****

**Kingdom of Saudi Arabia**

**Ministry of Education**

**King Faisal University**

**College of Computer Sciences & Information Technology**

**Registration System**

**by**

Abdulrahman essa Alkhalifa

Thamer Ismail Alkhattam

Omar Alhamed

yazeed almuslem

**Supervised by**

Dr. Hamad Naeem

**December, 2024**

**Table of Contents**

**1** **Introduction ……………………………………………………………….…… 3**

**2 Methodology ………………………………………………………………….….4**

**3 Results & Discussions ..…………………………………………………...…….9**

**4 Sample Inputs & Output ……………………………………………….………9**

**5** **Conclusion ..……………………………………………………………………13**

**Introduction**

**The Student Registration System is a comprehensive management tool designed to facilitate the enrollment and management of students and courses within an educational institution. In an era where efficient data handling and organization are crucial, this system offers a user-friendly interface for administrative tasks related to student registration and course management.**

**The primary objectives of the Student Registration**

**System are:**

**Student Management: Allow administrators to add, search, and display student records easily. This ensures that all student information is organized and accessible.**

**Course Management: Provide functionality to add and display courses available for enrollment.**

**Enrollment Tracking: Enable the assignment and removal of courses for individual students, ensuring that their academic records reflect their current enrollment status.**

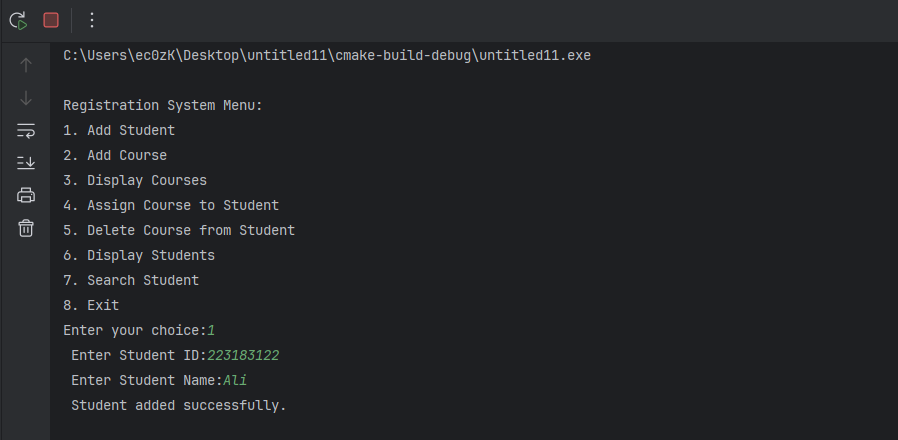
**Methodology**

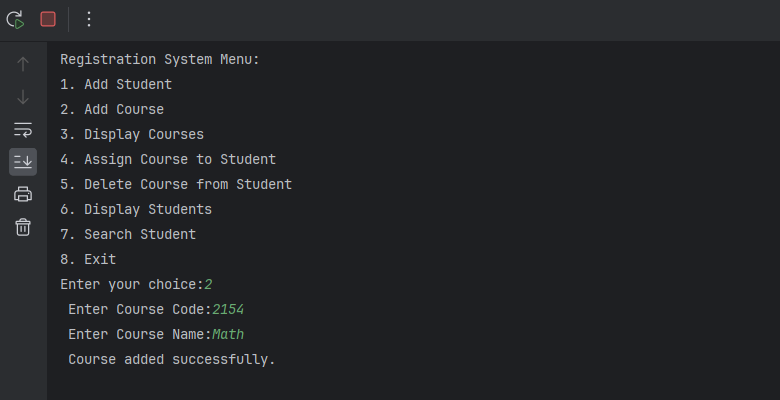
**Program Code:**

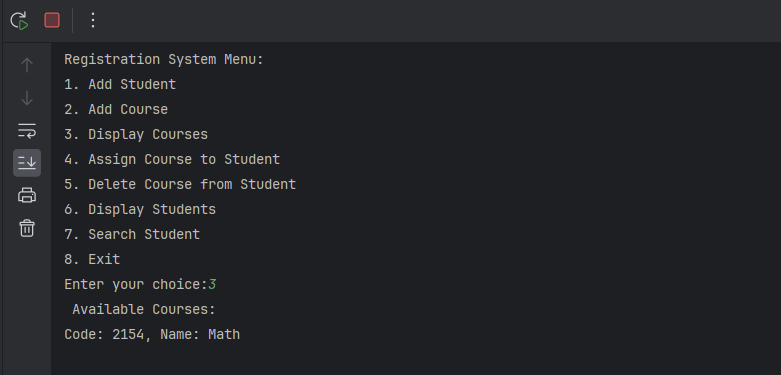
#include <iostream>  
#include <string>  
using namespace std;  
  
// Structure for Course Node  
struct Course {  
 string code;  
 string name;  
 Course\* next;  
};  
  
// Structure for Student Node  
struct Student {  
 string id;  
 string name;  
 Course\* enrolledCourses; // Pointer to a linked list of courses  
 Student\* next;  
};  
  
// Registration System Class  
class RegSystem {  
private:  
 Student\* studentHead; // Head of the linked list for students  
 Course\* courseHead; // Head of the linked list for courses  
  
public:  
 // Constructor to initialize the system  
 RegSystem() : studentHead(nullptr), courseHead(nullptr) {}  
  
 // Add a new student  
 void addStudent() {  
 string id, name;  
  
 cout << "Enter Student ID: ";  
 cin >> id;  
  
 // Check if the student already exists  
 Student\* temp = studentHead;  
 while (temp != nullptr) {  
 if (temp->id == id) {  
 cout << "Student with ID " << id << " already exists.\n";  
 return;  
 }  
 temp = temp->next;  
 }  
  
 cout << "Enter Student Name: ";  
 cin.ignore();  
 getline(cin, name);  
  
 // Create a new student node  
 Student\* newStudent = new Student{id, name, nullptr, nullptr};  
 newStudent->next = studentHead;  
 studentHead = newStudent;  
  
 cout << "Student added successfully.\n";  
 }  
  
 // Add a new course  
 void addCourse() {  
 string code, name;  
  
 cout << "Enter Course Code: ";  
 cin >> code;  
  
 // Check if the course already exists  
 Course\* temp = courseHead;  
 while (temp != nullptr) {  
 if (temp->code == code) {  
 cout << "Course with code " << code << " already exists.\n";  
 return;  
 }  
 temp = temp->next;  
 }  
  
 cout << "Enter Course Name: ";  
 cin.ignore();  
 getline(cin, name);  
  
 // Create a new course node  
 Course\* newCourse = new Course{code, name, nullptr};  
 newCourse->next = courseHead;  
 courseHead = newCourse;  
  
 cout << "Course added successfully.\n";  
 }  
  
 // Display all available courses  
 void displayCourses() {  
 if (courseHead == nullptr) {  
 cout << "No courses available.\n";  
 return;  
 }  
  
 cout << "Available Courses:\n";  
 Course\* temp = courseHead;  
 while (temp != nullptr) {  
 cout << "Code: " << temp->code << ", Name: " << temp->name << endl;  
 temp = temp->next;  
 }  
 }  
  
 // Assign a course to a student  
 void assignCourseToStudent() {  
 string studentID, courseCode;  
  
 cout << "Enter Student ID: ";  
 cin >> studentID;  
  
 // Find the student  
 Student\* student = studentHead;  
 while (student != nullptr && student->id != studentID) {  
 student = student->next;  
 }  
  
 if (!student) {  
 cout << "Student not found.\n";  
 return;  
 }  
  
 cout << "Enter Course Code: ";  
 cin >> courseCode;  
  
 // Check if the course exists  
 Course\* course = courseHead;  
 while (course != nullptr && course->code != courseCode) {  
 course = course->next;  
 }  
  
 if (!course) {  
 cout << "Course not found.\n";  
 return;  
 }  
  
 // Add course to the student's enrolled courses  
 Course\* newCourse = new Course{course->code, course->name, student->enrolledCourses};  
 student->enrolledCourses = newCourse;  
  
 cout << "Course assigned successfully.\n";  
 }  
  
 // Delete a specific course from a student's enrolled courses  
 void deleteCourseFromStudent() {  
 string studentID, courseCode;  
  
 cout << "Enter Student ID: ";  
 cin >> studentID;  
  
 // Find the student  
 Student\* student = studentHead;  
 while (student != nullptr && student->id != studentID) {  
 student = student->next;  
 }  
  
 if (!student) {  
 cout << "Student not found.\n";  
 return;  
 }  
  
 cout << "Enter Course Code to delete: ";  
 cin >> courseCode;  
  
 // Find and delete the course from the student's enrolled courses  
 Course\* current = student->enrolledCourses;  
 Course\* previous = nullptr;  
  
 while (current != nullptr && current->code != courseCode) {  
 previous = current;  
 current = current->next;  
 }  
  
 if (!current) {  
 cout << "Course not found for the student.\n";  
 return;  
 }  
  
 // Remove the course node  
 if (previous) {  
 previous->next = current->next;  
 } else {  
 student->enrolledCourses = current->next;  
 }  
  
 delete current;  
 cout << "Course removed successfully from student " << studentID << ".\n";  
 }  
  
