

# Student Enrollment Management Tool

Diosdado T. Aparri VI

Thomas Joaquin Bernardo

Brix Edmar Dela Rosa

BSIT 2 - 1

Ms. Mirafe Prospero

## TABLE OF CONTENT

<b>Title Page.....</b>	<b>2</b>
<b>Table of Contents.....</b>	<b>3</b>
<b>Introduction.....</b>	<b>4</b>
<b>Tool Overview.....</b>	<b>5</b>
<b>Topics Covered.....</b>	<b>6</b>
<b>Instructions for Using the tools .....</b>	<b>7</b>
<b>Conclusion.....</b>	<b>8</b>
<b>Reference.....</b>	<b>9</b>

## Introduction

The Student Enrollment Management Tool is a console-based application designed to streamline student records management. The tool allows educational institutions and administrators to efficiently handle essential student information, such as name, ID, and age, using a combination of advanced programming concepts like recursion, sorting algorithms, and object-oriented programming (OOP).

By leveraging Data Structures and Algorithms (DSA) principles, this tool ensures effective searching, sorting, and manipulation of student data.

### *Key Concepts:*

**Recursion:** This concept is used for searching through student records by recursively navigating through the list of students. Recursion helps break down the search task into smaller, more manageable parts.

**Quick Sort:** A highly efficient sorting algorithm that is used to organize student records based on either name or student ID. Quick Sort employs a divide-and-conquer approach to reorder student records.

**Object-Oriented Programming:** The tool utilizes classes to represent students and manage the entire enrollment system. This makes the code more modular, reusable, and easier to maintain.

## Target Audience:

This tool is primarily aimed at educational institutions, administrators, and school registrars who require a simple yet effective way to manage student enrollment data.

## Tool Overview

### Description:

The Student Enrollment Management Tool is a command-line application that allows administrators to manage a list of students. It supports the following functionalities:

**Add New Students:** Allows input of student details such as name, ID, and age.

**Search Student Records:** Utilizes recursion to search for students by name or ID.

**Sort Student Records:** Implements Quick Sort to arrange records by name or ID.

**Display Student Records:** Displays all student records currently stored in the system.

### Features:

Add new students with their name, ID, and age.

Search for students using recursion, which is an efficient method to find records based on a given search key (name or ID).

Sort student records by name or ID using the Quick Sort algorithm to arrange them in ascending order.

Display all student records in a user-friendly format.

### Scope Limitations:

The tool is console-based and does not have a graphical user interface (GUI).

It handles only basic student information (name, ID, and age).

The tool does not integrate with a database, meaning the data is only available during the runtime of the program.

## Topics Covered

### *Topic 1: Recursion (Search Functionality)*

#### *Description:*

*Recursion is the technique used for searching student records by traversing through the list of students and comparing each record to the provided search value.*

*Recursion involves a function calling itself until a base case is met.*

#### *Tool Functionality:*

*The function `searchStudentsRecursive` is defined to search for student records.*

*The function takes the list of students and the search key (name or ID) as input parameters and processes them recursively.*

#### Example:

Search for a student by name:

```
auto searchResults = system.searchStudents("name", "John Doe");
```

#### Output:

John Doe, 12345, 20

## *Topic 2: Sorting (Quick Sort)*

### Description:

Quick Sort is an efficient sorting algorithm that operates using the divide-and-conquer approach. The student list is partitioned into smaller sublists and sorted recursively. Quick Sort is well-suited for sorting large datasets due to its average-case time complexity of  $O(n \log n)$ .

### Tool Functionality:

The function `quickSortStudents` implements the Quick Sort algorithm to sort students based on either name or ID. It recursively divides the student list and sorts each partition until the entire list is sorted.

### Example:

Sorting by name:

```
system.sortStudents("name");
```

### Output:

Alice, 001, 21

Bob, 002, 22

### *Topic 3: Object-Oriented Programming (OOP)*

#### Description:

The tool utilizes Object-Oriented Programming (OOP) principles to organize code efficiently. The use of classes enables data encapsulation, abstraction, inheritance, and polymorphism.

