|  |  |
| --- | --- |
|  |  |

***Faculty of Engineering***

***Internet Programming Project Report***

***Student Exam Seating***

**CCE Department – Fourth year - English Section**

**Khalil Adib, 50474**

**Ali Ismail, 50494**

**Supervised By**

**Dr. Ahmad Koubeissi**

Monday, February 17

**Table of Content**

Table of Contents

[Abstract 4](#_Toc32800017)

[Introduction: 4](#_Toc32800018)

[Scope: 4](#_Toc32800019)

[More Details: 4](#_Toc32800020)

[Database System Design 6](#_Toc32800021)

[CDM Model: 6](#_Toc32800022)

[PDM Model: 6](#_Toc32800023)

[Table of Procedures: 7](#_Toc32800024)

[Website Design 11](#_Toc32800025)

[Login: 11](#_Toc32800026)

[Failure Login: 11](#_Toc32800027)

[Create Account: 12](#_Toc32800028)

[Students: 12](#_Toc32800029)

[Courses: 13](#_Toc32800030)

[Exams: 13](#_Toc32800031)

[Hall: 14](#_Toc32800032)

[Registration: 14](#_Toc32800033)

[Seating: 15](#_Toc32800034)

[Faculty: 15](#_Toc32800035)

[Department: 16](#_Toc32800036)

[Academic Year: 16](#_Toc32800037)

[Users: 17](#_Toc32800038)

[Flowchart 18](#_Toc32800039)

[Conclusion 20](#_Toc32800040)

**List of Figures**

[Figure 1 CMD label of the database 6](#_Toc32800041)

[Figure 2 PMD model of the database 6](#_Toc32800042)

[Figure 3 Login Webpage 11](#_Toc32800043)

[Figure 4 Not Instructor User 11](#_Toc32800044)

[Figure 5 New user account 12](#_Toc32800045)

[Figure 6 Student Page 12](#_Toc32800046)

[Figure 7 Course Page 13](#_Toc32800047)

[Figure 8 Exam Page 13](#_Toc32800048)

[Figure 9 Halls Page 14](#_Toc32800049)

[Figure 10 Registration Page 14](#_Toc32800050)

[Figure 11 Seating Page 15](#_Toc32800051)

[Figure 12 Faculty Page 15](#_Toc32800052)

[Figure 13 Department Page 16](#_Toc32800053)

[Figure 14 Academic Year Page 16](#_Toc32800054)

[Figure 15 Users Page 17](#_Toc32800055)

[Figure 16 Flowchart to find the best hall 18](#_Toc32800056)

# Abstract

Exam Hall Seating Arrangement System is developed for universities to simplify examination Hall allotment and seating arrangement. The propose of developing exam hall seating arrangement system is to get rid the traditional ways of conducting exams and save more time. Newly invented concept helps for the staffs to generate their exam hall arrangement easily.

**Keywords:** Hall arrangement.

## Introduction:

Examination Hall Management System is developed to target universities to simplify the allocation of examination halls. Previously, the head of department used to waste his time distributing courses over classes. This is the traditional way adapted by the majority, but now, the software will replace this method and distribute the required exams to the exam halls automatically with one press of a button.

## Scope:

The scope of the project is the design a web interface that will be given to a university for future use.

## More Details:

First, there are academic faculty and departments in the university. Each faculty has name and id. Each department belongs to a faculty and has id and name. The user manage the faculty, each faculty has username, password and email. Each faculty managed by 1 user, and each user can manage several faculty.

Students in university belong to a department and max 1 department. Each student has id, name.

Each hall belong to a faculty and each hall has id, number of seats, and status. The courses has name and id.

The most important thing, the exam has date, start time, end time, duration and type of exam. The exam occurs in hall in specific time and date. Each course has 2 type of exam, partial or final exam

The user can manage all the staff in the program related to the faculty he manage.

2 types of Reports should be generated from application: one that shows for a specific student, the exams dates and assigned rooms. Second that shows students assigned to each room during exams.