 // Display all students  
 void displayStudents() {  
 if (studentHead == nullptr) {  
 cout << "No students registered.\n";  
 return;  
 }  
  
 cout << "Registered Students:\n";  
 Student\* temp = studentHead;  
 while (temp != nullptr) {  
 cout << "ID: " << temp->id << ", Name: " << temp->name << endl;  
  
 // Display enrolled courses  
 Course\* course = temp->enrolledCourses;  
 if (!course) {  
 cout << " No enrolled courses.\n";  
 } else {  
 cout << " Enrolled Courses:\n";  
 while (course) {  
 cout << " Code: " << course->code << ", Name: " << course->name << endl;  
 course = course->next;  
 }  
 }  
  
 temp = temp->next;  
 }  
 }  
  
 // Search for a student and display their details  
 void searchStudent() {  
 string studentID;  
 cout << "Enter Student ID: ";  
 cin >> studentID;  
  
 Student\* student = studentHead;  
 while (student != nullptr && student->id != studentID) {  
 student = student->next;  
 }  
  
 if (!student) {  
 cout << "Student not found.\n";  
 return;  
 }  
  
 cout << "Student Details:\n";  
 cout << "ID: " << student->id << ", Name: " << student->name << endl;  
 cout << "Enrolled Courses:\n";  
 Course\* course = student->enrolledCourses;  
 if (!course) {  
 cout << "None\n";  
 } else {  
 while (course != nullptr) {  
 cout << "Code: " << course->code << ", Name: " << course->name << endl;  
 course = course->next;  
 }  
 }  
 }  
};  
  
int main() {  
 RegSystem system;  
 int choice;  
  
 do {  
 cout << "\nRegistration System Menu:\n"  
 << "1. Add Student\n"  
 << "2. Add Course\n"  
 << "3. Display Courses\n"  
 << "4. Assign Course to Student\n"  
 << "5. Delete Course from Student\n"  
 << "6. Display Students\n"  
 << "7. Search Student\n"  
 << "8. Exit\n"  
 << "Enter your choice: ";  
 cin >> choice;  
  
 switch (choice) {  
 case 1:  
 system.addStudent();  
 break;  
 case 2:  
 system.addCourse();  
 break;  
 case 3:  
 system.displayCourses();  
 break;  
 case 4:  
 system.assignCourseToStudent();  
 break;  
 case 5:  
 system.deleteCourseFromStudent();  
 break;  
 case 6:  
 system.displayStudents();  
 break;  
 case 7:  
 system.searchStudent();  
 break;  
 case 8:  
 cout << "Exiting program.\n";  
 return 0;  
 default:  
 cout << "Invalid choice. Try again.\n";  
 }  
 } while (true);  
  
