CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT
ON

Student Result Management System

Subject Name: Computer Programming

SUBJECT CODE : 25CAH - 101

SUBMITED BY:

Name : MAYANK SHARMA **UID : 25BCD10113**

Branch : BCA (DATA SCIENCE) **Section: 25BCD- 2B**

Semester : 1ST

SUBMITED TO:

Name : MR. Dharam Pal Sir **D.O.SUBMITION : 12-NOV-2025**

Designation: Asst. Professor

Sign:_____

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

1. Title

Student Result Management System

2. Introduction

The Student Result Management System is a C language–based application designed to manage and maintain the academic results of students efficiently.

This system helps in storing student information, calculating results, and displaying report cards in a structured format. It eliminates the need for manual record keeping, ensuring faster and more accurate result processing.

3. Objective

The main objective of this project is to:

Manage student academic records systematically.

Compute total marks, average, and grade automatically.

Enable easy addition, modification, and deletion of student data.

Store records permanently using file handling.

4. Scope

This system can be implemented in schools, colleges, and universities to manage student results electronically.

It serves as a base for larger education management systems and can be extended to include attendance and subject-wise performance analysis.

5. Software & Hardware Requirements

Language: C

Compiler: GCC / Turbo C / Code::Blocks

Operating System: Windows / Linux

Hardware: Minimum 2GB RAM, 500MB disk space

6. Modules of the Project

Add Student Record – Input student details and marks.

Display All Records – Show stored data in tabular form.

Search Record – Find a student by roll number.

Modify Record – Update marks or details.

Delete Record – Remove a student record.

Exit – Close the application.

7. Data Structure Used

```
struct Student {
    int rollno;
    char name[50];
    float marks[5];
    float total;
    float average;
    char grade;
};
```

8. Algorithm / Working

Start the program.

Display menu options.

Take user choice as input.

Perform operation (add, display, search, modify, delete).

Calculate total, average, and grade automatically.

Save data to file ("student.dat").

Return to menu until Exit is chosen.

Stop the program.

9. Source Code (C Program)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct Student {
    int rollno;
    char name[50];
```

```
    float marks[5];
    float total;
    float average;
    char grade;
};

void addStudent();
void displayStudents();
void searchStudent();
void calculateGrade(struct Student *s);
int main() {
    int choice;
    while (1) {
        printf("\n==== STUDENT RESULT MANAGEMENT SYSTEM ====\n");
        printf("1. Add Student Record\n");
        printf("2. Display All Records\n");
        printf("3. Search Student\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1: addStudent(); break;
            case 2: displayStudents(); break;
            case 3: searchStudent(); break;
            case 4: exit(0);
            default: printf("Invalid choice!\n");
        }
    }
    return 0;
```

```
}
void addStudent() {
    struct Student s;
    FILE *fp = fopen("student.dat", "ab");
    if (!fp) {
     printf("Error opening file!\n");
        return;
    }
    printf("Enter Roll No: ");
    scanf("%d", &s.rollno);
    printf("Enter Name: ");
    getchar();
    gets(s.name);
    s.total = 0;
    for (int i = 0; i < 5; i++) {
        printf("Enter Marks in Subject %d: ", i + 1);
        scanf("%f", &s.marks[i]);
        s.total += s.marks[i];
    }
    s.average = s.total / 5;
    calculateGrade(&s);
    fwrite(&s, sizeof(s), 1, fp);
    fclose(fp);
    printf("Record added successfully!\n");
}
void displayStudents() {
    struct Student s;
    FILE *fp = fopen("student.dat", "rb");
    if (!fp) {
```

```
        printf("No records found!\n");
        return;
    }
    printf("\n%-10s %-20s %-10s %-10s %-10s\n", "Roll No", "Name", "Total",
"Average", "Grade");
    while (fread(&s, sizeof(s), 1, fp)) {
        printf("%-10d %-20s %-10.2f %-10.2f %-10c\n", s.rollno, s.name, s.total,
s.average, s.grade);
    }
    fclose(fp);
}
void searchStudent() {
    struct Student s;
    FILE *fp = fopen("student.dat", "rb");
    int roll, found = 0;
    if (!fp) {
        printf("No records found!\n");
        return;
    }
    printf("Enter Roll No to Search: ");
    scanf("%d", &roll);
    while (fread(&s, sizeof(s), 1, fp)) {
        if (s.rollno == roll) {
            printf("\nRecord Found!\n");
            printf("Name: %s\nTotal: %.2f\nAverage: %.2f\nGrade: %c\n", s.name, s.total,
s.average, s.grade);
            found = 1;
            break;
        }
    }
```

```
    if (!found) printf("Record not found!\n");
    fclose(fp);
}
void calculateGrade(struct Student *s) {
    if (s->average >= 90) s->grade = 'A';
    else if (s->average >= 75) s->grade = 'B';
    else if (s->average >= 60) s->grade = 'C';
    else if (s->average >= 50) s->grade = 'D';
    else s->grade = 'F';
}
```

10. Sample Output

```
==== STUDENT RESULT MANAGEMENT SYSTEM ====
1. Add Student Record
2. Display All Records
3. Search Student
4. Exit
Enter your choice: 1
Enter Roll No: 101
Enter Name: Rahul
Enter Marks in Subject 1: 85
Enter Marks in Subject 2: 78
Enter Marks in Subject 3: 90
Enter Marks in Subject 4: 88
Enter Marks in Subject 5: 92
Record added successfully!
```

11. Advantages

Simple and user-friendly interface.

Eliminates manual record keeping.

Data is stored permanently.

Easy to extend and maintain.

12. Limitations

No password protection or role-based access.

Records are stored in binary form, not readable directly.

Limited error handling.

13. Future Enhancements

Add login and authentication.

Implement subject-wise performance charts.

Create a GUI-based version in C++ or Python.

Include automated grade analysis and ranking.

14. Conclusion

The Student Result Management System is a reliable and efficient project that demonstrates the use of key C programming concepts such as structures, file handling, loops, and conditionals. It provides a real-world simulation of how educational institutions can manage results digitally and forms a solid foundation for further software development.