* 

**CSE103: Structured Programming**

**[Spring 2023]**

**Project Report**

**School Management System**

**Course Code : CSE103**

**Course Title : Structured Programming**

**Section : 07**

**Group Number : 03**

**Submitted by:**

| **Student ID** | **Student Name** | **Contribution Percentage** |
| --- | --- | --- |
| 2023-1-60-043 | Sihab Bin Sarwer | 25% |
| 2023-1-60-045 | Maksudul Hasan Limon | 25% |
| 2023-1-60-046 | Redoanul Alam Mazumder | 25% |
| 2023-1-60-047 | Nadim Srabon | 25% |

1.Introduction

School Management System

The School Management System project aims to create a software solution that makes it easier to manage different tasks in a school, such as administrative work and academic processes. The system provides a digital platform to efficiently manage student records, teachers’ information and class routine involved in running a school.

Overview: To access the school management system program, the user must enter a specific user ID and password. Once the correct user ID and password are entered, the user will gain access to both the student module and the teacher module. The program utilizes a switch statement to create various options, each of which is implemented as a function. The functions within the program serve specific purposes and are described below.

A screen shot of a black screen

Description automatically generated with low confidence

A screenshot of a computer program

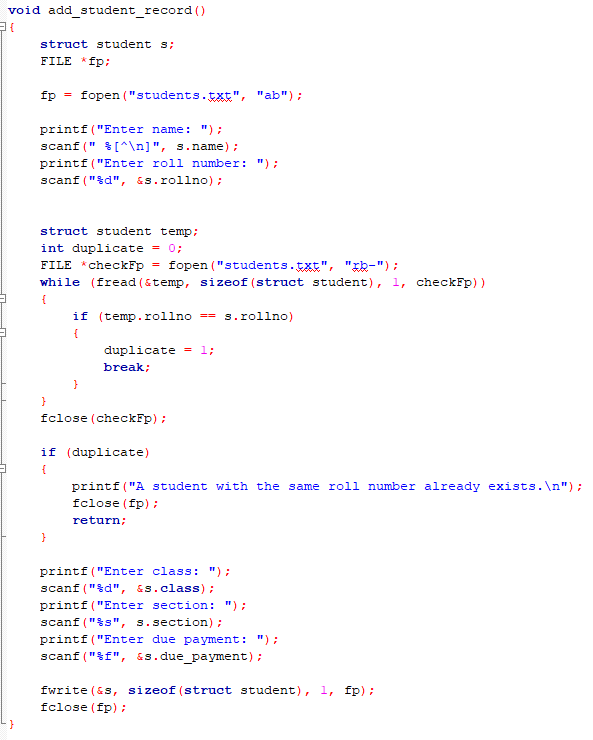
Description automatically generated with medium confidence

1. All function descriptions with output

**Function 1 & 6:**

A student's record is added to a file called "students.txt" using the function add\_student\_record(). The user is prompted to input the student's name, roll number, class, section, and due amount owed when using the function. To prevent duplicates, it then determines if a student with the same roll number already exists in the file.

There are no input parameters for the function. Through the console, it communicates with the user and ask input. This function adds the student's record to the "students.txt" file as its output.

A screenshot of a computer program

Description automatically generated with medium confidence

A teacher's record is added to a file called "teachers.txt" using function number two, add\_teacher\_record(). The user is prompted to enter the teacher's name, subject, degree, and ID number in the function. Additionally, it analyzes the file for duplicate ID numbers.

**Output:** A screen shot of a computer

Description automatically generated with medium confidenceA screen shot of a computer

Description automatically generated with low confidenceA black screen with white text

Description automatically generated with low confidence

A black background with white text

Description automatically generated with low confidence

**Function 2 & 7:**

**A picture containing text, screenshot, font, line

Description automatically generated**

The function viewAllStudentRecord() is used to display all the student records form the “student.txt” file. Similar to the first method, it opens the file in binary mode for reading and uses the fread() function to read each student's data from the file. Then, each student's details, including name, roll number, section, class, and amount owed, are printed. The function then carries on to the following iteration to read the data for the following student after outputting the information. It closes the file at the end. The "students.txt" file's contents, including information on every student, are printed by the function.

