PROJECT REPORT

STUDENT RECORD MANAGEMENT SYSTEM(SRMS)

DONE BY

NAME: B GOWTHAM

REGISTRATION NUMBER: AP2411001873

CSE 2nd YEAR, THIRD SEMESTER

SECTION - W

SUBMITTED TO

RAKESH RAMA RAJU

CODING SKILLS-1 (CSE 201)

ABSTRACT

The **Student Report Management System (SRMS)** is a consolebased application developed in the C programming language to efficiently manage student academic records. The system enables secure login and provides two roles: **Admin and User**. Admins can add, display, search, update, and delete student reports, whereas users can only view and search student data. All records are stored in external text files, ensuring portability and simplicity. This project demonstrates the use of **file handling, structures, conditional statements, and user authentication** in C programming. The system provides an easy and reliable solution for maintaining student academic details.

The **Student Report Management System (SRMS)** is a consolebased application developed in the C programming language to efficiently maintain academic information of students. The system enables secure login authentication and supports two separate user roles: **Admin** and **User**. Administrators are provided with complete access to manage student data, including operations such as adding new records, viewing the list of students, searching based on roll number, updating existing details, and deleting incorrect or outdated entries. Meanwhile, general users are restricted to viewing and searching student information only, ensuring controlled privilege management.

The system makes use of data persistence through **file handling operations**, storing user credentials as well as student data in text files without requiring any external database. By utilizing concepts such as **structures, loops, conditional logic, functions, and file handling**, SRMS presents a practical approach to digital record management. It enhances efficiency, reduces manual errors, and provides a user-friendly solution for institutions to maintain academic records securely and systematically.

INTRODUCTION

The Student Report Management System (SRMS) is designed to provide a simple yet structured method for storing and accessing student academic details digitally. Developed using the C programming language, the system provides basic but essential functionalities required for managing student records efficiently. It mainly focuses on storing each student's roll number, name, and academic marks in a secure and manageable format. Instead of relying on complex databases, SRMS uses text file storage to maintain the data permanently, which makes the application lightweight and easily deployable on any computer supporting the C environment.

The system incorporates user authentication, which ensures that only authorized individuals can access or modify student information. Two types of users are allowed to log into the system: Admin and User. The Admin has complete control over the student database, with privileges such as adding new student details, modifying existing data, and deleting inaccurate or outdated records. On the other hand, a regular User is limited to viewing and searching student data only, ensuring that critical information cannot be altered by unauthorized users. This role-based access control enhances data security and minimizes the risks of unauthorized modifications.

One of the key programming aspects of SRMS is the use of structures in C, which allow grouping of student information into a single data type. Along with structures, the project utilizes file handling, allowing data to be written, read, searched, updated, and deleted from text files. It also incorporates modular programming through user-defined functions, which improves readability, reusability, and maintenance of the code. Control statements such as loops, condition checking, and

string manipulation functions are used to validate data, compare credentials, and process user input efficiently.

OBJECTIVES

THE KEY OBJECTIVES OF THIS PROJECT ARE:

- To create a computerized system that efficiently maintains academic records of students.
- To provide a **secure login facility** to protect stored student information from unauthorized access.
- To implement **role-based access**, ensuring Admin users have full control while general users have limited privileges.
- To maintain student information such as **roll number, name, and marks** in a well-structured format.
- To replace traditional manual record-keeping with a **fast and error-free digital solution**.
- To allow **easy addition of new student data** without overwriting existing records.
- To provide a quick way to **search student records** based on roll number.
- To enable **updating of student details**, ensuring the database always remains accurate.
- To offer an option to **delete outdated or wrong records** efficiently without affecting other data.
- To use **file handling techniques** for storing student data permanently without using external databases.
- To demonstrate the real-world application of **C programming concepts**, including structures and modular programming.

