STUDENT INFORMATION MANAGEMENT SYSTEM

(COBOL OJT PROJECT)

GIC Myanmar Co. Ltd.

Submitted by

OJT GROUP 6

Khin Oo Thi Han

Khine Zin Nyunt

Khine Khine Oo

Kay Zin Thant

Nadi Linn

Supervised by

Mr. Htet Linn

Mentor, GIC Myanmar Co. Ltd.

SUBMISSION DATE: 25 JULY 2025

TABLE OF CONTENTS

ACKNOWLEDGEMENT

We would like to express our gratitude to Mr. Iwanaga San, CEO of GIC Myanmar Co. Ltd., for granting us the opportunity to join the internship program and undertake this OJT project. His support enabled us to apply our classroom knowledge in a real-world setting, which has been an invaluable experience.

We are also deeply thankful to Ms. Aye Thi Khaing, Ms. Thiri Soe, and Ms. Nann Thuzar for their guidance, insightful feedback, and continuous encouragement throughout the project presentation phase.

Special thanks go to Mr. Htet Linn for his mentorship and helpful advice, which greatly shaped the outcome of our project. We also appreciate our classmates for their teamwork, collaboration, and willingness to share knowledge throughout this journey.

This project was a collaborative effort, and we would like to acknowledge the specific roles undertaken by each group member:

- **Nadi Linn** led system design, Gantt chart planning, project timelines, and documentation.

- **Khine Zin Nyunt** managed the main menu and module integration as well as testing and debugging.

- **Khin Oo Thi Han** implemented search and delete functionalities.

- **Khine Khine Oo** developed the insert and edit record modules.

- **Kay Zin Thant** handled result display formatting and rank calculations.

- All members contributed to designing the overall data structures and project planning.

ABSTRACT

This report documents the design and implementation of a COBOL-based Student Information and Result Management System developed to manage student academic records across two semesters. The system is implemented and tested using Open COBOL IDE, providing a console-based interface that allows administrative users to add, edit, delete, view, and search student data stored in sequential files.

The application is modular in structure, consisting of a main control program and specialized subprograms for specific operations such as data insertion, deletion, searching, and report generation. All interactions are performed through a text-based interface using COBOL's ACCEPT and DISPLAY statements. While the system is optimized for moderate-sized datasets due to its use of sequential file processing and in-memory sorting, it ensures data integrity and reliability through safe file handling techniques. This system demonstrates effective academic data management within a traditional COBOL environment, supported by a modern development IDE.

CHAPTER 1

INTRODUCTION

Managing student academic records efficiently is essential for educational institutions, especially when dealing with multiple semesters and continuous assessment. This project presents a Student Information and Result Management System developed in **COBOL**, aimed at simplifying record-keeping tasks such as storing, updating, and reporting student performance data.

The system was built and tested using **OpenCOBOL IDE**, which provides a modern environment for developing traditional COBOL applications. Despite the use of a graphical development interface, the application itself remains **console-based**, interacting with users through a text-driven interface using COBOL's ACCEPT and DISPLAY statements.

Designed for two semesters, the system enables administrative users to manage student records stored in sequential files. It supports fundamental operations such as adding new records, editing or deleting existing entries, searching by student ID, and generating reports sorted by total marks. Each of these tasks is handled through modular COBOL subprograms, making the system organized and maintainable.

This report outlines the system's design, file structures, program flow, and implementation details. It also highlights the limitations of using sequential files and how temporary files are utilized to ensure data consistency. Through this project, the flexibility of COBOL in handling data-processing tasks is demonstrated within a modern IDE setup.

CHAPTER 2

SYSTEM OVERVIEW

The Student Information and Result Management System is a **modular, console-based application** developed in COBOL using the **OpenCOBOL IDE**. It is designed to manage academic records for students across **two semesters**, supporting essential operations such as data entry, record maintenance, and performance reporting.

The system interacts with the user via a simple text interface, where users can navigate through a menu-driven structure to perform operations like adding student data, updating marks, deleting records, and generating reports. Input and output are handled using COBOL's ACCEPT and DISPLAY statements, making the system lightweight and platform-independent.

