College Management System 1.0

Technical Manual

Developed By

Chandrakanth Ganji

Jai Singh

Akib Khan

Preet Sai Mutneja

Sambit Padhi

Index :

Introduction:

This application is made for efficient administration and management of college. It assists the authorities to manage and control various activities in college and at the same time it helps the students in efficient management of their time. It gives the faculty enough power and access to manage their courses easily so as to maximize their time for research activities without compromising on the quality of teaching. This application also enables guests to view the various programmes and facilities offered by the college.

Requirements:

This application is made for windows 7. It requires .NET 4.0 framework. The behaviour of this application on previous versions of .NET framework has not been tested and hence previous versions of .NET framework are unsupported. It also requires MySQL workbench 5.6 to host database.

Database:

The database consists of 9 tables:

1. course (course_code , credit , course_name , faculty_id , department)
2. course_reg(roll_no , semester , course_code , grade , credit)
3. department (dept_id , dept_name , description)
4. discussion_answer (post_id, question_id ,user_id ,post ,timestamp)
5. discussion_question (post_id , heading , post, user_id, timestamp)

6. faculty(faculty_id , name ,webmail , contact_no , address , position ,department , interest, dept_id)
7. login (id_login , user_type ,user_name , password)
8. noticeboard(notice_id , heading ,text ,topic ,timestamp)
9. student(roll_no, name, webmail ,dept_id, conatact , address ,admission_year)

Functions Used:

The class data(defined in data.h and data.cpp) acts as an interface for the application to interact with the database. Various Functions are available in this class. These functions have been made taking into account both the present and future needs of application. Effort has been taken that developer who has to implement a new functionality does not have to care about the details of interacting with the database. Various overloaded functions have been provided for convenience. All the database functions have a try-catch-finally exception handling mechanism. If any exception occurs during the execution then an appropriate Dialog box is displayed indicating the nature of exception. Connection to the database is closed as soon as the execution of query is over.

Data Members:

MySqlConnection^ conDatabase: Connection String for connecting to the MySql database.

Constructor:

data(): Initializes **conDatabase**

Functions:

bool **notOpen**(): Checks if the connection to the database is already open.

MySqlCommand^ **query**(String^ qry,MySqlConnection^ conDatabase): returns a MySqlCommand corresponding to the database query

Login Table:

1. int **getId**(String^ user_name): returns login id
2. String^ **getUserbyId**(int id): returns username

3. List< String ^ >^ **get_login_details**(int id): returns (idlogin,usertype,username,password)
4. bool **addLogin**(int user_type,String^ user_name,String^ password): returns true in case of successful addition of user false otherwise
5. bool **login**(String^ user_name,String^ password,int user_type): returns true in case of successful login false otherwise
6. bool **changePassword**(int id,String^ password): returns true if password changed successfully false otherwise
7. bool **changePassword**(String^ user_name,String^ password): returns true if password changed successfully
8. bool **removeLogin**(int id) : returns true if successful login

Courses table:

1. List< List< String^ >^ **>^ viewCourses_department**(String^ department): returns an array of (coursecode,credit,coursename,faculty_id)
2. List< List< String^ >^ >^ **viewCourses_faculty**(int fac_id): returns an array of (coursecode,credit,coursename,department)
3. List<String^>^ **viewCourses_Code**(String^ courseCode): returns (courseCode, Credit, course_name, fac_id, department)
4. bool **addCourse**(String ^course_code,int credit,String^ course_name,int faculty_id,String^ department): returns true if the course is added successfully
5. bool **removeCourse**(String ^courseCode): returns true in case of successful removal
6. bool **updateCourse**(String^ old_code,String ^course_code,int credit,String^ course_name,int faculty_id,String^ department): returns true in case of successful update
7. List<String^>^ **viewAllCourse**(): returns all courses

Faculty table

1. int **getFacultyId**(String^ webmail): returns faculty id

2. List< String^ >^ **faculty_details**(int facullty_id): returns (name,webmail,contact_no,address,position,department,interest,dept_id)
3. List< String^ >^ **faculty_details**(String^ webmail): returns (name,webmail,contact_no,address,position,department,interest,dept_id)
4. bool **add_faculty**(String^ name,String^ password,String^ webmail,String^ contact_no,String^ address,String^ position,String^ department,String^ interest,int dept_id) : returns true in case of successful addition , returns false otherwise
5. bool **remove_faculty**(int id): returns true in case of successful addition , returns false otherwise
6. bool **update_faculty**(int fac_id,String^ name,String^ webmail,String^ contact_no,String^ address,String^ position,String^ department,String^ interest,int dept_id): returns true in case of successful updation , returns false otherwise
7. String^ data::**getFacultyById**(int id): returns webmail of faculty
8. List<String^>^ **get_faculty_department**(String^ department): returns list of faculty of a given department
9. long long **getFacultyIDbyName**(String^ naam): returns faculty ID by name

Department table :

1. List< String^ >^ **department_details**(String^ department): returns (dept_id,dept_name,description)
2. List< String^ >^ **department_details**(int id): returns (dept_id,dept_name,description)
3. int **getDeptId**(String^ department): returns deptId
4. String^ **getDeptbyId**(int id): returns name of department
5. bool **addDepartment**(String^ dept_name,String^ desription): returns true in case of successful addition false otherwise
6. bool **updateDepartment**(int id,String^ description): returns true in case of successful updation , false otherwise
7. List<String^>^ **get_all_department()**: returns list of all departments

Student table :

1. bool **addStudent**(long long roll_no,String^ password,String^ name,String^ webmail,String^ department,String^ contact,String^ address,int adm_year): returns true in case of successful addition, false otherwise
2. bool **removeStudent**(long long roll_no): returns true in case of successful removal ,false otherwise
3. bool **updateStudent**(long long roll_no,String^ name,String^ webmail,String^ department,String^ contact,String^ address,int adm_year): returns true in case of successful updation, false otherwise
4. String^ **getStudentById**(long long roll_no): returns webmail of student
5. List< List< String^ >^ >^ **getStudentByDep**(String^ department): returns (name,admission_year) with admission year sorted in ascending order
6. List<String^>^ **getStudentByYear**(int year): returns a list of student
7. long long **getStudentRoll**(String^ webmail): returns roll_no of student
8. List<String^>^ **student_details**(long long roll_no): returns an array (name, webmail, dept_id, contact_no, address,admission year)

course_reg table

1. List<String^ >^ **getStudentByCourse**(String^ courseCode): returns student registered for particular course
2. double **getSpi**(long long roll_no,int sem): returns spi
3. double **getCpi**(long long roll_no): returns cpi
4. bool **addStudentToCourse**(long long roll_no,String^ courseCode,int sem): returns true in case of successful addition , false otherwise

