C Programming Project Report Template

TITLE PAGE

- **Project Title: STUDENT MANAGEMENT SYSTEM**
- **Course:** Programming in C (B.Tech 1st Semester)
- **Course Code: CSEG1041**
- **Submitted By:** PARAS RANA, 590027731
- **Submitted To:** MOHSIN F. DAR, SCHOOL OF COMPUTER SCIENCE
- **Date of Submission:** 24/11/2025

---

1. Problem Statement

The main problem this project tries to solve is keeping student records in one place without confusion. When data is written manually on paper or in scattered files, it becomes hard to update or find anything later. So the idea is to make a simple system in C where basic student details can be stored, searched and changed easily. This avoids mistakes and saves time by using a small digital record system instead of manual work.

---

2. Objective of the Project

* To build a simple program that can store and manage student details.
* To practise using C concepts like structures, functions, arrays, pointers and file handling.
* To create a small system where the user can add, edit, search or delete records.
* To learn how to save data permanently using text files.
* To understand how a real-life student database works in basic form.

---

3. Software / Tools Used

- Operating System: Windows 11
- IDE/Compiler: VS Code with GCC compiler

---

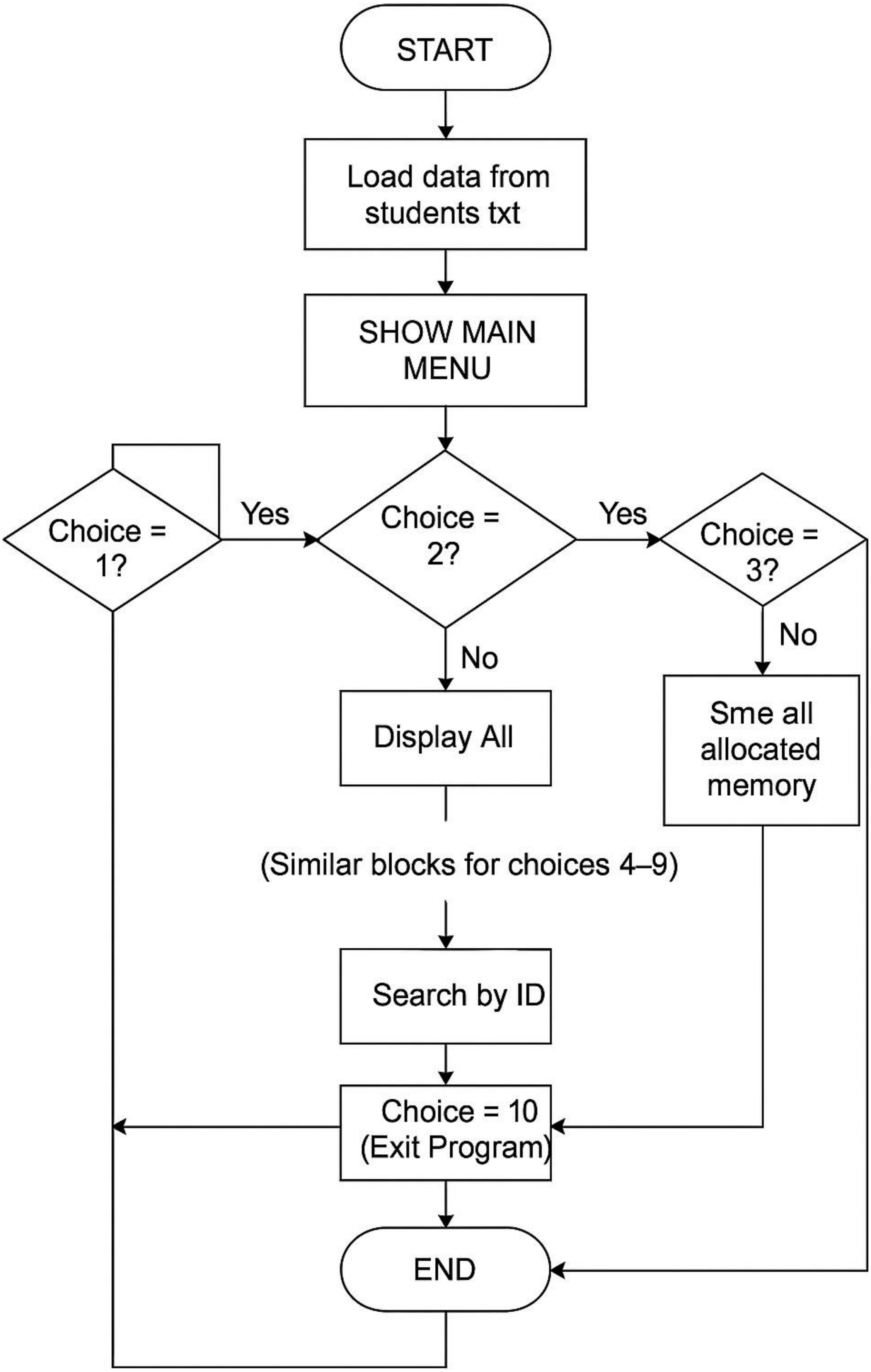
4. Algorithm

1. Start the program.
2. Load any old student data from the text file so the user can continue from last time.
3. Show the main menu on the screen.
4. Ask the user to choose an option.
5. Based on the option selected, call the correct function:
    - Add a new student
    - Display all students
    - Search a student by ID
    - Modify student details
    - Delete a student
    - Sort by ID or Name
    - Run extra demo functions
    - Show 2D marks table

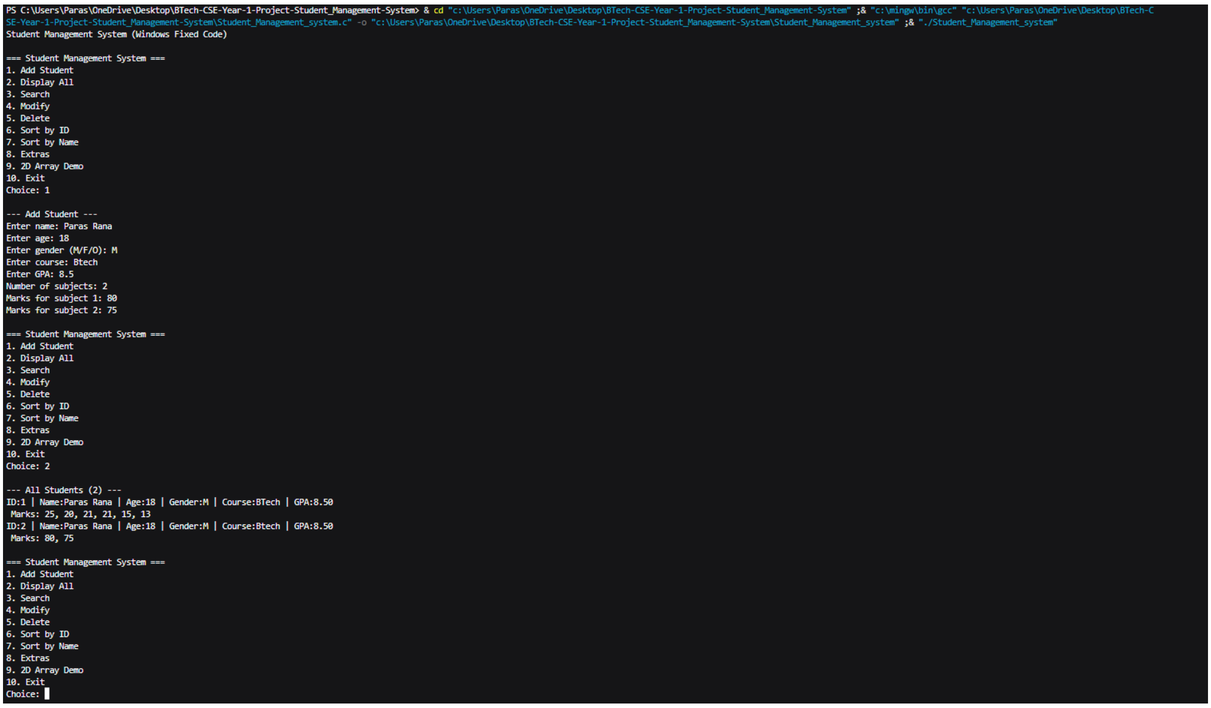
6. After performing the operation, save the updated data back to the file.
7. Show the menu again until the user chooses to exit.
8. Before ending, free all the memory used by the program.
9. Stop.

