BAHRIA UNIVERSITY ISLAMABAD

DEPARTMENT OF COMPUTER SCIENCE

Object Oriented Programming

Semester Project.

COURSE MANAGEMENT SYSTEM

Group Members:

SYED ABDUL HASEEB
(01-135251-029)

ARSIM ADIL
(01-135251-007)

MUHAMMAD ABDUL AZIZ
(01-135251-050)

ZARAR KHAN DURRANI
(01-135251-024)

SUBMITTED TO
SIR USMAN SHAFIQUE

15 December, 2025.

COURSE MANAGEMENT SYSTEM:

- ## INTRODUCTION:

  The Course Management System is a CLI based system which is used to manage the academic activities of a university. The system manages students, courses, instructors, and grades. The system allows students to enroll in courses, instructors to manage courses, and administrators to track academic records efficiently.

- ## OBJECTIVES:
  - To understand **object-oriented programming concepts**
  - To implement **classes and inheritance**
  - To implement concept of **Encapsulation, Polymorphism, static members, virtual functions, composition etc.**
  - To create a **menu-driven program**
  - To practice **real-world data management**

- ## SCOPE OF PROJECT:

  The system can manage a limited number of students, instructors, and courses.
  It is suitable for small-scale academic use. Data is stored temporarily during program execution and is not saved permanently. In Future, we can implement file handling to efficiently store data in a document file.

- ## Tools and Technologies Used:
  - Programming Language: **C++**
  - Compiler: **Visual Studio**
  - Platform: **Windows**
  - Concepts Used:
    - Classes and Objects
    - Inheritance
    - Arrays
    - Functions
    - Menu-based programming

- **PROJECT DETAILS:**

Classes Used:

1) **Person:**
   - Base Class.
   - Attributes: name, age.
   - Member function: display()

2) **Student:**
   - Derived class.
   - Inherits from Person
   - Member functions: enroll() , display() override

3) **Instructor:**
   - Derived class.
   - Inherits from Person
   - Member functions: assignCourse(), display() override

4) **Course:**
   - Class.
   - Member functions: addStudent(), display() override

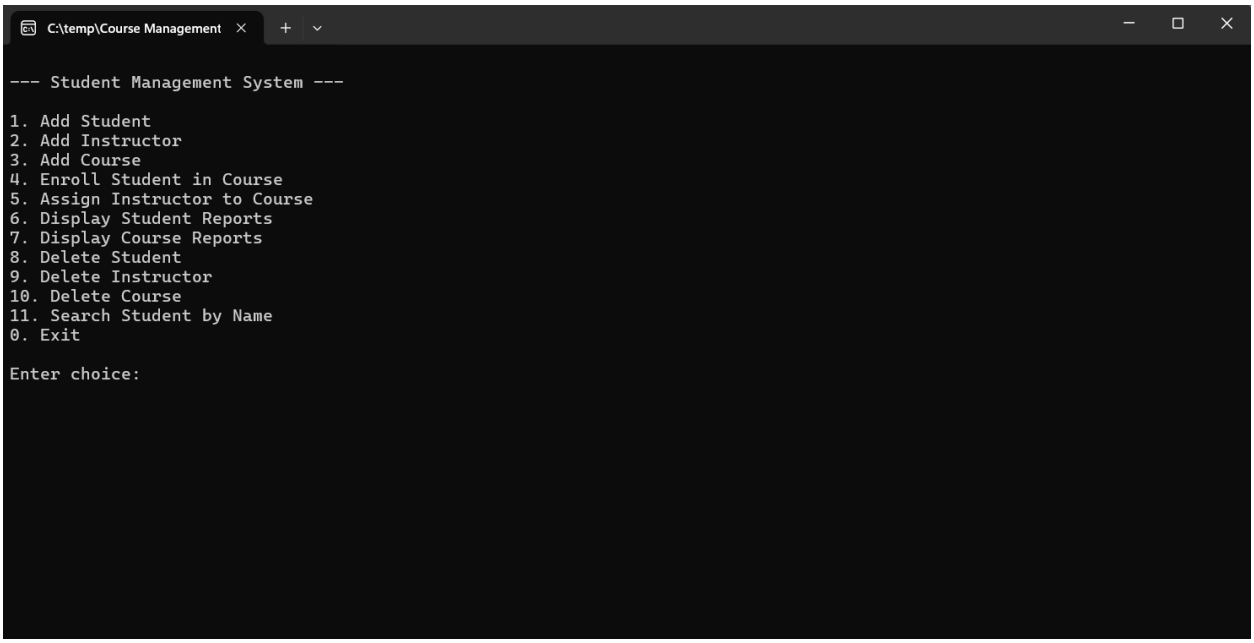
Main Menu:

- The main menu is a choice-based menu for user interface.
- It displays options to perform Add, Update, Delete and View information for Student, Courses and Instructors.

Procedure:

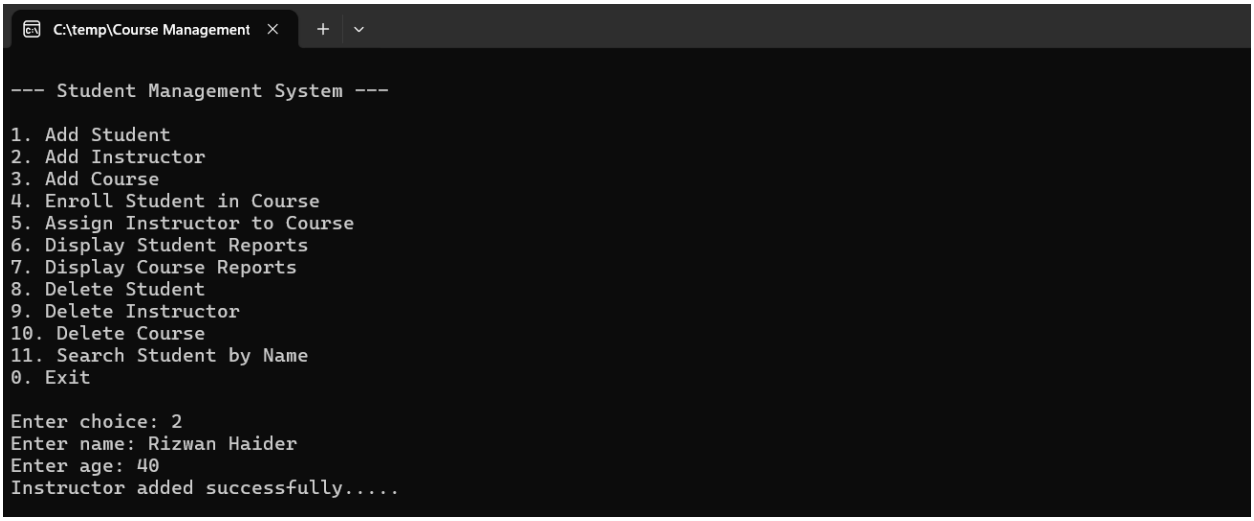
- Program starts and displays the main menu.
- The user selects based on choice of operation.
- User performs desired operation
- Program repeats until or unless the user presses exit.

SAMPLE OUTPUTS:



Main Menu

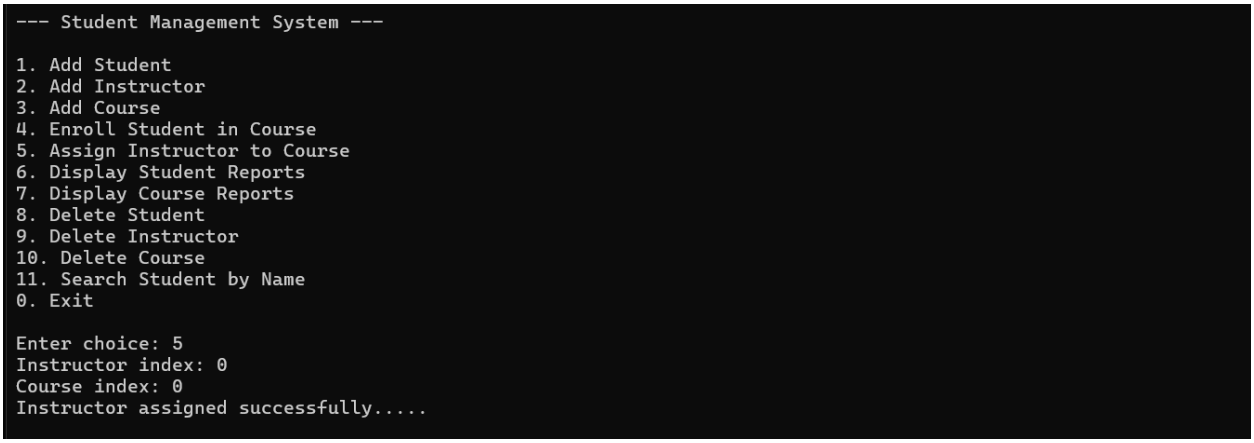
Add Student



Add Instructor

Add Course

Enroll Student in Course.

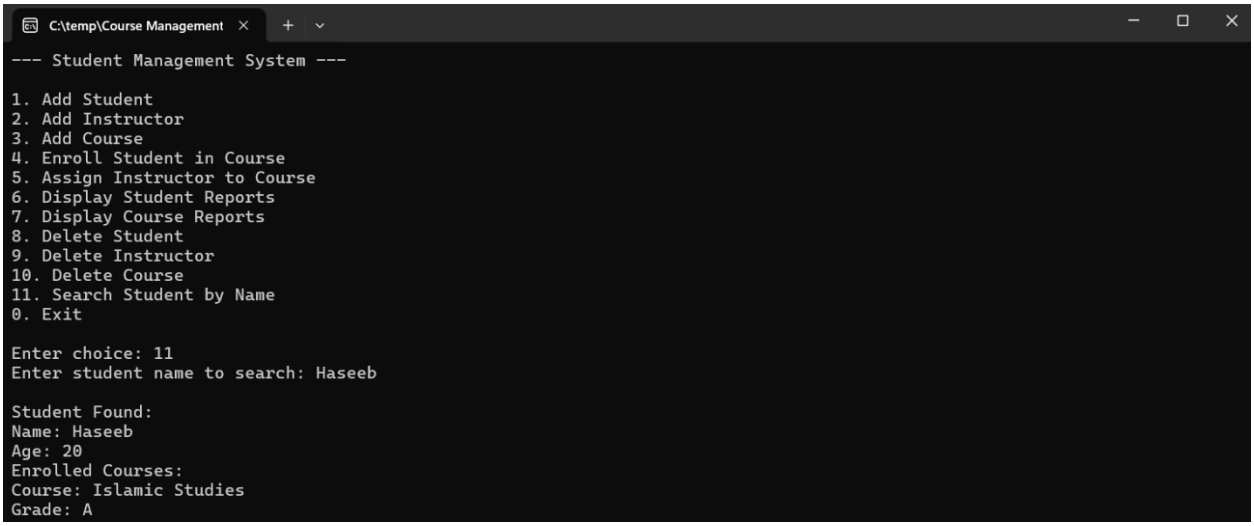


Assign Instructor to Course.

Display Student Reports.

```
--- Student Management System ---

1. Add Student
2. Add Instructor
3. Add Course
4. Enroll Student in Course
5. Assign Instructor to Course
6. Display Student Reports
7. Display Course Reports
8. Delete Student
9. Delete Instructor
10. Delete Course
11. Search Student by Name
0. Exit

Enter choice: 7

--- Course Reports ---

Course 1
Course Name: Islamic Studies
Course Code: 123
Instructor: Rizwan Haider
Enrolled Students: Haseeb ,
```

Course Reports.