Key Features

- **Student Registration**: Register a new student by entering their ID and name.
- **Add New Record**: Input student ID, name, and marks for Semester 1 and 2.
- **Search Record**: Retrieve a student's information using their unique ID.
- **Update Record**: Modify marks or personal information based on student ID.
- **Delete Record**: Remove a student's data from the system.
- **Generate Report**: Display all student records sorted by total marks per Semester, including grade and sorted by rank list.

Records are stored in **line-sequential files**:

- **"student_sem1.dat"** contains Semester 1 data
- **"student_sem2.dat"** contains Semester 2 data

To ensure safe updates and deletions, temporary files such as **“temp_sem1.dat”** and **“temp_sem2.dat”** are used during file rewriting processes, helping to maintain data integrity and prevent accidental loss

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

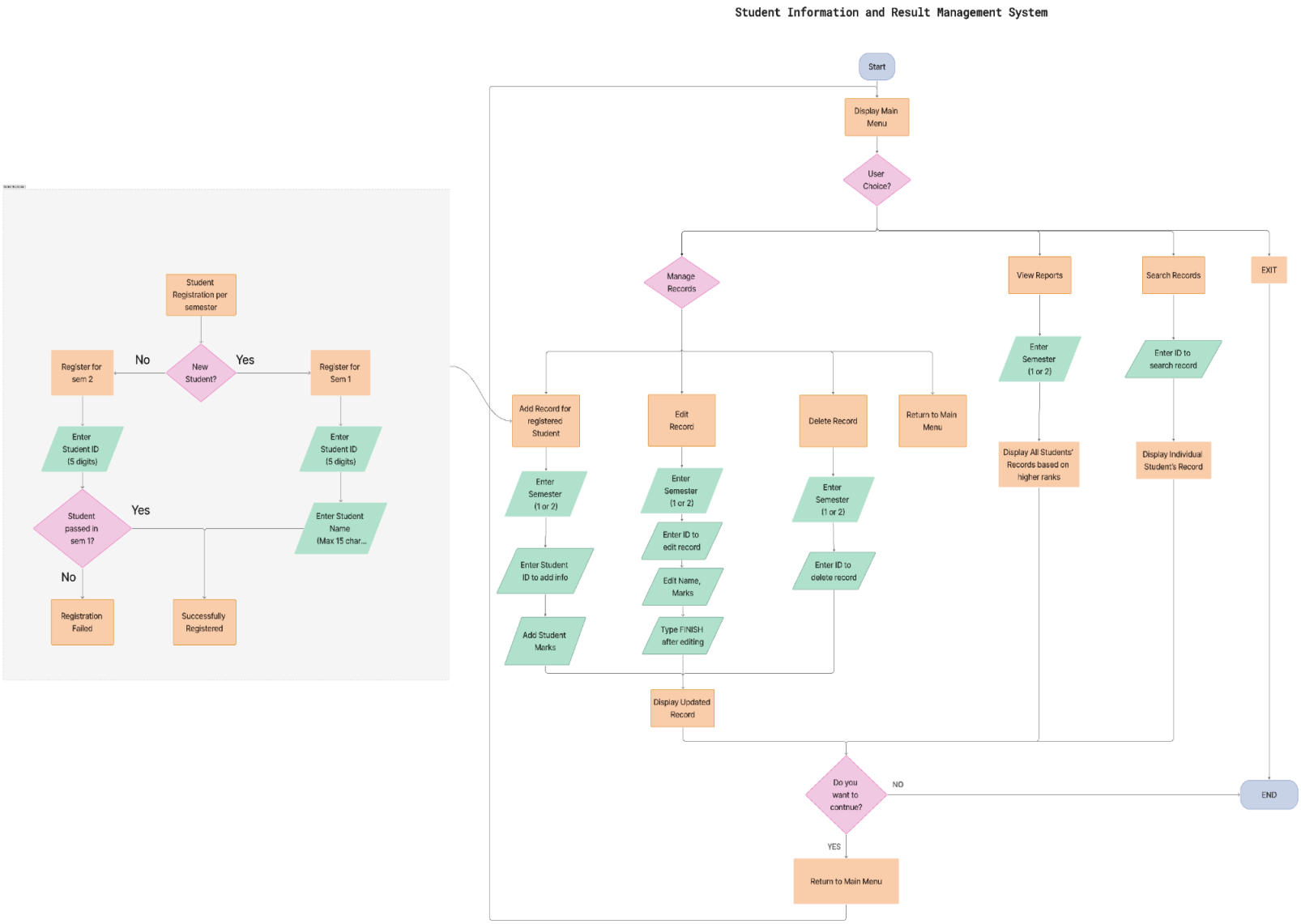

Figure 3.1. System Design

The flowchart above visually represents the overall workflow of the Student Information and Result Management System. It illustrates how the system handles user interaction, starting from the main menu and proceeding through key operations such as adding, editing, deleting, searching records, and generating reports.

Key Workflow Steps

- **Start & Display Main Menu**

  The system begins by displaying the main menu, offering options such as Manage Records, View Reports, Search Record, and Exit.
- **User Input and Validation**

  The user selects an option by entering the corresponding number. Input validation ensures that only valid selections proceed.
- **Manage Records**

  If the user chooses to manage records, a submenu is displayed allowing addition, editing, or deletion of student records. Each action invokes a respective subprogram.
- **Add Record**

  Allows the entry of student marks for a selected semester, but only for students who have already been registered. This ensures that result entries are securely tied to valid student records and prevents accidental creation of unknown or duplicate entries
- **Edit Record**

  Accepts a student ID, allows modification of data, and updates the records accordingly and displays updated records .
- **Delete Record**

  Receives a student ID, deletes the matching record using a temporary file to maintain data integrity.
- **View Reports**

  Loads student data, calculates totals and grades, sorts by total marks, ranks students, displays the report, and saves ranked data.

  If the students have the same ranks, it has to be ordered by student ID.

- **Search Record**

