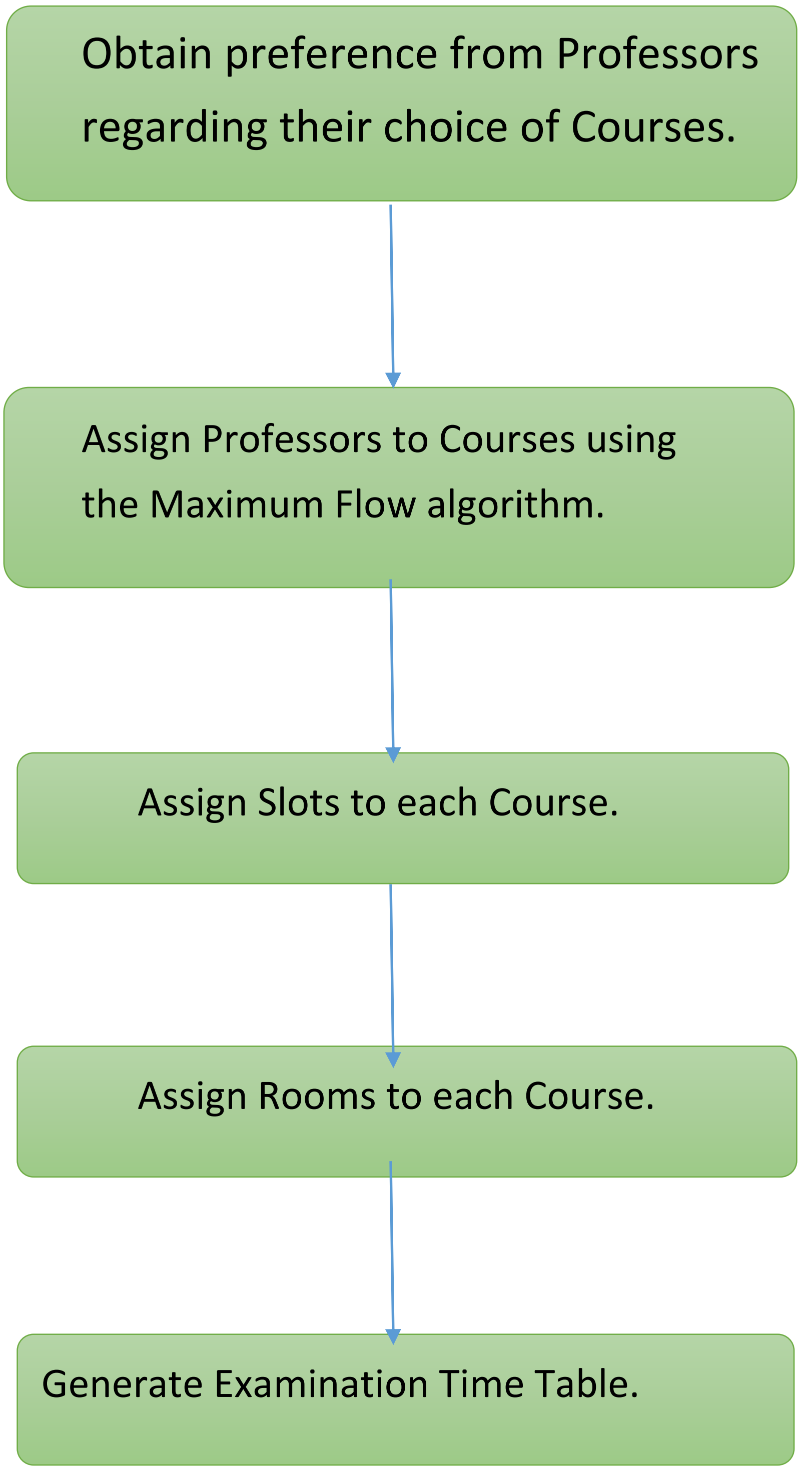
Technical Documentation

Academic Section Management System

IIT Guwahati

Piyush Gupta, Vineet Malik, Utkarsh Jain, HardikKatyal,AayushMakharia,HansrajPatel,Lucky,Sunny Kumar,Parvindar Singh, Rohan Aggarwal, Rishi Pathak,UmangKakkar,PranshuSrinivas,Sri Ram Reddy

CLASS TIME TABLE


Obtain preference from Professors regarding their choice of Courses.
Assign Professors to Courses using the Maximum Flow algorithm.
Assign Slots to each Course.
Assign Rooms to each Course.
Generate Examination Time Table.