5. bool **updateGrade**(int roll_no,String^ courseCode,int grade): returns true in case of successful updation, false otherwise
6. List<String^>^ **get_all_course_student**(int roll_no): returns all course for a student
7. List<List<String^>^>^ **get_all_grades**(int roll_no): returns an array of (semester,course_code,grade,credit)
8. bool **changeCredit**(String^ courseCode,int credit): returns true in case of successful change of credit , false otherwise

Discussion:

1. **List<String ^>^ getQuestionById(long long questionId)**:
   - **1.1.** This function is used to get attributes of a question.
   - **1.2.** **Arguments:**
     - **1.2.1. questionID:** Takes a 'long long' as it's argument. This ID is unique for every 'post' in discussion_question' table in 'database' database.
   - **1.3.** **Return Value:** Returns a list of Strings, each String is a column in 'discussion_question' table. Strings are returned in same order as in table. If no 'post' with this ID exists, it return a 'nullptr'.

2. **bool addQuestion(String^ questionHeading, String^ questionPost, String^ userID)**:
   - **2.1.** This function is used to add a new question in 'dicussion_question' table.
   - **2.2.** **Arguments:**
     - **2.2.1. questionHeading:** Every question is associated with a 'questionHeading' that provides an idea for the user to what is the post about.
     - **2.2.2. questionPost:** This is the original Post associated with question in 'discussion_question' table.
     - **2.2.3. userId:** Login ID is associated with each question. It provides the details of the person who posted the question.
   - **2.3.** **Return Value:** Returns 'true' if insertion is successful else it returns 'false'.

3. **bool deleteQuestion(long long questionID)**:
   - **3.1.** This function is used to delete an already existing question in 'discussion_question' table. It also deletes all the answers associated with it.
   - **3.2. Arguments:**
     - **3.2.1. questionID:** Takes a 'long long' as it's argument. This ID is unique for every 'post' in discussion_question' table in 'database' database.
   - **3.3. Return Value:** Returns 'true if deletion is successful else returns 'false'.

4. **List<List<String^ >^ >^ getAnswerById(long long questionID)**:
   - **4.1.** This function is used to get all answers associated with a question from 'discussion_answer' table. Each answer is identified by 'question_id' associated with it. There may be multiple answers associated with a question.
   - **4.2. Arguments:**
     - **4.2.1. questionID:** As stated above, each answer is identified by 'question_id' which may not be unique due to existence of multiple answers.
   - **4.3. Return Value:** Returns multiple answers for a 'question_id' with each answer being a list of String with all its attributes in same order as in 'discussion_answer' table.

5. **bool addAnswer(String^ answer, String^ userId, long long questionID)**:
   - **5.1.** This function is used to add a new answer for a question in 'discussion_answer' table.
   - **5.2. Arguments:**
     - **5.2.1. question:** Answer for a question as posted by a user for a question.
     - **5.2.2. userID:** Identity of user who posted the answer in form of Login ID.
     - **5.2.3. questionId:** Specifies the question for which the answer is posted.
   - **5.3. Return Value:** Returns 'true' if insertion is successful else it returns 'false'.

6. **bool deleteAnswer(long long answerID)**:
   - **6.1.** This function is used to delete an existing answer in 'discussion_answer' table.
   - **6.2. Arguments:**
     - **6.2.1. answerID:** Each answer is associated with unique 'post_id', from which an answer can be identified.
   - **6.3. Return Value:** Returns 'true' if deletion is successful, else returns 'false'.

7. **List<String^ >^ getNoticebyId(long long noticeID)**:
   - **7.1.** This function is used to get a Notice from /noticeboard' table.
   - **7.2. Arguments:**
     - **7.2.1. noticeID:** Each notice is identified by unique notice_id and can be queried using the same.
   - **7.3. Return Value:** It returns a list of strings with each string as an attribute of a notice. They are in same order as columns in 'noticeboard' table.

8. **bool addNotice(String^ noticeHeading, String^ noticeText, String^ noticeTopic)**:
   - **8.1.** This function is used add a new notice to 'noticeboard' table.
   - **8.2. Arguments:**
     - **8.2.1. noticeHeading:** Specifies heading of a notice being added.
     - **8.2.2. noticeText:** Specifies full notice to be added.
     - **8.2.3. noticeTopic:** Specifies topic of the notice to be added.
   - **8.3. Return Value:** Returns 'true' if insertion is successful else returns 'false'.

9. **bool deleteNoticeById(long long noticeID)**:
   - **9.1.** This function is used to delete a notice with a particular noticeId(unique for a notice).
   - **9.2. Arguments:**
     - **9.2.1. noticeID:** As stated earlier, noticeID is unique for a particular notice, hence used to locate a notice and delete it.
   - **9.3. Return Value:** returns 'true' if deletion is successful else returns 'false'.

10. List<List<String ^>^ >^ getTenQuestions(void):

10.1. This function is used to fetch latest ten questions (or less) posted, from ‘discussion_question’ table.

10.2. Arguments: None.

10.3. Return Value: Returns 2-d List of Strings, i-th list being attributes of i-th question. If number of questions is less than ten (suppose n), then no more than n lists are returned. If ‘discussion_question’ table is empty, then ‘nullptr’ is returned.

11. List<List<String^ >^ >^ getTenNotices(void):

11.1. This function is used to fetch latest ten notices (or less) posted, from ‘noticeboard’ table.

11.2. Arguments: None.

11.3. Return Value: Returns 2-d List of Strings, i-th list being attributes of i-th notice. If number of notices is less than ten (suppose n), then no more than n lists are returned. If ‘noticeboard’ table is empty, then ‘nullptr’ is returned.

