The National Teacher College

Finals in Data Structure
Registration System

By

Student Name:
John Albert David

Course:
Bachelor of Science in Information Technology

Block:
2.8 BSIT

Professor's Name:
Justin Louise R. Neypes

The National Teacher College

I. Introduction

1.1

The Student Registration System is a project developed using the C++ programming language to address common challenges in student registration and applicant management within higher education institutions. The system utilizes fundamental data structures such as arrays and stacks, along with sorting algorithms and file handling techniques, to efficiently store, organize, and manage student records.

The National Teacher College

The application enables users to register students, automatically generate unique student identification numbers based on the current date, sort student records by different criteria (Student ID, name, and result average), shortlist qualified applicants, undo the most recent registration, and store data persistently sing CSV files. The system is designed with a fixed capacity of 100 students per day, reflecting realistic operational constraints in public universities.

This project aligns with the United Nations Sustainable Development Goal (SDG) 4: Quality Education, specifically:

- **SDG 4.3**: Ensuring equal access for all to affordable and quality tertiary education.
- **SDG 4.a**: Improving education systems through student-centered and inclusive administrative solutions.

By introducing a structured and automated registration process, the system supports educational institutions in improving efficiency, accuracy, and accessibility in student admissions.

The National Teacher College

1.2

State universities in the Philippines consistently face challenges in managing student registration and applicant screening due to limited manpower, budget constraints, and a high volume of applicants. According to reports from the Commission on Higher Education (CHED), state universities and colleges (SUCs) accommodate thousands of applicants each academic year, often with administrative offices operating using manual or semi-digital systems. These conditions place significant pressure on registration staff and existing infrastructure.

In many state universities, registration processes are conducted based on fixed schedules with limited daily capacity. This results in long queues, overcrowded registration areas, delayed processing of student records, and increased likelihood of data entry errors. During peak enrollment periods, it is common for applicants to spend several hours waiting for their registration to be processed, while administrators struggle to organize, sort, and validate student information efficiently.

Furthermore, the lack of automated sorting and filtering mechanisms makes it difficult for institutions to quickly identify qualified applicants based on academic performance. Manual record handling also increases the risk of misplaced data and inconsistent record keeping, which can negatively affect both students and administrators.

This capstone project addresses these documented challenges by developing a Student Registration Sorting System capable of accommodating up to 100 students per day, even within a limited scheduling framework. By automating student registration, sorting, shortlisting, and data storage, the system provides a practical solution that reduces administrative workload, minimizes errors, and improves the overall efficiency and reliability of the registration process in Philippine state universities.

The National Teacher College

II. Requirements & Analysis

2.1
Functional Requirements (FR)

FR1	● Implemented using a fixed-size array Student students [MAX] to store student records. Input data is read via cin and getline, while persistent data is loaded from a CSV file using LoadFromCSVFile() and stored into the array.	**Prelim DSA**
FR2	● The system uses arrays for data storage, a stack (undo with push and pop) for undo operations, and sorting algorithms (bubble sort) for organizing student records by ID, name, and result average.	**Core DSA**

The National Teacher College

FR 3	● Implemented through advanced combined operations such as Shortlisting passer students, dynamic array reorganization, and undo based data reversal the shortlist passer function performs conditional filtering, in place array compaction, and record count recalculation. Additionally, sorting operations combined with stack-based undo functionality demonstrate coordinated use of arrays, stack, conditional logic, and algorithmic processing.	**Finals DSA**
FR4	● Implemented using formatted cout output in functions such as Display, ShortlistPasser, and confirmation messages after registration, sorting, undo, and exit operations.	**User Interface**

ID	Requirement	Metric
NFR1	**Performance:** Execution time of the core algorithm should be efficient.	The system uses in-memory arrays and simple sorting algorithms (bubble sort and std::sort) which are efficient for small to medium datasets (up to 100 records).
FR2	**Robustness**: The system shall handle invalid operations without crashing.	Basic validation is applied such as checking array limits (MAX), empty undo stack conditions, and file open checks in CSV load/save functions.
NFR3	**Maintainability**: The code shall be modular and easy to modify.	The system is divided into well-defined functions (Add, Sort, Display, SaveToCSVFile, etc.) with structured logic and consistent naming conventions.

