PROJECT REPORT

STUDENT RECORD MANAGEMENT SYSTEM (SRMS)

DONE BY

NAME: SRINJOY ADHIKARI

REGISTRATION NUMBER: AP24122230017

CSE 2ND YEAR IIIrd SEMESTER

MTech Integrated SECTION A

SUBMITTED TO

RAKESH RAMA RAJU

CODING SKILLS – I (CSE 201)

ABSTRACT

This project presents a Student Record Management System developed in the C programming language to offer a reliable, secure and easy-to-use solution for managing student information. The system stores key details such as roll number, name and marks in text files, ensuring that all data is safely preserved even after the program closes. It provides the essential operations required for effective record management, including adding new entries, viewing stored data, searching for specific students, updating information and deleting records when necessary.

A role-based login system forms the foundation of its security and usability. Users can log in as Admin, Staff, Guest or User and each role is granted only the level of access needed for its responsibilities. Admin users can perform all operations, staff users can view, update and search records, guest users are limited to display and searching information and user is limited to only viewing information. This structured approach not only protects the system from unauthorized modifications but also makes the interface simple and user-friendly for every type of user.

Overall, this project demonstrates the practical importance of core C programming concepts while showing how a straightforward application can effectively represent real-world data management.

INTRODUCTION

The Student Record Management System is a C-based application developed to efficiently manage and maintain student information using basic file-handling techniques. The system stores essential details such as roll number, name, and marks in external text files, ensuring that the data remains preserved even after the program is terminated. To ensure secure and organized access, the application incorporates a role-based login mechanism that restricts operations based on user privileges.

The system supports four distinct user roles: Admin, Staff, Guest and User. Admin users possess complete access and can add, update, delete, search, and view student records. Staff members are granted permissions to search, updating and viewing existing records, Guest users are limited to display and search the data and User is limited to only display the information. This layered permission structure promotes data integrity and prevents unauthorized modifications.

In addition to managing student data, the project demonstrates the practical application of key concepts in the C programming language, including structures, file operations, string handling, and conditional logic. Overall, the Student Management System serves as a foundational example of how real-world data management applications can be designed and implemented using fundamental programming principles.

OBJECTIVES

The main objectives of this project are:

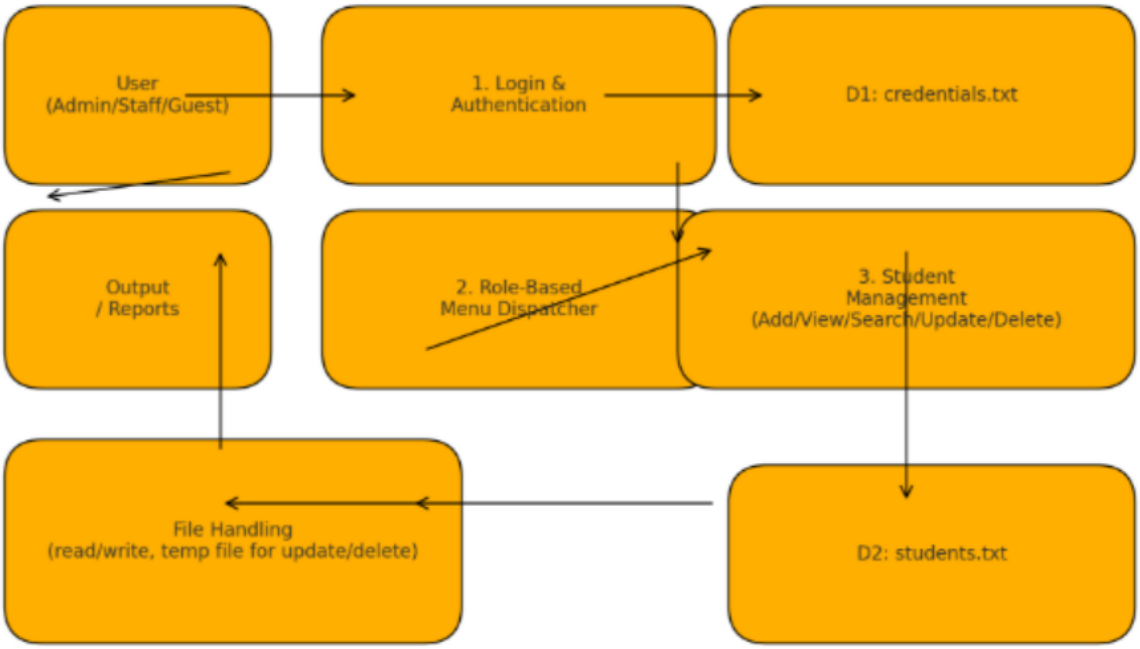
- To create an easy and efficient system for managing student information using C programming.
- To add a secure login system with different roles for Admin, Staff, Guest and User.
- To give each user type the right level of access so the system is used properly.
- To store student details in text files so the data can be saved for a long time and accessed easily.
- To allow Admin users to add, view, search, update and delete student records.
- To allow Staff users to view, update and search student information without making any changes.
- To give Guest users only display and search access to keep the data safe.
- To give User Role only display access to keep data safe.
- To show the practical use of file handling, structures and conditional statements in C.
- To protect the accuracy of the data by preventing unauthorized changes.
- To provide a simple, user-friendly, menu-based interface for smooth and easy navigation.