1. Login Page:

```
int a;
if(comboBox1->Text == "admin") a = 0;
else if(comboBox1->Text == "faculty") a = 1;
else if(comboBox1->Text == "student") a = 2;
else a = 3;
 data ^m=gcnew data();
if(m->login(textBox2->Text,textBox1->Text,a))
{
        if(a == 0)
                // MessageBox::Show("page will open soon");
                {
                admin1 ^f=gcnew admin1();
                        f->ShowDialog();
                }
        else if(a == 1)
                //faculty1->Show();
        {
                int d;
                d = m->getId(textBox2->Text);
                faculty1 ^f=gcnew faculty1(d);
                f->ShowDialog();
        }
        else if(a == 2)
        {
                int d;
                d = m->getId(textBox2->Text);
                stydent1 ^f=gcnew stydent1(d);
                f->ShowDialog();
        }
}
else  MessageBox::Show("login failed");
```

In function login we require 3 arguments username,password and usertype. Usertype is integer, we are taking admin as 0, faculty as 1 and student as 2. Login function is bool function, if usertype, usename and password matches it returns ‘true’ otherwise ‘false’. After login it redirects to appropriate forms.

2. Guest functions and features:

1. About: It describes the college for which software developed.
2. Noticeboard:

```
data^ database=gcnew data();
List< List<String^>^>^  courses=database>getTenNotices();
 for each( List<String^>^ course in courses)
                                    {
 this->dataGridView1->
 Row>Add(course[0],course[1],course[2],course[3],course[4],course[2]=="NU
 LL"?"Active":"Over");
                                    }
```

The function getTenNotices() returns latest 10 Notices and this notices are displayed on dataGridView. Courses is list of strings, here course is each row in courses. Course[0] is notice number, course[1] is heading, course[2] is text, course[3] is topic, course[4] is times at which notice posted.

3. Department:

```
this->comboBox1->DataSource=dat->get_all_department();
```

Get all departments and and store it into combo-box.

1. Courses:

```
String^ constring
L"datasource=localhost;port=3306;username=root;password=aquib12345";
                                        MySqlConnection^ conDataBase= gcnew
MySqlConnection(constring);
MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select
course_code,credit,course_name from database.course where department=
'"+dept+"';",conDataBase);
try{ MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
sda->SelectCommand = cmdDataBase ;
DataTable^ dbdataset = gcnew DataTable();
sda->Fill(dbdataset);
BindingSource^ bSource = gcnew BindingSource();
dbdataset->DefaultView->AllowDelete=false;
dbdataset->DefaultView->AllowEdit=false;
bSource->DataSource = dbdataset->DefaultView ;
f->dataGridView1->DataSource = bSource ;
sda->Update(dbdataset) ;
}catch(Exception^ex){
MessageBox::Show(ex->Message);}
```

First connect to database and find courses according to department. It will return list of array of strings. Store Course

information(course code,credit,course name) in dataGridView and show information.

2. Faculty:

```
String^ constring
L"datasource=localhost;port=3306;username=root;password=aquib12345"
;
MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);
MySqlCommand^ cmdDataBase =gcnew
MySqlCommand("select,name,webmail,contact_no,address,position,intere
st from database.faculty where department='"+dept+"';",conDataBase);
try{ MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
sda->SelectCommand = cmdDataBase ;
DataTable^ dbdataset = gcnew DataTable();sda->Fill(dbdataset);
BindingSource^ bSource = gcnew BindingSource();
dbdataset->DefaultView->AllowDelete=false;
dbdataset->DefaultView->AllowEdit=false;
bSource->DataSource = dbdataset->DefaultView ;
f->dataGridView1->DataSource = bSource ;
sda->Update(dbdataset) ;
 }catch(Exception^ex){
MessageBox::Show(ex->Message);}
```

First connect to database and find Faculty according to department. It will return array of strings. Store faculty information(name,webmail,contact no,address,position,interest) in dataGridView and show information.

3. Students

```
String^ constring
L"datasource=localhost;port=3306;username=root;password=aquib12345"
;
 MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);
 MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select
roll_no,name,webmail,contact,address,admission_year from
database.student where dept_id="+id+";",conDataBase);
 try{
 MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
sda->SelectCommand = cmdDataBase ;
 DataTable^ dbdataset = gcnew DataTable();
 sda->Fill(dbdataset);
BindingSource^ bSource = gcnew BindingSource();
dbdataset->DefaultView->AllowDelete=false;
dbdataset->DefaultView->AllowEdit=false;
 bSource->DataSource = dbdataset->DefaultView ;
```

```
f->dataGridView1->DataSource = bSource ;
sda->Update(dbdataset) ;
    }catch(Exception^ex){
MessageBox::Show(ex->Message);}
```

First connect to database and find Syudents according to department. It will return array of strings. Store student information(roll_no,name,webmail,contact,address, admission year) in dataGridView and show information.

3.Student function and feature:

1. Courses:

```
 data^ database=gcnew data();
String^ webmail=database->getUserbyId(id);
 long long roll_no=database->getStudentRoll(webmail);
 List< List<String^>^>^     courses=database>get_all_grades(roll_no);
 for each( List<String^>^ course in courses)
{this->dataGridView1->Rows-
>Add(course[0],course[1],course[2],course[3],course[2]=="0"?"Active":"Over");
                                    }
```

Data is header file where all functions are defined. We have used get_all_grade(roll_no) function. We initially had id for student. To get roll number first we get webmail of student using function getUserbyId(id). Then we use function getStudentRoll(webmail) to obtaine roll number. Then calling function get_all_grade(roll_no) which return array of strings which are semester, course code, grade , credit and status respectively. And this information is displayed as dataGridview.

2. Details:

```
data^ database=gcnew data();
        String^ webmail=database->getUserbyId(id);
        long long roll_no=database->getStudentRoll(webmail);
        List<String^>^ detail=database->student_details(roll_no);
        textBox1->Text=detail[0]; // Name
        textBox2->Text=System::Convert::ToString(roll_no); // Roll No
```

```
        textBox3->Text=detail[1]; // Webmail
        textBox4->Text=detail[3]; // contact_no
        textBox5->Text=detail[4]; // address
        textBox6->Text=System::Convert::ToString(database->getCpi(roll_no));
// CPI
        textBox7->Text=database-
>getDeptbyId(System::Convert::ToInt32(detail[2])); // department
        textBox8->Text=detail[5]; // admission year
        // name,webmail,dept_id,contact_no,address,admission year
```

Function student_details(roll no) returns list of strings with details of student (name,webmail,dept id,contact no,address,admission year) Roll number is got using webmail which is further got by id. Functions used were getUserbyId(id) and getStudentRoll(webmail);

