PROJECT REPORT

STUDENT RECORD MANAGEMENT SYSTEM (SRMS)

Name: Aniket Mahto
Registration No.:AP24110011701
Section: J

1. Introduction

A Student Record Management System (SRMS) is a digital platform designed to store, manage, and organize student-related information efficiently. Traditional manual systems rely on physical registers or Excel sheets, which are prone to errors, data loss, and difficulty in retrieving information.

This project focuses on developing a **simple, console-based desktop application** using **C++**, implementing **Object-Oriented Programming (OOP)** and **File Handling**.

The system provides basic functionalities such as:

- Adding student records
- Viewing all students
- Searching specific student data
- Deleting a student record
- Persistently storing data in a `students.txt` file

The design aims to be lightweight, easy to use, and educational for understanding core C++ concepts.

2. Problem Statement

Educational institutions often maintain student data manually, which is:

- Time-consuming
- Difficult to update
- Vulnerable to loss or damage
- Hard to search and sort

This project solves these issues by providing a **simple, automated solution** that uses a C++ program and text file to store detailed records of students.

3. Objectives

The primary objectives of the Student Record Management System are:

✓ Functional Objectives

1. To allow the user to add new student records.
2. To display all student records in a structured format.
3. To search for a student using their ID number.
4. To delete a student record with confirmation.
5. To store all records in a text file (`students.txt`) for permanent storage.
6. To automatically load data at program start and save changes at exit.

✓ Educational Objectives

1. To implement Object-Oriented Programming in real-world applications.
2. To understand and apply file handling using C++ streams (`ifstream`, `ofstream`).
3. To practice exception handling, input validation, and modular design.
4. To build a console-based mini-project with proper menu-driven flow.

4. Scope of the Project

This project is designed for small-scale educational needs such as:

- College lab assignments
- Small institutions
- Student data tracking in departments
- Beginner-level management systems

The system is intentionally kept simple and lightweight.

It does **not** use databases like MySQL, making it easy for beginners.

5. System Requirements

Software Requirements

Component	Version
Programming Language	C++17 or above
Compiler	GCC / MinGW / Clang / CodeBlocks
IDE (optional)	VS Code / CodeBlocks
Operating System	Windows / Linux / macOS

Hardware Requirements

Component	Minimum Requirement
RAM	2 GB
Processor	Dual Core or higher
Storage	A few kB (for text file)

6. System Design

This project follows **Object-Oriented Design**, using two main classes:

A. Student Class

Responsibilities:

- Store a single student's information
- Take input from the user
- Display the student details
- Convert student data into formatted text for file storage

Attributes:

Field	Type	Description
`id`	int	Unique student ID
`name`	string	Student name
`age`	int	Age of the student
`major`	string	Course or branch
`gpa`	double	Grade Point Average

Methods:

- `void input()`
- `void display() const`
- `string toLine() const`
- `static bool fromLine(...)`

B. StudentManager Class

Responsibilities:

- Maintain a collection of students
- Provide menu-driven operations
- Handle add/search/delete functions
- Load/save data from/to the text file

Attributes:

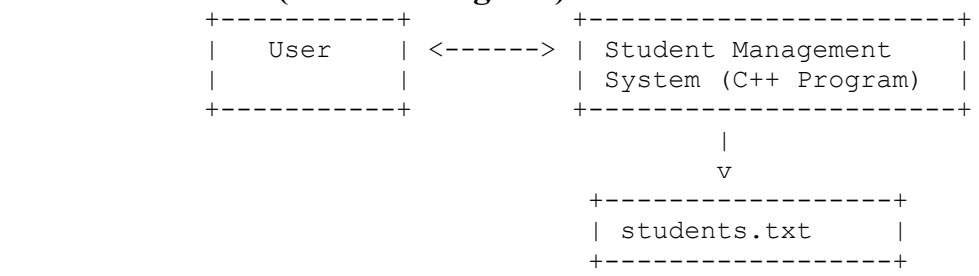
- `vector<Student> list`

Methods:

