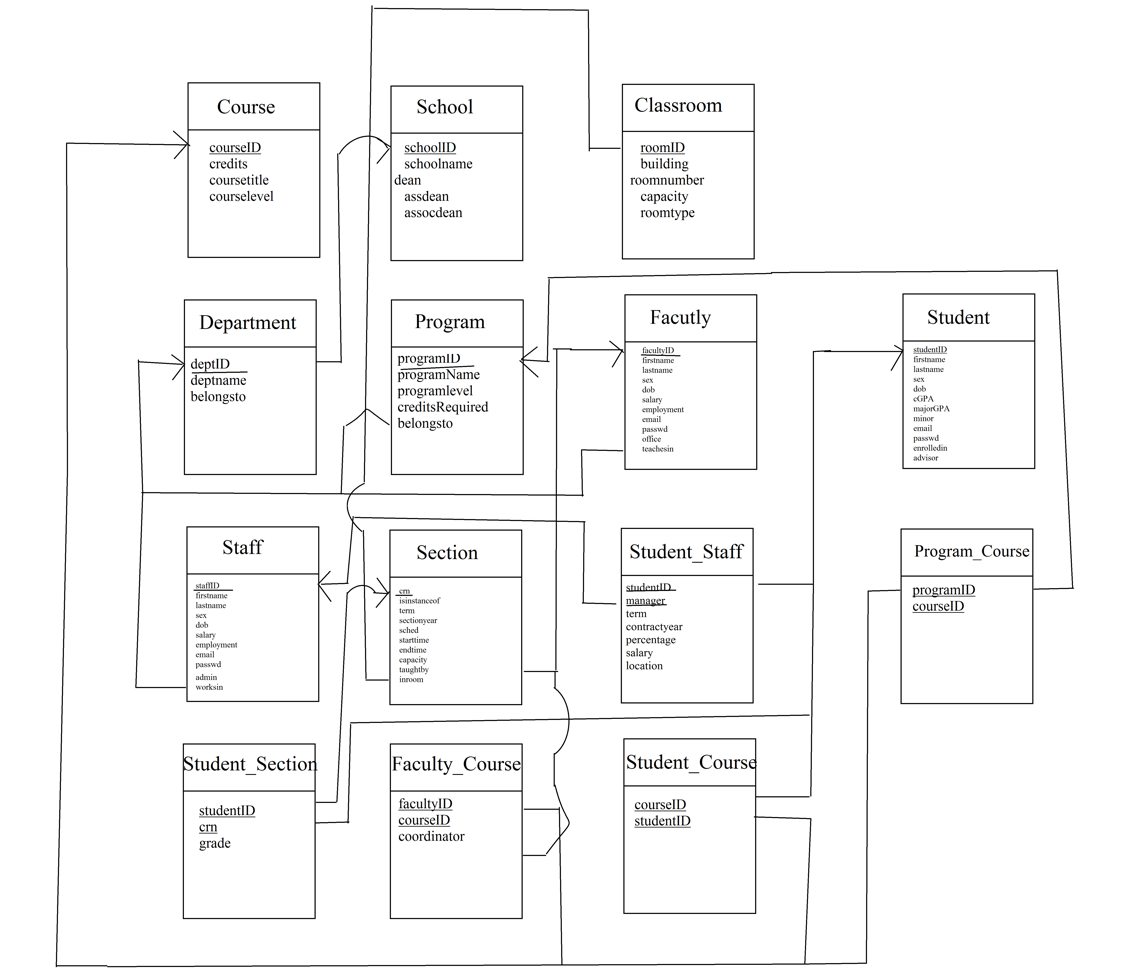
**Student\_Staff** is an intermediate relation created for student and staff since we have a maximum of n on both sides. This relation will contain the primary key of both student and staff relations as foreign keys but they will also act as a primary key. Other than these two attributes, we need to add the attributes of the relationship connecting both entities which is “isManager”.

**Student\_Section** is also an extra relation created to connect student and section. It will have the primary keys of both these entities as foreign keys, and the combination of these two will act as the primary key of the relation. We need to also add the attribute grade which is the attribute of the relation connecting the entities.

**Faculty\_Course** is a combination of entities faculty and course. The primary key of this relation is a composite key having both facultyID and courseID. An instructor can be the course coordinator that’s why we need to also add this attribute to the relation.

**Student\_Course** links both student and course entities together. The primary key of this relation is the combination of the IDs of both entities. This relation has no extra attributes to be added.

**Program\_Course** is a relation that connects both program and course entities. It has two foreign keys, the first being the programID which references the primary key of the relation Program. The second foreign key is courseID which references course. Same as the previous relation mentioned, this also has no extra attributes to be added to it.