MODULES

The system consists of the following major modules:

1. Login Module
   - Takes username and password.
   - Checks credentials from credentials.txt.
   - Identifies role: Admin, Staff, Guest or User.
2. Role-Based Menu Module
   - Shows menu according to user role.
   - Controls access to operations for Admin, Staff, Guest and User.
3. Student Addition Module (Admin Only)
   - Allows adding new student details.
   - Saves roll number, name, and marks in students.txt.
4. Display Students Module
   - Reads all records from students.txt.
   - Displays roll number, name, and marks.
5. Search Student Module
   - Searches for a student by roll number.
   - Shows matching record.
6. Update Student Module
   - Updates student name or marks.
   - Uses a temporary file to save updated data.
7. Delete Student Module (Admin Only)
   - Deletes a student record using roll number.
   - Rewrites file without the deleted entry.
8. File Handling Module
   - Handles reading and writing of credentials.txt and students.txt.

DATA FLOW DIAGRAM



Explanation

1. Login & Authentication:
   The system checks the entered username and password with credentials.txt and identifies the user role.
2. Role-Based Menu:
   Based on the role (Admin, Staff, Guest or User), the appropriate menu is shown and the chosen option is passed to the next process.
3. Student Management:
   Handles adding, viewing, searching, updating and deleting student records. Updates and deletions use a temporary file for safe rewriting.
4. File Handling:
   All student data is stored in *students.txt*, ensuring permanent storage and reliable read/write operations.

CODE & OUTPUTS

- **students.txt**

```
2|Max|130.00
3|Eleven|149.00
```

- **credentials.txt**

```
admin admin123 ADMIN
staff staff123 STAFF
guest guest123 GUEST
user user123 USER
```

- LOGIN & ADMIN MENU

```
void adminMenu() {
    int choice;
    char buf[10];
    for (;;) {
        printf("\n--- ADMIN MENU ---\n");
        printf("1. Add Student\n");
        printf("2. Display Students\n");
        printf("3. Search Student\n");
        printf("4. Update Student\n");
        printf("5. Delete Student\n");
        printf("6. Logout\n");
        printf("Enter Choice = ");
        fgets(buf, sizeof(buf), stdin);
        choice = atoi(buf);
        if (choice == 1) addStudent();
        else if (choice == 2) displayStudents();
        else if (choice == 3) searchStudent();
        else if (choice == 4) updateStudent();
        else if (choice == 5) deleteStudent();
        else if (choice == 6) { logout(); return; }
    }
}
```

```
===== LOGIN =====
Username: admin
Password: admin123

--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice =
```

After login, the Admin Menu appears, allowing the admin to add, view, search, update, or delete student records.

➢ ADD STUDENT

```c
void addStudent() {
    struct Student *arr;
    int count;
    struct Student s;
    char buffer[50];
    loadStudents(&arr, &count);
    printf("Enter Roll: ");
    fgets(buffer, sizeof(buffer), stdin);
    sscanf(buffer, "%d", &s.roll);
    printf("Enter Name: ");
    fgets(s.name, sizeof(s.name), stdin);
    s.name[strcspn(s.name, "\n")] = 0;
    printf("Enter Marks: ");
    fgets(buffer, sizeof(buffer), stdin);
    sscanf(buffer, "%f", &s.marks);
    arr = (struct Student*)realloc(arr, sizeof(struct Student) * (count + 1));
    arr[count] = s;
    count++;
    saveStudents(arr, count);
    free(arr);
    printf("Student Added...\n");
}
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 1
Enter Roll: 001
Enter Name: Raj
Enter Marks: 150
Student Added...
```

This screen allows the admin to add a new student record by entering the roll number, name, and marks. After submission, the record is successfully stored in the system.