Professor Course Allotment

File Name :*ProfAssignment.cpp*

Function Name :*ProfAssignment*

Input: *DeptID*

Return Type: *True/False*

Functions :

bpm()

Input: *Graph[][],seen[],professor,matchR[]*

Return Type: *True/False*

Description :*A DFS based recursive function that returns true if a matching for a professor is possible*

maxBPM()

Input: *Graph[][],matchR[]*

Return Type: *void*

Description :*Calculate the maximum number of matching from M professors to N Courses*

Calls :*bpm()*

ProfAssignment()

Input: *Department Code*

Return Type: *true if all professors can be assigned courses*

Description :*Assigns a professor a course of his choice if possible*

Calls :*maxBPM()*

Database Table:*CouresTT*

Database Rows Changed :

- *CoursesTT.ProfAssignedPreMidsem*
- *CoursesTT.ProfAssignedPostMidsem*

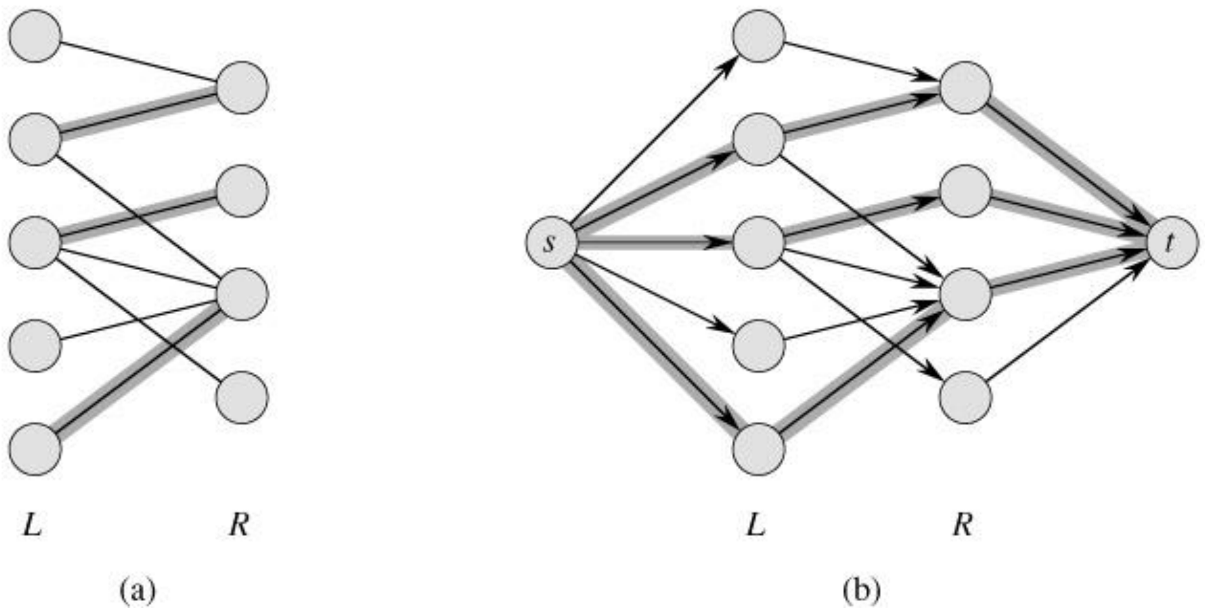
Algorithm Used :*Maximum Bipartite Matching*

Algorithm Description :

Maximum Bipartite Matching and Max Flow Problem

Maximum **B**ipartite **M**atching (**MBP**) problem can be solved by converting it into a flow network . Following are the steps.