2.2. Data Requirements (Description of input data structure and size)

- Data Requirements

The Student Registration Sorting System processes and manages student information using structured data types and fixed-size storage to ensure efficient access and manipulation.

Input Data Sources

The system accepts student data from two primary sources:

- User Input (Console):
  Student name,
  Student result average.
- CSV File Input:
  Previously saved student records are loaded from a CSV file upon program startup.

  Each record is parsed and stored into the students array.

The National Teacher College

Data Storage Structure

Student records are stored in a fixed-size array:

- Maximum number of students: 100 records per day
- Maximum undo operations: 100 records
- Data is processed entirely in memory using arrays
- Sorting and filtering operations are designed to work efficiently within these constraints

These limitations reflect realistic operational conditions in state universities and ensure the system remains efficient, manageable, and easy to maintain.

-

2.3. Complexity Analysis: Expected Time/Space complexity of the Core Algorithm (justify using Big O notation).

Operation	**Time complexity**	**Space complexity**
Add Student	O(1) Constant time Insertion into an array and stack push operations take constant time regardless of the number of students.	O(1) Only a single Student structure is created per operation.
Undo Operation	O(n) Linear time Searching for the student ID and shifting array elements requires traversing the array in the worst case.	O(1) Constant space No additional data structures are created.

The National Teacher College

Sorting	$O(n^2)$ / $O(n \log n)$ Bubble Sort: $O(n^2)$ - Quadratic time std::sort: $O(n \log n)$ - Average case Bubble sort is suitable due to the small dataset size ($n \leq 100$).	$O(1)$ Constant space Sorting is done in-place without auxiliary storage.
Shortlist	$O(n)$ Linear time Each student record is examined once.	$O(1)$ Constant space The operation modifies the existing array without allocating additional memory.
File I/O	$O(n)$ Linear time Each record is read from or written to the file once.	$O(1)$ Constant space Data is stored directly into the fixed-size array.

The National Teacher College

1. **Array (Primary Data Storage)**

Justification:
Arrays were chosen as the primary data structure because the system operates with a known and fixed maximum number of student records (100 students per day). Arrays provide fast access, simple implementation, and efficient memory usage, making them appropriate for small-scale registration systems commonly found in state universities.

Implementation Details:
The system uses a fixed-size array of Student structures:

The variable pupils is used to track the current number of registered students. All major operations adding students, sorting records, displaying data, shortlisting passers, and deleting records—are performed directly on this array.

2. **Stack (Undo Functionality)**

Justification:
A stack was selected to implement the undo feature because it follows the Last-In-First-Out (LIFO) principle, which naturally supports reversing the most

recent student registration. This ensures data integrity and allows users to correct recent input errors efficiently.

Implementation Details:

The system uses custom push() and pop() functions. Each newly registered student is pushed onto the stack, and when the undo option is selected, the most recent student record is popped and removed from the main student array.

3. **Sorting Algorithms (Data Organization)**

Justification:
Sorting algorithms were used to organize student records by Student ID, name, and result average, improving readability and supporting administrative decision-making. Sorting is essential in registration systems where ordered data presentation is required for evaluation and screening.