  Prompts for student ID and semester, searches the ranked results, and displays the individual student's details if found.

- **Exit**

  Terminates the program based on user request.

The system loops back to the main menu after completing any operation unless the user chooses to exit.

Temporary files play a crucial role in safely handling insertions and deletions.

The flowchart helps to understand the modular structure and sequential execution

CHAPTER 4

DETAILED DESCRIPTION OF CORE COMPONENTS

4.1. REGISTRATION PROCESS (REGISTRATION.cbl)

The student registration system enables the structured enrollment of students for Semester 1 and Semester 2. In Semester 1, students are registered by entering a unique 5-digit ID and name. The system checks for duplicates before saving the record with default subject marks and grade as shown in Figure 4.1. For Semester 2, registration is only allowed if the student has completed Semester 1 with a passing grade (A, B, or C) and is not already registered in Semester 2 as shown in Figure 4.1. Upon validation, the student's details are added, sorted, and saved, and a ranked report is generated to display all registered students by total marks and rank. This ensures proper academic flow and prevents duplicate or ineligible entries.

```
Select semester to register for:
1. SEM1 (New registration)
2. SEM2
Enter your choice (1 or 2):
1
Enter Student ID (5 digits):
00026
Enter Student Name:
H
Registration for SEM1 completed for ID: 00026

Process finished with exit code 0
```

```
Select semester to register for:
1. SEM1 (New registration)
2. SEM2
Enter your choice (1 or 2):
2
Enter Student ID (5 digits):
00028
Student ID 00028 not found in SEM1 records.

Process finished with exit code 0
```

```
Select semester to register for:
1. SEM1 (New registration)
2. SEM2
Enter your choice (1 or 2):
2
Enter Student ID (5 digits):
00026
Registration Failed for: Henry
Reason: Grade NA is not passing
Student ID 00026 not found in SEM1 records.
```

Figure 4.1. Student Registration

4.2. CONTROL PROGRAM (MAINFRAME.cbl)

The main control program, MAINFRAME.cbl, serves as the entry point of the system. It presents a menu-driven interface that allows users to choose operations such as adding, editing, deleting, viewing, or searching records. The interface, as shown in Figure 4.1, is built using COBOL's DISPLAY and ACCEPT statements, ensuring simple and accessible interaction. The program handles user input validation to avoid invalid selections and directs the workflow by calling relevant subprograms based on user choices. Submenus are available for deeper operations like editing or deleting specific records. The main program ensures smooth navigation and exits cleanly upon user request.