#### Tool Functionality:

Student Class: Represents an individual student with attributes such as name, ID, and age.

EnrollmentSystem Class: Manages the system's overall functionality, such as adding students, sorting, and searching.

#### Example:

Adding a new student:

### *Instructions for Using the Tool*

#### How to Install or Access the Tool

#### Prerequisites:

Ensure a C++ compiler is installed on your system. Recommended compilers:

g++ (GNU C++ Compiler) for Linux/Windows.

Xcode for macOS.

A terminal or command prompt to run the tool.

Downloading the Code:

Copy the source code provided into a file named `enrollment_tool.cpp`.

Alternatively, download the source file from a shared repository if provided.

Compilation:

Open a terminal or command prompt and navigate to the directory containing the `enrollment_tool.cpp` file.

Compile the code:

```
g++ -o enrollment_tool enrollment_tool.cpp
```

Running the Tool:

Execute the compiled program:

```
./enrollment_tool
```

For Windows Users:

Replace `./enrollment_tool` with `enrollment_tool.exe` when running the program.

## Step-by-Step Instructions for Key Features

### 1. Adding a New Student

Select Option 1 from the main menu: "Add Student".

Enter the student's name when prompted.

Enter the student's age.

The tool generates a unique ID for the student and confirms the addition.

Example:

Enter student name: John Doe

Enter student age: 20

Student added successfully! ID: S1

## 2. Searching for Students

Select Option 2 from the main menu: "Search Students".

Enter the search key (name or id).

Provide the search value corresponding to the key.

Example:

Enter search key (name/id): name

Enter search value: John Doe

Search Results:

John Doe, S1, 20

## 3. Sorting Student Records

Select Option 3 from the main menu: "Sort Students".

Specify the sort key (name or id).

Example:

Enter sort key (name/id): name

Students sorted successfully!

#### 4. Displaying All Students

Select Option 4 from the main menu: "Display Students".

The tool lists all enrolled students in the current order.

Example:

Alice, S1, 21

Bob, S2, 22

#### 5. Clearing the Screen

Select Option 5 from the main menu: "Clear Screen".

This feature clears the terminal to improve visibility.

#### 6. Exiting the Program

Select Option 6 from the main menu: "Exit".

The program terminates safely.

#### Screenshots and Diagrams

While this is a console-based tool, screenshots can help visualize the process.

Here's an example layout of the menu screen:

Main Menu:

## Student Enrollment System

1. Add Student
2. Search Students
3. Sort Students
4. Display Students
5. Clear Screen
6. Exit

Enter your choice: \_

Adding a Student:

\*\*\*\*\*ADD STUDENT\*\*\*\*\*

Enter student name: John Doe

Enter student age: 20

Student added successfully! ID: S1

## Conclusion

The Student Enrollment Management Tool effectively demonstrates the practical application of advanced programming concepts like recursion, sorting algorithms, and OOP. It enables administrators to manage student records efficiently and provides an easy-to-use console interface.

Effectiveness:

The recursive search ensures flexibility and accuracy for querying records.

Quick Sort guarantees fast sorting for large datasets.

OOP principles offer a scalable and maintainable codebase.

Potential Future Improvements:

Persistent Storage: Integrate a database or file system to store records permanently.

GUI Development: Create a graphical interface to enhance user interaction.

Advanced Features: Add options to update or delete student records.

Data Export/Import: Allow users to save and load student data as files.

## References

C++ Programming Concepts:

Official C++ Documentation: [cplusplus.com](http://cplusplus.com)

Recursion and Sorting Techniques: Data Structures and Algorithms by Mark Allen Weiss.

Object-Oriented Programming Principles: Programming in C++ by Bjarne Stroustrup.

Tool Design and Inspiration:

General Enrollment Systems: Research from educational management software.