---

5. Flowchart / Pseudocode


START
Load data from students txt
SHOW MAIN MENU
Choice = 1?
Yes
Choice = 2?
Yes
Choice = 3?
No
Display All
No
Sme all allocated memory
(Similar blocks for choices 4–9)
Search by ID
Choice = 10 (Exit Program)
END

6. Output Screenshots



```
PS C:\Users\Paras\OneDrive\Desktop\BTech-CSE-Year-1-Project-Student_Management-System> & cd "c:\Users\Paras\OneDrive\Desktop\BTech-CSE-Year-1-Project-Student_Management-System" ;& "c:\mingw\bin\gcc" "c:\Users\Paras\OneDrive\Desktop\BTech-C
SE-Year-1-Project-Student_Management-System\Student_Management_system.c" -o "c:\Users\Paras\OneDrive\Desktop\BTech-CSE-Year-1-Project-Student_Management-System\Student_Management_system" ;& "./Student_Management_system"
Student Management System (Windows Fixed Code)

=== Student Management System ===
1. Add Student
2. Display All
3. Search
4. Modify
5. Delete
6. Sort by ID
7. Sort by Name
8. Extras
9. 2D Array Demo
10. Exit
Choice: 1

--- Add Student ---
Enter name: Paras Rana
Enter age: 18
Enter gender (M/F/O): M
Enter course: Btech
Enter GPA: 8.5
Number of subjects: 2
Marks for subject 1: 80
Marks for subject 2: 75

=== Student Management System ===
1. Add Student
2. Display All
3. Search
4. Modify
5. Delete
6. Sort by ID
7. Sort by Name
8. Extras
9. 2D Array Demo
10. Exit
Choice: 2

--- All Students (2) ---
ID:1 | Name:Paras Rana | Age:18 | Gender:M | Course:BTech | GPA:8.50
 Marks: 25, 20, 21, 21, 15, 13
ID:2 | Name:Paras Rana | Age:18 | Gender:M | Course:Btech | GPA:8.50
 Marks: 80, 75

=== Student Management System ===
1. Add Student
2. Display All
3. Search
4. Modify
5. Delete
6. Sort by ID
7. Sort by Name
8. Extras
9. 2D Array Demo
10. Exit
Choice: 
```

