

PROJECT REPORT

STUDENT RECORD MANAGEMENT SYSTEM

(SRMS)

BY

Name of the Student:	Registration no:	Year/Brach/Section
L Krishna Prajeeth	AP24110010391	2nd/CSE/Z

Prepared in the partial fulfillment of

the Project Based Learning of Course

CSE 201 – Coding Skills-1



SRM UNIVERSITY, AP

DECEMBER 2025

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those who supported me throughout the development of this project.

First and foremost, I extend my heartfelt thanks to my faculty guide, **Mr. Prakash** (campus corporate connect trainer), for his constant guidance, encouragement, and valuable feedback at every stage of this work. His insights helped me understand concepts clearly and complete the project successfully.

I am also thankful to the Campus Corporate connect and the college administration for providing the necessary resources, facilities, and a supportive learning environment that made this project possible.

(Signature of the Course Instructor)

Mr. Prakash

Campus Corporate Connect 61

Abstract

This program is designed to collect and display a student's complete academic and personal information in a structured report format. It uses a Student structure that stores details such as roll number, name, age, course, email, phone number, marks of five subjects, total marks, percentage, and the final grade. The purpose of the program is to demonstrate how structures can be used to organize and process related data efficiently in C++.

The execution begins with taking personal details from the user, ensuring that each student can be uniquely identified. After this, the program collects marks for five subjects. These marks are stored in an array within the structure. A member function `calculateGrade()` is then used to process the data. It adds all the marks to calculate the total and divides the sum by five to obtain the percentage.

Once the percentage is computed, the program assigns a grade based on a simple grading scale:

- 90% and above – A
- 75% to 89% – B
- 60% to 74% – C
- 40% to 59% – D
- Below 40% – F

After the calculations, the program prints a clear and well-formatted student report. This report includes all the personal details entered earlier, along with the total marks, percentage, and grade. This makes it easy for anyone to understand the student's performance at a glance.

Overall, the program highlights how structures, arrays, loops, and conditional statements can be used together to build simple data-management applications. It effectively converts user input into an organized academic report, demonstrating both data handling and basic result processing in C++.

Table of Contents

1. Introduction
2. Objectives
3. System Requirements
4. Functional Requirements
5. Non-functional Requirements
6. System Design
 - o Overview
 - o Module decomposition
 - o Class diagram (textual)
7. Data Structures and Algorithms
8. Implementation Details
 - o Key classes and functions
 - o File format
 - o Error handling
9. Sample Code (core snippets)
10. User Manual
 - How to compile
 - How to run
 - Example session
11. Test Plan and Test Cases
12. Results and Screenshots (example I/O)
13. Future Enhancements
14. Conclusion
15. References

1. Introduction

Student Record Management Systems (SRMS) simplify storing and retrieving student information such as roll number, name, age, course, and grades. This project provides a console-based C++ application suitable for small institutions or as an educational demonstration of file-based persistence and object-oriented programming.

2. Objectives

- Design and implement a user-friendly console application in C++.
- Store student records persistently using file I/O.
- Provide CRUD operations (Create, Read, Update, Delete).
- Implement search and sorting functions.
- Demonstrate robust input validation and error handling.

3. System Requirements

Hardware

- Any PC with minimum 256 MB RAM and 50 MB disk space.

Software

- C++ compiler (g++ recommended, version supporting C++11 or later).
- Operating system: Windows / Linux / macOS.

4. Functional Requirements

- Add new student record.
- Display all student records.
- Search student by roll number or name.
- Update existing student record.
- Delete a student record.
- Sort records by roll number, name, or GPA.
- Persist and load records from file at startup/shutdown.

5. Non-functional Requirements

- Console-based UI for simplicity.
- Reliable file operations with error messages.
- Reasonable performance ($O(n \log n)$ sorting with `std::sort`).
- Maintainable and modular C++ codebase.

6. System Design

Overview

The system follows a modular design with separate classes for Student, StudentManager (handles operations and file I/O), and a simple Menu driving the user interaction.

Class diagram (textual)

7. Data Structures and Algorithms

Purpose of the Data Structure

- To store **complete student details**
- To hold **marks of 5 subjects**
- To compute and store **total, percentage, and grade**

Flow of Algorithm (Summary)

1. Input → Student Details
2. Input → 5 Marks
3. Compute → Total
4. Compute → Percentage
5. Compute → Grade
6. Output → Complete Report

8. Implementation Details

*A structure is used to keep all student data together

*Input is taken using `cin` and `getline()`

* A loop is used to read 5 marks

*The function `calculateGrade()` computes the results

*Final output shows the student's details, total, percentage, and grade

9. Key design decisions

1. A **structure** is used to keep all student details together.
2. An **array** stores the 5 subject marks to make calculations easy.
3. A separate **calculateGrade() function** is used to compute total, percentage, and grade.

File format (CSV)

```
roll,name,age,course,EMAIL,phone,marks  
1,John Doe,20,CS,john@gmail.com,9999888810,8.5
```

Error handling

- Check file open success and display clear messages.
- Handle malformed lines gracefully (skip with logged warning).

10. Sample Code (core snippets)

Below are simplified, well-commented code snippets. Full implementation should include additional validation and robustness.

```
#include <iostream>  
  
#include <cstring>  
  
using namespace std;
```

```
struct Student {  
  
    int rollNumber;  
  
    char name[50];  
  
    int age;
```

```

char course[30];
float marks[5];
float totalMarks;
float percentage;
char grade;
char email[50];
char phone[15];

void calculateGrade() {
    totalMarks = 0;

    for(int i = 0; i < 5; i++) {
        totalMarks += marks[i];
    }

    percentage = totalMarks / 5;

    // Calculate grade
    if (percentage >= 90)
        grade = 'A';
    else if (percentage >= 75)
        grade = 'B';
    else if (percentage >= 60)
        grade = 'C';
    else if (percentage >= 40)
        grade = 'D';
    else
        grade = 'F';
}

```

```
    }  
};  
  
int main() {  
    Student s;  
  
    cout << "Enter Roll Number: ";  
    cin >> s.rollNumber;  
  
    cin.ignore(); // clear buffer  
  
    cout << "Enter Name: ";  
    cin.getline(s.name, 50);  
  
    cout << "Enter Age: ";  
    cin >> s.age;  
  
    cin.ignore();  
  
    cout << "Enter Course: ";  
    cin.getline(s.course, 30);  
  
    cout << "Enter Email: ";  
    cin.getline(s.email, 50);  
  
    cout << "Enter Phone: ";  
    cin.getline(s.phone, 15);
```

```

cout << "Enter marks for 5 subjects:\n";
for(int i = 0; i < 5; i++) {
    cout << "Subject " << i+1 << ": ";
    cin >> s.marks[i];
}

// Calculate grade and percentage
s.calculateGrade();

// Output result
cout << "\n===== STUDENT REPORT =====\n";
cout << "Roll Number: " << s.rollNumber << endl;
cout << "Name: " << s.name << endl;
cout << "Age: " << s.age << endl;
cout << "Course: " << s.course << endl;
cout << "Email: " << s.email << endl;
cout << "Phone: " << s.phone << endl;

cout << "Total Marks: " << s.totalMarks << endl;
cout << "Percentage: " << s.percentage << "%" << endl;
cout << "Grade: " << s.grade << endl;

return 0;
}

```

11. Conclusion

This Student Record Management System demonstrates core software engineering principles: requirement analysis, modular design, implementation in C++, persistent storage, and basic algorithms. It can be extended into a full-featured system with a database and GUI.

12. References

- C++ Reference (cplusplus.com / cppreference.com)
 - “Programming: Principles and Practice Using C++” — Bjarne Stroustrup
-

End of report.