# Database System Design

## CDM Model:

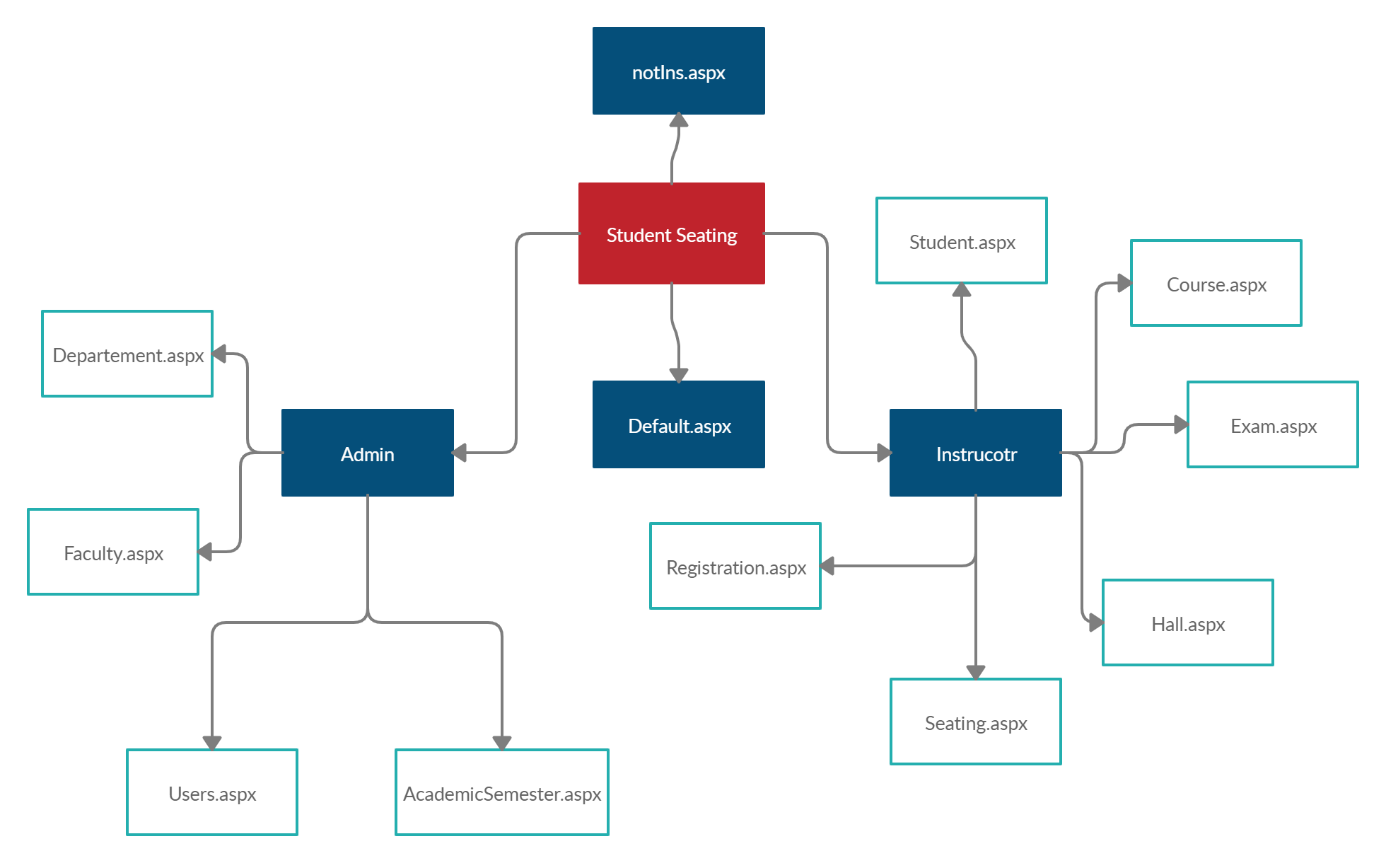


Figure 1 CMD label of the database

## PDM Model:

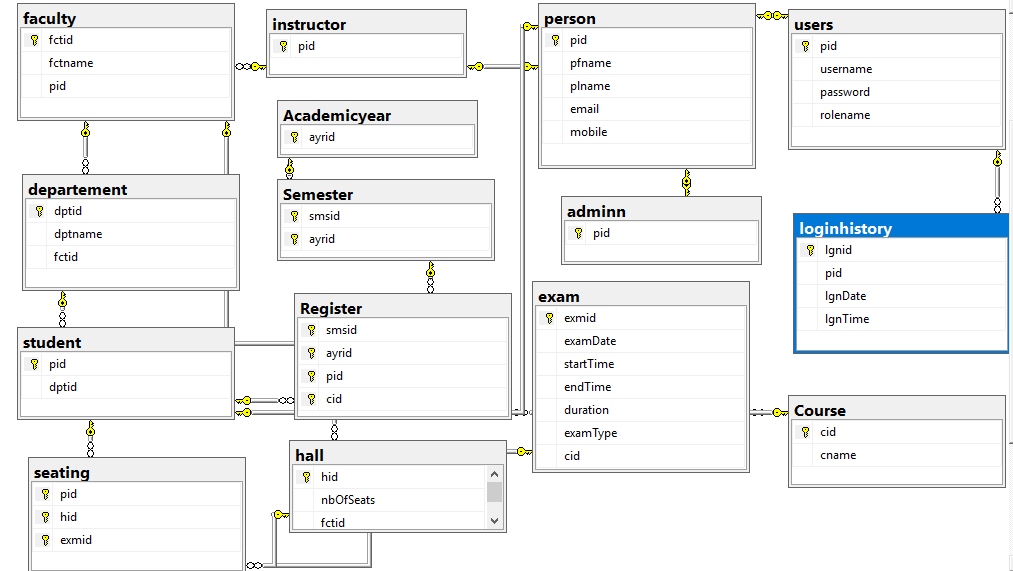
****

Figure 2 PMD model of the database

## Table of Procedures:

|  |  |
| --- | --- |
| Name of Procedure | Result |
| select\_all\_faculty | Faculty ID, Faculty Name |
| users\_select\_all | ID and Name if the user is instructor |
| acadyear\_select\_all | Academic Year, Semester |
| select\_instructors\_Fullname | Name, ID of an instructor |
| faculty\_insert | Insert to the faculty table new fctid, fctname, pid |
| faculty\_delete | Delete an faculty by enter fctid |
| faculty\_update | Update name of an faculty by enter the fctname and its fctid |
| select\_departement\_by\_fctid | Select an department represent to a special faculty(fctid) |
| insert\_departement | Insert an department(dptid,dptname) represent to a special faculty(fctid) |
| delete\_departement | delete an department(dptid,dptname,students) represent to a special faculty(fctid) |
| departement\_update | Update name of an department by enter its dptid |
| insert\_academicYear | Insert an academic year to the table Academic Year |
| insert\_FallSems | Insert a fall semester(semsid) according to a special academicYear(ayrid) in the semester table |
| insert\_SpringSems | Insert a spring semester(semsid) according to a special academicYear(ayrid) in the semester table |
| check\_username | Check if this username is an user |
| insert\_new\_user | Add new user to the user table by enter (pid, username, password, 'new user') |
| select\_all\_users | Select ID, username, rolename |
| insert\_new\_person | Add new person |
| insert\_into\_instructor | Add an instructor |
| make\_instructor | Make an user rolename as instructor |
| get\_id\_by\_username | Display the id of an user |
| remove\_instructor | delete instructor |
| make\_instructor\_user | return the instructor rolename from an instructor to a ‘new user’ |
| delete\_user | Delete user |
| select\_student\_in\_faculty | Select a student(Name, ID, Department) |
| insert\_new\_std | Add new student(pid, dptid) to the student table |
| std\_update | Change name, id of a student |
| select\_all\_courses | Display courses |
| insert\_new\_course | Add new course(name, id) to the course table |
| course\_update | Change the name of the course |
| select\_max\_exmID | Display for maximum exam ID |
| insert\_new\_exm | insert into exam table(exmid, examDate, startTime, endTime, duration, examType, cid) |
| select\_all\_exm | Display the (exmid, examDate, startTime, endTime, duration, examType, cid) of all exams |
| select\_hall\_by\_fctid | Display the ID hall and its available seats |
| insert\_new\_hall | insert into hall table a new(hid, nbOfSeats, fctid) |
| delete\_hall | Delete a hall |
| hall\_update | Update the number of available seats |
| register\_new\_course | Add new registered course into register table |
| view\_registered\_course | Display the ID and Name of courses registered |
| delete\_registered\_course | Delete an registered course |
| select\_number\_of\_student\_in\_course | Display the number of students in each course that registered |
| select\_count\_hall\_by\_fctid | Display number of hall in a faculty |
| get\_exam\_info\_by\_course\_info | Display (exmid, examDate, startTime, endTime, duration, examType, cid) of an exam for a course |
| check hall | Display the information of an exam of a hall |
| select\_student\_register\_in\_course | Display the ID of student that registered in a course |
| get\_examid\_by\_course\_info | Display ID of exam that represent to a course |
| student\_seating | To distribute students into halls |
| select\_hall\_student\_exam | To display the hall and its exam information in it |

*And these some of procedures code:*

* *create procedure select\_hall\_student\_exam*

*@pid int, @sms varchar(50), @aca varchar(50), @examType varchar(50)*

*as*

*select seating.hid as 'Hall', exam.examDate as 'Date', exam.startTime as 'Start', exam.endTime as 'End', exam.duration as 'Duration'*

*from seating, exam*

*where seating.exmid = exam.exmid and seating.pid = @pid and exam.examType = @examType*

*and seating.pid in*

*(select Register.pid*

*from Register, Course*

*where Register.cid = Course.cid*

*and Register.smsid = @sms*

*and Register.ayrid = @aca);*

*go*

* *create procedure select\_hall\_student\_exam*

*@pid int, @sms varchar(50), @aca varchar(50), @examType varchar(50)*

*as*

*select seating.hid as 'Hall', exam.examDate as 'Date', exam.startTime as 'Start', exam.endTime as 'End', exam.duration as 'Duration'*

*from seating, exam*

*where seating.exmid = exam.exmid and seating.pid = @pid and exam.examType = @examType*

*and seating.pid in*

*(select Register.pid*

*from Register, Course*

*where Register.cid = Course.cid*

*and Register.smsid = @sms*

*and Register.ayrid = @aca);*

*go*

# Website Design

## Login:

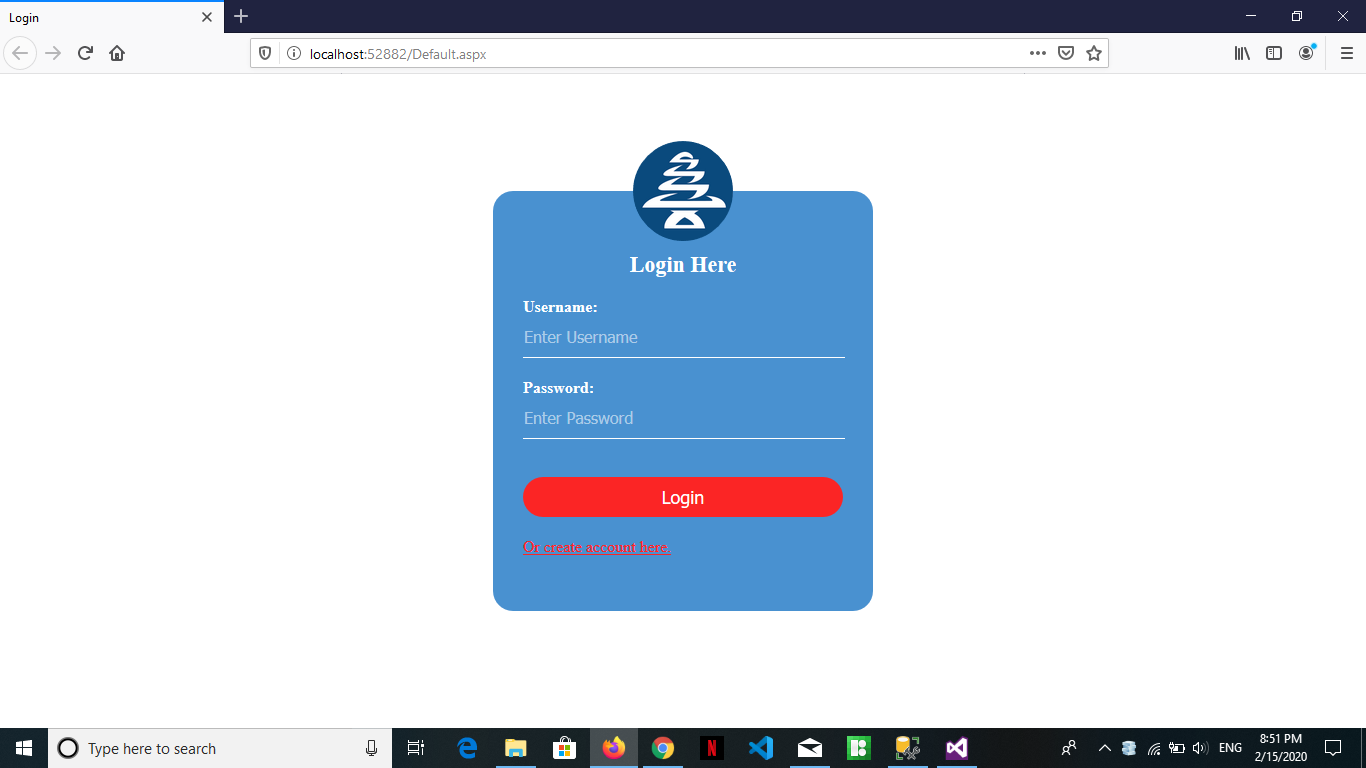
****

Figure 3 Login Webpage

This page is the first webpage that shown when the user need to work on the application. This page is for to ensure that no other instructor can open the webpage.