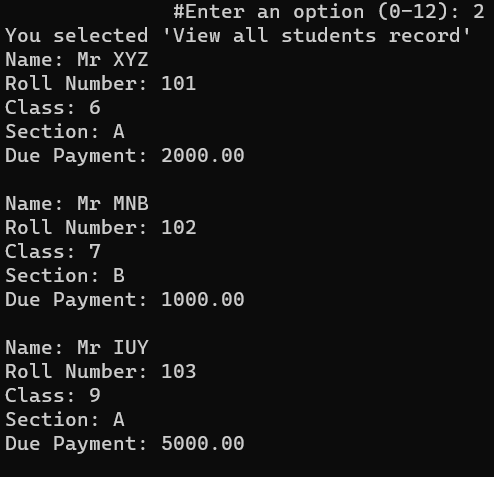
A picture containing text, line, font, screenshot

Description automatically generated

The function viewAllTeacherRecord() is used to display all the teacher records from the "teachers.txt" file. The fread() function is used to read the data for each teacher from the file after opening it in binary mode for reading. The information for each instructor, including their name, subject, degree, and ID number, is then printed. The function then clears the data in the struct variable "s1" to make room for the following iteration after publishing the information. It closes the file at the end. The function displays all of the teacher's data that was saved in the "teachers.txt" file.

**Output:**

A picture containing text, screenshot, font

Description automatically generated

**Function 3 & 8:**

The function, search\_student\_by\_roll() reads a user-inputted roll number, opens a file called "students.txt" in binary mode, and then looks for a matching roll number in the file. In the event that a match is made, it prints the student's information (name, roll number, class, section, and amount owed) and then exits. It prints a "not found" message if no match is discovered.

A screenshot of a computer code

Description automatically generated with low confidence

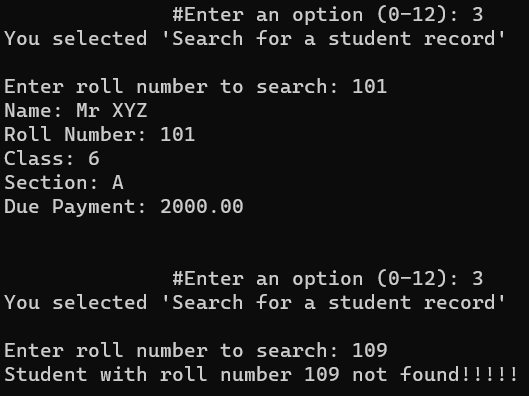
A picture containing text, screenshot, font

Description automatically generated

And, search\_teacher\_by\_id() it reads an ID number provided from the user, opens a file called "teachers.txt" in binary mode, and then looks for a matching ID number in the file. In the event that a match is made, it prints the teacher's information (name, subject, degree, and ID number) and then goes back. In the absence of a match, it prints “not found” option.

Both functions use file handling operations to read data from binary files. They iterate through the records in the file using a loop and compare the search key (roll number for students and ID number for teachers) with each record in the file. If a match is found, the corresponding details are printed, and the file is closed.

**Output:**

A picture containing text, screenshot, font

Description automatically generated

**Function 4 & 9:**

**A screenshot of a computer code

Description automatically generated with medium confidence**

The "students.txt" file, which contains student records, can be edited with this function void edit(). The user is prompted to input the student's roll number whose record has to be changed. The file is then opened in "rb+" mode, allowing both reading and writing.

The function uses fread() to read each record from the file, then compares the roll number to the user's input. If a match is made, the system displays the student's current information and asks the user to submit new data.

The function uses fseek() to shift the file pointer to the start of the current record after receiving the new data, and fwrite() to replace the old record with the new data. The notice that the record has been successfully changed is then displayed.