Implementation Details:
The system implements Bubble Sort for sorting operations triggered by the user and uses std::sort when saving records to a CSV file:

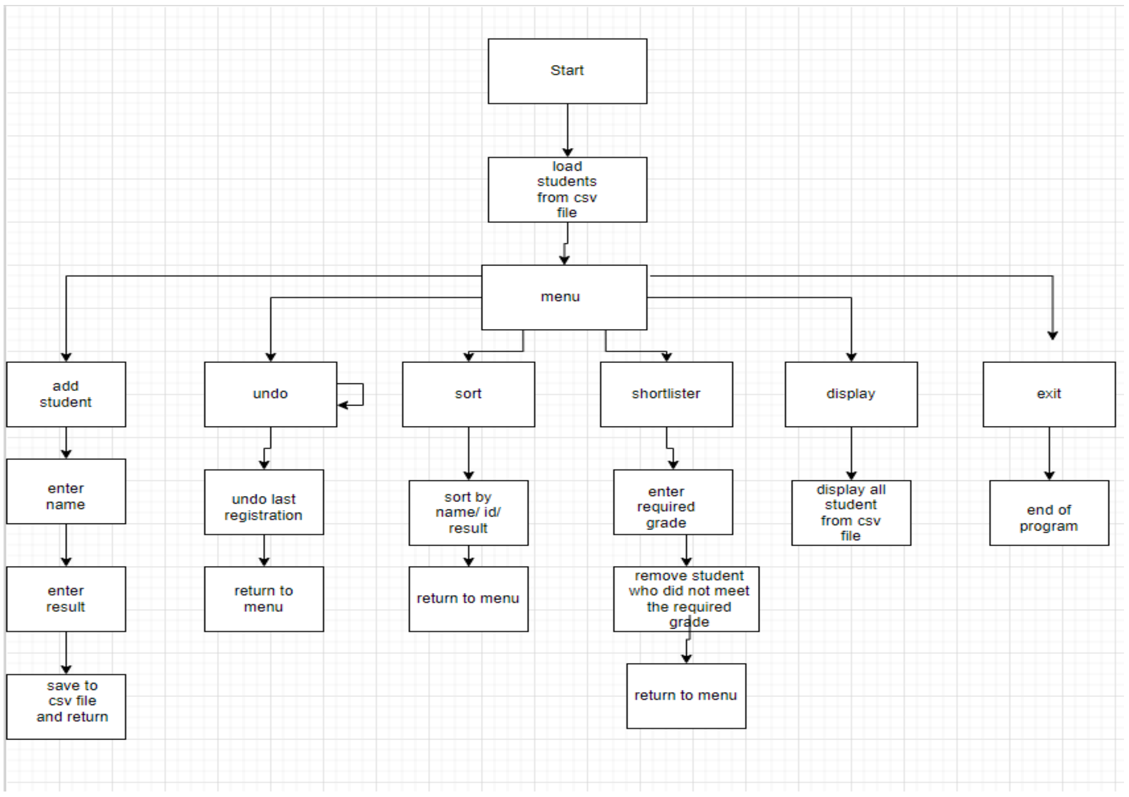
The National Teacher College

- Sorting by Student ID
- Sorting by Student Name
- Sorting by Result Average

All sorting operations are performed directly on the students array, ensuring in-place sorting without additional memory overhead.

The National Teacher College

3.2. Algorithm Flowchart: Include the Flowchart for the system's most complex function (the core algorithm using a Finals concept)



3.3. Module Breakdown: Define the custom C++ classes and how they interact.

1. **Student struct:**

- Purpose:
  Represents a single student with a unique ID, name, and result average.

- Interactions: Stored in the global students array, passed to the push() and pop() functions for undo functionality, used in sorting, displaying, CSV saving/loading, and shortlisting operations.

2. Undo:

- Purpose: Maintains a history of added students to allow undoing the last registration.

- Interactions: Called from Add() to record new students (push: Adds a student to the undo stack), called from Last() to undo a registration (pop: Removes and returns the last student from the stack), works directly with the Student struct.

Core modules:

3. Add()

- Purpose: Register a new student.
- Process: heck if the student array is full, input name and resultAverage, generate unique ID based on current date and student count, store student in students array, push student to undo stack,
  save to CSV.