 return 0;  
}

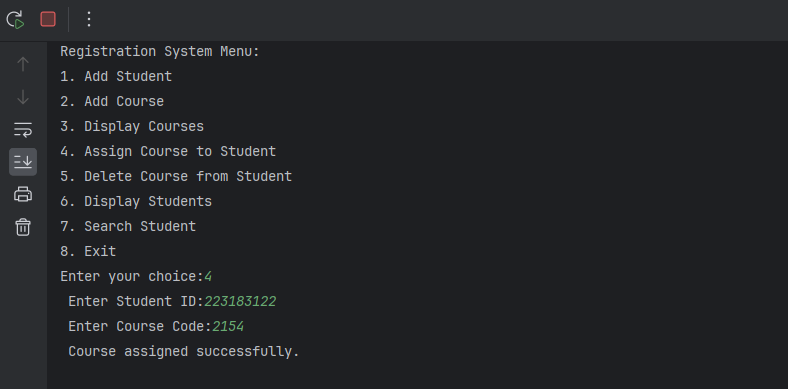
**Results and discussion**

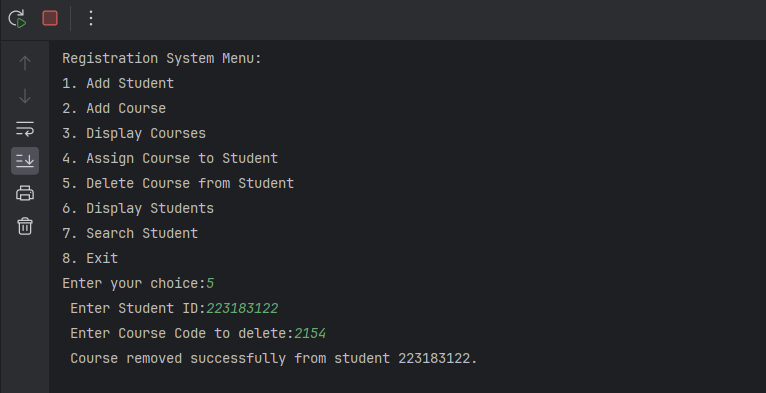
**Sample Inputs & Outputs:**

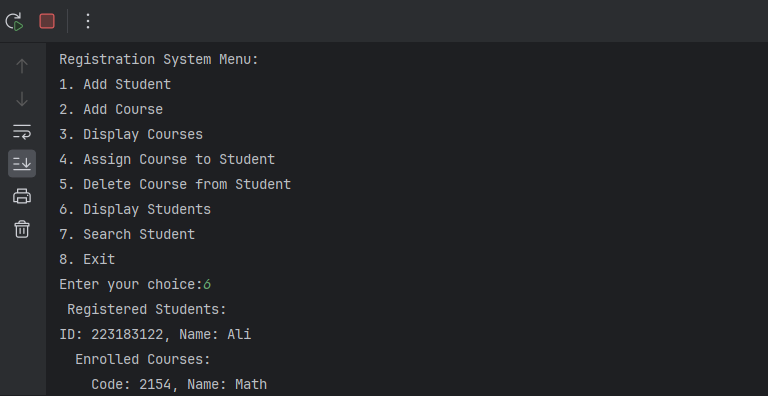
**Add Student: Administrators can register new students by entering their ID and name, with checks in place to prevent duplicate entries.**

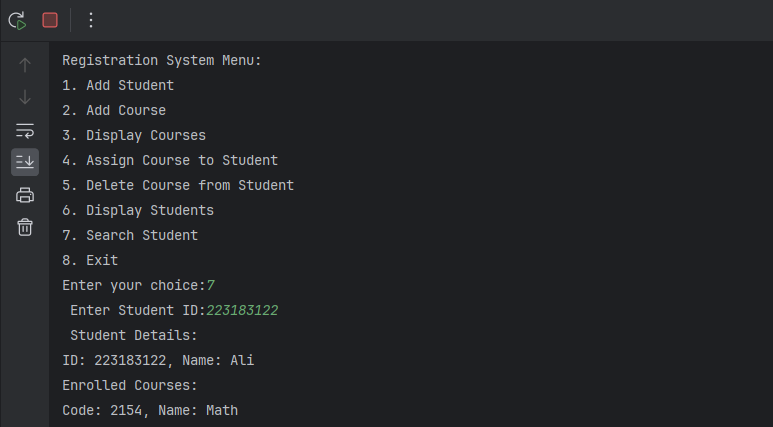
**Add Course: New courses can be added to the system with unique codes and names, ensuring that the course catalog is comprehensive and up-to-date.**

**Display Courses and Students: The system can list all available courses and registered students, making it easy to view and manage the educational offerings and student body.**

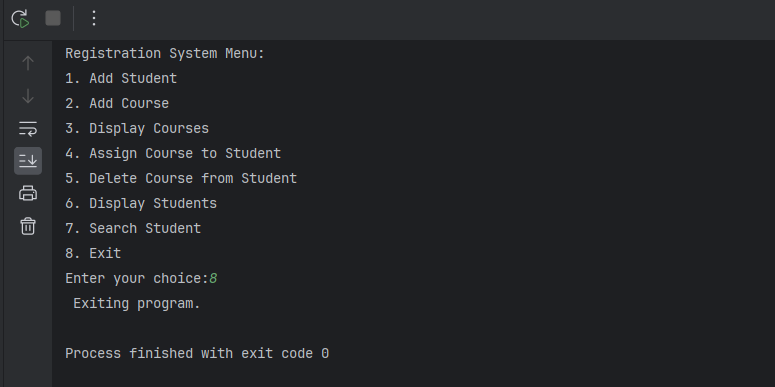
**Assign Course to Student&** **Delete Course from Student: Allows for the assignment of courses to students and the ability to remove courses as needed, keeping students' records current.**



**Display Students: To display the list of registered students in the Student Registration System, and display each student's ID, name, and their enrolled courses, if any.**

**Search Student: Users can search for specific students to view their details and enrolled courses.**

**Exit: The exit functionality in the Student Registration System is implemented to allow users to terminate the program.**

****

**Conclusion**

**The Student Registration System is an essential tool for educational institutions aiming to streamline their administrative processes. By efficiently managing student and course data, the system not only saves time for administrators but also enhances the overall educational experience for students. Its structured approach to data organization ensures that all stakeholders can access and manage information effortlessly.**