## Failure Login:

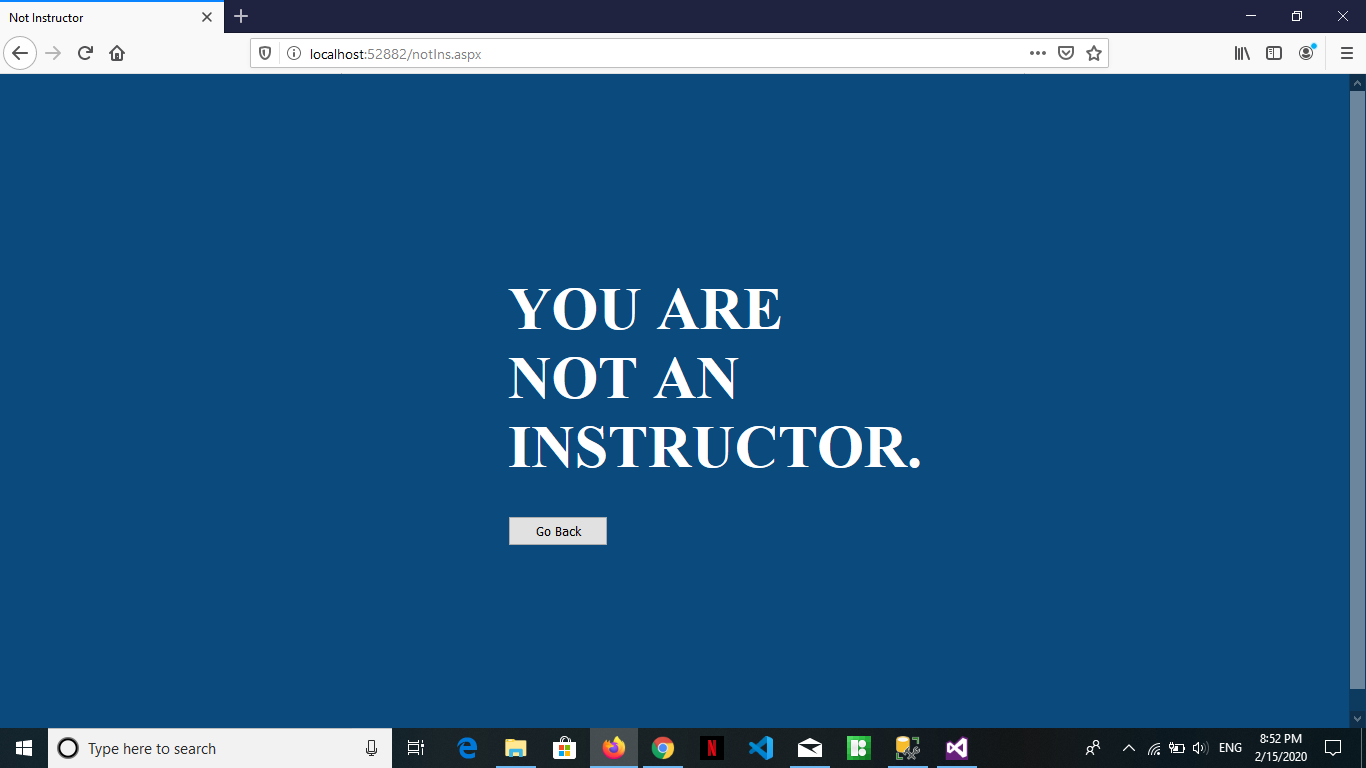
****

Figure 4 Not Instructor User

This page open when the user is not one of the instructor that can use the website.

## Create Account:

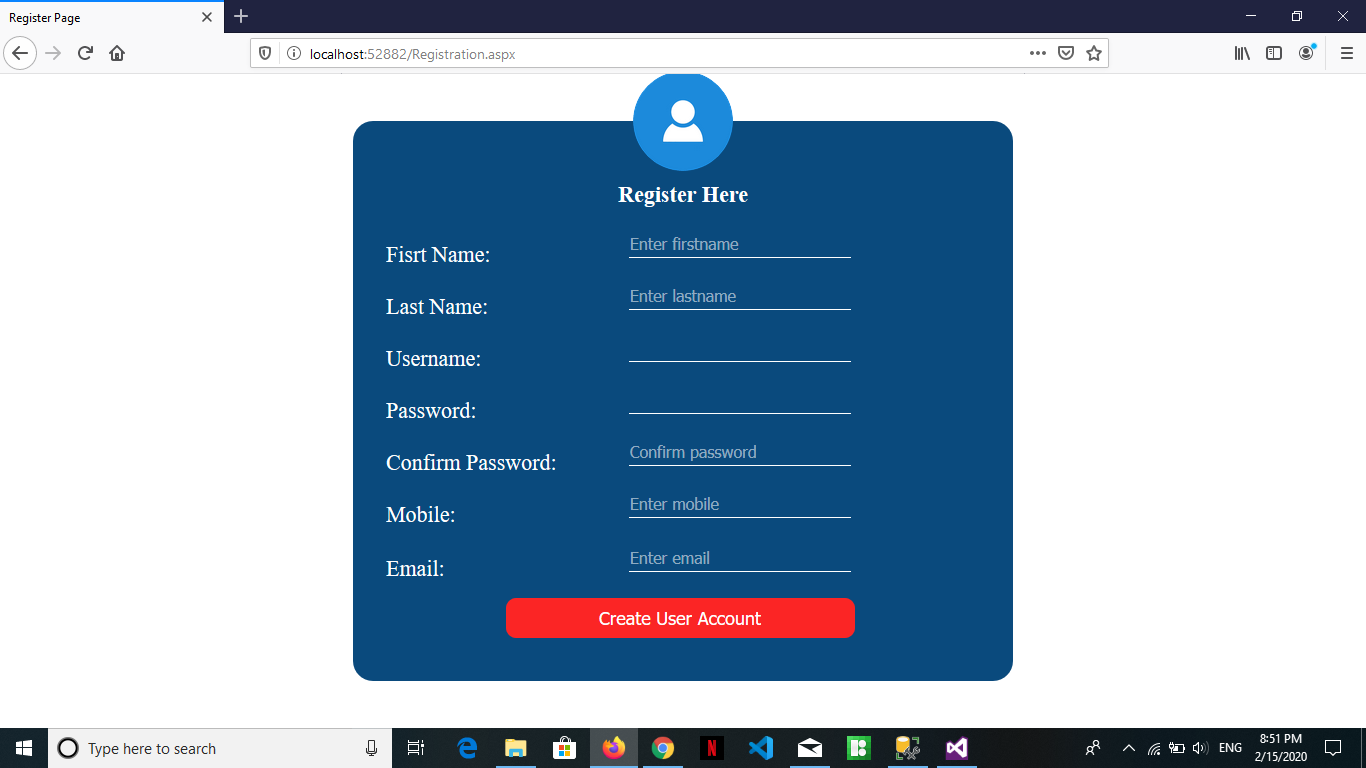
****

Figure 5 New user account

Here any one can make an account to use the webpage, but after take a permission from the admin.