1. ***Build a Flow Network***
   There must be a source and sink in a flow network. So we add a source and add edges from source to all applicants. Similarly, add edges from all jobs to sink. The capacity of every edge is marked as 1 unit.
2. We use Ford-Fulkerson algorithm to find the maximum flow in the flow network built in step 1. The maximum flow is actually the MBP we are looking for.



A matching in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint. A maximum matching is a matching of maximum size (maximum number of edges). In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matchings for a given Bipartite Graph.

Let us first define input and output forms. Input is in the form of Edmonds matrix which is a 2D array bpGraph[M][N] with M rows (for M course Applicants) and N columns (for N courses). The value bpGraph[i][j] is 1 if i'th professor is interested in j'thcourse, otherwise 0.Output is number maximum number of people that can get courses.

In bpm(), we one by one try all course that an applicant 'u' is interested in until we find a course, or all courses are tried without luck. For every course we try, we do following.
If a course is not assigned to anybody, we simply assign it to the applicant and return true. If a course is assigned to somebody else say x, then we recursively check whether x can be assigned some other course. To make sure that x doesn't get the same course again, we mark the course 'v' as seen before we make recursive call for x. If x can get other course, we change the applicant for course 'v' and return true. We use an array maxR[0..N-1] that stores the applicants assigned to different courses.

If bmp() returns true, then it means that there is an augmenting path in flow network and 1 unit of flow is added to the result in maxBPM().

Limitations :

- *Each professor can take only one course in the morning and one in the evening*
- *Each course that the professor applies for is equally preferred by the professor*
- *HOD has to to assign professor for those courses manually for which no professor can be allocated*
- *Maximum number of professors in a department is 100*

Slot Assignment

File Name :*slotAllocation.cpp*

Function Name :*assignSlots*

Input: *void*

Return Type: *int(not necessary)*

Functions :

assignSlots()

Input: *void*

Return Type: *True/False*

Description :*Function assigns the slots to all courses in the*

Database Table Used: *CoursesTT*

Database Rows Changed :*CoursesTT.SlotAssigned*

Algorithm Used :*Greedy Allocation of slots*

Algorithm Description :

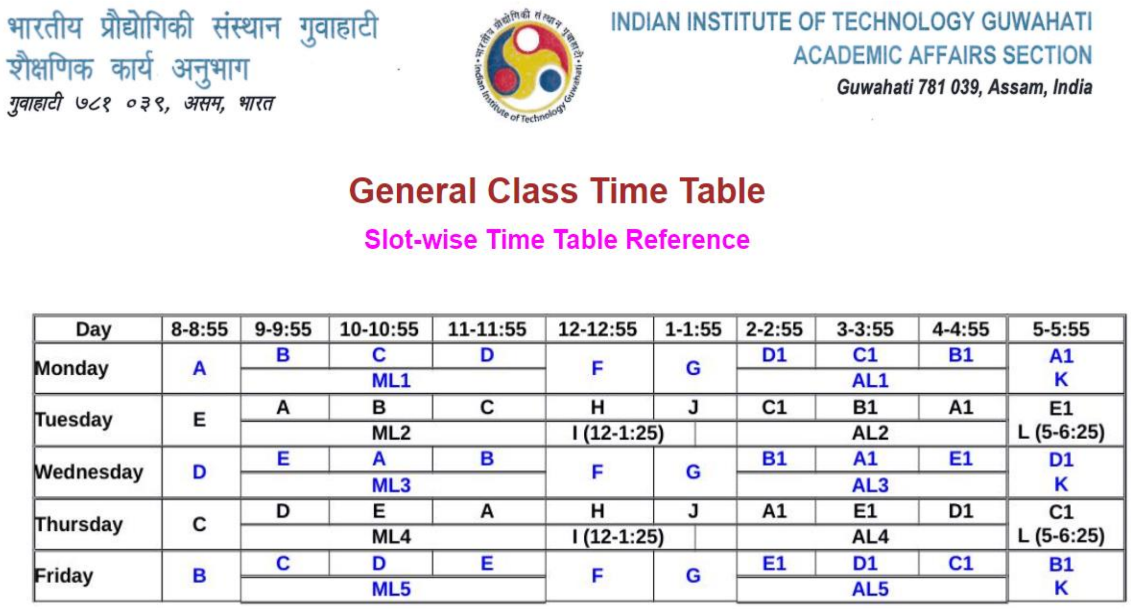
Time slots are allocated in a greedy fashion. Courses having more number of registered batches are considered first and slots are assigned to them.