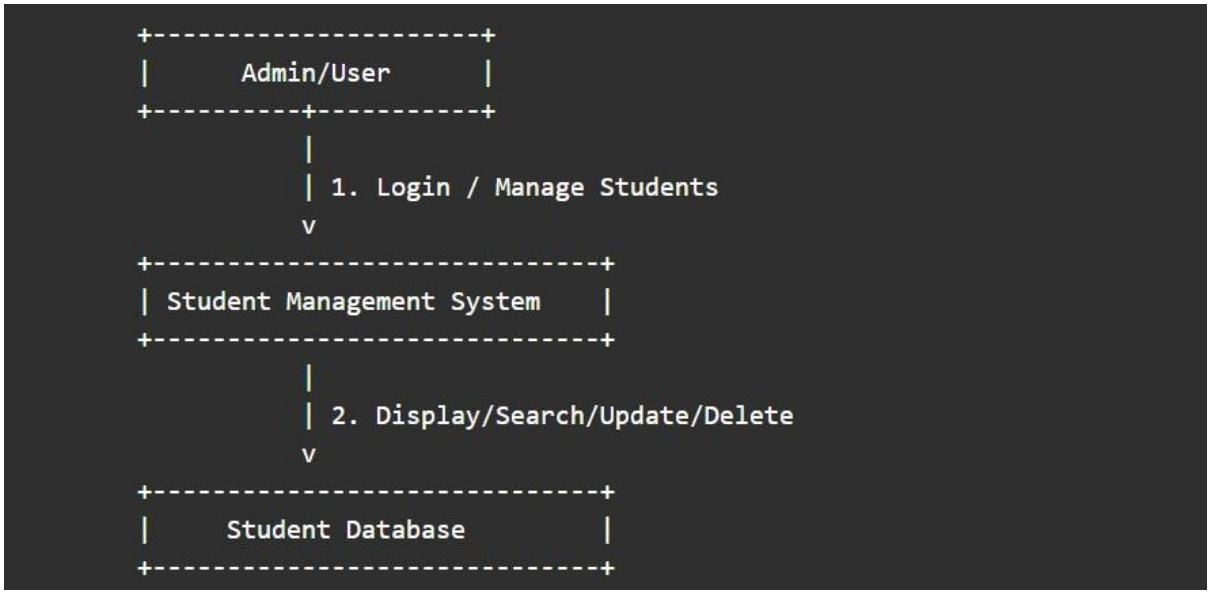
MODULES

Module Name	Description
Login Module	Authenticates user credentials and supports rolebased access (Admin/User).
Admin Module	Allows administrators to add, view, search, update, and delete student records.
User Module	Allows viewing and searching of student data only (restricted access).
Add Student Module	Stores new student data (Roll No., Name, Marks).
View Records Module	Displays all students' stored records from the file.
Search Module	Searches for a student using their roll number.
Update/Delete Modules	Admins can modify or remove stored student data.

File Storage Module	Handles credential storage (credentials.txt) and student records (students.txt).

DATA FLOW DIAGRAM

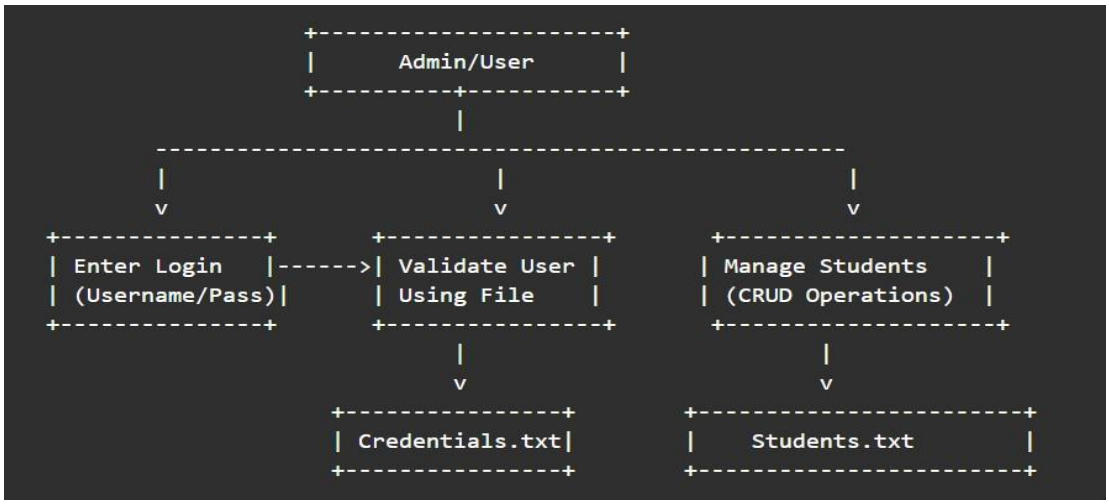
LEVEL -1 DATA FLOW DIAGRAM



Explanation

- User/Admin interacts with the SRMS.
- System processes the request and accesses stored files.