Search Student

CODE:

VIEW ON GITHUB.

```
#include <iostream>
#include <string>
using namespace std;
const int MAX_STUDENTS = 10;
const int MAX_INSTRUCTORS = 4;
const int MAX_COURSES = 4;
// Base Class Person--------
class Person {
protected:
    string name;
    int age;

public:
    Person(string n = "", int a = 0)
    {
        name = n;
        age = a;
    }

    virtual void display()
    {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
    }

    string getName()
    {
        return name;
    }

    virtual ~Person() {}
};

// Course Class-----------
class Course
{
public:
    string name;
    int code;
    string instructorName;
    Person* students[MAX_STUDENTS];
    int studentCount;

    Course(string n = "", int c = 0)
    {
        name = n;
        code = c;
        instructorName = "";
        studentCount = 0;
    }

    void addStudent(Person* s)
    {
        if (studentCount < MAX_STUDENTS)
```

```
        {
            students[studentCount] = s;
            studentCount++;
        }
    }

    void display()
    {
        cout << "Course Name: " << name << endl;
        cout << "Course Code: " << code << endl;

        if (instructorName == "")
        {
            cout << "Instructor: None" << endl;
        }
        else
        {
            cout << "Instructor: " << instructorName << endl;
        }

        cout << "Enrolled Students: ";
        if (studentCount == 0)
        {
            cout << "None" << endl;
        }
        else
        {
            for (int i = 0; i < studentCount; i++)
            {
                cout << students[i]->getName() << " , ";
            }
            cout << endl;
        }
    }
};

// Student Class----------
class Student : public Person
{
public:
    int rollNumber;
    Course* enrolled[MAX_COURSES];
    char grades[MAX_COURSES];
    int courseCount;

    Student(string n = "", int a = 0, int r = 0)
    {
        name = n;
        age = a;
        rollNumber = r;
        courseCount = 0;
    }

    int getRollNumber()
    {
        return rollNumber;
    }
```

```
    void enroll(Course* c, char grade)
    {
        if (courseCount < MAX_COURSES)
        {
            enrolled[courseCount] = c;
            grades[courseCount] = grade;
            courseCount++;
            c->addStudent(this);
        }
    }

    void display() override
    {
        cout << "Roll Number: " << rollNumber << endl;
        Person::display();
        cout << "Enrolled Courses:" << endl;

        if (courseCount == 0)
        {
            cout << "None" << endl;
        }
        else
        {
            for (int i = 0; i < courseCount; i++)
            {
                cout << "Course: " << enrolled[i]->name << endl;
                cout << "Grade: " << grades[i] << endl;

                if (grades[i] > 'B')
                {
                    cout << "Status: Will have to improve their grade." << endl;
                }
            }
        }
    }
};

// Instructor Class------------
class Instructor : public Person
{
public:
    Course* teaching[MAX_COURSES];
    int teachingCount;

    Instructor(string n = "", int a = 0)
    {
        name = n;
        age = a;
        teachingCount = 0;
    }

    void assignCourse(Course* c)
    {
        if (teachingCount < MAX_COURSES)
        {
            teaching[teachingCount] = c;
            teachingCount++;
            c->instructorName = name;
```

```
        }
    }

    void display() override
    {
        Person::display();
        cout << "Teaching Courses:" << endl;

        if (teachingCount == 0)
        {
            cout << "None" << endl;
        }
        else
        {
            for (int i = 0; i < teachingCount; i++)
            {
                cout << teaching[i]->name << endl;
            }
        }
    }
};

// Main Function-----------
int main()
{
    Student students[MAX_STUDENTS];
    Instructor instructors[MAX_INSTRUCTORS];
    Course courses[MAX_COURSES];

    int studentCount = 0;
    int instructorCount = 0;
    int courseCount = 0;

    int choice;

    do
    {
        cout << "\n--- Student Management System ---" << endl << endl;
        cout << "1. Add Student" << endl;
        cout << "2. Add Instructor" << endl;
        cout << "3. Add Course" << endl;
        cout << "4. Enroll Student in Course" << endl;
        cout << "5. Assign Instructor to Course" << endl;
        cout << "6. Display Student Reports" << endl;
        cout << "7. Display Course Reports" << endl;
