1. Introduction

The C++ Student Enrollment System serves as a comprehensive solution for managing student information within educational institutions. This report outlines the design, implementation, and functionality of the system, focusing on the creation of a Student structure and key functions for enrollment management.

2. Objectives

The primary objectives of the C++ Student Enrollment System are as follows:

- **Structured Student Definition:** Develop a Student structure with fields such as name, age, class, and subjects.
- **Enrollment Functions:** Implement functions to facilitate the enrollment process, including adding students, displaying the list of enrolled students, searching for students by name, and calculating the total number of students enrolled.

3. System Design

3.1 Student Structure Definition

The heart of the system lies in the Student structure, providing a foundation for storing essential student details:

```
struct Student {
    std::string name;
    int age;
    std::string class_;
    std::vector<std::string> subjects;
};
```

The structure includes fields such as `name` (student's full name), `age`, `class_` (student's class), and `subjects` (a vector to store enrolled subjects).

3.2 Functions

3.2.1 Enroll Students

void enrollStudent(std::vector<Student>& students);

This function enables the user to input student information and stores it in the system. It ensures a systematic approach to capturing key details.

3.2.2 Display Enrolled Students

void displayEnrolledStudents(const std::vector<Student>& students);

Displays a comprehensive list of all students currently enrolled. This function aids in monitoring and verification of student data.

3.2.3 Search Student by Name

void searchStudentByName(const std::vector<Student>& students, const std::string& name);

Facilitates the quick retrieval of a student's details by searching with their name. This enhances the accessibility of individual student records.

3.2.4 Calculate Number of Students Enrolled

int calculateNumberOfStudentsEnrolled(const std::vector<Student>& students);

Determines and returns the count of students currently enrolled. This function provides a statistical overview of the student population.

4. Implementation

The implementation phase involves coding the Student structure and functions in C++, ensuring their seamless integration and functionality within the system. The use of vectors for storing student data allows for dynamic and efficient management of information.

5. Testing

Thorough testing is conducted to validate the accuracy and reliability of each function. Unit testing, integration testing, and system testing are performed to identify and rectify any potential bugs or issues.

6. User Manual

A detailed user manual is provided to guide administrators and users on the effective utilization of the system. It includes step-by-step instructions for each function, ensuring a smooth user experience.

7. Conclusion

The C++ Student Enrollment System successfully achieves its objectives by providing a structured approach to student information management. The combination of the Student structure and enrollment functions offers a user-friendly interface, streamlining the enrollment process and facilitating efficient data retrieval.

8. Future Enhancements

Future enhancements may include the incorporation of additional features such as course registration, dynamic memory management, and integration with a larger student management system. These improvements aim to further enhance the system's capabilities and adaptability to evolving educational needs.

9. Acknowledgments

We extend our appreciation to the developers and users who contributed valuable insights during the development and testing phases of this project. Their feedback and collaboration have significantly enriched the quality of the C++ Student Enrollment System.

This project report provides a comprehensive overview of the C++ Student Enrollment System, detailing its design, objectives, implementation, and future enhancements. The system serves as an effective tool for educational institutions seeking an organized and efficient approach to student information management.

Code:

```
#include <iostream>
#include <string>

using namespace std;

// Define the Student structure
struct Student {
   string name;
   int age;
   string className;
   string subjects[5];  // Assuming a student can have up to 5 subjects
};

const int MAX_STUDENTS = 100;  // Maximum number of students the system can handle

// Function prototypes
void enrollStudent(Student students[], int& enrolledCount);
void displayEnrolledStudents(const Student students[], int enrolledCount);
void searchStudentByName(const Student students[], int enrolledCount);
int calculateEnrolledStudents(const int enrolledCount);

int main() {
   Student students[MAX_STUDENTS];
   int enrolledCount = 0;
```

```
    char choice;
    do {
        cout << "1. Enroll Student\n"
             << "2. Display Enrolled Students\n"
             << "3. Search Student by Name\n"
             << "4. Calculate Number of Students Enrolled\n"
             << "5. Exit\n"
             << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case '1':
                enrollStudent(students, enrolledCount);
                break;

            case '2':
                displayEnrolledStudents(students, enrolledCount);
                break;

            case '3':
                searchStudentByName(students, enrolledCount);
                break;

            case '4':
                cout << "Number of students enrolled: " << calculateEnrolledStudents(enrolledCount) << endl;
                break;

            case '5':
                cout << "Exiting program. Thank you!\n";
                break;

            default:
                cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != '5');

    return 0;
}

// Function to enroll a student
void enrollStudent(Student students[], int& enrolledCount) {
    if (enrolledCount < MAX_STUDENTS) {
        cout << "Enter student details:\n";
        cout << "Name: ";
        cin.ignore();  // Ignore any remaining newline characters in the buffer
        getline(cin, students[enrolledCount].name);
```