## 3.3 SQL implementation

To create the database, we used MySQL. The implementation of the database was done by following the LDM schematic, where 14 tables were created following the corresponding 14 relations that are in the LDM. The attributes we created were the same as the attributes of the relations in the LDM itself. After creating the tables in the database, we inserted some values in order to use it and test queries on it.

# 4 User Interface

## 4.1 General Overview

When first accessing our application, the first thing that appears is a login page, which is made of a grid pane and contains two text fields and a button which allows you to login. The user enters his username and password in the text fields and then clicks login and then the user is taken to his page. For each type of user, we have created a specific customized scene. Each scene is represented with a border pane, the top of this pane has the name of the user in addition to his ID and other details depending on the type of user. The top also contains a Detail button for the user to see his personal details, in addition to a Sign out button. The bottom also contains buttons and combo boxes which displays to the user what are the different functionalities he has. Each button and combo box in our interface has its own listener which does a specific predefined command once clicked/selected. On each click/selection, the content of the center of the border pane changes.

Our project has multiple functionalities that the user can benefit from. These functionalities mainly target the student, staff (admin and non admin) in addition to faculty. To interact with the database, we created many buttons that have different functionalities.

## 4.2 Student

A student has to first enter his login credentials in order to access his own page. After accessing the student main page, the interface offers a variety of buttons:

1. “Current Sections” shows the student in which courses is he enrolled this semester
2. “Enroll” shows the current available courses, and when you select a course you click the button “Register in Selected Section” to register the course.

Note that when a section is full, a student can not register in it, and an error message will pop telling the student that the section is full.

1. “Drop” button allows a student to drop any course he’s currently registered in
2. “Grades” button shows the student the final grade he took in all previous courses
3. “Financial Aid” button shows the student the details regarding the financial aid
4. “Change password” does what the name says, it allows the student to change his account password

When the enroll button is clicked, the center of the border pane of the page will be made of another border pane, where the first border pane contains the buttons that are available, and the other one is a pane to display the answer.

## 4.3 Faculty

Regarding the user interface, a faculty has to first enter his login credentials in order to access his own page. After accessing the student main page, the interface offers a variety of buttons:

a. “View Courses” allows the faculty member to view the course that he’s teaching

b. “View Sections” will work when the instructor selects one of the courses he’s teaching. This button will show the instructor the information regarding the sections of the course selected.

c. “View Students” will show the list of students enrolled in a specific section

d. “View all sections” will allow the user to see all the sections of all previous course he has given.

e. “View advisees” will allow the instructor to see the information of the students that he advises.

e. “Change password” has the same functionality as the change password in student.

## 4.3 Non Admin staff

A non admin staff has to also enter his login credentials. The main page offers mainly two buttons:

1. “View Details” which allows the member to view all his details.
2. “View FA students” will allow the staff member to view all the information of all the students that he supervises.
3. “View Selected Student” is a button that will show up when the FA students are shown, one student is selected and the the user can view the information of this student after clicking this button.
4. “Change password” has the same functionality.

## 3.4 Admin Staff

Regarding the user interface, after entering his login credentials, the main page offers a big amount of options. We first created 5 combo-boxes where each combo-box contains options regarding the title of the combo-box.

Student has all the options related to the student him self:

1. “View All Students” allows the admin to view all the students that are enrolled in the university
2. “View Student’s courses” allows the admin to view all the courses taken by a specific student
3. “Add student” allows the admin to add a student to the database
4. “Delete student” allows the admin to delete a student from the database
5. “View FA students” will allow the staff member to view all the information of all the students that he supervises.
6. “View all sections” allows the admin to view all the sections of all the courses offered

Instructor has all the options related to the faculty:

1. “View all instructors” allows the admin to see all the faculty members of the university
2. “View instructor’s courses” allows the admin to view all the courses being taught by an instructor after selecting the name he wants.
3. “Add instructor” allows the admin to add a faculty to the database
4. “Delete instructor” allows the admin to delete a faculty from the database
5. “Change instructor’s salary” allows the admin to change the salary of a specific instructor

Course has the options related to the courses:

1. “View courses” allows the admin to view all the courses at the university
2. “View courses sections” allows user to view the different sections of a course after being selected
3. “Add course” allows the admin to add a course to the database
4. “Delete course” allows the admin to delete a course from the database

Section displays all information regarding all sections that were given at the university.

1. “View All Section” allows the user to view all given sections in all academic semesters at the unviersiy
2. “View Section’s students” allows the user to see the students registered in a specific course
3. “Add section” allows the admin to add a course section to the database
4. “Delete section” allows the admin to delete a course section from the database

# Staff combo-box displays all the options related to staff members:

1. “View all staff” allows the admin to see all the registered staff members at the university
2. “View staff’s students” allows the admin to see the student assistants of a specific member
3. “Add staff” allows the admin to add a staff to the database
4. “Delete staff” allows the admin to delete a staff from the database
5. “Change staff’s status” allows an admin to grant admin privileges to another staff
6. “Change staff’s salary” will change the salary of a specific staff

Other than these options, an admin has a button called “Expert”. This button allows the admin to manipulate the database by directly writing the query he wants in a text field which will be sent to the database, processed and then an output will be displayed.