## Students:

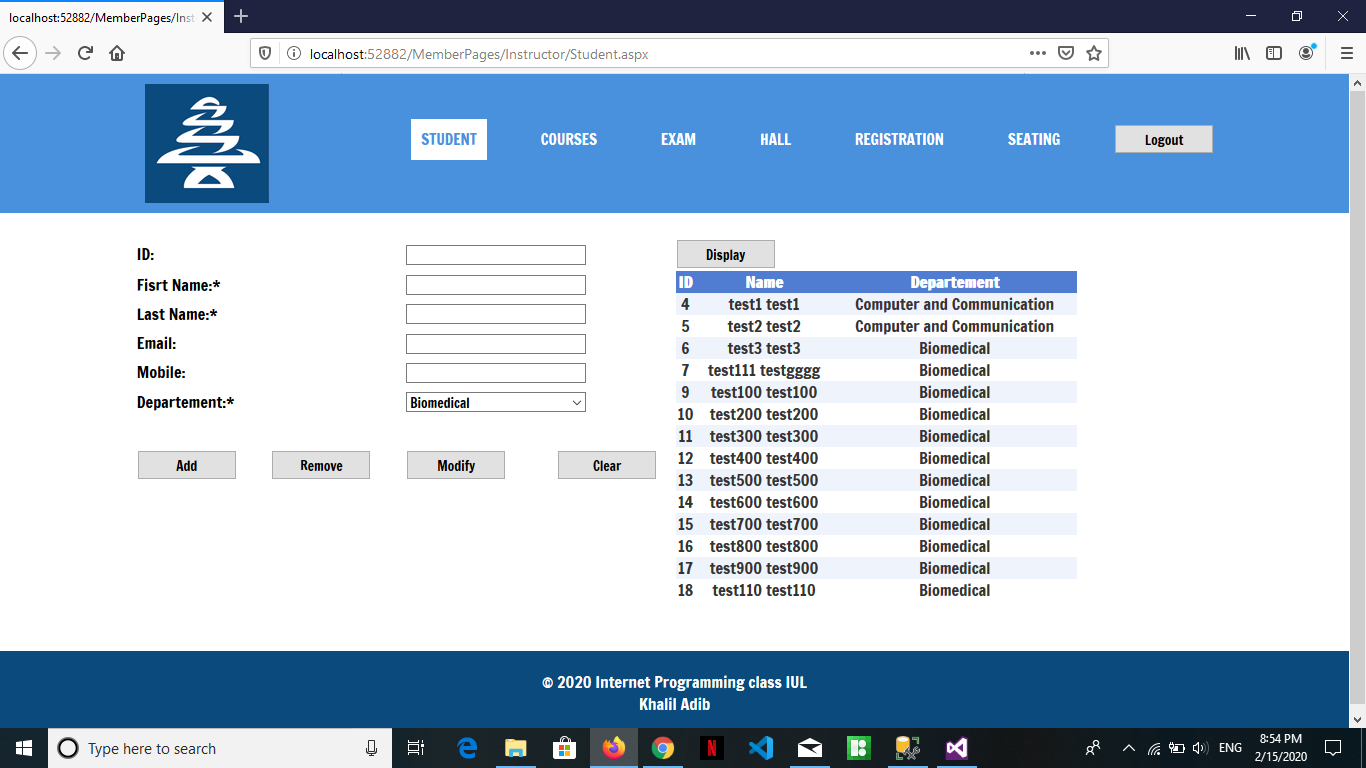
****

Figure 6 Student Page

This webpage is used for add, delete, modify…any student in any department he registered. (Instructor webpage)

## Courses:

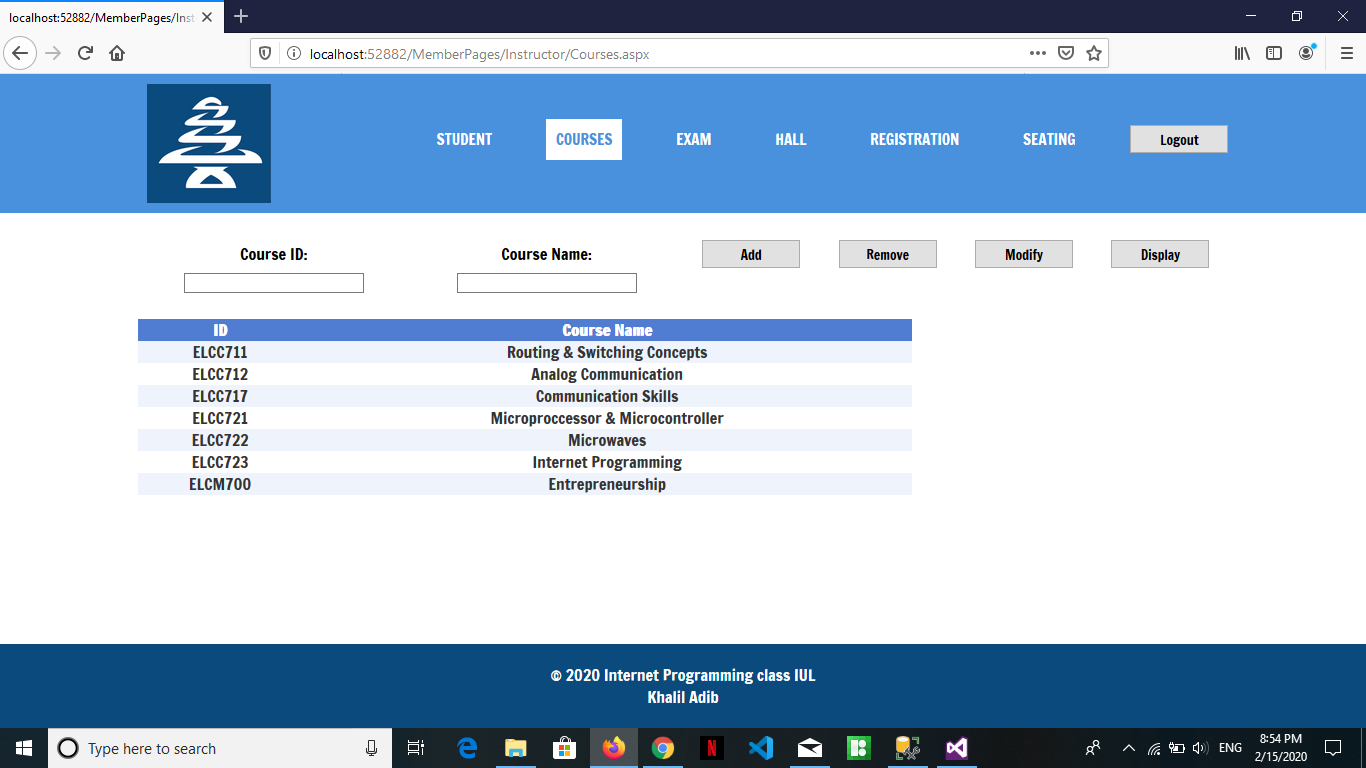
****

Figure 7 Course Page

This page for add, remove, modify or display any course the user need. .(Instructor webpage)