        cout << "8. Delete Student" << endl;
        cout << "9. Delete Instructor" << endl;
        cout << "10. Delete Course" << endl;
        cout << "11. Search Student by Roll Number" << endl;
        cout << "0. Exit" << endl << endl;
        cout << "Enter choice: ";
        cin >> choice;

        // Add a Student-----
        if (choice == 1)
        {
            string n;
            int a, r;
```

```
        cout << "Enter name: ";
        cin.ignore();
        getline(cin, n);
        cout << "Enter age: ";
        cin >> a;
        cout << "Enter roll number: ";
        cin >> r;

        students[studentCount] = Student(n, a, r);
        studentCount++;
        cout << "Student Added Successfully...." << endl;
    }

    // Add An Instructor----
    else if (choice == 2)
    {
        string n;
        int a;
        cout << "Enter name: ";
        cin.ignore();
        getline(cin, n);
        cout << "Enter age: ";
        cin >> a;

        instructors[instructorCount] = Instructor(n, a);
        instructorCount++;
        cout << "Instructor Added Successfully...." << endl;
    }

    // Add Course-----
    else if (choice == 3)
    {
        string n;
        int c;
        cout << "Enter course name: ";
        cin.ignore();
        getline(cin, n);
        cout << "Enter course code: ";
        cin >> c;

        courses[courseCount] = Course(n, c);
        courseCount++;
        cout << "Course Added Successfully...." << endl;
    }

    // Enroll Student---
    else if (choice == 4)
    {
        int s, c;
        char g;
        cout << "Student index: ";
        cin >> s;
        cout << "Course index: ";
        cin >> c;
        cout << "Grade: ";
        cin >> g;

        students[s].enroll(&courses[c], g);
```

```
        cout << "Student enrolled into Course successfully.... " << endl;
    }

    // Assign an instructor------
    else if (choice == 5) {
        int i, c;
        cout << "Instructor index: ";
        cin >> i;
        cout << "Course index: ";
        cin >> c;

        instructors[i].assignCourse(&courses[c]);
        cout << "Instructor assigned to Course successfully...." << endl;
    }

    // Display student list----
    else if (choice == 6)
    {
        cout << "Displaying Student Lists...." << endl;
        for (int i = 0; i < studentCount; i++)
        {
            cout << "\nStudent " << i + 1 << endl;
            students[i].display();
        }
    }

    // Display all course details---
    else if (choice == 7)
    {
        cout << "Displaying Course Details...." << endl;
        for (int i = 0; i < courseCount; i++)
        {
            cout << "\nCourse " << i + 1 << endl;
            courses[i].display();
        }
    }

    // Delete a student---
    else if (choice == 8)
    {
        int index;
        cout << "Enter student index: ";
        cin >> index;

        for (int i = index; i < studentCount - 1; i++)
        {
            students[i] = students[i + 1];
        }
        studentCount--;
        cout << "Student Deleted Successfully...." << endl;
    }

    // Delete instructor----
    else if (choice == 9)
    {
        int index;
        cout << "Enter instructor index: ";
        cin >> index;
```

```
            for (int i = index; i < instructorCount - 1; i++) {
                instructors[i] = instructors[i + 1];
            }
            instructorCount--;
            cout << "Instructor Deleted Successfully...." << endl;
        }

        // Delete course----
        else if (choice == 10)
        {
            int index;
            cout << "Enter course index: ";
            cin >> index;

            for (int i = index; i < courseCount - 1; i++)
            {
                courses[i] = courses[i + 1];
            }
            courseCount--;
            cout << "Course Deleted Successfully...." << endl;
        }

        // Search Student by Roll Number----
        else if (choice == 11)
        {
            int roll;
            cout << "Enter roll number: ";
            cin >> roll;

            bool found = false;
            for (int i = 0; i < studentCount; i++) {
                if (students[i].getRollNumber() == roll) {
                    students[i].display();
                    found = true;
                    break;
                }
            }

            if (!found)
                cout << "Student not found." << endl;
        }

    } while (choice != 0);

    cout << "Program has been ended." << endl;
    return 0;
}
```