# 5 Experimental Evaluation

In this section we will provide some query tests to show how our project works, since our project contains a lot of queries, so we decided to chose some from each page we have. As we mentioned earlier, our project is mainly composed of buttons to help the user chose what he wants, and at the same time admin has an expert option which lets him write any query he wants. Each button in our interface has a listener which tells the program what to do when this specific button is pressed.

Student page features:

“Details” Button: SELECT \* FROM STUDENT WHERE studentID = loginID, where logicID is a variable created in java in order to know which student is logged in, so we can use this information in order to specify which information we want to get the info from.

“Enroll” Button: SELECT distinct courseID,credits, coursetitle, courselevel FROM section S , COURSE C where S.isinstanceof = C.courseID and term = 'fall' and sectionyear = 2016 and c.courseID not in (Select se.isinstanceof from STUDENT\_SECTION S , Section se WHERE S.crn = se.crn AND grade = 'In Progress' AND studentID= loginID .

This query will display all the available courses for the user, so all he has to do is to simply select one of the and click “Register in selected section”.

“Drop” Button:

First of all, when clicking the drop button, the courses currently registered by the student will appear using this query:

SELECT se.\* from STUDENT\_SECTION S , Section se WHERE S.crn = se.crn AND grade = 'In Progress' AND studentID=loginID ;

In addition to a “drop selected section” button, and this button when clicked will perform the following query:

DELETE FROM STUDENT\_SECTION WHERE studentID=loginID AND crn =item.getCrn() where item is a variable of type section which stores the selected section, and from this selected section we get the crn of the course.

Faculty page features:

“View Courses” Button: SELECT DISTINCT co.\* from section S , FACULTY\_COURSE C , Course co where S.isinstanceof = C.courseID and S.isinstanceof = co.courseID and term = 'Fall' and sectionyear = 2016 AND facultyID =loginID

“View All Sections” Button: SELECT \* FROM SECTION WHERE taughtby = loginID

“View Advisees” Button : SELECT \* FROM STUDENT WHERE advisor =loginID

Staff page features:

“View FA Students” Button: SELECT SS.studentID, SS.manager, S.firstname, S.lastname, SS.term, SS.contractyear, SS.percentage, SS.salary, SS.location from STUDENT\_STAFF SS, STUDENT S where SS.studentID = S.studentID AND manager= loginID

“Change Password” Button: UPDATE staff set passwd = passwordTF.getText() where staffID = loginID

passwordTF.getText() is a function that gets the password that was inserted in the text field by the user.

Admin page features:

“View Courses” option in combo-box “Course”: SELECT \* FROM SECTION WHERE taughtby = idTF.getText()

“View Section’s Students” option in combo-box “Section”: SELECT Student.\* FROM STUDENT\_SECTION JOIN STUDENT on crn =item.getCrn() AND student.studentID = student\_section.studentID , where item.getCrn() is a function that gets the CRN of the selected course.

“Add Instructor” option in combo-box “Instructor”: INSERT INTO FACULTY VALUES(idTF.getText() ,fnTF.getText() , lnTF.getText(), sexTF.getText() , dobTF.getText() ,salaryTF.getText(),employmentTF.getText() ,emailTF.getText(), passwordTF.getText(), officeTF.getText(), teachesinTF.getText() )

where all of these getText methods will get the texts that are added by the user to the text fields.

“Delete Student” option in combo-box “Student”: DELETE FROM STUDENT WHERE studentID =item.getId()

# 6 Extra features

Due to the lack of time, we were not able to improve more our project, but we can add many extra features that improve the system even more. First of all, we can improve the admin and make him be able to create a new school and a new program, he can also add new students to any class. Also, an admin can update any thing he likes, like for example he can update the information of a student or a faculty member. In addition to that, we could also create an expert part for each user but each user has his own privileges depending on his type. For example a student’s expert scene allows the student to see his courses, register courses etc.. but does not allow him to add a staff member or add a course.

# 7 Conclusion

In conclusion, we need to add that regardless of the huge amount of time spent in implementing this project, from the start till the end, this project has a lot of features that can be added to all the users, but due to time limitations and lack of knowledge of things we did not learn yet, we were not able to add everything we though off. This project simply reflects the basic things a university database can do, and can surly be improved a lot more.

In general, since this project is a hands on experience, we learned more about creating a manipulating databases, we also learned about JavaFx and how to create Graphical User interfaces.