3. Discussion:

```
tableLayoutPanel1->Controls->Clear();
                                tableLayoutPanel1->Controls-
>Add(buttonDiscuss, 10);
                                x=nullptr;
                                data^ f = gcnew data();
                                x= f->getTenQuestions();
                                int num_posts=x->Count;
                                //array<System::Windows::Forms::Button^ >^
buttons_delete = gcnew array<System::Windows::Forms::Button^
>(num_posts);
                                array<System::Windows::Forms::Label^
labels_timestamp = gcnew array<System::Windows::Forms::Label^
>(num_posts);
                                array<System::Windows::Forms::Label^ >^
labels_posts = gcnew array<System::Windows::Forms::Label^ >(num_posts);
                                array<System::Windows::Forms::Label^ >^
labels_numans = gcnew array<System::Windows::Forms::Label^
>(num_posts);
                                for(int i=0; i<num_posts; ++i) {
                                        if(x[i]!=nullptr) {
                                                labels_timestamp[i] = gcnew
System::Windows::Forms::Label();
                                                labels_posts[i] = gcnew
System::Windows::Forms::Label();
                                                labels_numans[i]=gcnew
System::Windows::Forms::Label();
                                                labels_timestamp[i]->Text = "@" +
x[i][3] + " " + x[i][4];
```

```
                                                labels_timestamp[i]->Dock =
System::Windows::Forms::DockStyle::Fill;
                                                labels_timestamp[i]->Font = gcnew
System::Drawing::Font(labels_timestamp[i]->Font->FontFamily, 9);
                                                labels_posts[i]->Text = x[i][1];
                                                labels_posts[i]->Dock =
System::Windows::Forms::DockStyle::Fill;
                                                labels_posts[i]->Font = gcnew
System::Drawing::Font(labels_posts[i]->Font->FontFamily, 9);
                                                if((f-
>getAnswerById(Convert::ToInt64(x[i][0])) != nullptr)) {
                                                        labels_numans[i]->Text =
Convert::ToString(f->getAnswerById(Convert::ToInt64(x[i][0]))->Count) + " " +
"Answers.";
                                                } else {
                                                        labels_numans[i]->Text =
"Waiting for an Answer.";
                                                }
                                                labels_numans[i]->Dock =
System::Windows::Forms::DockStyle::Fill;
                                                labels_numans[i]->Font = gcnew
System::Drawing::Font(labels_numans[i]->Font->FontFamily, 9);
                                                tableLayoutPanel1->Controls-
>Add(labels_timestamp[i], 0, i);
                                                tableLayoutPanel1->Controls-
>Add(labels_posts[i], 1, i);
                                                tableLayoutPanel1->Controls-
>Add(labels_numans[i], 2, i);
                                        }
                                }
                                if(num_posts>0) labels_posts[0]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost0);
                                if(num_posts>1) labels_posts[1]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost1);
                                if(num_posts>2) labels_posts[2]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost2);
                                if(num_posts>3) labels_posts[3]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost3);
                                if(num_posts>4) labels_posts[4]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost4);
                                if(num_posts>5) labels_posts[5]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost5);
                                if(num_posts>6) labels_posts[6]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost6);
                                if(num_posts>7) labels_posts[7]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost7);
                                if(num_posts>8) labels_posts[8]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost8);
                                if(num_posts>9) labels_posts[9]->Click += gcnew
System::EventHandler(this, &faculty4::handlepost9);
```

This function used to add discussion with argument heading, post and user id. You can also give answer to particular question already present in discussion group. This function takes argument question id, user id and post (answer).

4. Noticeboard:

```
data^ database=gcnew data();
List< List<String^>^>^  courses=database>getTenNotices();
 for each( List<String^>^ course in courses)
                                    {
this->dataGridView1->
Row>Add(course[0],course[1],course[2],course[3],course44],course[2]=="NUL
L"?"Active":"Over");
                                    }
```

The function getTenNotices() returns latest 10 Notices and this notices are displayed on dataGridView. Courses is list of strings, here course is each row in courses. Course[0] is notice number, course[1] is heading, course[2] is text, course[3] is topic, course[4] is times at which notice posted.

:

```
#include "faculty2.h"
#include "faculty3.h"
#include "faculty4.h"
#include "faculty5.h"
#include "data.h"
#include "noticeboard.h"
```

This code includes all the form which is reachable from this form faculty one form.

```
private: System::Void faculty1_Load(System::Object^  sender, System::EventArgs^
e) {
                                    data^ t=gcnew data();
                                    String^ naam=t->getUserbyId(id);
                                    label2->Text = naam;
                          }
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e) {
//                        this->Hide();
                          faculty2^ nf=gcnew faculty2(id);
                          nf->ShowDialog();
```

```
        }
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^
e) {

    //          this->Hide();
                faculty3^ n=gcnew faculty3(id);
                n->ShowDialog();

        }
private: System::Void button3_Click(System::Object^  sender, System::EventArgs^
e) {
          //    this->Hide();
                faculty4^ n=gcnew faculty4(id);
                n->ShowDialog();
        }
private: System::Void button4_Click(System::Object^  sender, System::EventArgs^
e) {
                // this->Hide();
                noticeboard^ n=gcnew noticeboard();
                n->ShowDialog();
        }
private: System::Void pictureBox1_Click(System::Object^  sender,
System::EventArgs^  e) {
        }
private: System::Void button5_Click(System::Object^  sender, System::EventArgs^
e) {
                faculty5^ n=gcnew faculty5(id);
                n->ShowDialog();
        }
};
}
```

These are the code which after click the bottom provided on faculty1 form to go another faculty form.

```
#include "data.h"
```

This code include the data h form which communicate to database.

This is included in faculty2 form.

```
private: System::Void faculty2_Load(System::Object^  sender, System::EventArgs^
e) {

                data^ database=gcnew data();

                String^ webmail = database->getUserbyId(id);
                int fac_id = database->getFacultyId(webmail);
```

```
                        List< List<String^>^>^ courses=database->viewCourses_faculty(fac_id);
                        for each( List<String^>^ course in courses)
                        {
                                this->dataGridView1->Rows->Add(course[0],course[1],course[2],course[3]);
                        }

                }
```

These are the code which take information to database and print in faculty2 form. For this we use data grid view function which print course details in from.

```
private: System::Void faculty3_Load(System::Object^  sender, System::EventArgs^  e) {
                        data^ database=gcnew data();

                        String^ webmail = database->getUserbyId(id);
                        List<String^>^ detail=database->faculty_details(webmail);
                        textBox1->Text=detail[0]; // Name

                        textBox2->Text=detail[1]; // Webmail
                        textBox3->Text=detail[2]; // contact_no
                        textBox4->Text=detail[4]; // position
                        textBox5->Text=detail[5]; //department
                        textBox6->Text=detail[6];//interest area
                        textBox7->Text=detail[3];//address
                        textBox8->Text=detail[7];//dept_id
                        //textBox8->Text=detail[5]; // admission year
                }
```

These are code print faculty details on form. This code matches the faculty user id and his username from database and prints his details.

This will print faculty details in textbox on form.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
                        data^ t=gcnew data();
                        String^ webmail = t->getUserbyId(id);
                        int fac_id = t->getFacultyId(webmail);
                        int dep_id = System::Convert::ToInt64(textBox8->Text);
                        t->update_faculty(fac_id,textBox1->Text,textBox2->Text,textBox3->Text,textBox7->Text,textBox4->Text,textBox5->Text,textBox6->Text,dep_id);
```

```
MessageBox::Show("details updated");
```

This code provides facility to update information of faculty after click the update bottom given on form. He/she can update his/her new information in database.

```
private: System::Void faculty4_Load(System::Object^  sender, System::EventArgs^
e) {
                                    data^ f = gcnew data();
                                    List<List<String^ >^ >^ x= f->getTenQuestions();
                                    int num_posts=x->Count;

 array<System::Windows::Forms::Label^ >^ labels_timestamp = gcnew
array<System::Windows::Forms::Label^ >(num_posts);
array<System::Windows::Forms::Label^ >^ labels_posts = gcnew
array<System::Windows::Forms::Label^ >(num_posts);
array<System::Windows::Forms::Label^ >^ labels_numans = gcnew
array<System::Windows::Forms::Label^ >(num_posts);
                                    MessageBox::Show(x[0][4]);
                                    for(int i=0; i<num_posts; ++i) {
                                            if(x[i]!=nullptr) {
labels_timestamp[i] = gcnew System::Windows::Forms::Label();
labels_posts[i] = gcnew System::Windows::Forms::Label();
labels_numans[i]=gcnew System::Windows::Forms::Label();
labels_timestamp[i]->Text = "@" + x[i][3] + " " + x[i][4];
labels_timestamp[i]->Dock = System::Windows::Forms::DockStyle::Fill;
labels_posts[i]->Text = x[i][1];
labels_posts[i]->Dock = System::Windows::Forms::DockStyle::Fill;
if((f->getAnswerById(Convert::ToInt64(x[i][0])) != nullptr)) {
labels_numans[i]->Text = Convert::ToString(f-
>getAnswerById(Convert::ToInt64(x[i][0]))->Count) + " " + "Answers.";
                                                    } else {
labels_numans[i]->Text = "Waiting for an Answer.";
                                                    }
labels_numans[i]->Dock = System::Windows::Forms::DockStyle::Fill;
tableLayoutPanel1->Controls->Add(labels_timestamp[i], 0, i);
tableLayoutPanel1->Controls->Add(labels_posts[i], 1, i);
tableLayoutPanel1->Controls->Add(labels_numans[i], 2, i);
                                            }
                                    }
                            }
```

This is code for discussion. Faculty can discus to student related his/her problem.

```
private: System::Void faculty5_Load(System::Object^  sender, System::EventArgs^
e) {
                                    List<String^>^ courses=gcnew List<String^>();
                                    data^ database = gcnew data();
                                    String^ webmail = database->getUserbyId(id);
                                    int fac_id = database->getFacultyId(webmail);
                                    String^ constring =
L"datasource=localhost;port=3306;username=root;password=aquib12345";
                                    MySqlConnection^ conDatabase= gcnew
MySqlConnection(constring);

                                    MySqlCommand^ cmdDatabase =gcnew
MySqlCommand("select course_code from database.course where faculty_id=
"+fac_id+";",conDatabase);
                                    MySqlDataReader^ myReader;
                                    try{
                                            if(database->notOpen())
                                                    conDatabase->Open();
                                             myReader=cmdDatabase->ExecuteReader();
                                             while(myReader->Read())
                                             {
                                                    courses->Add(myReader->GetString(0));

                                             }
                                    }
                                    catch(Exception^ ex)
                                    {
                                            MessageBox::Show(ex->Message);
                                    }
                                    finally
                                    {
                                            if(!(database->notOpen()))
                                                    conDatabase->Close();
                                    }

                                    this->comboBox1->DataSource = courses;


                            }
private: System::Void comboBox2_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {

                            }
private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e) {
                            data^ f = gcnew data();
                            this->comboBox2->DataSource = f->getStudentByCourse(this-
>comboBox1->Text);
                    }
```

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e) {
                         data^ f = gcnew data();
                         int a = System::Convert::ToInt64(this->comboBox2->Text);
                         int b = System::Convert::ToInt64(textBox1->Text);
                        if(f->updateGrade(a,this->comboBox1->Text,b))
                        {
                                 MessageBox::Show("grade updated succesfully");
                        }
                        else {   MessageBox::Show("update not succed");       }
                }
};
}
```

From given above code faculty can update student cpi spi in which course what grade he/she got through his/her roll number .

5.Admin functions and Features:

admin1.h :

```
private: System::Void admin1_Load(System::Object^  sender, System::EventArgs^
e) {

}
```

When admin is logged in admin form is loaded.

```
private: System::Void btn_logout_Click(System::Object^  sender,
System::EventArgs^ e) {

                this->Close();

 }
```

When logout button is clicked admin form closes.

```
private: System::Void student_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

 {

                admin1_student^ adm_stu=gcnew admin1_student();
                adm_stu->ShowDialog();

 }
```

When the linklabel student is clicked admin_student form is loaded.

```
private: System::Void faculty_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

 {

                admin1_faculty^ adm_fac=gcnew admin1_faculty();
                adm_fac->ShowDialog();

 }
```

When the linklabel faculty is clicked admin_faculty form is loaded.

```
private: System::Void dept_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

                admin1_dept^ adm_dept=gcnew admin1_dept();
                    adm_dept->ShowDialog();

}
```

When the linklabel department is clicked admin_dept form is loaded.

```
private: System::Void coursereg_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

                admin1_courgereg ^f=gcnew admin1_courgereg();
                    f->ShowDialog();

 }
```

When the linklabel course registration is clicked admin_coursereg form is loaded.

