**GROUP MEMBERS:**

**AIMEN NADEEM 2022-BSE-001**

**ZUFRA JAHAN 2022-BSE-37**

**AYESHA FARRUKH 2022-BSE-38**

**Data Analysis And Algorithm**

**Database connectivity**

**DATABASE: COLLEGE MANAGEMENT SYSTEM**

using MySql.Data.MySqlClient;

using System.Drawing;

namespace NEW\_DATABASE

{

public partial class Form1 : Form

{

private MySqlConnection con;

public Form1()

{

InitializeComponent();

string connstring = "server=localhost;uid=root;pwd=zuaa3306#;database=collegemanagementsystem";

con = new MySqlConnection(connstring);

try

{

con.Open();

}

catch (MySqlException ex)

{

MessageBox.Show("Connection failed: " + ex.Message);

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

string connstring = "server=localhost;uid=root;pwd=zuaa3306#;database=collegemanagementsystem";

MySqlConnection con = new MySqlConnection();

con.ConnectionString = connstring;

con.Open();

MessageBox.Show("Connected to MySQL database successfully!");

}

catch (MySqlException ex)

{

MessageBox.Show("Connection failed: " + ex.Message);

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

string name = textBox1.Text;

string Id = textBox2.Text;

string contact = textBox3.Text;

string role = textBox4.Text;

string email = textBox5.Text;

string sql = "INSERT INTO administrator (ID, Contact, Role, Email, Name) " +

"VALUES (@Id, @contact, @role, @email, @name)";

using (MySqlCommand cmd = new MySqlCommand(sql, con))

{

cmd.Parameters.AddWithValue("@Id", Id);

cmd.Parameters.AddWithValue("@Contact", contact);

cmd.Parameters.AddWithValue("@Role", role);

cmd.Parameters.AddWithValue("@Email", email);

cmd.Parameters.AddWithValue("@Name", name);

cmd.ExecuteNonQuery();

MessageBox.Show("Record inserted successfully!");

}

}

catch (MySqlException ex)

{

MessageBox.Show(ex.ToString());

}

}

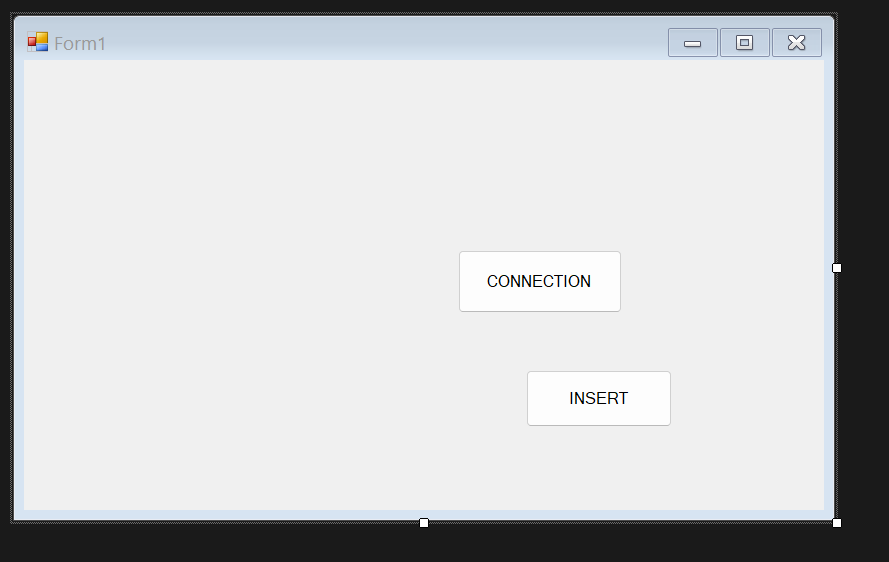
private void label2\_Click(object sender, EventArgs e)