## Exams:

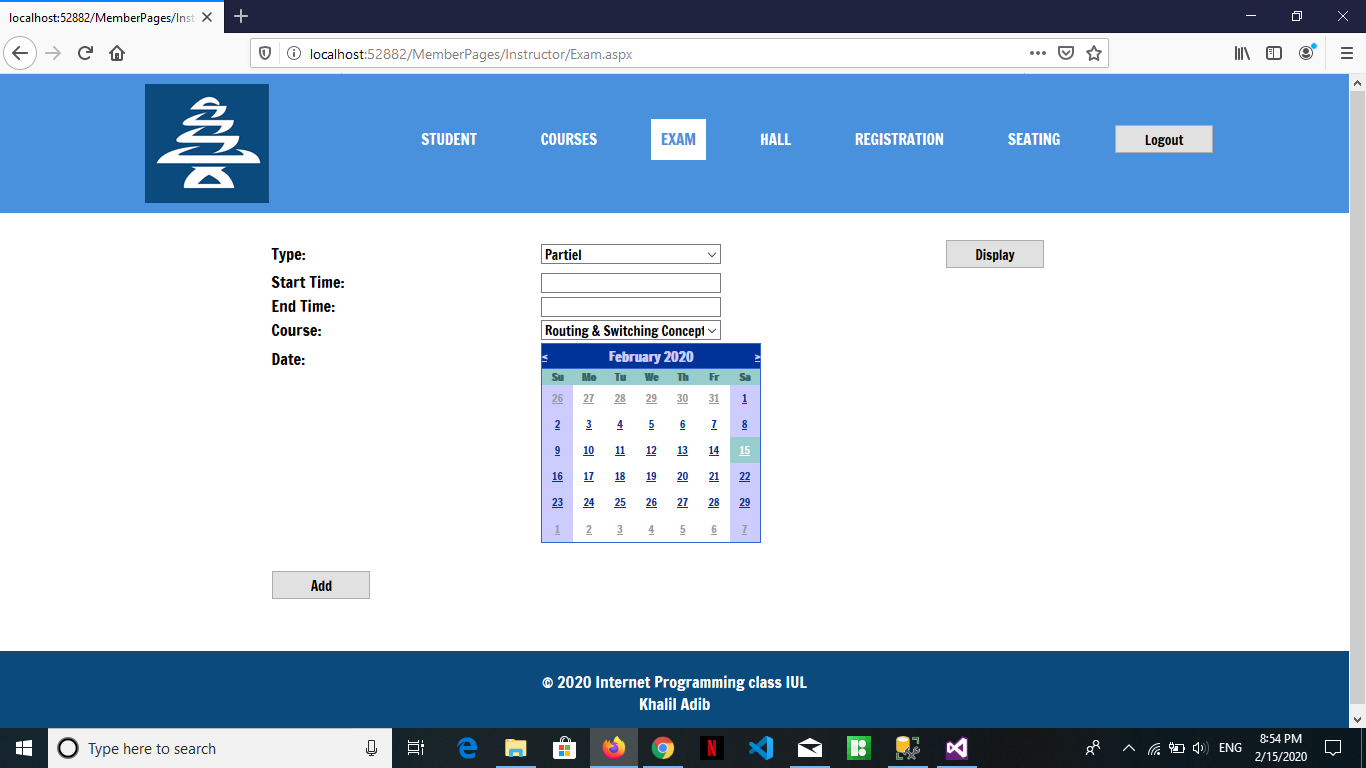
****

Figure 8 Exam Page

This page is for add an exam with its time and date specified. .(Instructor webpage)

## Hall:

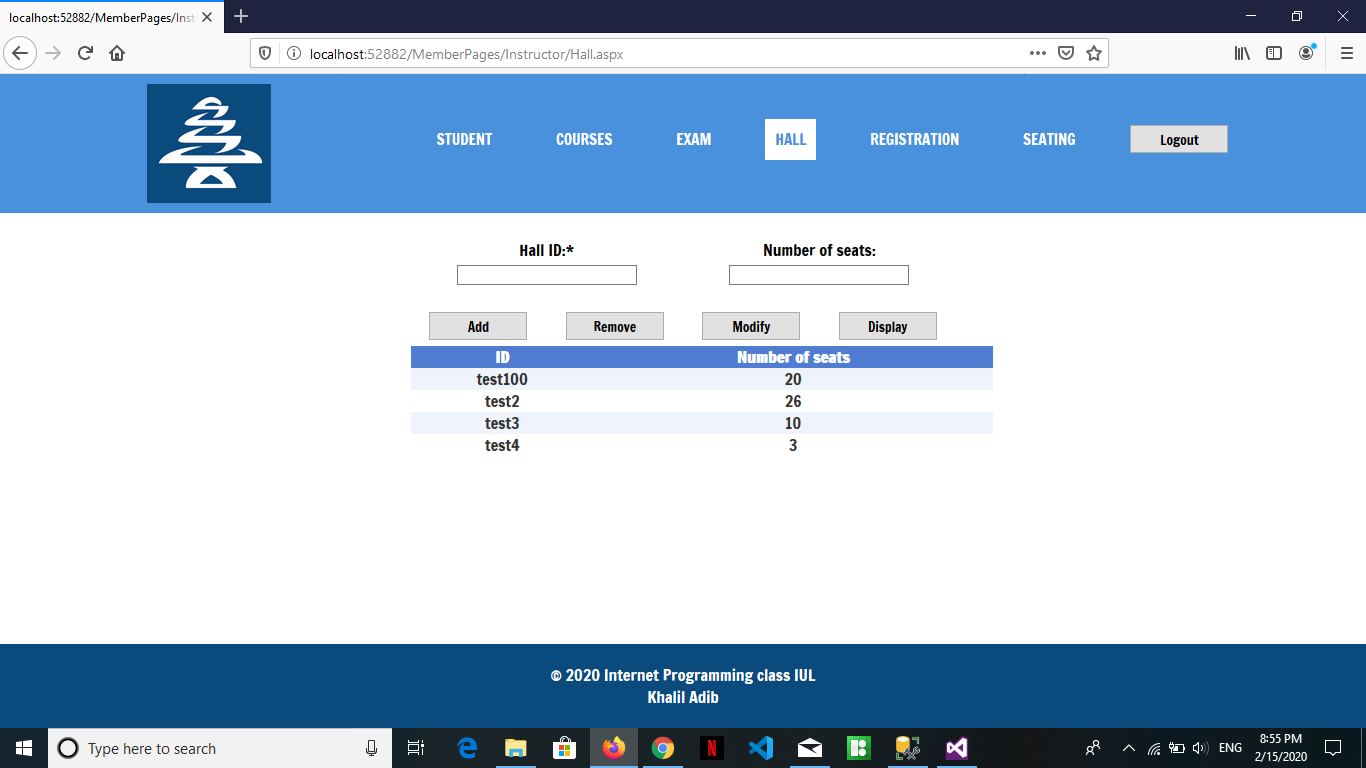
****

Figure 9 Halls Page

This page used to add, remove, modify or display the ID room and its number of seats available in it. (Instructor webpage)

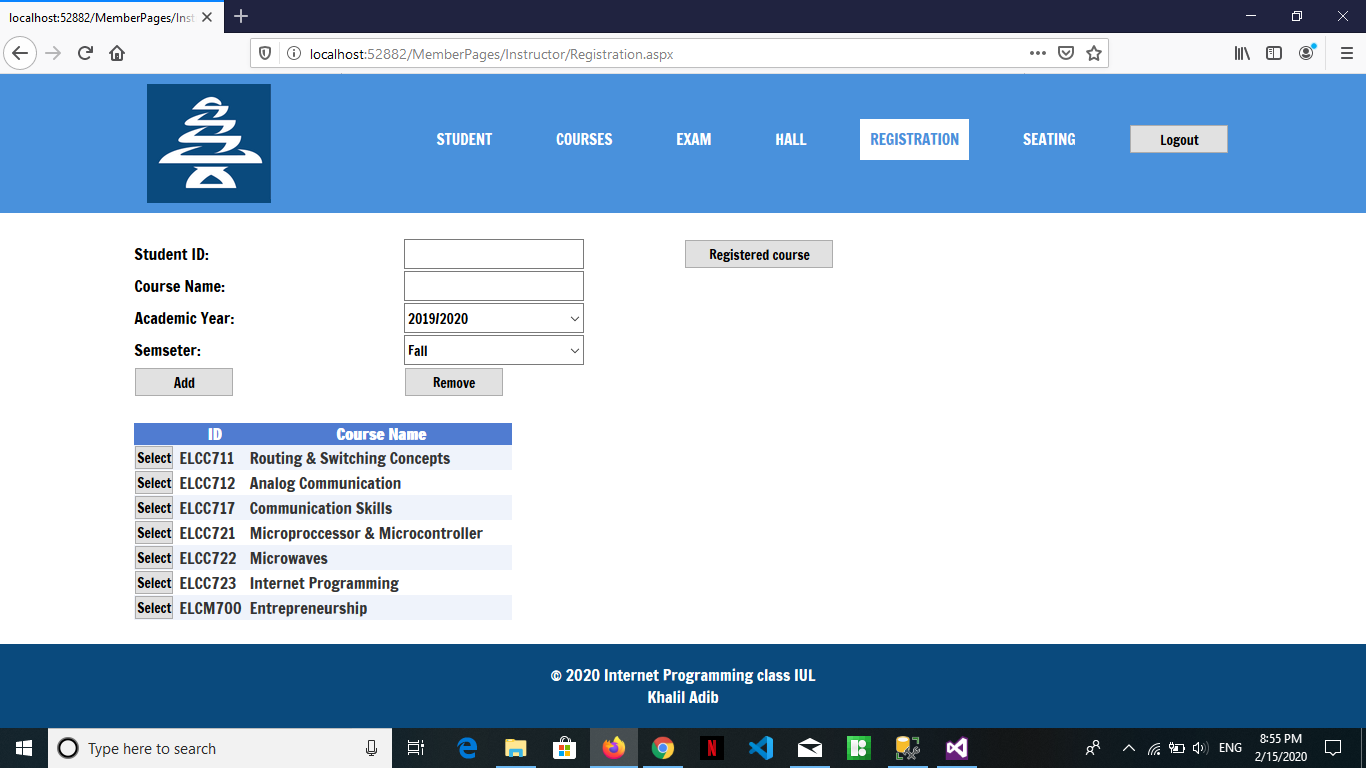
Registration: ****

Figure 10 Registration Page

This page is for add or remove a students and his course name, academic year, semester. (Instructor webpage)