The user enters the roll number of the student who has to be edited into this function using scanf(). The result is a message stating whether or not the record was successfully found and modified.

A screenshot of a computer program

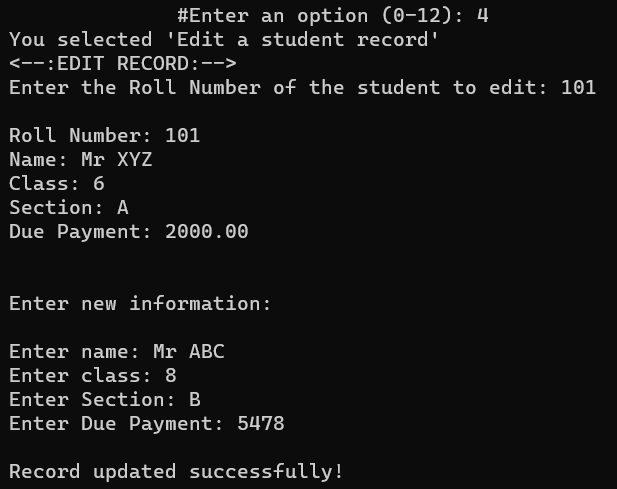
Description automatically generated with medium confidence

The "teachers.txt" file's teacher records can be edited using this function void edit\_teacher\_record(). It requires the user to input the ID of the teacher whose record has to be modified, just like edit() does. In order to facilitate reading and writing, it additionally opens the file in "rb+" mode.

Each teacher record is read from the file by the function, which then checks the ID against the user's input. If a match is made, the system displays the teacher's current information and asks the user to submit new information.

The function uses fseek() to shift the file pointer to the start of the current record after receiving the new data and fwrite() to update the record with the new data. The notice that the record has been successfully changed is then displayed.

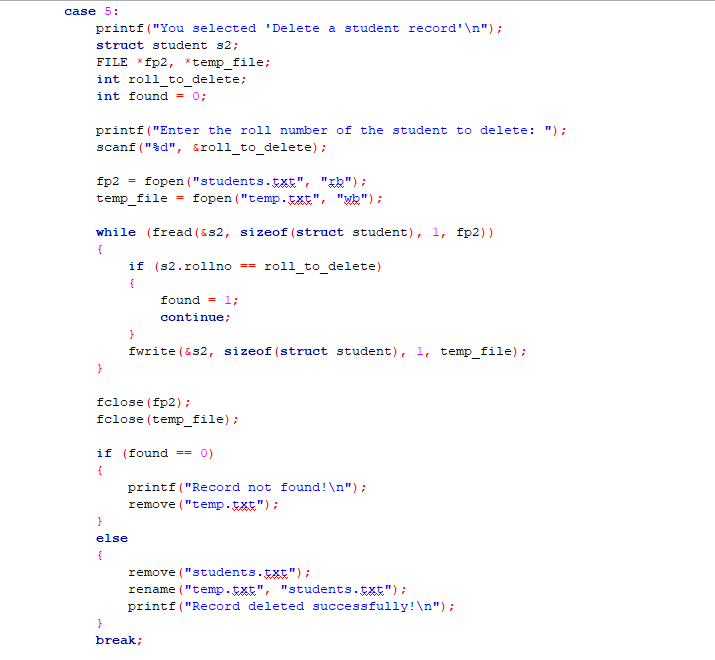
The ID of the teacher to be edited is the input to this function and is supplied by the user through scanf(). The result is a message stating whether or not the record was successfully found and modified.

**Output:**

A screen shot of a computer

Description automatically generated with medium confidence

**Function 5 & 10:**



Using this function, a student record can be removed from the "students.txt" file. The user is prompted to erase the student's roll number. It reads each student record from the input file, copies all records to a temporary file, except for the one that needs to be erased. The desired record is then effectively deleted after the original file has been replaced with the temporary file. If the record cannot be located, it deletes the temporary file and displays the relevant notice.