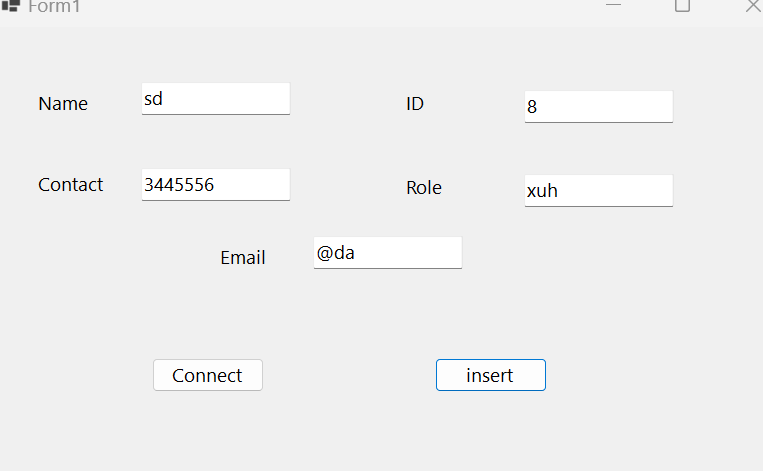
{

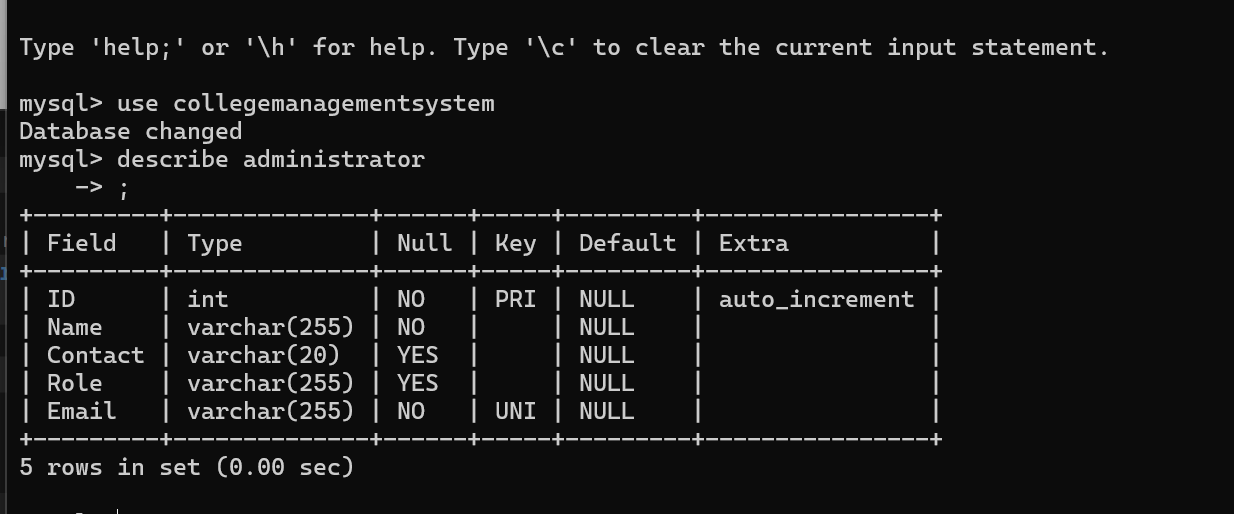
}

}

}







**College management system code**

#include <iostream>

#include <chrono>

#include <string>

using namespace std;

using namespace std::chrono;

const int MAX\_ENTRIES = 100;

class Student {

public:

int ID;

string Name;

string Address;

string Email;

string Contact;

Student() : ID(-1) {}

};

class Faculty {

public:

int ID;

string Name;

string Email;

string Address;

string Qualification;

string Contact;

Faculty() : ID(-1) {}

};

class Course {

public:

int ID;

string Title;

string Description;

int Credit\_Hours;

Course() : ID(-1) {}

};

class Administrator {

public:

int ID;

string Name;

string Contact;

string Role;

string Email;

Administrator() : ID(-1) {}

};

class Admission {

public:

int ID;

int Student\_ID;

string Application\_Submission\_Date;

Admission() : ID(-1) {}

};

class Fee {

public:

int ID;

double Amount;

string Payment\_Date;

int Admission\_ID;

Fee() : ID(-1) {}

};

class Assignment {

public:

int ID;

int Course\_ID;

string Description;

string Deadline;

Assignment() : ID(-1) {}

};

class Student\_Assignment {

public:

int Student\_ID;

int Assignment\_ID;

string Grades;

Student\_Assignment() : Student\_ID(-1), Assignment\_ID(-1) {}

};

class Report {

public:

int ID;

int Student\_ID;

int Course\_ID;

string Grades;

Report() : ID(-1) {}

};

class Scholarship {

public:

int ID;

string Name;

double Award\_Amount;

Scholarship() : ID(-1) {}

};

class Payment {

public:

int ID;

int Admission\_ID;

double Amount;

string Payment\_Date;

Payment() : ID(-1) {}

};

class CollegeManagementSystem {

private:

Student students[MAX\_ENTRIES];

Faculty faculties[MAX\_ENTRIES];

Course courses[MAX\_ENTRIES];

Administrator administrators[MAX\_ENTRIES];

Admission admissions[MAX\_ENTRIES];