## Seating:

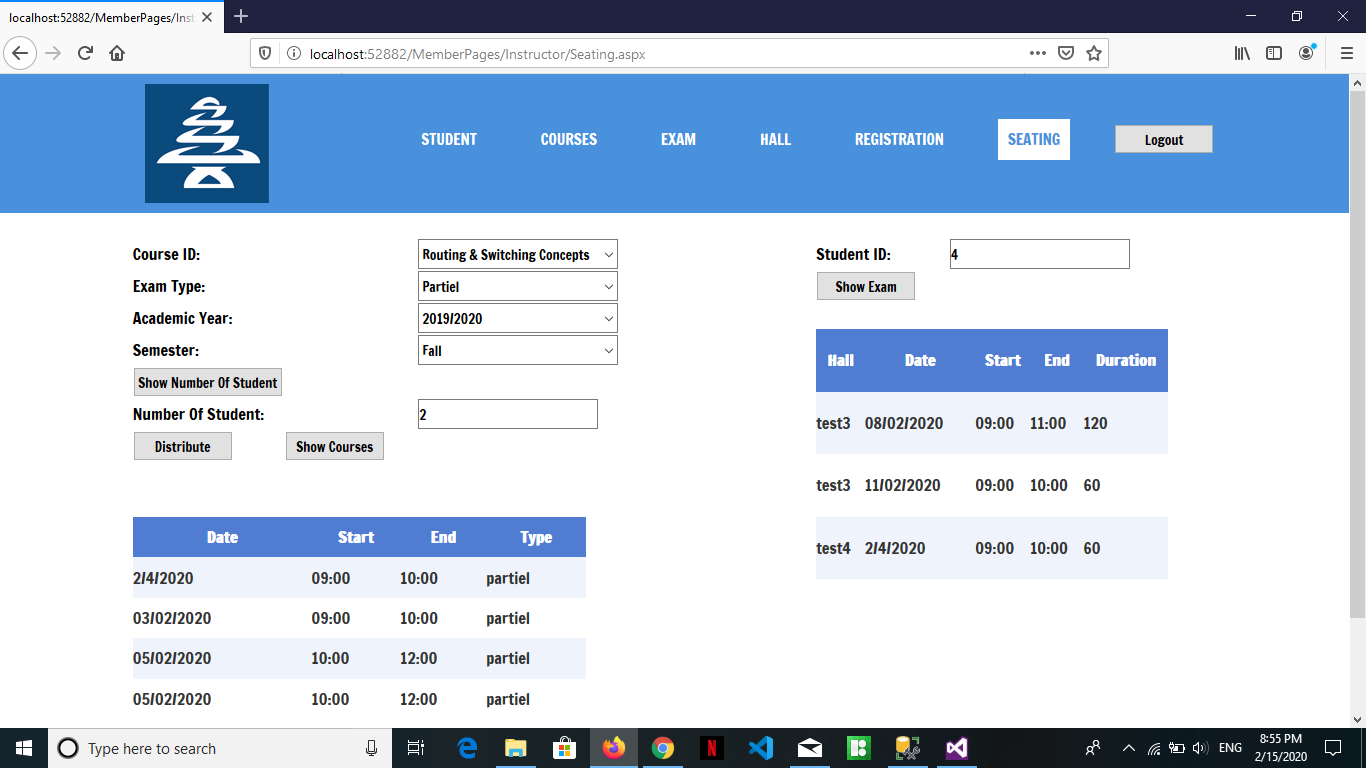
****

Figure 11 Seating Page

In this page the instructor can know how many students are registered in each course, and know the exam course, date and its hall respectively to each student from entering his ID. (Instructor webpage)

## Faculty:

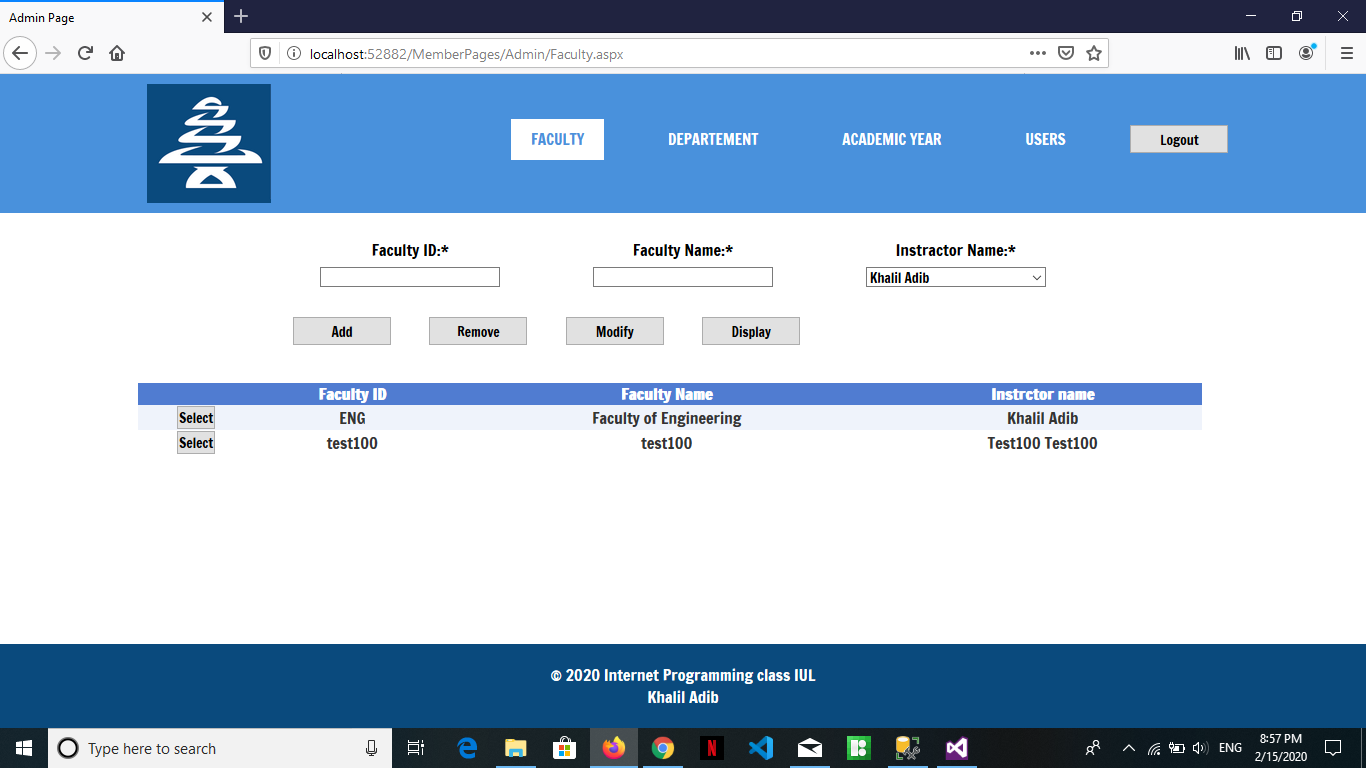
****

Figure 12 Faculty Page

Here the user can add, remove, modify and display any faculty he want in his university by entering its ID, Name and its special instructor. (Admin webpage)