4.3. RECORD INSERTION (INSERT.cbl)

The INSERT.cbl module is responsible for collecting and storing student information. After selecting the semester, users are prompted to input the number of entries. The system generates unique student IDs by reading the existing data and incrementing the highest ID found. Students' names and six subject marks are collected with strict validation to ensure data accuracy. The program then calculates total marks and assigns grades based on predefined thresholds (A, B, C, or F). Records are temporarily written to a staging file before being appended to the main semester data file, thereby preventing data corruption. This process is illustrated in Figure 4.2.

```
**********************************
*Student Record Management System*
**********************************
1. Manage Records
2. View Reports
3. Search Record
4. Exit
Enter your choice (1-4):
1
------------------------------------------------------------
Manage Records Menu
1. Add Record
2. Edit Record
3. Delete Record
4. Back to Main Menu
Enter Manage Option (1-4):
1
Select Semester (1 or 2):
1
Enter Student ID to search:
00026
Student found: Henry
Enter mark for CST11101   (0 to 100):
10
Enter mark for CST11201   (0 to 100):
10
Enter mark for CST11401   (0 to 100):
```

Figure 4.2. Main Menu & Record Insertion

4.4. RECORD EDITING (EDIT.cbl)

The EDIT.cbl module provides functionality to update existing student records. Upon entering a student ID and selecting the semester, the system locates the corresponding record and displays editable fields. Users can selectively modify fields such as the student's name or individual subject marks. Each new value is validated to ensure it meets expected input criteria. Once the user types "Finish", the program recalculates total marks and updates the student's grade accordingly. All records, including the modified one, are written to a temporary file and later used to replace the original data file, preserving data integrity. Figure 4.3 demonstrates the record editing process.

```
Enter Student ID to Edit:
00024
Student Found: Nadi Lin
Which field do you want to edit?(NAME,SUB1 to 6)
Or type FINISH to end editing.
NAME
Editing Started.
Enter new name (max 15 characters):
Lin
Which field do you want to edit?(NAME,SUB1 to 6)
Or type FINISH to end editing.
finish
Finished editing.
===== Edited Record =====
ID: 00024
Name: Lin
Marks:
  CST11101: 056
  CST11201: 056
  CST11401: 056
  CST11501: 056
  CST11601: 056
  CST11701: 056
Total: 336
Grade: C
==========================
Updating ranked results...
SEMESTER I
=============================================================================================
StudentID   Name                          1101 1201 1401 1501 1601 1701  Total  Grade  Rank
=============================================================================================
```

Figure 4.4. Record Editing

4.5. RECORD DELETION (DELETE-RECORD.cbl)

The deletion process is handled by DELETE-RECORD.cbl. This subprogram prompts the user to enter a student ID and the semester of the record to be deleted. It reads through the corresponding data file and copies all records—except the one marked for deletion—into a temporary file. After successful verification, the original file is deleted and replaced by the temporary one. This ensures that only the targeted record is removed while maintaining the consistency of the remaining data. A status flag is returned to indicate the success or failure of the deletion. Figure 4.4 outlines the deletion workflow.

```
------------------------------------------------------
Manage Records Menu
1. Add Record
2. Edit Record
3. Delete Record
Enter Manage Option (1-3):
3
Select Semester (1 or 2):
1
Enter Student ID to Delete:
00026
Record deleted successfully.
SEMESTER I
=============================================================================================
StudentID   Name                          1101 1201 1401 1501 1601 1701  Total  Grade  Rank
=============================================================================================
00004       Diana Miller                   96   95   97   98   99   94    579    A      1
00009       Ian Gray                       93   95   94   96   92   91    561    A      2
00016       Paige Foster                   92   93   94   95   90   91    555    A      3
00007       George White                   91   90   92   93   94   90    550    A      4
00013       Michael Carter                 88   87   89   90   91   88    533    A      5
```