Fee fees[MAX\_ENTRIES];

Assignment assignments[MAX\_ENTRIES];

Student\_Assignment student\_assignments[MAX\_ENTRIES];

Report reports[MAX\_ENTRIES];

Scholarship scholarships[MAX\_ENTRIES];

Payment payments[MAX\_ENTRIES];

int studentCount;

int facultyCount;

int courseCount;

int administratorCount;

int admissionCount;

int feeCount;

int assignmentCount;

int studentAssignmentCount;

int reportCount;

int scholarshipCount;

int paymentCount;

public:

CollegeManagementSystem() : studentCount(0), facultyCount(0), courseCount(0), administratorCount(0),

admissionCount(0), feeCount(0), assignmentCount(0), studentAssignmentCount(0), reportCount(0),

scholarshipCount(0), paymentCount(0) {}

void addStudent() {

if (studentCount < MAX\_ENTRIES) {

Student student;

student.ID = studentCount + 1;

cout << "Enter Student Name: ";

cin.ignore();

getline(cin, student.Name);

cout << "Enter Student Address: ";

getline(cin, student.Address);

cout << "Enter Student Email: ";

getline(cin, student.Email);

cout << "Enter Student Contact: ";

getline(cin, student.Contact);

students[studentCount++] = student;

}

}

void addFaculty() {

if (facultyCount < MAX\_ENTRIES) {

Faculty faculty;

faculty.ID = facultyCount + 1;

cout << "Enter Faculty Name: ";

cin.ignore();

getline(cin, faculty.Name);

cout << "Enter Faculty Email: ";

getline(cin, faculty.Email);

cout << "Enter Faculty Address: ";

getline(cin, faculty.Address);

cout << "Enter Faculty Qualification: ";

getline(cin, faculty.Qualification);

cout << "Enter Faculty Contact: ";

getline(cin, faculty.Contact);

faculties[facultyCount++] = faculty;

}

}

void addCourse() {

if (courseCount < MAX\_ENTRIES) {

Course course;

course.ID = courseCount + 1;

cout << "Enter Course Title: ";

cin.ignore();

getline(cin, course.Title);

cout << "Enter Course Description: ";

getline(cin, course.Description);

cout << "Enter Credit Hours: ";

cin >> course.Credit\_Hours;

courses[courseCount++] = course;

}

}

void addAdministrator() {

if (administratorCount < MAX\_ENTRIES) {

Administrator admin;

admin.ID = administratorCount + 1;

cout << "Enter Administrator Name: ";

cin.ignore();

getline(cin, admin.Name);

cout << "Enter Administrator Contact: ";

getline(cin, admin.Contact);

cout << "Enter Administrator Role: ";

getline(cin, admin.Role);

cout << "Enter Administrator Email: ";

getline(cin, admin.Email);

administrators[administratorCount++] = admin;

}

}

void addAdmission() {

if (admissionCount < MAX\_ENTRIES) {

Admission admission;

admission.ID = admissionCount + 1;

cout << "Enter Student ID for Admission: ";

cin >> admission.Student\_ID;

cout << "Enter Application Submission Date (YYYY-MM-DD): ";

cin.ignore();

getline(cin, admission.Application\_Submission\_Date);

admissions[admissionCount++] = admission;

}

}

void addFee() {

if (feeCount < MAX\_ENTRIES) {

Fee fee;

fee.ID = feeCount + 1;

cout << "Enter Amount: ";

cin >> fee.Amount;

cout << "Enter Payment Date (YYYY-MM-DD): ";

cin.ignore();

getline(cin, fee.Payment\_Date);

cout << "Enter Admission ID: ";

cin >> fee.Admission\_ID;

fees[feeCount++] = fee;

}

}

void addAssignment() {

if (assignmentCount < MAX\_ENTRIES) {

Assignment assignment;

assignment.ID = assignmentCount + 1;

cout << "Enter Course ID: ";

cin >> assignment.Course\_ID;

cout << "Enter Assignment Description: ";

cin.ignore();

getline(cin, assignment.Description);

cout << "Enter Assignment Deadline (YYYY-MM-DD): ";

getline(cin, assignment.Deadline);

assignments[assignmentCount++] = assignment;

}

}

void addStudentAssignment() {

if (studentAssignmentCount < MAX\_ENTRIES) {

Student\_Assignment studentAssignment;

cout << "Enter Student ID: ";

cin >> studentAssignment.Student\_ID;

cout << "Enter Assignment ID: ";

cin >> studentAssignment.Assignment\_ID;

cout << "Enter Grades: ";

cin.ignore();

getline(cin, studentAssignment.Grades);

student\_assignments[studentAssignmentCount++] = studentAssignment;