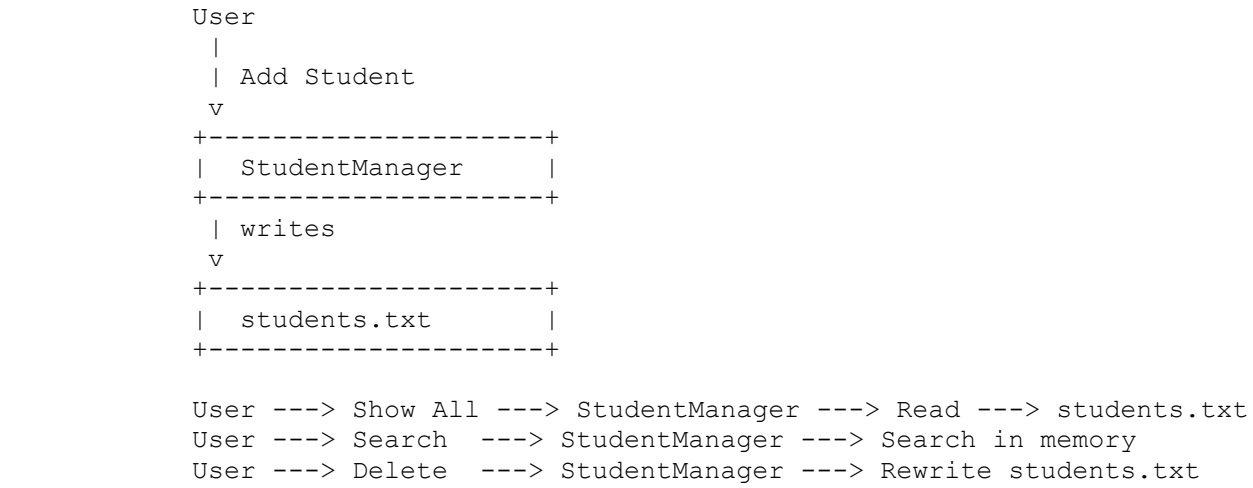
- `addStudent()`
- `showAll()`
- `searchStudent()`
- `deleteStudent()`
- `loadFromFile()`
- `saveOnExit()`

7. Data Flow Diagram (DFD)

Level 0 (Context Diagram)



Level 1 DFD



8. File Format

Each student record is stored in a text file using the format:

`ID|Name|Age|Major|GPA`

Example:

```
101|Rahul Sharma|19|CSE|8.75
102|Sneha Kapoor|20|IT|9.10
103|Aarav Roy|18|BCA|7.90
```

This format makes it easy to parse using `getline()` and `stringstream`.

9. Program Workflow

Startup

1. Load existing records from `students.txt`
2. Display main menu

Main Menu Choices

1. Add Student
2. Show All Students
3. Search Student
4. Delete Student
5. Exit

Exit

- Automatically save all records

10. Testing & Validation

Test Cases

Test Case	Input	Expected Result
Add Student	Correct information	Stored in file and displayed
Add Duplicate ID	ID already exists	Error message
Search Student	Valid ID	Correct student details displayed
Search Student	Invalid ID	"Student not found"
Delete Student	Valid ID + confirmation	Record removed from file
Show All	Multiple records	Displayed in table format

11. Advantages of SRMS

✓ From User Perspective

- Easy to use
- Simple menu system
- Fast searching
- Permanent data storage

✓ From Developer Perspective

- Good practice for OOP
- Demonstrates file handling
- Flexible for future enhancements
- No external database dependency

12. Limitations

- No update/edit functionality
- No role-based user authentication
- Not suitable for very large datasets
- Not a GUI-based system
- Data stored in plain text (not encrypted)

13. Future Enhancements

- Add update/modify student record
- Add sorting (by name, GPA, ID)
- Add login system
- Add admin/teacher roles
- Add CSV or database (MySQL/SQLite) storage
- Add GUI using Qt/GTK/wxWidgets
- Add pagination for large records
- Auto-generate IDs instead of manual entry

14. Conclusion

The Student Record Management System is a successful implementation of a small-scale database management tool using C++. The project effectively demonstrates the use of:

- Classes & Objects
- Vectors
- File Handling
- Menu-driven programming
- User-friendly input validation

This system is ideal for academic learning and can be extended into a full-fledged management system with more features.