- Interactions: Uses Student struct and push().

4. Last()

- Purpose: Undo last student registration.

- Process: Pop the last student from the undo stack, find and remove the student from the students array, decrement the pupil count.
- Interactions: Uses Student struct, pop(), and modifies the students array.

5. **Sort(int a)**

- Purpose: Sort students by ID, name, or result average.
- Interactions: Directly operates on the students array and uses nested loops for Bubble Sort.

6. **Display()**
   - Purpose: Show all registered students.
   - Interactions: Reads students array and prints data.

7. **SaveToCSVFile() & LoadFromCSVFile()**
   - Purpose: Persist student data between sessions.
   - Interactions: SaveToCSVFile() writes students array to a CSV file, loadFromCSVFile() reads CSV and repopulates students array and undo stack.

8. **ShortlistPasser()**
   - Purpose: Filter students who meet a passing grade.

- Interactions: Operates on students array, updates pupils count after removing non-passers, Calls SaveToCSVFile() to update CSV.

9. Main Module (main())

- Purpose: Provides the user interface and main control flow.
- Process: Load students from CSV on startup, display menu and read user input, call appropriate function based on choice, loop until the user selects exit.
- Interactions: Calls all functional modules: Add(), Last(), Sort(), Display(), ShortlistPasser(), and CSV operations, uses students array and undo stack.

The National Teacher College

5.1. Conclusion

5.2. Individual Contributions (Detailed breakdown of each member's assigned module/class.)

Conclusion

The Student Registration Sorting System successfully addresses the challenges faced by state universities in the Philippines in managing student registration and applicant screening. By leveraging C++ programming, fundamental data structures such as arrays and stacks, sorting algorithms, and file handling techniques, the system provides an automated, efficient, and reliable solution for registering up to 100 students per day.

The system's key functionalities—including automatic student ID generation, sorting by multiple criteria, shortlisting qualified applicants, undoing recent registrations, and persistent data storage using CSV files—demonstrate its effectiveness in reducing administrative workload, minimizing errors, and improving overall operational efficiency.

Moreover, the project supports the United Nations Sustainable Development Goal (SDG) 4 on Quality Education by promoting equitable access to higher education and introducing student-centered administrative solutions. By implementing a structured and automated registration process, this system enhances accessibility, accuracy, and transparency in student admissions.

Overall, this capstone project not only provides a practical tool for Philippine state universities but also serves as a foundation for future enhancements, such as expanding daily capacity, incorporating search

The National Teacher College

functions, or integrating database systems to further improve scalability, usability, and administrative effectiveness , adding search capabilities, implementing graphical user interfaces (GUI).

The system, documentation, and presentation is all made by Mr. John Albert David.

The National Teacher College

REFERENCES:

1) Overcapacity and admission denial in SUCs
At least 168,000 students were denied enrollment in state universities due to capacity issues—showing that demand exceeds what universities can handle. Philstar.com
Philippine Star — "168,000 fail to enter SUCs amid capacity issues"

https://www.philstar.com/headlines/2025/10/03/2477189/168000-fail-enter-sucs-amid-capacity-issues

2) SUC admission and retention challenges recognized by CHED
The Commission on Higher Education (CHED) has launched research into admission system challenges and retention in state universities, indicating systemic issues with how admissions are currently handled. GMA Network
🔗 *GMA News Online — "CHED launches study on student admission, retention in SUCs"*

https://www.gmanetwork.com/news/topstories/nation/897227/ched-launches-study-on-student-admission-retention-in-sucs/story/

Evidence of Manual Enrollment Challenges (Secondary Source)

3) Manual enrollment process causes long queues and inefficiencies
While focused on one institution, this reflects a broader trend in Philippine colleges where traditional manual enrollment leads to long queues, delays, and errors. Studocu
Studocu summary — "Manual enrollment process inefficiencies"
(Excerpt from a study discussing general enrollment struggles)