## Department:

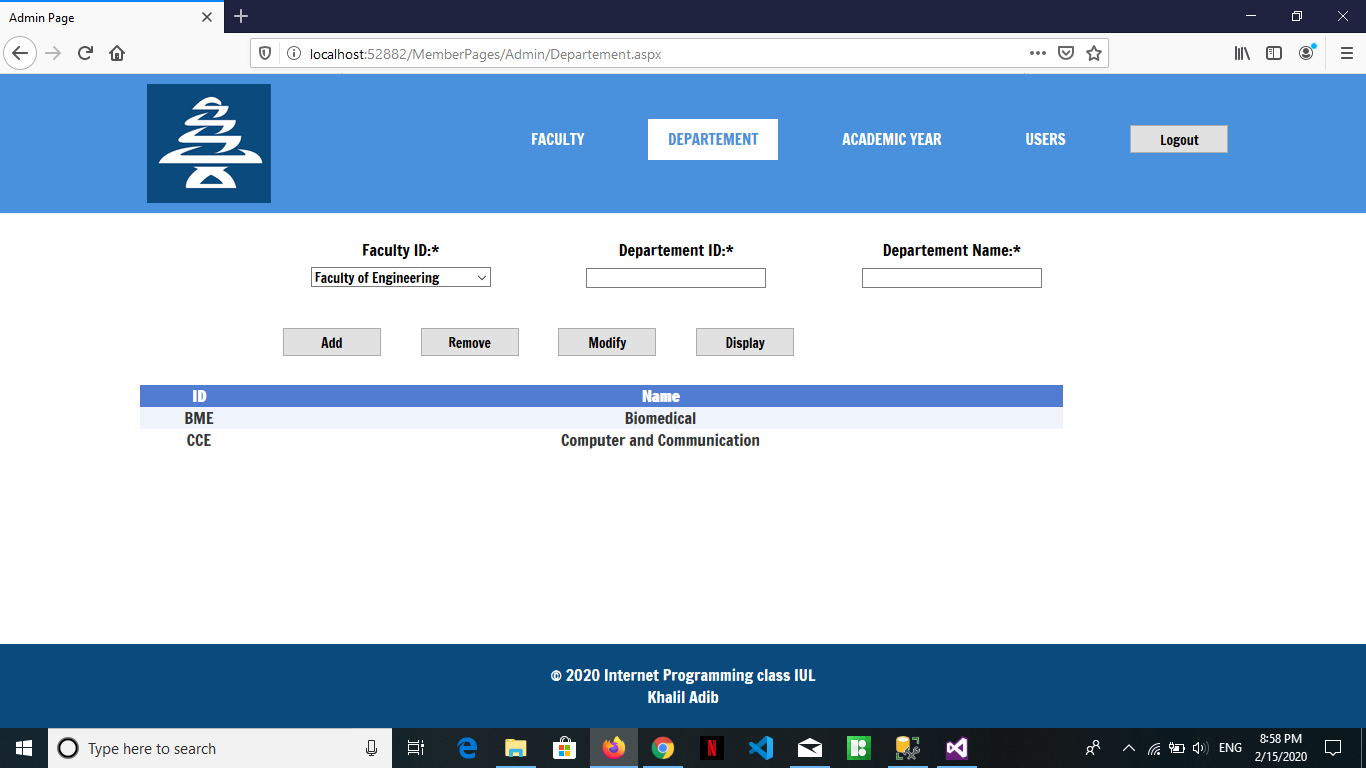
****

Figure 13 Department Page

Adding, removing, modifying and displaying a department is established in this page by entering the ID of its relative faculty and its ID and Name. (Admin webpage)

## Academic Year:

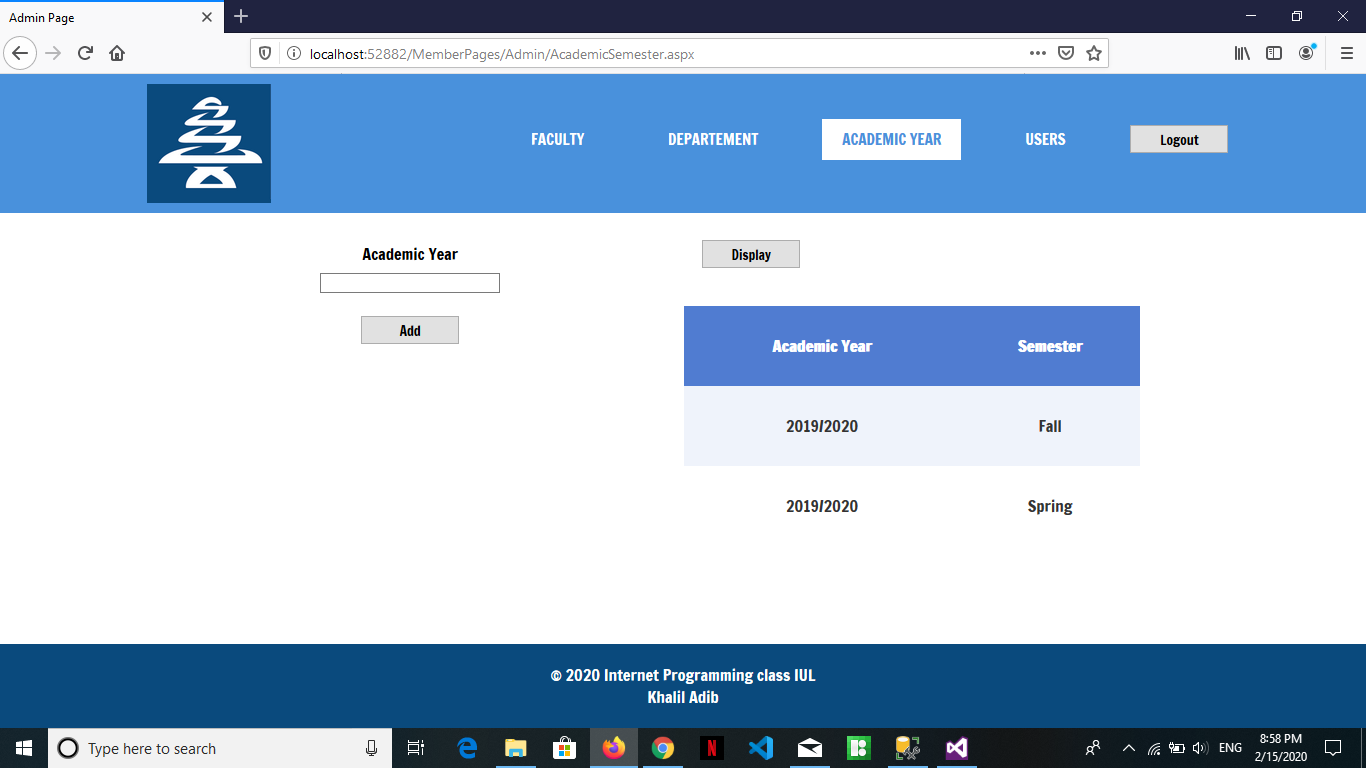
****

Figure 14 Academic Year Page

In this page the user can add a new academic year or display the current registrations year. (Admin webpage)

## Users:

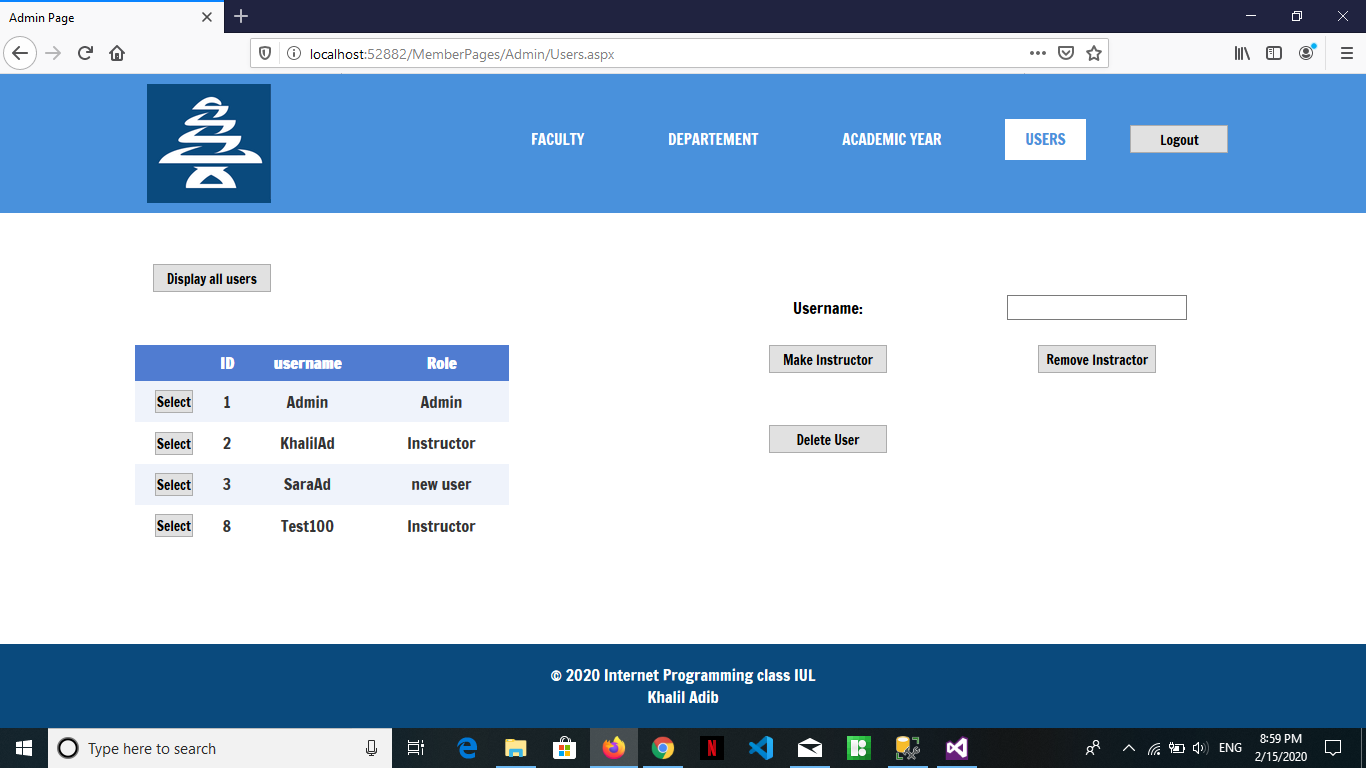
****

Figure 15 Users Page

This page is especially for admin can see the users that can use the Webpage and the admin can give permission for new user or delete his permission. (Admin webpage)

# Flowchart

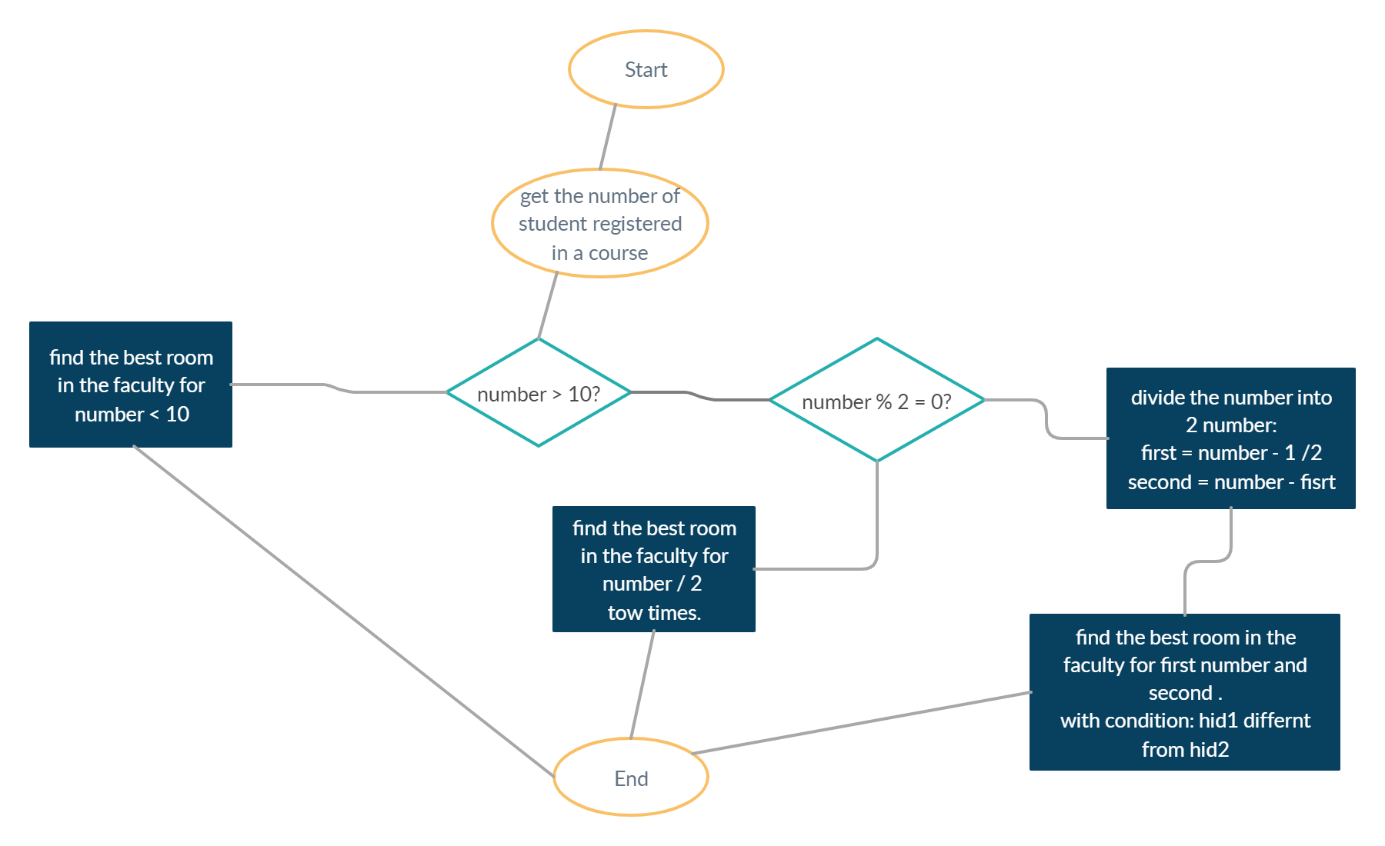


Figure 16 Flowchart to find the best hall

First of all, we check the number of students registered in the course. If the number is less than 10, we put all the student in the same hall. We created a method called **bestHall** to find the best Hall in the faculty. If the number is great than 10, the program check if the number even, we divided over 2 and search for the **bestHall** for 2 groups of student related to the same course exam, by using the same method. But if the number is odd, the program create 2 variables, first one is equal to (number -1)/2, and the second is equal to (number – first), and distribute the 2 groups is the same way, by using **bestHall**.

public static string bestHall(string fctid, string date, string start, string end, int number, string hid1)

{

DBU.callproc("exec select\_count\_hall\_by\_fctid @fctid = '" + fctid + "';");

int nbHall = int.Parse(DBU.ds.Tables[0].Rows[0].ItemArray[0].ToString());

int i = 0, j = 0, diff = 0;

DBU.callproc("exec select\_hall\_by\_fctid @fctid = '" + fctid + "';");

int min = int.Parse(DBU.ds.Tables[0].Rows[0].ItemArray[1].ToString());

min = min - number;

while (i < nbHall)

{

DBU.callproc("exec select\_hall\_by\_fctid @fctid = '" + fctid + "';");

string hids = DBU.ds.Tables[0].Rows[i].ItemArray[0].ToString();

int seats = int.Parse(DBU.ds.Tables[0].Rows[i].ItemArray[1].ToString());

diff = seats - number;

if (diff > 0 || diff == 0)

{

if (diff < min || diff == min)

{

DBU.callproc("check\_hall @hid = '" + hids + "', @date = '" + date + "', @start = '" + start + "', @end = '" + end + "';");

int test;

try

{

test = int.Parse(DBU.ds.Tables[0].Rows[0].ItemArray[0].ToString());

}

catch

{

test = 0;

}

if (test + number < seats || test + number == seats)

{

if (!string.Equals(hids, hid1))

{

j = i;

min = diff;

}

}

}

}

i++;

}

DBU.callproc("exec select\_hall\_by\_fctid @fctid = '" + fctid + "';");

string hid = DBU.ds.Tables[0].Rows[j].ItemArray[0].ToString();

return hid;

}

This is the method, it take as parameter, the faculty id, course id, date of the exam, start and end time of the exam, the number of student and finally a hall id.

It search for the best hall by subtract the number of seats in the hall by the number of student, and search for the minimum number of all hall in the faculty.

In each time find a new minimum, they check for the availably of this hall in the date and time donated, if there is no exam at the same time, all is fine, if there is an exam, the calculate the number of student in the exam + the number of student in the exam I want to distribute there student, if the number less than or equal the number of seats, all is fine, if not, the hall not available.

For the hall id take as parameter, in case where the number is great than 10, the 2 groups of students will distribute over 2 different hall, because of that, I check for the hall id in the method.

In case of number < 10, I put “” in the hid take as parameter.

Finally, the function return the id of the best hall.

# Conclusion

In this project, we faced some difficulties and it took us a long time and long thinking until we reached the solution of the methods. If we had more time, we would work more than that and feel more on it. This project can be used by the university in exams times and may be modified and developed in the future so that the students have the ability to use it and perform the process of registration a special account for them and see their exam class, day and time.