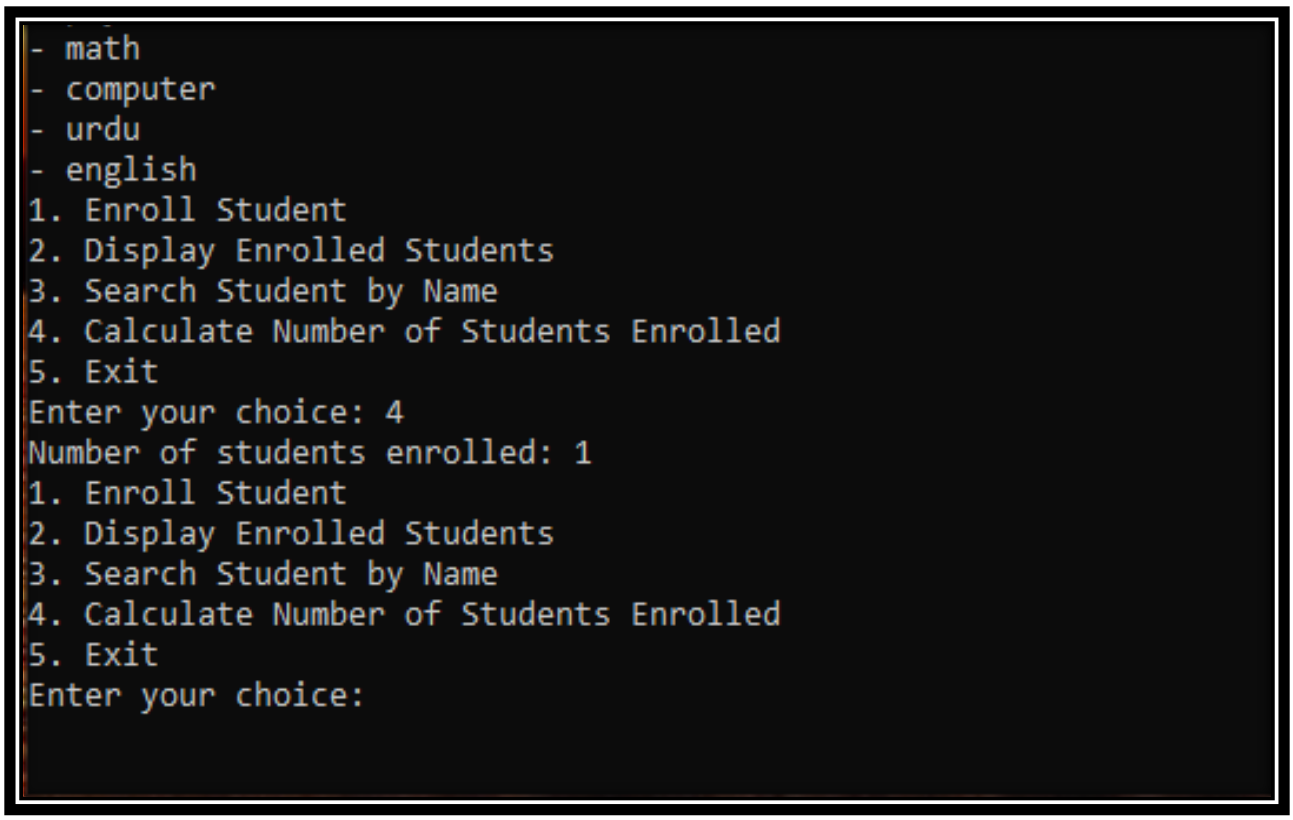
```
        cout << "Age: ";
        cin >> students[enrolledCount].age;

        cout << "Class: ";
        cin.ignore();  // Ignore any remaining newline characters in the buffer
        getline(cin, students[enrolledCount].className);

        cout << "Enter up to 5 subjects (type 'done' to finish):\n";
        int subjectCount = 0;
        while (subjectCount < 5) {
            cout << "Subject " << subjectCount + 1 << ": ";
            getline(cin, students[enrolledCount].subjects[subjectCount]);

            if (students[enrolledCount].subjects[subjectCount] == "done") {
                break;
            }

            subjectCount++;
        }

        cout << "Student enrolled successfully!\n";
        enrolledCount++;
    } else {
        cout << "Maximum number of students reached. Cannot enroll more students.\n";
    }
}

// Function to display enrolled students
void displayEnrolledStudents(const Student students[], int enrolledCount) {
    if (enrolledCount > 0) {
        cout << "List of enrolled students:\n";
        for (int i = 0; i < enrolledCount; i++) {
            cout << "Student " << i + 1 << ":\n";
            cout << "Name: " << students[i].name << "\n";
            cout << "Age: " << students[i].age << "\n";
            cout << "Class: " << students[i].className << "\n";
            cout << "Subjects:\n";
            for (int j = 0; j < 5 && students[i].subjects[j] != ""; j++) {
                cout << "- " << students[i].subjects[j] << "\n";
            }
            cout << "-----------------\n";
        }
    } else {
        cout << "No students enrolled yet.\n";
    }
}

// Function to search for a student by name
```

```
void searchStudentByName(const Student students[], int enrolledCount) {
   cout << "Enter the name of the student to search: ";
   cin.ignore();  // Ignore any remaining newline characters in the buffer
   string searchName;
   getline(cin, searchName);

   bool found = false;
   for (int i = 0; i < enrolledCount; i++) {
      if (students[i].name == searchName) {
         cout << "Student found:\n";
         cout << "Name: " << students[i].name << "\n";
         cout << "Age: " << students[i].age << "\n";
         cout << "Class: " << students[i].className << "\n";
         cout << "Subjects:\n";
         for (int j = 0; j < 5 && students[i].subjects[j] != ""; j++) {
            cout << "- " << students[i].subjects[j] << "\n";
         }
         found = true;
         break;
      }
   }

   if (!found) {
      cout << "Student not found.\n";
   }
}

// Function to calculate the number of enrolled students
int calculateEnrolledStudents(const int enrolledCount) {
   return enrolledCount;
}
```