When courses are allocated each batch it is offered to is considered to find the first common empty slot for each batch.

When a slot is assigned to a batch that slot is marked as occupied for that batch.

Assumptions :

- *Each professor is free at whenever the slot is allocated*
- *A batch does not have more than 5 classroom courses in the morning and more than 5 classroom courses in the evening*
- *One person can take only one minor*
- *Slot template is fixed and cannot be changed*
- *Room for labs are fixed and are assigned manually by the departments*



भारतीय प्रौद्योगिकी संस्थान गुवाहाटी
शैक्षणिक कार्य अनुभाग
गुवाहाटी ७८१ ०३९, असम, भारत

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
ACADEMIC AFFAIRS SECTION
Guwahati 781 039, Assam, India

General Class Time Table

Slot-wise Time Table Reference

Day	8-8:55	9-9:55	10-10:55	11-11:55	12-12:55	1-1:55	2-2:55	3-3:55	4-4:55	5-5:55
Monday	A	B	C	D	F	G	D1	C1	B1	A1
			ML1					AL1		K
Tuesday	E	A	B	C	H	J	C1	B1	A1	E1
			ML2		I (12-1:25)			AL2		L (5-6:25)
Wednesday	D	E	A	B	F	G	B1	A1	E1	D1
			ML3					AL3		K
Thursday	C	D	E	A	H	J	A1	E1	D1	C1
			ML4		I (12-1:25)			AL4		L (5-6:25)
Friday	B	C	D	E	F	G	E1	D1	C1	B1
			ML5					AL5		K

[Slots are assigned according to the standard template for Time Table used in the institute.]

Room Assignment

File Name :*roomAllocation.cpp*

Function Name :*assignRooms*

Input: *void*

Return Type: *int(not necessary)*

Functions :

countRegisteredCourse()

Input: *String ^ CourseID*

Return Type: *int*

Description :*Finds the number of students enrolled in a particular course*

Database Table Used: *StudentCourses*

Database Rows Changed : *None*

compr()

Input: *pair<int,string>a,pair<int,string>b*

Return Type: *bool*

Description :*return true if a.first<b.first*

Database Table Used: *None*

Database Rows Changed : *None*

assignRooms()

Input: *void*

Return Type: *void*

Description :*Function assigns rooms for all courses*

Database Table Used: *CourrsesTT,Rooms*

Database Rows Changed :*CoursesTT.RoomAssigned*

Algorithm Used : *Greedy Allocation of Rooms*

Algorithm Description :

For each course find an unoccupied room with capacity greater than or equal to registered students in the timeslot of the course.

That room is allocated to the course and is marked as occupied for that slot

Requirements :

- *Rooms of enough capacity are available*
- *Enough rooms are available*

Limitations :

- *Special Room Equipment not considered*
- *Student movement distances not minimised*

Exam Room Allotment

File Name :*ExamRoomAllotment.cpp*

Function Name :*assignExamRooms*

Input: *void*

Return Type: *int(not necessary)*

Functions :

assignExamRooms()

Input: *void*

Return Type: *True/False*

Description :*Function assigns the Exam Room and Slot to all courses for examination*

Database Table Used: *CoursesTT,ExamTT*

Database Rows Changed :*ExamTT**

Algorithm Used :*Greedy Allocation of ExamRooms*

Algorithm Description :

Each exam is held according to its course time slot.

For each time slot courses are considered in increasing count of students registered.

Each Room is divided into 4 parts and sorted into vectors by increasing capacity.

Remove rooms from the vector once allocated

For a course room allotment is done in the same slot the slot is emptied.

Move to next slot in a circular order

Limitations :

- *Exam day is predicided*
- *Full occupancy of rooms is not ensured*

FORMS

Homepage
StudentForm
StudentCourses
StudentProfile
DetailedStudentProfile
Search

INTRODUCTION

Creating a primary interface(Homepage) and provide a profile for each user and searching the users

Homepage

· User can login into his profile
· There are three types of users namely student,professor,staff
· Search the database for required information about other students,professors,staff.

TOOLS

txtUsername:textbox for username to login
txtPassword:textbox for password to login
login:logs in the user if the credentials match.
textBox1:textbox for searching
comboBox1:selecting department
radiobutton1,radiobutton2,radiobutton3:selecting for student,professor,staff respectively

Functions and subroutines

- loginBtn_Click(System::Object^ sender, System::EventArgs^ e):when the user click login button in homepage database connection is opened and txtUsername->Text and txtPassword->Text is matched with the database data and if the data matches StudenForm is displayed hiding homepage.
- Hide():Hides the form
- ShowDialog():shows the Form
- helpBtn_Click(System::Object^ sender, System::EventArgs^ e):loads search form
- searchBtn_Click(System::Object^ sender, System::EventArgs^ e):displays necessary tools required to search

Queries

Select * from Student where [Username] ='"+txtUsername-> Text+"';:selects rows from database where username in the database is matched with txtUsername->Text

StudenForm

- Displays the profile of user default
- User can access his courses information(MY COURSES) and Search the database for required information about other students,professors,staff.
- Logging out

TOOLS

textBox1:textbox for searching
comboBox1:selecting department
radiobutton1,radiobutton2,radiobutton3:selecting for student,professor,staff respectively

Functions

- StudentForm_Load(System::Object^ sender, System::EventArgs^ e):loads StudentProfile form
- MyCoursesBtn_Click(System::Object^ sender, System::EventArgs^ e):loads StudentCourses Form
- helpBtn_Click(System::Object^ sender, System::EventArgs^ e):loads search form
- searchBtn_Click(System::Object^ sender, System::EventArgs^ e):displays necessary tools required to search
- logoutBtn_Click(System::Object^ sender, System::EventArgs^ e):logs out user and displays homepage

StudentProfile

- Displays the name,rollno,programme,email,nationality,gender,hostel of the user
- Additonal information can be viewed by clicking additional information button

Tools

Additional information:displaysDetailedStudentProfile form

Funtions

- StudentProfile_Load(System::Object^ sender, System::EventArgs^ e):connects to database and displays the information of the student in the respective labels
- ExtraInfoBtn_Click(System::Object^ sender, System::EventArgs^ e):displays the DeatailedStudentProfile form hiding Student Profile

Queries

- Select * from Student where [Username] ='"+usrnm+"';:selects rows from database where username in the database is matched with usrnm(username of user)

DetailedStudentProfile

- Displays the additional information of the user
- User can edit his additional information

Tools

Edit:all the editable information texboxes are enabled

Funtions

- DetailedStudentProfile_Load(System::Object^ sender, System::EventArgs^ e)

:connects to database and displays the information of the student in the respective texboxes

- editBtn_Click(System::Object^ sender, System::EventArgs^ e):all the editable textboxes are enabled and update button and cancel button is shown
- UpdateBtn_Click(System::Object^ sender, System::EventArgs^ e):updates the information in database
- cancelBtn_Click(System::Object^ sender, System::EventArgs^ e):cancels the update button and all the textboxes are set to readonly and update and cancel buttons are hidden and edit and back button are shown
- backBtn_Click(System::Object^ sender, System::EventArgs^ e):shows the StudentProfile form
- button1_Click(System::Object^ sender, System::EventArgs^ e):user can upload a photo form the resources folder
- MobileValidator(String^ mobile):checks wherther the mobile number is valid or not
- GenderValidator(String^ Gender):checks whether the gender is valid or not