---

admin1_student.h :

```
        private: System::Void admin1_student_Load(System::Object^  sender,
System::EventArgs^  e)

 {

 }
```

When the student linklabel in admin1 form is clicked admin1_student is loaded.

```
        private: System::Void add_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)
```

```
{
        admin1_student_add^ adm_stu_add=gcnew admin1_student_add();
        adm_stu_add->ShowDialog();
}
```

When the add student linklabel in admin1_student form is clicked admin1_student_add is loaded.

```
        private: System::Void remove_linkLabel_LinkClicked(System::Object^
sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)
{
        admin1_student_remove^ adm_stu_remv=gcnew admin1_student_remove();
        adm_stu_remv->ShowDialog();
}
```

When the modify student record linklabel in admin1_student form is clicked admin1_student_remove is loaded.

---

admin1_student_add.h :

```
private: System::Void admin1_student_add_Load(System::Object^  sender,
System::EventArgs^  e)
{
         data ^f=gcnew data();
        this->dept_comboBox->DataSource=f->get_all_department();
}
```

When student linklabel in add student is clicked admin1_student_add form is loaded with the departments in dept_combobox.

```
private: System::Void addstu_button_Click(System::Object^  sender,
System::EventArgs^  e)

{


          data ^f=gcnew data();

         long long r_no= System::Convert::ToInt32(rollno_textBox->Text);

         int ad_year=System::Convert::ToInt32(admyear_textBox->Text);

         if(f->addStudent(r_no,passwd_textBox->Text,stuname_textBox-
>Text,webmailid_textBox->Text,dept_comboBox->Text,contactno_textBox-
>Text,Presentadd_textBox->Text,ad_year))

                        {

                                MessageBox::Show("Student added successfully");

                        }

                        else

                        {

                                MessageBox::Show("Student not added");

                        }

}
```

When add button is clicked it sends the corresponding details of a student to database and the information is stored in student table.

---

admin1_student_remove.h :

```
private: System::Void re_load()
```

```
 {
 String^ constring =
L"datasource=localhost;port=3306;username=root;password=root";
 MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);
MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select * from
database.student;",conDataBase);


try
{
 MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
 sda->SelectCommand = cmdDataBase ;
dbdataset = gcnew DataTable();
 sda->Fill(dbdataset);
 BindingSource^ bSource = gcnew BindingSource();
 bSource->DataSource = dbdataset ;
 this->dataGridView1->DataSource = bSource ;
 sda->Update(dbdataset) ;
 }
catch(Exception^ex)
{
 MessageBox::Show(ex->Message);}
}
```

Using the attribute in the combobox1 students can be searched and when the remove button is clicked corresponding information of that student is removed from the database. When the update button is clicked then the details of that student can be updated.

```
private: System::Void admin1_student_remove_Load(System::Object^  sender,
System::EventArgs^  e)

{

this->comboBox1->DropDownStyle=ComboBoxStyle::DropDownList;

re_load();

}
```

When the modify student record linklabel in admin1_student form is clicked admin1_student_remove is loaded with drop down list having Roll_no, Name, Webmail, Admission_Year, Dept_Id and the reload function is called.

```
private: System::Void dataGridView1_CellContentClick(System::Object^  sender,
System::Windows::Forms::DataGridViewCellEventArgs^  e)

 {

 try

{

          if(e->ColumnIndex==0)

          {

        int roll_n=System::Convert::ToInt32(this->dataGridView1->Rows[e-
>RowIndex]->Cells[2]->Value);

          admin1_student_updateinfo^ update_inf=gcnew
admin1_student_updateinfo(roll_n);

                  update_inf->ShowDialog();

                  re_load();

          }

          else if(e->ColumnIndex==1)

         {

             int roll_n=System::Convert::ToInt32(this->dataGridView1->Rows[e-
>RowIndex]->Cells[2]->Value);

                 data^ database=gcnew data();
```

```
                database->removeStudent(roll_n);
                re_load();
            }
}
catch(Exception^ ex)
{
MessageBox::Show("Empty Table or No Search Results");
}
}
```

Searches according to the attribute present in the combobox and the entry in the text box and displays student records. If the update button is used , admin1_student_updateinfo form is loaded with the student details where the admin can change some details of that student and remove button is used to remove particular student from the database.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e)
{
        DataView^ dv=gcnew DataView(dbdataset);
        if(comboBox1->Text=="Name" || comboBox1->Text=="Webmail")
        dv->RowFilter=String::Format("{0} like '%{1}%'",comboBox1-
>Text,searchBox->Text);
        else if(comboBox1->Text=="Dept_Id" || comboBox1->Text=="Roll_no" ||
comboBox1->Text=="Admission_Year")
        {
                try
                 {
                        dv->RowFilter=String::Format("{0}={1}",comboBox1-
>Text,System::Convert::ToInt32(searchBox->Text));
                 }
                catch(Exception^)
```

```
                {
                    MessageBox::Show("Not a Valid Search Input");
                }
            }
            this->dataGridView1->DataSource=dv;
}
```

Searches student list according to the attribute in combobox in this form.

---

admin1_faculty.h :

```
private: System::Void admin1_faculty_Load(System::Object^  sender,
System::EventArgs^  e)
{
}
```

This form loads when faculty linklabel is clicked in admin1.h .

```
private: System::Void add_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)
{
    admin1_faculty_add^ adm_fac_add=gcnew admin1_faculty_add();
    adm_fac_add->ShowDialog();
}
```

When the add faculty linklabel is clicked admin1_faculty_add is loaded.

```
private: System::Void remove_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)
{
    admin1_faculty_remove^ adm_fac_remv=gcnew admin1_faculty_remove();
    adm_fac_remv->ShowDialog();
```

```
}
```

When the modify faculty record linklabel is clicked admin1_faculty_remove is loaded.

---

admin1_faculty_add.h :

```
private: System::Void btn_add_Click(System::Object^  sender, System::EventArgs^
e)

{

        data ^f=gcnew data();

        int d_id=f->getDeptId(dept_comboBox->Text);

        f->add_faculty(stuname_textBox->Text ,passwd_textBox-
>Text,webmailid_textBox->Text,contactno_textBox->Text,facadd_textBox-
>Text,posn_textBox->Text,dept_comboBox->Text,intarea_textBox->Text, d_id);

}
```

When the add button is clicked the information of corresponding faculty is added to database in faculty table.

