C PROJECT

PROJECT MADE BY ARNAB ROY AND MEDHA BHATNAGAR

PROJECT ON STUDENT MANAGEMENT PROGRAM

# School Management System Project Report

## 1. Project Overview and Student Details

This project implements a console-based School Management System (SMS) designed to manage student records efficiently. The program utilizes the C programming language to demonstrate proficiency in foundational concepts such as data structures and persistent file storage.

|  |  |
| --- | --- |
| **Detail** | **Value** |
| **Student Name** | Arnab Roy, Medha Bhatnagar |
| **SAP ID** | 590023460, 590023682 |
| **Course** | First Semester, College Level C Programming |
| **Project File Name** | school\_management.c |

## 2. Core Technical Implementation

### A. Structure Definition (struct Student)

The core of the data management is the struct Student. This user-defined data type bundles all necessary student attributes into a single, cohesive unit. This approach streamlines data handling and ensures that a complete record is read or written with a single file operation.

**Data Fields:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Purpose** |
| roll\_number | int | Unique identifier used for searching, modifying, and deleting records. |
| name | char[] | Stores the student's full name. |
| student\_class | char[] | Records the student's current academic grade or class. |
| address | char[] | Detailed contact address. |
| total\_score | float | Academic performance score. |
| fee\_paid | bool | Financial status: true (Paid) or false (Not Paid). |

### B. Data Persistence via Binary File Handling

Data is permanently stored in the binary file student\_records.dat. Binary file handling is preferred over text files for structures in C because it allows entire records to be read and written quickly and accurately, preserving the structure's exact memory layout.

**Key File Functions:**

* **fopen():** Opens the file in specific modes ("ab", "rb", "r+b") based on the required operation (Append, Read, Read/Write).
* **fwrite():** Writes the entire struct Student (a block of memory) to the file.
* **fread():** Reads the entire struct Student from the file into a memory variable.
* **fseek():** Used in modify\_student() to precisely locate and jump back to a specific record's position for in-place overwriting.

## 3. Detailed Functionality Breakdown

The program operates using a menu-driven approach, where the main() function cycles through options managed by the following key functions:

|  |  |  |
| --- | --- | --- |
| **Function Name** | **File Operation Mode** | **Description of Operation** |
| main() | N/A | Manages the primary execution loop and menu flow, directing control to the selected functions until the user chooses to exit (Option 6). |
| **add\_student()** | ab (Append Binary) | Collects all data fields from the user, performs basic validation (e.g., ensuring Roll Number is a positive integer), and appends the new record to the end of student\_records.dat. |
| **view\_all\_students()** | rb (Read Binary) | Iterates through every record in the data file sequentially, displaying the details of each student using the print\_student\_data utility function. |
| **search\_student()** | rb (Read Binary) | Takes a Roll Number, reads records until a match is found, displays the matching student's details, and then stops reading the file. |
| **modify\_student()** | r+b (Read and Write Binary) | Locates a record by Roll Number. Once found, it prompts the user for updated values (Score and Fee Status). It then uses fseek() to reset the file pointer and fwrite() to overwrite the old data block with the new data. |
| **delete\_student()** | rb and wb | Implements deletion by copying all records *except* the targeted one from the original file to a temporary file (temp\_records.dat). It then removes the original file (remove()) and renames the temporary file (rename()) to replace it. |
| clear\_input\_buffer() | N/A | A necessary utility to clear leftover characters in the input stream, ensuring that input functions like fgets() work correctly after scanf(). |

4. PROBLEM STATEMENTS

**The Drawback:** You are using binary files (fread and fwrite), which are very fast but very strict. The program relies on reading the file in perfect blocks of sizeof(struct Student).

**The Implication:** If the program crashes while writing, or if someone opens and modifies the student\_records.dat file externally (even by adding a single extra character), the file's structure is ruined. The next time you run the program, fread() will read the wrong data block lengths, potentially treating part of a name as a score, and the entire database could become unreadable.

## 5. Acknowledgment (Corollary)

The successful development and robust implementation of data persistence techniques in this C project were highly dependent on the structured learning environment and expert guidance provided.

**Special Acknowledgment to Professor Vinod Kumar** for teaching the fundamental concepts that enabled the successful implementation of data structures and binary file handling, making this complete, functional project possible.