Student Management System Documentation

Under the guidance of
Eng.Keroles Shenouda

By

Abdelrahman El-Badawy Fawzy
January 2024

Contents

0.1 Introduction

The Student Management System (SMS) project aims to provide a simple yet efficient solution for managing student records in an educational institution. This system allows administrators and staff members to perform various tasks related to student management, such as adding new students, updating existing records, searching for specific students, and deleting student records when necessary.

In educational institutions, maintaining accurate and up-to-date student records is crucial for administrative purposes, academic planning, and communication with students and their families. Traditional paper-based methods of student record management are often cumbersome, time-consuming, and prone to errors. Therefore, an automated system like the SMS project can streamline the process, improve data accuracy, and enhance overall efficiency.

The SMS project is implemented in the C programming language, providing a lightweight and platform-independent solution that can be easily integrated into existing systems or used as a standalone application. It leverages concepts of data structures, file handling, and user input/output to offer a user-friendly interface for managing student records.

This documentation provides an overview of the SMS project, including its features, implementation details, usage instructions, and future enhancements. It serves as a guide for developers, administrators, and other stakeholders involved in the deployment and maintenance of the SMS system.

0.2 Features

List and describe the features of the SMS project, such as:

- Adding a student
- Adding a student from a file
- Finding a student by roll number, first name, or course ID
- Deleting a student by roll number
- Updating student information by roll number
- Showing all students

0.3 Implementation

The Student Management System (SMS) project is implemented in C programming language. Below are the main components and implementation details:

0.3.1 Main Function

The `main()` function serves as the entry point of the program. It initializes the FIFO buffer, which acts as a data structure to store student records. It then enters a loop to present a menu of options to the user and executes the corresponding functionality based on the user's choice.

0.3.2 Student Structure

The `Student_t` structure represents a student record. It contains fields such as roll number, first name, last name, course ID, and any other relevant information about a student.

0.3.3 FIFO Buffer

The FIFO buffer, represented by the `FIFO_BUF_t` structure, is used to store the student records. It is implemented as a circular buffer to efficiently manage memory and allow for constant-time insertion and deletion operations.

0.3.4 Functions

Various functions are implemented to perform operations on the student records stored in the FIFO buffer. These functions include:

- `AddStudent()`: Adds a new student record to the FIFO buffer.
- `AddStudentFromFile()`: Adds student records from a file to the FIFO buffer.
- `searchByRollNumber()`: Finds a student record by roll number.

- `FindStudentByFirstName()`: Finds a student record by first name.
- `FindStudentByCourseId()`: Finds a student record by course ID.
- `FindTotalNumberOfStudents()`: Counts the total number of students in the FIFO buffer.
- `DeleteStudentByRollNumber()`: Deletes a student record by roll number.
- `UpdateStudentByRollNumber()`: Updates a student record by roll number.
- `ShowAllStudents()`: Displays all student records stored in the FIFO buffer.

These functions are called from the `main()` function based on the user's choice from the menu.

0.4 Dependencies

The Student Management System (SMS) project relies on the following standard C libraries for its functionality:

- `stdio.h`: This header file provides input and output functionality, including functions for reading from and writing to files, as well as standard input/output streams like `printf()` and `scanf()`.
- `stdint.h`: This header file defines a set of standard integer types with specified widths, such as `int8_t`, `uint16_t`, etc. These types ensure consistent behavior across different platforms and compilers.
- `string.h`: This header file provides various string manipulation functions, such as `strcpy()`, `strcat()`, and `strlen()`. These functions are used for working with character arrays and strings in the SMS project.

These standard C libraries are essential for implementing core functionality and handling input/output operations in the SMS project. They are widely supported and available on most C compilers and platforms.

0.5 Code Documentation

The code for the Student Management System project can be found in the GitHub repository at the following link:

GitHub Repository

Below is an overview of the main components and functions:

0.5.1 proj2.c

The `proj2.c` file contains the main function of the program. It serves as the entry point and orchestrates the execution flow of the Student Management System. Key functions and operations performed in this file include:

- Initialization of the FIFO buffer.
- Presentation of a menu to the user for selecting different tasks.
- Invocation of functions based on the user's choice to perform various operations like adding, searching, updating, or deleting student records.

0.5.2 Student.h

The `Student.h` header file defines the structure `Student_t` representing a student record. It includes fields such as roll number, first name, last name, course ID, etc. Functions related to student operations may also be declared in this file.

0.5.3 FIFO.h

The `FIFO.h` header file declares the FIFO (First In, First Out) buffer data structure used for storing student records. It may include function prototypes for initializing the FIFO buffer, adding elements, deleting elements, and other operations.

0.5.4 Other Files

Review other files in the repository, such as `Student.c` and `FIFO.c`, which likely contain the implementations of functions declared in the corresponding header files. Document the purpose and functionality of each function implemented in these files.

This documentation provides an overview of the main components and functionality of the Student Management System project. For detailed documentation of individual functions and their parameters, refer to the source code files.