Queries

- Select * from Student where [Username] ='"+usrnm+"';:selects rows from database where username in the database is matched with usrnm(username of user)
- Update [Student] set [SecondaryEmail]='" + this->e->Text->Trim() + "',[PhoneNo]='" + this->f->Text->Trim() + "',[PassportSizePhotograph]='"+"Resources\\"+this->usrnm+".jpg"+"',[Nationality]='" + this->i->Text->Trim() + "',[Hostel]='" + this->m->Text->Trim() + "',[RoomNo]='" + this->n->Text->Trim() + "',[Address]='" + this->o->Text->Trim() + "',[Pincode]='" + this->p->Text->Trim() + "',[FatherName]='" + this->r->Text->Trim() + "',[MotherName]='" + this->s->Text->Trim() + "',[BloodGroup]='" + this->t->Text->Trim() + "',[Height]='" + this->u->Text->Trim() + "',[Weight]='" + this->v->Text->Trim() + "',[AadhaarNumber]='" + this->w->Text->Trim() + "',[PassportNumber]='" + this->x->Text->Trim() + "'where [Username]= '" + usrnm + "' "; : Updates the information into datbase where the username is matched

Search

- Can be done in homepage and studentfom
- Displays the results in dynamic buttons and clicking on the buttons displays the information about the searched result

Tools

Dynamic buttons

Funtions

- UserControlSearch_Load(System::Object^ sender, System::EventArgs^ e):connects to database and searches for the text in textbox(textBox1) where the department is selected by comboBox1 and the usertype is selected by radiobuttons and displays the FULLNAME in dynamically created buttons .If the textbox is empty it displays the FULLNAME whole selected department and usertype results in dynamically created buttons
- creator(inti,String^ firstnm,String^ middlenm,String^ lastnm,String^ usrnm):creates buttons of mentioned size and displays buttons at desired location and the usercontrolsearch is based on usertype
- clickbtnStudent(System::Object^ sender, System::EventArgs^ e):dynamic buttons are labeled as clickbtnstudent when the searched usertype is student .on clicking on particular button displays the required information of student

- clickbtnOthers(System::Object^ sender, System::EventArgs^ e):dynamic buttons are labeled as clickbtnstudent when the seachedusertype is either professor or staff.clicking on particular button displays the required information of user

Queries

- "Select * from "+ table +" where [FirstName] Like '%"+ textbox +"%' or [MiddleName] Like '%"+ textbox +"%' or [LastName] Like '%"+ textbox +"%' or [RollNumber] = '"+ textbox +"' and [DeptID] ='"+ combobox +"';" : selects rows from student table for textbox text and where department matches combobox
- "Select * from "+ table +" where [Username] ='"+ username +"';" : elects rows from database where username in the database is matched with username(username of user)
- select [Post] from "+table1+" where [Username] ='"+ username +"';" : selects the Post coloumn in from table1(either profstatus or staffstatus) where username in the database is matches withe username(searched result)
- "Select [Nationality] from "+table+" where [Username] ='"+ username +"';" : selects the Nationality coloumn in from table1(either professor or staff) where username in the database is matches withe username(searched result)

StudentCourses

- Displays the current courses taken by the student and previous semester courses and grades
- Drop and changing the courses

Tools

Dynamic buttons

Funtions

- StudentCourses(String ^text) **:**initialises present_sem and present_year
- StudentCourses_Load(System::Object^ sender, System::EventArgs^ e):calls the coursebuttongenerator function and previousbuttongenerator function
- generateCourseButtons():connects to database and the course id ,course grade of each course is read and calls course_btn_creator(course_btn_index, course_id, course_grade, con) function which generates dynamic buttons with coursegrade,courseid,coursename,coursecredits as text of buttons
- generate_next_btn():generates a dynamic button when course information of the nextsemester is needed
- generate_prev_btn():generates a dynamic button when course informaton of the previous semester is needed
- btn_course_Click(System::Object ^ sender, System::EventArgs^ e): student can drop or change the course
- generate_drop_change_controls():generates drop and change buttons
- generate_ok_cancel_buttons():genearates ok and cancel dynamic buttons
- drop_btn_Click(System::Object^ sender, System::EventArgs^ e):clears the user controls and calls generateCourseButtons and generate_drop_changefuncitons
- change_btn_Click(System::Object^ sender, System::EventArgs^ e):student can change from one course to another and when the change is done it calls generateCourseButtons and genearate_ok_cancel_buttons() functions