Figure 4.5. Record Deletion

4.6. RECORD SEARCH (SEARCH-RECORD.cbl)

The SEARCH-RECORD.cbl module enables users to retrieve individual student information using a unique student ID. The program searches through pre-ranked result files, which contain already sorted data by total marks. If a matching record is found, it displays all relevant details including subject marks, total score, grade, and rank. If no match is found, the system informs the user accordingly. The process is straightforward and optimized for quick look-up, as shown in Figure 4.5.

```
------------------------------------------------------
Student Management System
1. Manage Records
2. View Reports
3. Search Record
4. Exit
Enter your choice (1-4):
3
Enter Student ID to Search:
00026
Select Semester (1 or 2):
1
SEARCH RESULT - SEMESTER I
=============================================================================================
StudentID   Name                          1101 1201 1401 1501 1601 1701  Total  Grade  Rank
=============================================================================================
00026       Ashley                         34   44   55   66   77   77    353     C     21
Do you want to continue? (Y/N):
```

Figure 4.6. Search and Display Individual's record

4.7. DISPLAY REPORT MODULE (VIEW.cbl)

The VIEW.cbl subprogram handles the generation and display of full student performance reports for each semester. It loads all records into memory after validating their structure and content. Total marks are calculated, and grades are assigned for each student. The module then applies a Bubble Sort algorithm to rank students in descending order based on their total scores. Ranked reports are presented on the console and simultaneously written to output files for future searches and recordkeeping. This module forms the basis for viewing overall performance, as seen in Figure 4.6.

```
Student Management System
1. Manage Records
2. View Reports
3. Search Record
4. Exit
Enter your choice (1-4):
2
Select Semester (1 or 2):
1
SEMESTER I
```

StudentID	Name	1101	1201	1401	1501	1601	1701	Total	Grade	Rank
00004	Diana Miller	96	95	97	98	99	94	579	A	1
00009	Ian Gray	93	95	94	96	92	91	561	A	2
00016	Paige Foster	92	93	94	95	90	91	555	A	3
00007	George White	91	90	92	93	94	90	550	A	4
00013	Michael Carter	88	87	89	90	91	88	533	A	5
00002	Bob Smith	88	90	87	89	90	88	532	A	6
00018	Rachel Irwin	87	89	88	86	85	87	522	A	7
00015	Oliver Evans	85	86	87	84	82	83	507	A	8
00008	Hannah Black	84	83	82	85	80	81	495	A	9
00011	Kevin Adams	82	84	83	80	81	85	495	A	10
00005	Ethan Brown	80	81	79	82	83	80	485	A	11
00001	Alice Johnson	75	78	80	82	79	80	474	A	12
00022	kay zin thant	78	78	78	78	78	78	468	A	13
00014	Natalie Davis	77	78	76	75	77	74	457	A	14
00017	Quentin Harris	75	74	76	78	77	75	455	A	15
00012	Laura Bennett	78	76	75	74	72	73	448	B	16
00019	Samuel James	70	69	71	73	72	74	429	B	17
00003	Charlie Lee	69	71	73	70	72	74	429	B	18
00006	Fiona Green	70	72	68	69	71	70	420	B	19
00021	khin oo thi han	67	67	67	67	67	67	402	B	20
00026	Ashley	34	44	55	66	77	77	353	C	21
00024	Nadi Lin	56	56	56	56	56	56	336	C	22
00025	Rosie	45	45	45	45	45	45	270	F	23

```
Ranked results for Semester 1 saved to ranked_results_sem1.dat
Do you want to continue? (Y/N):
```

Figure 4.7. Display All Students' Records per Semester

CHAPTER 5

DESIGN CONSIDERATIONS AND CONSTRAINTS

The design of the Student Information and Result Management System focuses on simplicity, modular structure, and compatibility with COBOL's procedural programming model. While the system successfully fulfills its core objectives, several constraints were considered during development.

The system relies on a **sequential file structure**, where student records are stored in line-sequential data files. This design is simple and portable across different systems, but it does not support direct record access or indexing. As a result, operations like searching or updating a record require reading the entire file line by line, which may become inefficient for larger datasets.