- DISPLAY STUDENT

```c
void displayStudents() {
    struct Student *arr;
    int count, i;
    loadStudents(&arr, &count);
    if (count == 0) {
        printf("No records...\n");
        return;
    }
    for (i = 0; i < count; i++) {
        printf("%d | %s | %.2f\n", arr[i].roll, arr[i].name, arr[i].marks);
    }
    free(arr);
}
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 2
1 | Raj | 150.00
```

This output shows the list of all stored student records. It displays roll numbers, names, and marks in a structured table format.

- SEARCH STUDENT

```c
void searchStudent() {
    struct Student *arr;
    int count, roll, i;
    char buffer[50];
    loadStudents(&arr, &count);
    printf("Enter Roll to search: ");
    fgets(buffer, sizeof(buffer), stdin);
    sscanf(buffer, "%d", &roll);
    for (i = 0; i < count; i++) {
        if (arr[i].roll == roll) {
            printf("%d | %s | %.2f\n", arr[i].roll, arr[i].name, arr[i].marks);
            free(arr);
            return;
        }
    }
    printf("Not found...\n");
    free(arr);
}
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 3
Enter Roll to search: 001
1 | Raj | 150.00
```

This screen lets the user search for a student using the roll number. If the record exists, the student's details are displayed.

➢ UPDATE STUDENT

```
void updateStudent() {
    struct Student *arr;
    int count, roll, i;
    char buffer[50];
    loadStudents(&arr, &count);
    printf("Enter Roll to update: ");
    fgets(buffer, sizeof(buffer), stdin);
    sscanf(buffer, "%d", &roll);
    for (i = 0; i < count; i++) {
        if (arr[i].roll == roll) {
            printf("Enter New Name: ");
            fgets(arr[i].name, sizeof(arr[i].name), stdin);
            arr[i].name[strcspn(arr[i].name, "\n")] = 0;
            printf("Enter New Marks: ");
            fgets(buffer, sizeof(buffer), stdin);
            sscanf(buffer, "%f", &arr[i].marks);
            saveStudents(arr, count);
            free(arr);
            printf("Updated...\n");
            return;
        }
    }
    printf("Not found...\n");
    free(arr);
}
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 4
Enter Roll to update: 001
Enter New Name: VECNA
Enter New Marks: 111
Updated...
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 2
1 | VECNA | 111.00
```

The update screen enables modification of an existing student's details. The admin can update the student's name and marks after entering the roll number.

➢ DELETE STUDENT

```c
void deleteStudent() {
    struct Student *arr;
    int count, roll, i, j;
    char buffer[50];
    loadStudents(&arr, &count);
    printf("Enter Roll to delete: ");
    fgets(buffer, sizeof(buffer), stdin);
    sscanf(buffer, "%d", &roll);
    for (i = 0; i < count; i++) {
        if (arr[i].roll == roll) {
            for (j = i; j < count - 1; j++) arr[j] = arr[j + 1];
            count--;
            saveStudents(arr, count);
            free(arr);
            printf("Deleted...\n");
            return;
        }
    }
    printf("Not found...\n");
    free(arr);
}
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 5
Enter Roll to delete: 001
Deleted...
```

```
--- ADMIN MENU ---
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter Choice = 2
2 | Max | 130.00
3 | Eleven | 149.00
```

This output shows the result of deleting a student record. The system removes the matching roll number from the database and confirms the action.

CONCLUSION

The Student Record Management System is a simple program that helps store and manage student information in an easy and organised way. Users can add, view, search, update, and delete student records depending on their role. Admin, Staff, Guest and User users have different levels of access, which helps keep the records safe and prevents unwanted changes. The system is menu-driven and easy to understand, making it simple for anyone to use.

This project shows how basic C programming concepts like structures, functions, loops, conditions, and simple file storage can be used to build a small working application. It performs all the main tasks needed for handling student records in a clear and reliable way. Overall, the Student Record Management System is a good example of how useful applications can be created using simple and fundamental C programming techniques.