- ok_btn_Click(System::Object^ sender, System::EventArgs^ e):if change is made it shows the appropriate messagebox
- cancel_btn_Click(System::Object^ sender, System::EventArgs^ e):cancels the change and calls the generateCoursesfunciton
- prev_btn_Click(System::Object^ sender, System::EventArgs^ e):course information of the previous semester is displayed
- next_btn_Click(System::Object^ sender, System::EventArgs^ e):course information of the next semester(next to the current viewed semester) is displayed

Queries

- Select * from StudentCourses where [Username] ='"+usrnm+"' AND [Session] = '"+session+"' AND [Semester] = '"+semesterName+"' : selects rows from the StudentCourses table in database where username and session and semester are matched
- Select * from CourseList where [CourseID] ='"+course_id+"':selects rows from the Courselist table in database where CourseID is matched

PROFESSOR PORTAL

FORMS

- Professor homepage
- Update Profile
- Send Grades
- Send Course Preference
- TA Request
- Timetable
- Courses
- Current Students
- View Notification
- Add Notification
- Course Adjustment Approval

Update Profile

Functions:

- btnPhoto_Click
- btnUpdate_Click
- professor_updateProfile_Load

Queries:

- UPDATE Professor SET [Password]='"+ this->txtPassword->*Text* +"',[SecondaryEmail]='"+ this->txtSecEmail->*Text* +"',[PhoneNo]='"+ this->txtPhoneNo->*Text* +"',[Address]='"+ this->txtRoomNo->*Text* +"',[EmergencyContact]='"+ this->txtEmergencyNo->*Text* +"',[PermanentAddress]='"+ this->txtPermanentAddress->*Text* +"',[HomePincode]='"+ this->txtPincode->*Text* +"',[DoYouHavePassport]='"+ this->comboBoxPassportStatus->*Text*

+"',[PassportNumber]='"+ this->txtPassportNumber->*Text* +"',[PassportIssuedByCountry]='"+ this->txtPassportCountry->*Text* +"',[PassportExpiryDate]='"+ this->txtPassportExpiry->*Text* +"',[HomePageLink]='"+ this->txtHomepageLink->*Text* +"',[PassportSizePhotograph]='"+ "Resources\\" + this->usrnm+".jpg" +"' where [Username]='"+this->usrnm+"' ;

- SELECT * FROM Professor WHERE [Username]='"+this->usrnm+"'

Send Grades

Functions:

- txtb_student_TextChanged
- button1_Click
- dataGridView1_CellContentClick
- FillTable()
- FillCombo
- professor_sendGrades_Load

Queries:

- Select * from ProfCourses where [Username] = '" + this->usrnm + "' and [Session] like '20%" +(ltm->*tm_year*-100)+"%'"+";
- Select * from StudentCourses where [CourseID] = '" + course + "' and [Session] like '20%" +(ltm->*tm_year*-100)+"%'"+";
- Update StudentCourses set [Grade]='"+ this->comboBox2->*Text* +"' where [Username]='"+ this->txtb_student->*Text* +"' and [Session] like '20%" +(ltm->*tm_year*-100)+"%' and [CourseID] = '"+ this->cmb_selectcourse->*Text* +"';
- Select * from Student where [Username] = '" + this->txtb_student->*Text* +"';

TA REQUEST

Functions:

- professor_ta_manage_Load
- dataGridView1_CellContentClick
- sendRequest_Click

Queries:

- insert into Notification(SenderUsername,SendTime,SendDate,Message,OtherReceivers,Type) values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+req+"','"+"Admin"+"','Request');
- Select Username,RollNumber,FirstName,LastName from Phd where [Professor] = '"+this->usrnm+"' and [Status] = 'Active' "+";\