To safely handle deletion and modification of records, **temporary files** such as temp_sem1.dat and temp_sem2.dat are used. These temporary files are used to reconstruct data by copying all valid records and then replacing the original files. While this approach ensures data integrity and reduces the risk of data corruption, it increases the number of file operations and overall processing time.

Another design limitation is the use of a **Bubble Sort algorithm** for ranking student records based on total marks. Bubble Sort was chosen for its simplicity and ease of implementation in COBOL; however, it is not efficient for large datasets due to its $O(n^2)$ time complexity. This sorting method is suitable only for small to medium-sized record sets, such as those in a classroom or academic batch.

The **console-based interface**, implemented using COBOL's ACCEPT and DISPLAY statements, was selected for its wide platform compatibility and minimal resource usage. However, it lacks modern user interface elements such as graphical components or mouse support, which can limit the user experience.

Lastly, the system was designed with a **modular structure**, with each major operation (insertion, deletion, editing, searching, and viewing) implemented as a separate subprogram. This promotes code clarity, easier debugging, and future extensibility—for instance, supporting more semesters or adding export features in later versions.

CHAPTER 6

TESTING AND VALIDATION

Testing was an essential part of the system development process to ensure correctness, reliability, and user-friendliness. A variety of testing techniques were applied to cover functional, boundary, and integration aspects of the system.

Functional testing was performed on all major modules. The insertion module was tested to ensure accurate data entry, grade calculation, and the generation of unique student IDs. The deletion module was validated by removing specific records and confirming that the remaining data was correctly preserved. The search functionality was tested by querying both existing and non-existent student IDs. The view module was assessed for correct ranking, sorting, and total mark calculations.

Strong emphasis was placed on **input validation**. Marks were restricted to valid numeric ranges (0 to 100), and menu options were checked to prevent invalid selections. Student IDs were also verified for uniqueness and format consistency to avoid duplication or logical conflicts.

To test system robustness, several **edge cases** were considered. These included attempting to delete a non-existent student, searching records when the file was empty, or entering invalid characters for marks or names. The system responded correctly in all these cases, providing appropriate error messages or handling the situation gracefully.

The accuracy of **file operations** was also verified. Temporary files were created and replaced the original files only after a successful update or deletion. Records persisted accurately between program runs, and no data loss or corruption was detected during repeated insertions and deletions.

Finally, a review of the **user experience** was conducted. The menu structure was found to be intuitive, with clear instructions and feedback messages. The formatting of reports was consistent, and navigation between different modules was smooth, contributing to the overall usability of the system.

CHAPTER 7

CONCLUSIONS

The Student Information and Result Management System, developed using COBOL in the Open COBOL IDE, successfully achieves its primary goal of managing academic records in a structured and modular manner. The application allows users to perform essential operations such as adding new records, updating or deleting existing entries, generating ranked reports, and searching for student details — all through a user-friendly console interface.

Its modular design, with clearly defined subprograms, promotes maintainability and separation of concerns. The use of sequential files ensures reliable data storage and retrieval, while temporary files safeguard data integrity during critical operations like insertion and deletion.

While the system is effective for small to medium-sized datasets, certain limitations—such as the reliance on bubble sort and the lack of indexed file support—highlight areas for potential improvement. Despite these constraints, the system performs efficiently for the intended educational context and demonstrates the practical use of COBOL for real-world data management tasks.

Future Improvements

To enhance the system's performance and usability, several improvements can be considered. Replacing the current bubble sort with a more efficient algorithm, such as merge sort, would improve scalability for larger datasets. Implementing indexed file handling could significantly reduce search times compared to sequential access. Additionally, upgrading the user interface from a basic console to a screen-based or graphical interface would improve user experience. Lastly, incorporating authentication and role-based access control would enhance security, especially in multi-user environments.

LISTS OF FIGURES

REFERENCES

- *https://www.ibm.com/docs/en*
- *https://www.geeksforgeeks.org/search/?gq=cobol*
- *https://www.geeksforgeeks.org/cobol/what-is-cobolcommon-business-oriented-language/*