}

}

void addReport() {

if (reportCount < MAX\_ENTRIES) {

Report report;

report.ID = reportCount + 1;

cout << "Enter Student ID: ";

cin >> report.Student\_ID;

cout << "Enter Course ID: ";

cin >> report.Course\_ID;

cout << "Enter Grades: ";

cin.ignore();

getline(cin, report.Grades);

reports[reportCount++] = report;

}

}

void addScholarship() {

if (scholarshipCount < MAX\_ENTRIES) {

Scholarship scholarship;

scholarship.ID = scholarshipCount + 1;

cout << "Enter Scholarship Name: ";

cin.ignore();

getline(cin, scholarship.Name);

cout << "Enter Award Amount: ";

cin >> scholarship.Award\_Amount;

scholarships[scholarshipCount++] = scholarship;

}

}

void addPayment() {

if (paymentCount < MAX\_ENTRIES) {

Payment payment;

payment.ID = paymentCount + 1;

cout << "Enter Admission ID: ";

cin >> payment.Admission\_ID;

cout << "Enter Amount: ";

cin >> payment.Amount;

cout << "Enter Payment Date (YYYY-MM-DD): ";

cin.ignore();

getline(cin, payment.Payment\_Date);

payments[paymentCount++] = payment;

}

}

void displayStudents() {

cout << "Students:" << endl;

for (int i = 0; i < studentCount; ++i) {

cout << "ID: " << students[i].ID << ", Name: " << students[i].Name << ", Email: " << students[i].Email << endl;

}

}

void displayFaculties() {

cout << "Faculties:" << endl;

for (int i = 0; i < facultyCount; ++i) {

cout << "ID: " << faculties[i].ID << ", Name: " << faculties[i].Name << ", Email: " << faculties[i].Email << endl;

}

}

void displayCourses() {

cout << "Courses:" << endl;

for (int i = 0; i < courseCount; ++i) {

cout << "ID: " << courses[i].ID << ", Title: " << courses[i].Title << endl;

}

}

void displayAdministrators() {

cout << "Administrators:" << endl;

for (int i = 0; i < administratorCount; ++i) {

cout << "ID: " << administrators[i].ID << ", Name: " << administrators[i].Name << ", Role: " << administrators[i].Role << endl;

}

}

void displayAdmissions() {

cout << "Admissions:" << endl;

for (int i = 0; i < admissionCount; ++i) {

cout << "ID: " << admissions[i].ID << ", Student ID: " << admissions[i].Student\_ID << ", Submission Date: " << admissions[i].Application\_Submission\_Date << endl;

}

}

void displayFees() {

cout << "Fees:" << endl;

for (int i = 0; i < feeCount; ++i) {

cout << "ID: " << fees[i].ID << ", Amount: " << fees[i].Amount << ", Payment Date: " << fees[i].Payment\_Date << endl;

}

}

void displayAssignments() {

cout << "Assignments:" << endl;

for (int i = 0; i < assignmentCount; ++i) {

cout << "ID: " << assignments[i].ID << ", Course ID: " << assignments[i].Course\_ID << ", Description: " << assignments[i].Description << ", Deadline: " << assignments[i].Deadline << endl;

}

}

void displayStudentAssignments() {

cout << "Student Assignments:" << endl;

for (int i = 0; i < studentAssignmentCount; ++i) {

cout << "Student ID: " << student\_assignments[i].Student\_ID << ", Assignment ID: " << student\_assignments[i].Assignment\_ID << ", Grades: " << student\_assignments[i].Grades << endl;

}

}

void displayReports() {

cout << "Reports:" << endl;

for (int i = 0; i < reportCount; ++i) {

cout << "ID: " << reports[i].ID << ", Student ID: " << reports[i].Student\_ID << ", Course ID: " << reports[i].Course\_ID << ", Grades: " << reports[i].Grades << endl;

}

}

void displayScholarships() {

cout << "Scholarships:" << endl;

for (int i = 0; i < scholarshipCount; ++i) {

cout << "ID: " << scholarships[i].ID << ", Name: " << scholarships[i].Name << ", Award Amount: " << scholarships[i].Award\_Amount << endl;

}

}

void displayPayments() {

cout << "Payments:" << endl;

for (int i = 0; i < paymentCount; ++i) {

cout << "ID: " << payments[i].ID << ", Admission ID: " << payments[i].Admission\_ID << ", Amount: " << payments[i].Amount << ", Payment Date: " << payments[i].Payment\_Date << endl;

}

}

};

int main() {

auto start = high\_resolution\_clock::now();