COURSES

Functions:

- professor_currentCourses_Load
- comboBox1_SelectedIndexChanged

Queries:

- Select * from ProfCourses where [Username] = '" + this->usrnm + "' and [Session] ='"+comboBox1->*Text*+"';
- Select * from CourseList where (others arguments are appended dynamacally)

CURRENT STUDENTS

Functions:

- professor_current_students_Load
- comboBox1_SelectedIndexChanged

Queries:

- Select * from StudentCourses where [CourseID] = '" + course + "' and [Session] like '20%" +(ltm->*tm_year*-100)+"%'"+";
- Select Username,RollNumber,FirstName,LastName,FieldOfSpecialisation from Phd where [Professor] = '"+this->usrnm+"' and [Status] = 'Active' "+";
- select distinct [CourseID] from CourseList;

VIEW NOTIFICATIONS:

Functions:

- viewNotifications_btn_Click
- dataGridView1_CellContentClick

Queries:

- Select [NotificationID],[SenderUsername],[SendTime],[SendDate],[Message] from Notification where [OtherReceivers]='"+this->usrnm+ "' and [Type]='"+"Message"+"';

ADD NOTIFICATIONS:

Functions:

- rb_message_CheckedChanged
- rb_request_CheckedChanged
- professor_addnotif_2_Load
- btn_post_Click
- btn_submit_Click
- comboBox1_SelectedIndexChanged
- professor_addnotif_3_Load

Queries:

- insert into Notification(SenderUsername,SendTime,SendDate,Message,DeptID,[Session],Programme,Type) values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+":"+ltm->*tm_sec*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+messageText->*Text*+"','"+checkedListBox1->*Items*[i]+"','"+sessionCombo->*Text*+"','"+programmeCombo->*Text*+"','Message');
- insert into Notification(SenderUsername,SendTime,SendDate,Message,OtherReceivers,Type) values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+messageText->*Text*+"','"+otherText->*Text*+"','Message');
- select distinct [CourseID] from CourseList;
- insert into Notification(SenderUsername,SendTime,SendDate,Message,OtherReceivers,Type)

values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+req+"','"+"Admin"+"','Request');

COURSE ADJUSTMENT APPROVAL

Functions:

- professor_course_adjustment_Load
- dataGridView1_CellContentClick
- approve_Click
- deny_Click

Queries:

- Select NotificationID,SenderUsername,SendTime,SendDate,Message from Notification where [OtherReceivers] = '"+this->usrnm+"' and [Type] = 'Request' "+";
- Delete from StudentCourses where [username]='"+textBox1->*Text*+"'and [CourseID]='"+strarr[1]+"'and [Session]='"+"2017-18"+"' ;
- Select Username from ProfCourses where [CourseID]='"+strarr[3]+"'and [Session]='"+"2017-18"+"' ;
- update Notification set [Message]='"+msg2+"',[OtherReceivers]='"+textBox1->*Text*+"' where [NotificationID]='"+nID;
- update StudentCourses set [CourseID]='"+strarr[3]+"' where [Username]='"+textBox1->*Text*+"' and [CourseID]='"+strarr[1]+"' and [Session]='"+"2017-18"+"';
- insert into Notification(SenderUsername,SendTime,SendDate,Message,OtherReceivers,Type) values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+strarr[1]+" changed successfully"+"','"+textBox1->*Text*+"','Message');
- insert into Notification(SenderUsername,SendTime,SendDate,Message,OtherReceivers,Type) values('"+this->usrnm+"','"+ltm->*tm_hour*+":"+ltm->*tm_min*+"','"+ltm->*tm_mday*+"/"+(ltm->*tm_mon*+1)+"/"+(ltm->*tm_year*+1900)+"','"+strarr[1]+" request denied"+"','"+textBox1->*Text*+"','Message');
- update Notification set [Message]='"+msg2+"' where [NotificationID]="+nID+";

WARNINGS

- **Entering any special character in any of the textboxes will lead to crashing of the application.**
- **It is vulnerable to SQL Injection Attack.**