LEVEL -2 DATA FLOW DIAGRAM



Explanation

- After login, control is transferred based on role:
  - Admin: Full access (Add, View, Search, Update, Delete).
  - User: Only View and Search.
- All actions read/write from **students.txt**.

CODE & OUTPUTS

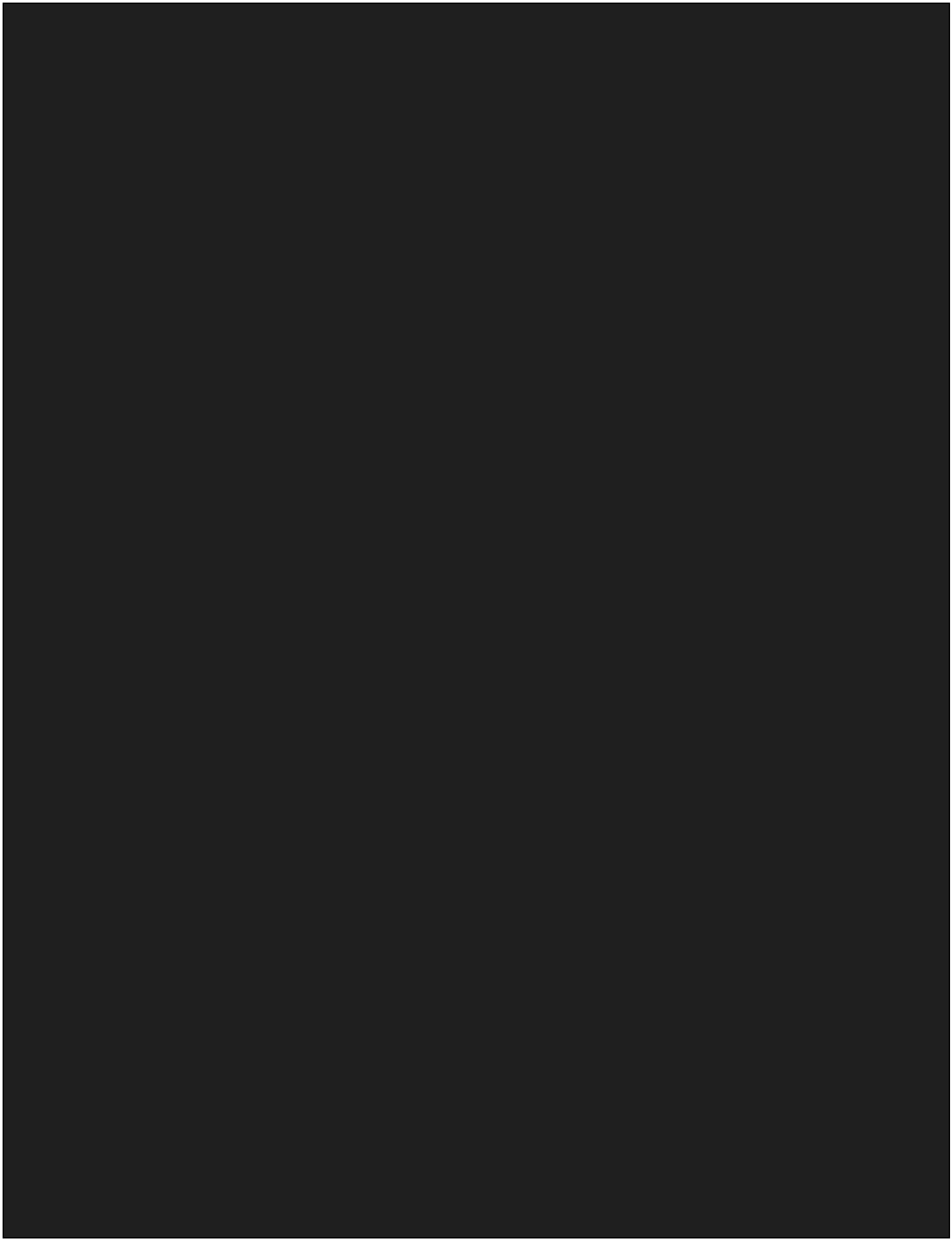
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define CREDENTIAL_FILE "credentials.txt"
#define STUDENT_FILE "students.txt"

struct student {
int roll;     char
name[50];
float marks;
};

char currentUser[20]; char
currentRole[20];

void createCredentials();
int loginSystem();
```

```c
void mainMenu(); void
adminMenu(); void
userMenu(); void
addStudent(); void
displayStudents(); void
searchStudent(); void
updateStudent(); void
deleteStudent();

int main() {
    createCredentials();

    if (loginSystem())
mainMenu();     else
        printf("Login failed!\n");

    return 0;
}

/* ---------------- CREATE CREDENTIALS ---------------- */
void createCredentials() {
    char choice, user[20], pass[20], role[10];

    printf("Create login credentials? (y/n): ");
scanf(" %c", &choice);

    if (choice == 'y' || choice == 'Y') {
printf("Enter username: ");         scanf("%s",
user);

        printf("Enter password: ");
scanf("%s", pass);

        printf("Enter role (admin/user): ");
scanf("%s", role);

        FILE *fp = fopen(CREDENTIAL_FILE, "w");
fprintf(fp, "%s %s %s\n", user, pass, role);
fclose(fp);

        printf("Credentials created!\n");
    }
```

```
    FILE *f = fopen(STUDENT_FILE, "r");
if (!f) {
```

```
        f = fopen(STUDENT_FILE,
"w");         fclose(f);     } else {
fclose(f);
    }
}

/* ---------------- LOGIN SYSTEM ---------------- */ int
loginSystem() {
    char username[20], password[20], fuser[20], fpass[20], frole[20];

    printf("\n===== LOGIN =====\n");
printf("Username: ");
scanf("%s", username);
printf("Password: ");
scanf("%s", password);

    FILE *fp = fopen(CREDENTIAL_FILE,
"r");     if (!fp) {
printf("Credentials missing!\n");
return 0;
    }

    while (fscanf(fp, "%s %s %s", fuser, fpass, frole) != EOF) {         if
(strcmp(username, fuser) == 0 && strcmp(password, fpass) == 0) {
strcpy(currentUser, fuser);             strcpy(currentRole, frole);
fclose(fp);             return 1;
        }     }
fclose(fp);
return 0;
}

/* ----------------- MAIN MENU -----------------
*/ void mainMenu() {     if (strcmp(currentRole,
"admin") == 0)         adminMenu();     else
        userMenu();
}

/* ---------------- ADMIN MENU ---------------- */
void adminMenu() {     int ch;
```

```c
    do {         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: ");         scanf("%d",
&ch);

        switch (ch) {            case 1:
addStudent(); break;            case 2:
displayStudents(); break;            case 3:
searchStudent(); break;            case 4:
updateStudent(); break;            case 5:
deleteStudent(); break;            case 6:
return;            default: printf("Invalid
choice!\n");
        }
    } while (1);
}

/* ---------------- USER MENU ---------------- */
void userMenu() {    int ch;

    do {         printf("\n===== USER MENU
=====\n");         printf("1. Display
Students\n");         printf("2. Search
Student\n");         printf("3. Logout\n");
printf("Enter choice: ");
scanf("%d", &ch);

        switch (ch) {            case 1:
displayStudents(); break;            case 2:
searchStudent(); break;            case 3:
return;            default: printf("Invalid
choice!\n");
        }
    } while (1);
}

/* ---------------- ADD STUDENT ----------------
*/ void addStudent() {    struct student st;
```



```
    printf("Enter     roll     number:     ");
scanf("%d", &st.roll);

    printf("Enter full name: ");
getchar();     fgets(st.name,
sizeof(st.name), stdin);
st.name[strcspn(st.name, "\n")] = 0;

    printf("Enter       marks:       ");
scanf("%f", &st.marks);

    FILE *fp = fopen(STUDENT_FILE, "a");     fprintf(fp,
"%d|%s|%.2f\n", st.roll, st.name, st.marks);     fclose(fp);

    printf("Student added!\n");
}

/* ---------------- DISPLAY STUDENTS ----------------
*/ void displayStudents() {     struct student st;
    FILE *fp = fopen(STUDENT_FILE, "r");

    printf("\nRoll\tName\t\tMarks\n");
    printf("-------------------------------------\n");

    while (fscanf(fp, "%d|%[^|]|%f", &st.roll, st.name, &st.marks) != EOF)
printf("%d\t%-15s %.2f\n", st.roll, st.name, st.marks);

    fclose(fp);
}

/* ---------------- SEARCH STUDENT ----------------
*/ void searchStudent() {     int roll, found = 0;
printf("Enter roll: ");     scanf("%d", &roll);

    struct student st;
    FILE *fp = fopen(STUDENT_FILE, "r");
```

```
    while (fscanf(fp, "%d|%[^|]|%f", &st.roll, st.name, &st.marks) != EOF)
{       if (st.roll == roll) {          printf("Found: %d  %s  %.2f\n",
st.roll, st.name, st.marks);          found = 1;
```

```
            break;
        }
    }

    if (!found)
        printf("Not found!\n");

    fclose(fp);
}

/* ---------------- UPDATE STUDENT ----------------
*/ void updateStudent() {     int roll, found = 0;
printf("Enter roll to update: ");     scanf("%d",
&roll);

    struct student st;
    FILE *fp = fopen(STUDENT_FILE, "r");
    FILE *tmp = fopen("temp.txt", "w");

    while (fscanf(fp, "%d|%[^|]|%f", &st.roll, st.name, &st.marks) != EOF)
{        if (st.roll == roll) {            found = 1;

            printf("New name: ");
getchar();            fgets(st.name,
sizeof(st.name), stdin);
st.name[strcspn(st.name, "\n")] = 0;

            printf("New marks: ");
scanf("%f", &st.marks);
        }        fprintf(tmp, "%d|%s|%.2f\n", st.roll, st.name,
st.marks);     }

    fclose(fp);
fclose(tmp);

    if (found) {
remove(STUDENT_FILE);
rename("temp.txt", STUDENT_FILE);
printf("Updated!\n");
    } else {
remove("temp.txt");
        printf("Roll not found!\n");
```

```
        }
    }
```

```c
/* ---------------- DELETE STUDENT ----------------
*/ void deleteStudent() {     int roll, found = 0;
printf("Enter roll to delete: ");     scanf("%d",
&roll);

    struct student st;
    FILE *fp = fopen(STUDENT_FILE, "r");
    FILE *tmp = fopen("temp.txt", "w");

    while (fscanf(fp, "%d|%[^|]|%f", &st.roll, st.name, &st.marks) != EOF)
{         if (st.roll == roll) {             found = 1;             continue;
        }         fprintf(tmp, "%d|%s|%.2f\n", st.roll, st.name,
st.marks);     }

    fclose(fp);
fclose(tmp);

    if (found) {
remove(STUDENT_FILE);
rename("temp.txt", STUDENT_FILE);
printf("Deleted!\n");
    } else {
remove("temp.txt");
        printf("Roll not found!\n");
    }
}
```

OUTPUT :

```
Create login credentials? (y/n): y
Enter username: admin
Enter password: admin123
Enter role (admin/user): admin
Credentials created!

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

1.ADDING STUDENT

```
===== ADMIN MENU =====
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter choice: 1
Enter roll number: 101
Enter full name: Ramesh
Enter marks: 78
Student added!
```

```
===== ADMIN MENU =====
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter choice: 1
Enter roll number: 102
Enter full name: Rakesh
Enter marks: 89
Student added!
```

2.DISPLAY STUDENT

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

Roll     Name                 Marks
-----------------------------------
101      Ramesh               78.00
102      Rakesh               89.00
```

3.SEARCH & UPDATE

```
===== ADMIN MENU =====
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter choice: 3
Enter roll: 101
Found: 101  Ramesh  78.00
```

```
===== 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: 102
New name: Rakesh
New marks: 92
Updated!
```

4.DELETE & LOGOUT

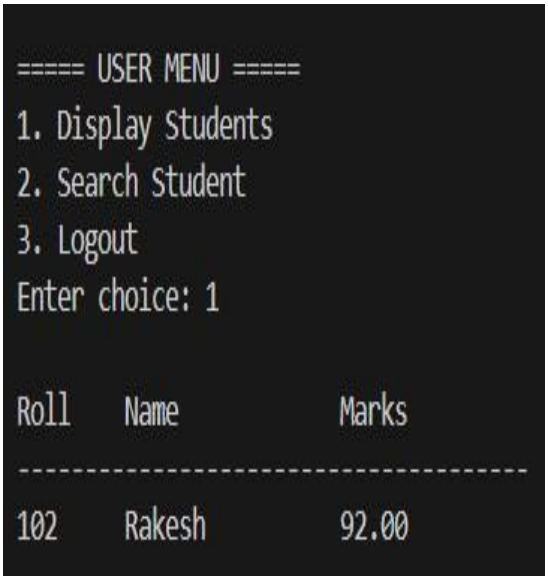
```
===== ADMIN MENU =====
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter choice: 5
Enter roll to delete: 101
Deleted!
```

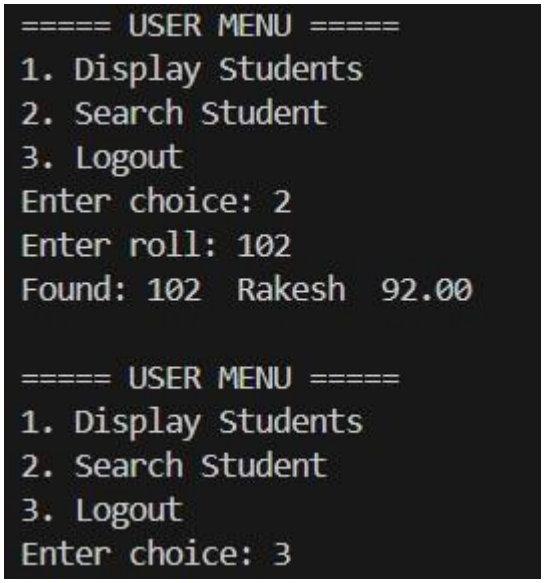
```
===== ADMIN MENU =====
1. Add Student
2. Display Students
3. Search Student
4. Update Student
5. Delete Student
6. Logout
Enter choice: 6
```

5.STAFF LOGIN

```
Create login credentials? (y/n): y
Enter username: staff
Enter password: staff123
Enter role (admin/user): user
Credentials created!

===== LOGIN =====
Username: staff
Password: staff123
```

6.SEARCH & LOGOUT





credentials.txt

admin admin123 admin

staff staff123 staff guest

guest123 guest

students.txt

101 Ramesh 78

102 Rakesh 92

CONCLUSION

The Student Report Management System (SRMS) is an effective and user-friendly solution designed to simplify the management of student academic information. The project successfully demonstrates how fundamental C programming concepts such as structures, file handling, decision making, and modular coding techniques can be applied to develop real-time applications. Through its secure login facility and role-based access control, SRMS ensures proper data privacy by allowing different access rights to administrators, staff, and guest users. All academic records are stored in text files, making the system lightweight, portable, and easy to use without requiring any external database.

The system provides all essential data operations such as adding, updating, deleting, searching, and viewing student details. It significantly reduces the chances of human error compared to manual record management and enhances efficiency by offering quick retrieval of student information. This project highlights the importance of digitizing academic data and serves as a strong foundation for further enhancements, such as integrating databases, graphical interfaces, encryption, and cloud support. Overall, SRMS proves to be an impactful and practical software application that can be adopted by educational institutions of different scales to streamline their record-keeping process.