CollegeManagementSystem cms;

int choice;

do {

cout << "1. Add Student" << endl;

cout << "2. Add Faculty" << endl;

cout << "3. Add Course" << endl;

cout << "4. Add Administrator" << endl;

cout << "5. Add Admission" << endl;

cout << "6. Add Fee" << endl;

cout << "7. Add Assignment" << endl;

cout << "8. Add Student Assignment" << endl;

cout << "9. Add Report" << endl;

cout << "10. Add Scholarship" << endl;

cout << "11. Add Payment" << endl;

cout << "12. Display Students" << endl;

cout << "13. Display Faculties" << endl;

cout << "14. Display Courses" << endl;

cout << "15. Display Administrators" << endl;

cout << "16. Display Admissions" << endl;

cout << "17. Display Fees" << endl;

cout << "18. Display Assignments" << endl;

cout << "19. Display Student Assignments" << endl;

cout << "20. Display Reports" << endl;

cout << "21. Display Scholarships" << endl;

cout << "22. Display Payments" << endl;

cout << "23. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cms.addStudent();

break;

case 2:

cms.addFaculty();

break;

case 3:

cms.addCourse();

break;

case 4:

cms.addAdministrator();

break;

case 5:

cms.addAdmission();

break;

case 6:

cms.addFee();

break;

case 7:

cms.addAssignment();

break;

case 8:

cms.addStudentAssignment();

break;

case 9:

cms.addReport();

break;

case 10:

cms.addScholarship();

break;

case 11:

cms.addPayment();

break;

case 12:

cms.displayStudents();

break;

case 13:

cms.displayFaculties();

break;

case 14:

cms.displayCourses();

break;

case 15:

cms.displayAdministrators();

break;

case 16:

cms.displayAdmissions();

break;

case 17:

cms.displayFees();

break;

case 18:

cms.displayAssignments();

break;

case 19:

cms.displayStudentAssignments();

break;

case 20:

cms.displayReports();

break;

case 21:

cms.displayScholarships();

break;

case 22:

cms.displayPayments();

break;

case 23:

cout << "Exiting..." << endl;

break;

default:

cout << "Invalid choice. Please try again." << endl;

}

} while (choice != 23);

auto end = high\_resolution\_clock::now();

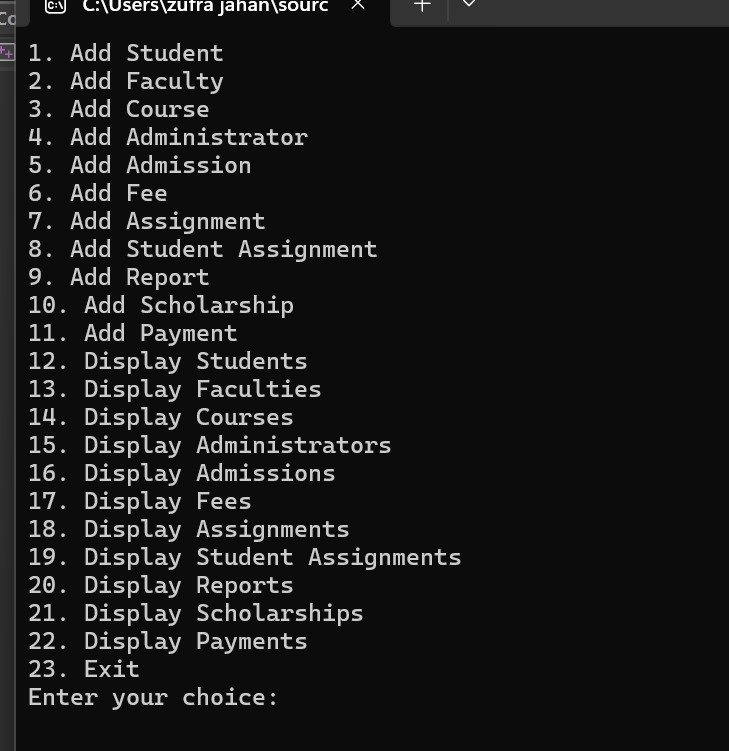
auto duration = duration\_cast<seconds>(end - start);

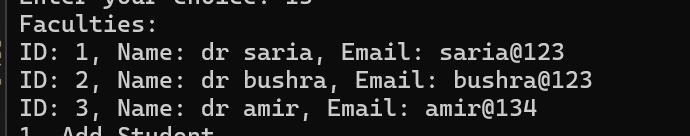
cout << "\n\nTime Complexity: " << duration.count() << " seconds\n";

return 0;

}

**Output**

****

****

****