This program defines a `Student` structure, allows for enrolling students, displays enrolled students, searches for a student by name, and calculates the number of enrolled students. The data is stored in an array of `Student` structures, and various functions handle different aspects of the program's functionality. Please note that this is a basic example, and you can expand and enhance it based on your specific requirements

Output:

```
1. Enroll Student
2. Display Enrolled Students
3. Search Student by Name
4. Calculate Number of Students Enrolled
5. Exit
Enter your choice: 1
Enter student details:
Name: moiz
Age: 20
Class: 12
Enter up to 5 subjects (type 'done' to finish):
Subject 1: physics
Subject 2: math
Subject 3: computer
Subject 4: urdu
Subject 5: english
Student enrolled successfully!
1. Enroll Student
2. Display Enrolled Students
3. Search Student by Name
4. Calculate Number of Students Enrolled
5. Exit
Enter your choice: 2
List of enrolled students:
Student 1:
Name: moiz
Age: 20
Class: 12
Subjects:
- physics
```


- math
- computer
- urdu
- english
1. Enroll Student
2. Display Enrolled Students
3. Search Student by Name
4. Calculate Number of Students Enrolled
5. Exit
Enter your choice: 4
Number of students enrolled: 1
1. Enroll Student
2. Display Enrolled Students
3. Search Student by Name
4. Calculate Number of Students Enrolled
5. Exit
Enter your choice: