A REPORT

ON

STUDENT RECORD MANAGEMENT SYSTEM

SUBMITTED BY

Name of the student	Registration No.
K. Lava Kumar	AP24110011409
B. Pradeep	AP24110011432
D. Venu Gopal	AP24110011446
C. Jashua Jordan	AP24110011457

Prepared in the partial fulfilment of the

Project Based Learning of course CSE 201

SRM UNIVERSITY, AP

DECEMBER 2025

CERTIFICATE

This is to certify that the project titled **"Student Record Management System"** is a work carried out by all the team members as a required academic component of the course CSE 201: coding skills-1 during semester 3 of academic year 2025-2026 in the department of Computer Science and Engineering at SRM University, AP. This work was executed under the guidance and is deemed to have successfully met all the course requirements.

(Signature of the Course Instructor)

Course Instructor,
Department of Coding Skill,
SRM University, AP

Acknowledgement

I take this opportunity to express my sincere gratitude to all those who have contributed directly or indirectly to the successful completion of this project titled **"Student Record Management System."** This project would not have been possible without the valuable guidance, encouragement, and continuous support of many individuals.

First and foremost, I would like to express my deep sense of gratitude to our professor, **Prakash**, for his constant guidance, valuable suggestions, and moral support throughout the development of this project. His technical expertise, constructive feedback, and motivation played a vital role in shaping this project and bringing it to a successful completion.

I would like to express my sincere appreciation to my teammates and friends for their cooperation, support, and constructive suggestions which helped me overcome many difficulties during the project work. Their constant motivation and teamwork made this project a memorable learning experience.

With regards,
Team Members

ABSTRACT

The Student Record Management System is a software-based application developed to store, manage, and maintain student academic and personal information in an efficient and organized manner. In many educational institutions, student records are traditionally maintained using manual methods such as registers and files. This process is time-consuming, prone to human errors, and becomes difficult to manage as the number of students increases. To overcome these limitations, this project presents a computerized solution for student data management.

The system enables the administrator to perform essential operations such as adding new student records, viewing existing records, updating student information, deleting unnecessary records, and searching student details using unique identifiers like roll number or name. All data is stored securely in a database, ensuring fast access, accuracy, and data integrity. A user-friendly interface allows easy interaction with the system, even for users with basic technical knowledge.

The Student Record Management System can be widely used in schools, colleges, universities, and training institutes. Additionally, the system can be enhanced in the future by integrating advanced features such as online student portals, attendance tracking, fee management, result processing, and cloud-based data storage. Overall, this project offers a practical and modern solution for overcoming the limitations of manual student record keeping and contributes towards the digital transformation of educational institutions.

The main objective of this project is to reduce manual work, minimize errors, and improve efficiency in maintaining student records. The Student Record Management System is highly useful for schools, colleges, and educational institutions. It also provides scope for future enhancements such as attendance management, result processing, fee management, and cloud-based data storage.

TABLE OF CONTENTS

INTRODUCTION

In today's digital era, information plays a vital role in the effective functioning of every organization, especially in the field of education. Educational institutions such as schools, colleges, and universities deal with a large volume of student information including personal details, academic records, attendance, and performance data. Traditionally, this information has been maintained manually using registers, files, and documents. Although this method has been followed for many years, it has become inefficient due to the increasing number of students and the growing need for quick and accurate access to data. Manual record-keeping is time-consuming, prone to errors, difficult to update, and lacks proper security.

With the rapid development of computer technology and digital systems, most organizations are shifting from manual processes to automated solutions. A Student Record Management System is one such computerized solution designed to manage student information in a systematic, efficient, and secure manner. This system helps in storing, retrieving, updating, and deleting student records with ease. By using a centralized database, all student-related information can be accessed instantly, which improves the overall efficiency of administrative operations.

This project focuses on designing and implementing a user-friendly and reliable student management system using modern programming techniques and database technology. The system provides basic functionalities such as adding new student records, viewing existing records, updating information, deleting unwanted records, and searching for specific student details. It serves as an effective tool for managing student data in an organized manner.

PROBLEM STATEMENT

In many educational institutions, student records are still maintained using traditional manual methods such as paper files, registers, and spreadsheets. As the number of students increases every year, managing large volumes of data through these methods becomes highly difficult and inefficient. Manual record-keeping is time-consuming and requires significant effort to store, retrieve, update, and maintain student information. It also increases the chances of human errors such as incorrect entries, data duplication, and misplacement of important records.

Another major problem with the manual system is the lack of proper data security. Student records can be easily accessed, altered, or lost due to unauthorized handling, physical damage, or natural causes. Searching for a particular student's information from a large set of records is also a slow and tedious process. Moreover, generating reports and maintaining updated records becomes complicated with the traditional approach.

These limitations highlight the need for a reliable, efficient, and secure computerized solution to manage student data. Therefore, the Student Record Management System is proposed to overcome the drawbacks of the existing manual system and to provide an automated platform that ensures accuracy, fast access, secure storage, and efficient management of student information for educational institutions.

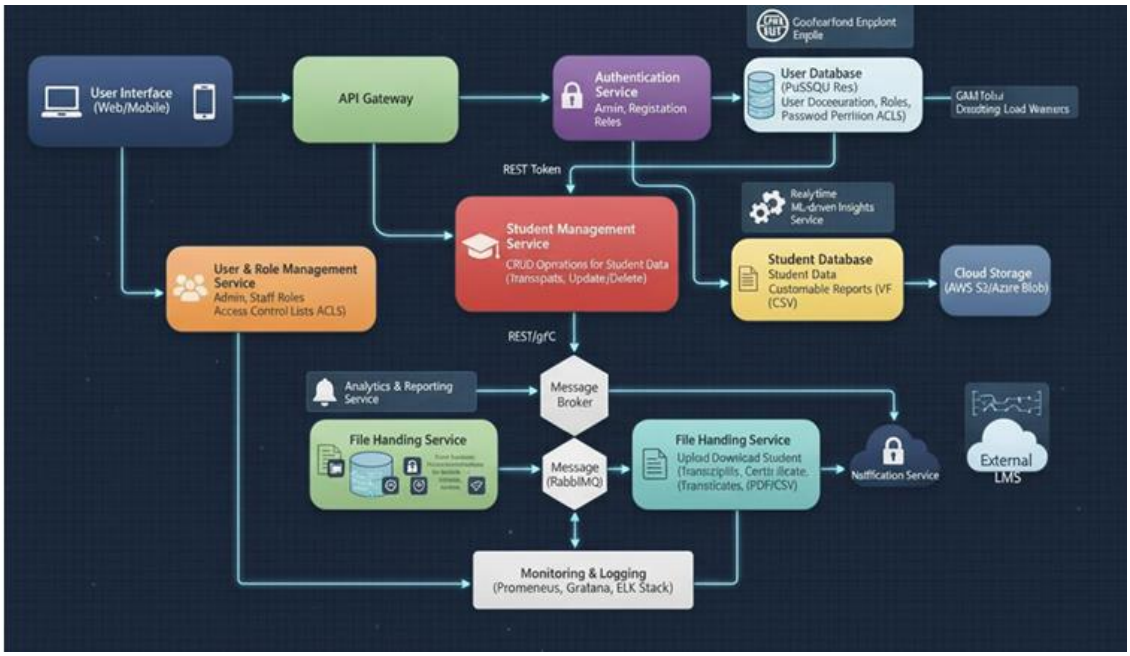
OBJECTIVES

The primary objective of the Student Record Management System is to design and develop a reliable and efficient computerized solution for managing student information in educational institutions. The system aims to replace the traditional manual method of record keeping with a digital platform that ensures accuracy, security, and fast access to data.

One of the important objectives of this project is to reduce the amount of paperwork and manual effort involved in maintaining student records. By automating the process, the system helps in saving time and minimizing human errors that commonly occur in manual data entry and record maintenance. The project also aims to provide a user-friendly interface so that administrative staff can operate the system easily without requiring advanced technical knowledge.

Another key objective is to ensure secure storage of student information by implementing proper authentication and data validation mechanisms. The system is designed to prevent unauthorized access and to maintain the integrity and confidentiality of student data. In addition, the project aims to provide quick searching and retrieval of student records based on unique identifiers such as roll number or name. Overall, the main goal of this project is to improve the efficiency, accuracy, and reliability of student record management through the effective use of computer technology.

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), 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.

SAMPLE CODE

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

struct Student {
    int id;
    char name[50];
    char course[50];
    char year[20];
    char address[100];
};

struct Complaint {
    int student_id;
    char message[200];
};

// FUNCTION DECLARATIONS
void adminMenu();
void studentMenu(int stID);
void addStudent();
void viewStudents();
void modifyStudent();
void deleteStudent();
void raiseComplaint(int stID);
void viewComplaints();
```

```
void viewOwnRecord(int stID);

// ----------------- MAIN MENU -----------------
int main() {
    int choice, stID;

    while (1) {
        printf("\n========= STUDENT RECORD MANAGEMENT SYSTEM =========\n");
        printf("1. Admin Login\n");
        printf("2. Student Login\n");
        printf("3. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                adminMenu();
                break;

            case 2:
                printf("Enter Student ID: ");
                scanf("%d", &stID);
                studentMenu(stID);
                break;

            case 3:
                printf("Exiting...\n");
                exit(0);
```

```
            default:
                printf("Invalid choice!\n");
        }
    }

    return 0;
}

// ------------------ ADMIN MENU ------------------
void adminMenu() {
    int choice;

    while (1) {
        printf("\n========== ADMIN MENU ==========\n");
        printf("1. Add Student Record\n");
        printf("2. View All Records\n");
        printf("3. Modify Student Record\n");
        printf("4. Delete Student Record\n");
        printf("5. View Complaints\n");
        printf("6. Back to Main Menu\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addStudent(); break;
            case 2: viewStudents(); break;
            case 3: modifyStudent(); break;
            case 4: deleteStudent(); break;
```

```
            case 5: viewComplaints(); break;
            case 6: return;
            default: printf("Invalid choice!\n");
        }
    }
}

// ----------------- STUDENT MENU -----------------
void studentMenu(int stID) {
    int choice;

    while (1) {
        printf("\n========== STUDENT MENU ==========\n");
        printf("1. View My Record\n");
        printf("2. Raise Complaint\n");
        printf("3. Back to Main Menu\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: viewOwnRecord(stID); break;
            case 2: raiseComplaint(stID); break;
            case 3: return;
            default: printf("Invalid choice!\n");
        }
    }
}

// ----------------- ADD STUDENT -----------------
```

```
void addStudent() {
    struct Student s;
    FILE *fp = fopen("students.txt", "ab+");  // create file if not exists

    if (!fp) {
        printf("Error opening file!\n");
        return;
    }

    printf("Enter ID: ");
    scanf("%d", &s.id);
    printf("Enter Name: ");
    scanf(" %[^\n]s", s.name);
    printf("Enter Course: ");
    scanf(" %[^\n]s", s.course);
    printf("Enter Year: ");
    scanf(" %[^\n]s", s.year);
    printf("Enter Address: ");
    scanf(" %[^\n]s", s.address);

    fwrite(&s, sizeof(s), 1, fp);
    fclose(fp);

    printf("Record added successfully!\n");
}

// ---------- VIEW ALL STUDENTS ----------
void viewStudents() {
    struct Student s;
```

```
    FILE *fp = fopen("students.txt", "rb");

    if (!fp) {
        printf("No student records found!\n");
        return;
    }

    printf("\n========= ALL STUDENT RECORDS =========\n");

    while (fread(&s, sizeof(s), 1, fp)) {
        printf("\nID: %d\nName: %s\nCourse: %s\nYear: %s\nAddress: %s\n",
               s.id, s.name, s.course, s.year, s.address);
    }

    fclose(fp);
}

// ---------------- MODIFY STUDENT ----------------
void modifyStudent() {
    struct Student s;
    FILE *fp = fopen("students.txt", "rb");

    if (!fp) {
        printf("No student records found!\n");
        return;
    }

    FILE *temp = fopen("temp.txt", "wb");
    if (!temp) {
```

```
    printf("Error creating temp file!\n");
    fclose(fp);
    return;
}

int id, found = 0;
printf("Enter ID to modify: ");
scanf("%d", &id);

while (fread(&s, sizeof(s), 1, fp)) {
    if (s.id == id) {
        found = 1;
        printf("Enter new name: ");
        scanf(" %[^\n]s", s.name);
        printf("Enter new course: ");
        scanf(" %[^\n]s", s.course);
        printf("Enter new year: ");
        scanf(" %[^\n]s", s.year);
        printf("Enter new address: ");
        scanf(" %[^\n]s", s.address);
    }
    fwrite(&s, sizeof(s), 1, temp);
}

fclose(fp);
fclose(temp);

remove("students.txt");
rename("temp.txt", "students.txt");
```

```
    if (found)
        printf("Record updated successfully!\n");
    else
        printf("Record not found!\n");
}

// ----------------- DELETE STUDENT -----------------
void deleteStudent() {
    struct Student s;
    FILE *fp = fopen("students.txt", "rb");

    if (!fp) {
        printf("No student records found!\n");
        return;
    }

    FILE *temp = fopen("temp.txt", "wb");
    if (!temp) {
        printf("Error creating temp file!\n");
        fclose(fp);
        return;
    }

    int id, found = 0;
    printf("Enter ID to delete: ");
    scanf("%d", &id);

    while (fread(&s, sizeof(s), 1, fp)) {
```

```
        if (s.id != id)
            fwrite(&s, sizeof(s), 1, temp);
        else
            found = 1;
    }

    fclose(fp);
    fclose(temp);

    remove("students.txt");
    rename("temp.txt", "students.txt");

    if (found)
        printf("Record deleted successfully!\n");
    else
        printf("Record not found!\n");
}

// ----------------- RAISE COMPLAINT -----------------
void raiseComplaint(int stID) {
    struct Complaint c;
    FILE *fp = fopen("complaints.txt", "ab+");  // create file if not exists

    if (!fp) {
        printf("Error opening complaints file!\n");
        return;
    }

    c.student_id = stID;
```

```
    printf("Enter your complaint: ");
    scanf(" %[^\n]s", c.message);

    fwrite(&c, sizeof(c), 1, fp);
    fclose(fp);

    printf("Complaint submitted successfully!\n");
}

// ------------------ VIEW COMPLAINTS ------------------
void viewComplaints() {
    struct Complaint c;
    FILE *fp = fopen("complaints.txt", "rb");

    if (!fp) {
        printf("No complaints found!\n");
        return;
    }

    printf("\n========= ALL COMPLAINTS =========\n");

    while (fread(&c, sizeof(c), 1, fp)) {
        printf("\nStudent ID: %d\nComplaint: %s\n",
               c.student_id, c.message);
    }

    fclose(fp);
}
```

```
// ---------- VIEW OWN RECORD ----------
void viewOwnRecord(int stID) {
    struct Student s;
    FILE *fp = fopen("students.txt", "rb");

    if (!fp) {
        printf("No student records found!\n");
        return;
    }

    while (fread(&s, sizeof(s), 1, fp)) {
        if (s.id == stID) {
            printf("\n========= YOUR RECORD =========\n");
            printf("ID: %d\nName: %s\nCourse: %s\nYear: %s\nAddress: %s\n",
                   s.id, s.name, s.course, s.year, s.address);
            fclose(fp);
            return;
        }
    }

    printf("Record not found!\n");
    fclose(fp);
}
```

Functions Present in the Code:

Main System Functions

1. **main()**
    a. Displays main menu
    b. Takes user to Admin or Student section

Admin-Related Functions

2. **adminMenu()**
    a. Shows admin options (Add, View, Modify, Delete, Complaints)
3. **addStudent()**
    a. Adds a new student record
    b. Saves to **students.txt**
4. **viewStudents()**
    a. Displays all student records
    b. Reads from **students.txt**
5. **modifyStudent()**
    a. Edits an existing student record
    b. Uses a temp file to rewrite data
6. **deleteStudent()**
    a. Deletes a student record
    b. Copies all except target ID into a temp file
7. **viewComplaints()**
    a. Shows all student complaints
    b. Reads from **complaints.txt**

Student-Related Functions

8. **studentMenu(int stID)**
    a. Shows student-specific options
    b. Uses student ID from login
9. **viewOwnRecord(int stID)**
    a. Displays only the logged-in student's record
10. **raiseComplaint(int stID)**

- Lets a student submit a complaint
- Saves to **complaints.txt**

RESULT(Output)

```
========= STUDENT RECORD MANAGEMENT SYSTEM =========
1. Admin Login
2. Student Login
3. Exit
Enter your choice: 1

========== ADMIN MENU ==========
1. Add Student Record
2. View All Records
3. Modify Student Record
4. Delete Student Record
5. View Complaints
6. Back to Main Menu
Enter choice: 1

Enter ID: 101
Enter Name: John Smith
Enter Course: BSCS
Enter Year: 2nd Year
Enter Address: 123 Street, City
Record added successfully!

========== ADMIN MENU ==========
1. Add Student Record
2. View All Records
3. Modify Student Record
4. Delete Student Record
5. View Complaints
6. Back to Main Menu
Enter choice: 2

========= ALL STUDENT RECORDS =========

ID: 101
Name: John Smith
Course: BSCS
Year: 2nd Year
Address: 123 Street, City
```

```
========= STUDENT RECORD MANAGEMENT SYSTEM =========
1. Admin Login
2. Student Login
3. Exit
Enter your choice: 2
Enter Student ID: 101

========== STUDENT MENU ==========
1. View My Record
2. Raise Complaint
3. Back to Main Menu
Enter choice: 1

======== YOUR RECORD ========
ID: 101
Name: John Smith
Course: BSCS
Year: 2nd Year
Address: 123 Street, City

========== STUDENT MENU ==========
1. View My Record
2. Raise Complaint
3. Back to Main Menu
Enter choice: 2

Enter your complaint: Classroom projector not working.
Complaint submitted successfully!

========= STUDENT RECORD MANAGEMENT SYSTEM =========
1. Admin Login
2. Student Login
3. Exit
Enter your choice: 1

========== ADMIN MENU ==========
1. Add Student Record
2. View All Records
3. Modify Student Record
4. Delete Student Record
5. View Complaints
6. Back to Main Menu
Enter choice: 5
```

```
========= STUDENT RECORD MANAGEMENT SYSTEM =========
1. Admin Login
2. Student Login
3. Exit
Enter your choice: 3
Exiting...
```

CONCLUSION

The Student Record Management System (SRMS) developed using the C programming language successfully fulfills the primary objective of managing student information in a systematic, accurate, and efficient manner. This project provides a user-friendly platform to store, retrieve, update, and delete student records, thereby eliminating the drawbacks of a manual record-keeping system.

Through the implementation of core programming concepts such as structures, functions, file handling, loops, and conditional statements, the system ensures permanent data storage and easy accessibility of records. The use of file handling allows student data to be preserved even after the program is terminated, making the system reliable and practical for real-world applications.

The project also improves operational efficiency by reducing human errors, saving time, and ensuring better data organization. Furthermore, the menu-driven interface makes the application simple to use even for users with basic computer knowledge. This system can be effectively used in schools, colleges, and training institutes for managing student details.

In conclusion, the Student Record Management System is a successful implementation of a real-time application using C programming. It demonstrates how fundamental programming concepts can be applied to solve practical problems. With further enhancements such as a graphical user interface (GUI), database integration, and security features, this system can be upgraded into a fully functional professional-level application.

REFERENCE

1. E. Balagurusamy, Programming in ANSI C, 8th Edition, McGraw-Hill Education, New Delhi, 2019.

2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, 2nd Edition, Prentice Hall, 1988.

3. Yashavant P. Kanetkar, Let Us C, 16th Edition, BPB Publications, New Delhi, 2016.

4. S. Malik, C Programming: From Problem Analysis to Program Design, Cengage Learning, 2014.

5. Reema Thareja, Data Structures Using C, 2nd Edition, Oxford University Press, 2014.

6. Herbert Schildt, C: The Complete Reference, 4th Edition, McGraw-Hill, 2000.

7. Rajib Mall, Fundamentals of Software Engineering, PHI Learning, 2014.

8. Pressman, R. S., Software Engineering: A Practitioner's Approach, 7th Edition, McGraw-Hill, 2010.