```
private: System::Void admin1_faculty_add_Load(System::Object^  sender,
System::EventArgs^  e)

 {

         data ^f=gcnew data();

      this->dept_comboBox->DataSource=f->get_all_department();

 }
```

This form is loaded with the departments in the department table in the database.

---

admin1_faculty_remove.h :

```
private: System::Void re_load()
 {
        String^ constring =
L"datasource=localhost;port=3306;username=root;password=root";
        MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);
        MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select * from
database.faculty;",conDataBase);
        try
       {
        MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
       sda->SelectCommand = cmdDataBase ;
       dbdataset = gcnew DataTable();
        sda->Fill(dbdataset);
        BindingSource^ bSource = gcnew BindingSource();
        bSource->DataSource = dbdataset ;
        this->dataGridView1->DataSource = bSource ;
        sda->Update(dbdataset) ;
        }
       catch(Exception^ex)
       {
        MessageBox::Show(ex->Message);}
       }
```

Using the attribute in the combobox1 students can be searched and when the remove button is clicked corresponding information of that student is removed from the database. When the update button is clicked then the details of that student can be updated.

```
private: System::Void admin1_faculty_remove_Load(System::Object^  sender,
System::EventArgs^  e)
```

```
{
   this->comboBox1->DropDownStyle=ComboBoxStyle::DropDownList;
  re_load();
}
```

When the modify faculty record linklabel in admin1_faculty form is clicked admin1_faculty_remove is loaded with drop down list having Name, Webmail, Position, Department and the reload function is called.

```
private: System::Void dataGridView1_CellContentClick(System::Object^  sender,
System::Windows::Forms::DataGridViewCellEventArgs^  e)
{
    try{
                if(e->ColumnIndex==0)
                {

                int fac_id=System::Convert::ToInt32(this->dataGridView1->Rows[e-
>RowIndex]->Cells[2]->Value);

                admin1_faculty_updateinfo^ update_inf=gcnew
admin1_faculty_updateinfo(fac_id);
                update_inf->ShowDialog();
                re_load();
                }
                else if(e->ColumnIndex==1)
                {
                int fac_id=System::Convert::ToInt32(this->dataGridView1->Rows[e-
>RowIndex]->Cells[2]->Value);
                data^ database=gcnew data();
                database->removeStudent(fac_id);
```

```
            re_load();
            }
        }
        catch(Exception^ ex)
        {
        MessageBox::Show("Empty Table or No Search Results");
        }
}
```

Searches according to the attribute present in the combobox and the entry in the text box and displays faculty records. If the update button is used , admin1_faculty_updateinfo form is loaded with the faculty details where the admin can change some details of the corresponding faculty and remove button is used to remove particular faculty from the database.

```
private: System::Void search_button_Click(System::Object^  sender,
System::EventArgs^  e)
{
        DataView^ dv=gcnew DataView(dbdataset);
        if(!String::IsNullOrEmpty(textBox1->Text) &&
!String::IsNullOrEmpty(comboBox1->Text))
       dv->RowFilter=String::Format("{0} like '%{1}%'",comboBox1->Text,textBox1-
>Text);
        else
        dv->RowFilter=String::Format("1=1");
        this->dataGridView1->DataSource=dv;
}
```

Using this button we can search the faculty details according to the attribute present in the combobox.

---

admin1_dept.h :

```
private: System::Void Courses_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

 admin1_dept_courses ^f=gcnew admin1_dept_courses();

 f->ShowDialog();

}
```

Loads admin1_dept_courses when couse details linklabel in admin1_dept is clicked.

```
private: System::Void adddept_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

 admin1_dept_adddept ^f=gcnew admin1_dept_adddept();

 f->ShowDialog();

}
```

Loads admin1_dept_adddept form when add department linklabel in admin1_dept is clicked.

```
private: System::Void upddept_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

admin1_dept_upddept ^f=gcnew admin1_dept_upddept();

f->ShowDialog();

}
```

Loads admin1_dept_upddept form when update department linklabel in admin1_dept is clicked.

---

admin1_dept_courses.h :

```
private: System::Void addcourse_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

 {

 admin1_dept_coursedetails_addcourse ^adcrse=gcnew
admin1_dept_coursedetails_addcourse();

 adcrse->ShowDialog();

 }
```

Loads admin1_dept_coursedetails_addcourse form when Add course linklabel in admin1_dept_courses is clicked.

```
private: System::Void remcourse_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

{

admin1_dept_courses_modify^ modify=gcnew admin1_dept_courses_modify();

modify->ShowDialog();

}
```

Loads admin1_dept_courses_modify form when modify course linklabel in admin1_dept_courses is clicked.

---

admin1_dept_coursedetails_addcourse.h :

```
private: System::Void Addcrse_button_Click(System::Object^  sender,
System::EventArgs^  e)

{
```

```
 data ^f=gcnew data();

 int bar = System::Convert::ToInt32(credit_textBox->Text);

if(f->addCourse(coursecode_textBox->Text,bar,coursename_textBox-
>Text,fac_id,dept_comboBox->Text))

MessageBox::Show("Course Added Successfully");

else

MessageBox::Show("Course Addition Failure");

}
```

Adds the course details to the course table in the database.

```
private: System::Void admin1_dept_coursedetails_addcourse_Load(System::Object^
sender, System::EventArgs^  e)

{

 data ^f=gcnew data();

this->dept_comboBox->DataSource=f->get_all_department();

}
```

When this form loads the departments in the department table are loaded in the combobox.

```
private: System::Void facname_comboBox_SelectedIndexChanged(System::Object^
sender, System::EventArgs^  e)

 {

  data ^f=gcnew data();

  fac_id=f->getFacultyIDbyName(facname_comboBox->Text);

 }
```

Gets the faculty id by faculty name ,the one which is selected in facname_comboBox.

---

admin1_dept_coursedetails_modifycourse.h :

```
private: System::Void re_load()

{

        String^ constring =
L"datasource=localhost;port=3306;username=root;password=root";

        MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);

        MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select * from
database.course;",conDataBase);

        try

         {

                MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();

                sda->SelectCommand = cmdDataBase ;

                DataTable^ dbdataset = gcnew DataTable();

                sda->Fill(dbdataset);

                BindingSource^ bSource = gcnew BindingSource();

                dbdataset->DefaultView->AllowDelete=false;

                dbdataset->DefaultView->AllowEdit=false;

                bSource->DataSource = dbdataset->DefaultView ;

                this->dataGridView1->DataSource = bSource ;

                sda->Update(dbdataset) ;

         }

        catch(Exception^ex)

         {

          MessageBox::Show(ex->Message);}

         }

}
```

When the update button is used admin1_dept_courses_update form is loaded with the corresponding contents where we can change them

accordingly and the contents changed are overwritten in course table in database and remove vbutton is used to remove that course.