0.6 Images

0.6.1 Add Student Function

```
Welcome To Student Management System
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 1
Enter the student's Roll number:
13
Enter the student's first Name:
Abdoo
Enter the student's last Name:
badawy
Enter the student's GPA:
3.9
Enter the student's registered course number 1:
3
Enter the student's registered course number 2:
4
Enter the student's registered course number 3:
5
Enter the student's registered course number 4:
3
Enter the student's registered course number 5:
2
Students Added Successfully
```

Figure 1: Screenshot of the AddStudent function in action.

0.6.2 AddStudentFromFile Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 2
Students Added Successfully
```

Figure 2: Screenshot of the AddStudentFromFile function in action.

0.6.3 searchByRollNumber Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 3
Please Enter Student Roll Number:
13
Student Found
Student Roll Number: 13
Student First Name: Abdoo
Student Last Name: badawy
Student GPA: 1073741824
Student Course 1 ID: 3
Student Course 2 ID: 4
Student Course 3 ID: 5
Student Course 4 ID: 3
Student Course 5 ID: 2
```

Figure 3: Screenshot of the searchByRollNumber function in action.

0.6.4 FindStudentByFirstName Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 4
Please Enter Student First Name: Abdoo
Student Found
Student Roll Number: 13
Student First Name: Abdoo
Student Last Name: badawy
Student GPA: 3.900000
Student Course 1 ID: 3
Student Course 2 ID: 4
Student Course 3 ID: 5
Student Course 4 ID: 3
Student Course 5 ID: 2
```

Figure 4: Screenshot of the FindStudentByFirstName function in action.

0.6.5 FindStudentByCourseId Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 5
Please Enter Student Course ID:
12
Student Found
Student Roll Number: 14352
Student First Name: Laila
Student Last Name: ElBadawy
Student GPA: 1.000000
Student Course 1 ID: 12
Student Course 2 ID: 10
Student Course 3 ID: 2
Student Course 4 ID: 53
Student Course 5 ID: 2333
```

Figure 5: Screenshot of the FindStudentByCourseId function in action.

0.6.6 FindTotalNumberOfStudents Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 6
Total Number Of Students: 2
```

Figure 6: Screenshot of the FindTotalNumberOfStudents function in action.

0.6.7 DeleteStudentByRollNumber Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 7
Please Enter Student Roll Number:
13
Student Deleted Successfully
```

Figure 7: Screenshot of the DeleteStudentByRollNumber function in action.

0.6.8 UpdateStudentByRollNumber Function

```
proj2.exe [C/C++ Application]  [pid: 366]
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 8
Please Enter Student Roll Number:
14352
Student Found
Student Roll Number: 14352
Student First Name: Laila
Student Last Name: ElBadawy
Student GPA: 1.000000

--------------------------------------------------------------------------------
Student Course 1 ID: 12
Student Course 2 ID: 10
Student Course 3 ID: 2
Student Course 4 ID: 53
Student Course 5 ID: 2333

--------------------------------------------------------------------------------
Enter the student's first Name:
lyly
Enter the student's last Name:
badawy
Enter the student's GPA:
3.55
Please Enter Student Course 1 ID: 3
Please Enter Student Course 2 ID: 5
Please Enter Student Course 3 ID: 6
Please Enter Student Course 4 ID: 3
Please Enter Student Course 5 ID: 4
```

Figure 8: Screenshot of the UpdateStudentByRollNumber function in action.

0.7 ShowAllStudents Function

```
Please choose the task
1. Add Student
2. Add Student From File
3. Find Student By roll Number
4. Find Student By First Name
5. Find Student By Course Id
6. Find Total Number Of Students
7. Delete Student By Roll Number
8. Update Student By Roll Number
9. Show All Students
10. Exit
Your Choice: 9
FIFO elements:
Roll Number :14352 ||First Name :Laila ||Last Name :ElBadawy ||GPA :1.000000 ||Course 1 ID :12 ||Course 2 ID :10 ||Course 3 ID
```

Figure 9: Screenshot of the ShowAllStudents function in action.

0.8 Conclusion

In conclusion, the Student Management System (SMS) project provides a robust solution for managing student records in educational institutions. By leveraging the C programming language and implementing efficient data structures and algorithms, the SMS system offers administrators and staff members a user-friendly interface to perform various tasks related to student management.

Throughout the development of the SMS project, key objectives have been achieved, including:

- Streamlining the process of adding, updating, searching, and deleting student records.
- Enhancing data accuracy and reliability by utilizing appropriate data structures like FIFO buffers.
- Improving user interaction through a menu-based interface, ensuring ease of use and accessibility.
- Providing a scalable and maintainable solution that can be adapted to the evolving needs of educational institutions.

Moving forward, there are opportunities for further enhancements and optimizations to the SMS project. These may include:

- Implementing additional features such as student attendance tracking, grade management, or course scheduling.
- Enhancing error handling and validation mechanisms to ensure data integrity and security.
- Optimizing performance and memory usage for larger datasets or high-traffic scenarios.
- Incorporating user feedback and iterative improvements to refine the user experience.

Overall, the SMS project represents a significant step towards modernizing student record management systems and facilitating efficient administrative processes in educational institutions. By embracing innovation and continuous improvement, the SMS system has the potential to make a positive impact on the education sector and contribute to the success of students, faculty, and staff alike.