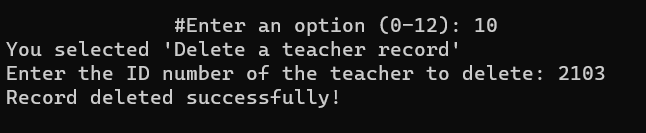
Based on the supplied ID number, this method removes a teacher entry from the "teachers.txt" file. Each teacher record is read from the file, and a check is made to see if the ID number matches the one that has to be erased. In the event that a match is discovered, it forgoes copying the record to a temporary file, thereby erasing it from the original file.

The input file and temporary file are closed once all the records have been processed. The temporary file is deleted and a message is displayed if no matching records are discovered. When a record is erased, the temporary file updates the file by taking the place of the original file. The notice then indicates whether or not the record was successfully removed.

A screenshot of a computer program

Description automatically generated with low confidence

**Output:**

A screenshot of a computer

Description automatically generated with medium confidence

**Lacking of Our Program:**

1. The lack of a password change feature within our program poses a significant limitation to user convenience and security. As it stands, users are unable to modify their passwords directly through the program's user interface. This limitation necessitates manual intervention by modifying the source code, which is not a user-friendly solution. Incorporating a dedicated password change functionality within the program would enhance user experience and ensure better adherence to security best practices. By providing a user-friendly interface for password modification, we can empower users to manage their account security effectively, without the need for technical intervention.
2. Our program's alignment and appearance can be improved to create a more visually appealing and cohesive presentation. By focusing on design elements and implementing a consistent and intuitive user interface, we can enhance usability and navigation. These improvements will contribute to a positive user experience, increasing engagement and satisfaction with our C code-based program.
3. The current source code of our program lacks input validation functionality, which is crucial for verifying the validity and appropriateness of user inputs. Implementing input validation would enable us to handle erroneous or malicious inputs effectively, enhancing the reliability and security of our program. By incorporating robust validation mechanisms, we can prevent common issues such as data corruption, program crashes, and potential vulnerabilities. Furthermore, proper input validation will contribute to a more seamless and error-free user experience, promoting overall program stability and user satisfaction.

**Future Implementation:**

1. By implementing input validation, we can ensure that users enter accurate and valid data. This validation process will help us prevent erroneous or invalid inputs from affecting the functionality and integrity of our program. It will enable us to enforce specific criteria or rules for data entry, reducing the risk of errors, inconsistencies, or unexpected behaviors. With input validation in place, our program can handle user input more effectively and provide a smoother user experience.
2. Improving error handling through the implementation of suitable error messages and exception handling to enhance user experience and program stability.
3. Enhancing the search functionality by introducing advanced options such as searching by member name and ID, which will provide users with more flexibility and accuracy in retrieving desired information.
4. Adding sorting features to the program that allow users to arrange the project list based on various attributes such as date, priority, or alphabetical order. This will provide users with the ability to organize and navigate the project list more efficiently, improving overall usability and user experience.
5. Implementing user authentication and access control mechanisms to ensure that only authorized users have access to the program and specific operations. By incorporating user authentication, the program will require users to provide valid credentials before gaining access, thereby enhancing security and preventing unauthorized usage. Access control features will further enable administrators to define user roles and permissions, restricting certain operations to authorized individuals and maintaining data integrity and confidentiality.

**Conclusion:** The process of finishing this project has been immensely enjoyable and rewarding. We were given the chance to learn more about project management systems and put our C programming talents to the test. We learned useful lessons about managing data structures, performing file I/O operations, and creating menu-driven programs during the process.  
  
We faced and overcame a number of obstacles when executing the project, mainly in file processing and functionality. We may use the lessons we learned from these obstacles to make futureimprovements.

The knowledge and abilities we have gained will unquestionably be extremely helpful in our future undertakings.

We are really proud of the time and effort we have put into this project as we complete it. It has greatly improved our programming skills and expanded our knowledge of the C programming language. We feel accomplished and eager for the upcoming programming challenges as a result of this experience.