```
private: System::Void dataGridView1_CellContentClick(System::Object^  sender,
System::Windows::Forms::DataGridViewCellEventArgs^  e)

{

        try

         {

           if(e->ColumnIndex==0)

                {

                String^ courseCode=System::Convert::ToString(this->dataGridView1-
>Rows[e->RowIndex]->Cells[2]->Value);

                admin1_dept_courses_update^ up=gcnew
admin1_dept_courses_update(courseCode);

                up->ShowDialog();

                re_load();

                }

                else if(e->ColumnIndex==1)

                {

                data^ f=gcnew data();

                f->removeCourse(System::Convert::ToString(this->dataGridView1-
>Rows[e->RowIndex]->Cells[2]->Value));

                re_load();

                }

        }

        catch(Exception^)

        {

        MessageBox::Show("Empty Table or Row");

        }

}
```

When update button is used new form admin1_dept_courses_update is loaded with that course details where the admin can change some of the details of that course .And the remove button is used to remove that course.

---

admin1_dept_adddept.h :

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e)
 {
 data^ database=gcnew
 data();
 if(String::IsNullOrWhiteSpace(textBox1->Text) ||
String::IsNullOrWhiteSpace(textBox1->Text) || String::IsNullOrEmpty(textBox1-
>Text) || String::IsNullOrWhiteSpace(textBox2->Text) ||
String::IsNullOrEmpty(textBox2->Text) )
 {
MessageBox::Show("Invalid Strings");
 }
 else
 {
 if(database->addDepartment(textBox1->Text,textBox2->Text))
MessageBox::Show("Department Added Successfully");
 else
 MessageBox::Show("Some error in addition");
 }
}
```

Department and information about it is added in this form.

---

admin1_dept_upddept.h :

```
private: System::Void admin1_dept_upddept_Load(System::Object^  sender,
System::EventArgs^  e)

{

 this->comboBox1->DropDownStyle=ComboBoxStyle::DropDownList;

 data^ database=gcnew data();

 this->comboBox1->DataSource=database->get_all_department();

}
```

Loads the combobobox1 with the departments in the department table in the database.

```
private: System::Void comboBox1_SelectedIndexChanged(System::Object^  sender,
System::EventArgs^  e)

 {

 data^ database=gcnew data();

 textBox1->Text=(database->department_details(comboBox1->Text))[1];

 }
```

Loads the textbox1 with the department description in the department table in the database.

---

admin1_coursereg.h :

```
private: System::Void addstu_linkLabel_LinkClicked(System::Object^  sender,
System::Windows::Forms::LinkLabelLinkClickedEventArgs^  e)

 {

 admin1_coursereg_addstudent ^f=gcnew admin1_coursereg_addstudent();
```

```
 f->ShowDialog();
 }
```

When the modify course records is clicked loads new form admin1_coursereg_addstudent.

---

admin1_coursereg_addstudent.h :

```
private: System::Void re_load()
 {
 String^ constring =
L"datasource=localhost;port=3306;username=root;password=root";
 MySqlConnection^ conDataBase= gcnew MySqlConnection(constring);
 MySqlCommand^ cmdDataBase =gcnew MySqlCommand("select * from
database.student;",conDataBase);
 try
 {
 MySqlDataAdapter ^ sda = gcnew MySqlDataAdapter();
 sda->SelectCommand = cmdDataBase ;
dbdataset = gcnew DataTable();
 sda->Fill(dbdataset);
 BindingSource^ bSource = gcnew BindingSource();
dbdataset->DefaultView->AllowDelete=false;
dbdataset->DefaultView->AllowEdit=false;
bSource->DataSource = dbdataset->DefaultView ;
this->dataGridView1->DataSource = bSource ;
sda->Update(dbdataset) ;
}
```

```
catch(Exception^ex)

{

MessageBox::Show(ex->Message);

}

}
```

This function reloads te form with the contents which are received according to the text in combobox1.

```
private: System::Void admin1_coursereg_addstudent_Load(System::Object^
sender, System::EventArgs^  e)

{

re_load();

this->comboBox1->DropDownStyle=ComboBoxStyle::DropDownList;

}
```

Loads the form with dept_id and roll_noattributes in combobox1.

```
private: System::Void dataGridView1_CellContentClick(System::Object^  sender,
System::Windows::Forms::DataGridViewCellEventArgs^  e)

{

if(e->ColumnIndex==0 && !String::IsNullOrEmpty(System::Convert::ToString(this-
>dataGridView1->Rows[e->RowIndex]->Cells[4]->Value)))

{

admin1_coursereg_addstudent_add^ add_st=gcnew
admin1_coursereg_addstudent_add(System::Convert::ToInt32(this->dataGridView1-
>Rows[e->RowIndex]->Cells[2]->Value));

add_st->ShowDialog();

re_load();

}

else if((e->ColumnIndex==1 &&
!String::IsNullOrEmpty(System::Convert::ToString(this->dataGridView1->Rows[e-
>RowIndex]->Cells[3]->Value))))
```

```
{
admin1_coursereg_addstudent_changegrade^ grad=gcnew
admin1_coursereg_addstudent_changegrade(System::Convert::ToInt32(this-
>dataGridView1->Rows[e->RowIndex]->Cells[2]->Value));
grad->ShowDialog();
re_load();
}
}
```

When the add course button is used then new form admin1_coursereg_addstudent_add is loaded with a combobox which has coursecode from the course table and semster is added and modify grade button is used to load a new form admin1_coursereg_addstudent_changegrade which has a combobox with course codes and textbox with the corresponding student's roll_no and grade is selected from the the combobox.

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^
e)
{
 DataView^ dv=gcnew DataView(dbdataset);
 if(this->comboBox1->Text=="Roll_no" || this->comboBox1->Text=="dept_id")
 {
int num;
if(Int32::TryParse(textBox1->Text,num))
{
dv->RowFilter=String::Format("{0}={1}",comboBox1->Text,num);
}
else
{
if(!String::IsNullOrEmpty(textBox1->Text))
MessageBox::Show("Invalid Search");
```

```
dv->RowFilter=String::Format("1=1");

}

this->dataGridView1->DataSource=dv;

}

}
```

Reloads the form with details according to the combobox1 text and the text in textbox1.