7. Source Code

```
/*
  Student Management System (Mini Project)
  ----------------------------------------
  This is my C programming mini project. I made it using the
  concepts we learned in class in all units (except Unit VI).
  I tried to keep the code simple so I can understand and
  explain it in viva.

  Name:  PARAS RANA
  SAP ID: 590027731
  Batch: B48
  College: UPES, Dehradun
  Submission Date: 20 Nov 2025
  Course: C Programming (1st Sem)
*/

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

#define DATAFILE "students.txt"
#define MAX_NAME 64
#define MAX_COURSE 48
```

```
#define LINE_SZ 512

/* enums for gender and a simple status */
typedef enum {
    GENDER_M = 'M',
    GENDER_F = 'F',
    GENDER_O = 'O'
} Gender;

typedef enum {
    OK = 0,
    ERR = 1
} Status;

/* union just to cover syllabus */
typedef union {
    int roll_number;
    char room[10];
} ExtraInfo;

/* structure */
typedef struct {
    int id;
    char name[MAX_NAME];
    int age;
    char gender;
    char course[MAX_COURSE];
    float gpa;

    int marks_count;
    int *marks;

    ExtraInfo extra;
} Student;

/* Globals */
Student *arr = NULL;
int count = 0;
int capacity = 0;

/* Function declarations */
void trim_newline(char *s);
void read_line(const char *prompt, char *buf, int n);
int read_int(const char *prompt);
float read_float(const char *prompt);
char read_gender(const char *prompt);

void ensure_capacity(void);
void free_student_marks(Student *s);

void load_from_file(void);
int save_to_file(void);

int find_index_by_id(int id);
int get_next_id(void);

void add_student(void);
void display_all(void);
void search_student(void);
void modify_student(void);
void delete_student(void);
void sort_by_id(void);
void sort_by_name(void);

void print_student(const Student *s);

long factorial(long n);
void pointer_demo(void);
void extras_demo(void);
void marks_matrix_demo(void);

void menu(void);

/* ---------------- INPUT HANDLING ---------------- */

void trim_newline(char *s) {
    int n = strlen(s);
    while (n > 0 && (s[n-1]=='\n' || s[n-1]=='\r' || s[n-1]==' '))
    {
        s[n-1] = '\0';
        n--;
    }
}

void read_line(const char *prompt, char *buf, int n) {
    printf("%s", prompt);
    if (fgets(buf, n, stdin) == NULL) {
        buf[0] = '\0';
        return;
    }
    trim_newline(buf);
}

int read_int(const char *prompt) {
    char tmp[64];
    while (1) {
        read_line(prompt, tmp, sizeof(tmp));
        if (strlen(tmp)==0) continue;

        char *end;
        long v = strtol(tmp, &end, 10);
        if (end == tmp || *end != '\0') {
            printf("Invalid integer.\n");
            continue;
        }
        return (int)v;
    }
}

float read_float(const char *prompt) {
    char tmp[64];
    while (1) {
        read_line(prompt, tmp, sizeof(tmp));
        if (strlen(tmp)==0) continue;

        char *end;
        float v = strtof(tmp, &end);
        if (end == tmp || *end != '\0') {
            printf("Invalid number.\n");
            continue;
        }
        return v;
    }
}

char read_gender(const char *prompt) {
    char tmp[8];
    while (1) {
        read_line(prompt, tmp, sizeof(tmp));
        char c = toupper((unsigned char)tmp[0]);
        if (c=='M' || c=='F' || c=='O') return c;
        printf("Enter M/F/O only.\n");
    }
}

/* ---------------- DYNAMIC ARRAY ---------------- */

void ensure_capacity(void) {
    if (capacity == 0) {
        capacity = 4;
        arr = malloc(sizeof(Student)*capacity);
    } else if (count >= capacity) {
        capacity *= 2;
        arr = realloc(arr, sizeof(Student)*capacity);
    }
}

void free_student_marks(Student *s) {
    if (s->marks) free(s->marks);
    s->marks = NULL;
    s->marks_count = 0;
}

/* ---------------- FILE HANDLING (WINDOWS SAFE) ---------------- */

void load_from_file(void) {

    FILE *fp = fopen(DATAFILE, "r");
    if (!fp) return;

    char line[LINE_SZ];

    while (fgets(line, sizeof(line), fp)) {

        trim_newline(line);
        if (strlen(line)==0) continue;

        Student s;
        s.marks = NULL;

        /* strtok version (Windows supports this) */
        char *tok;

        tok = strtok(line, "|");
        if (!tok) continue;
        s.id = atoi(tok);

        tok = strtok(NULL, "|");
        strcpy(s.name, tok);

        tok = strtok(NULL, "|");
        s.age = atoi(tok);

        tok = strtok(NULL, "|");
        s.gender = tok[0];

        tok = strtok(NULL, "|");
        strcpy(s.course, tok);

        tok = strtok(NULL, "|");
        s.gpa = atof(tok);
        tok = strtok(NULL, "|");
        s.marks_count = atoi(tok);

        tok = strtok(NULL, "|"); // marks list

        if (s.marks_count > 0 && tok != NULL) {
            s.marks = malloc(sizeof(int)*s.marks_count);

            int i = 0;
            char *mtok = strtok(tok, ",");
            while (mtok && i < s.marks_count) {
                s.marks[i++] = atoi(mtok);
                mtok = strtok(NULL, ",");
            }
        }

        ensure_capacity();
        arr[count++] = s;
    }

    fclose(fp);
}

int save_to_file(void) {
    FILE *fp = fopen(DATAFILE, "w");
    if (!fp) return ERR;

    for (int i=0; i<count; i++) {
        Student *s = &arr[i];

        fprintf(fp, "%d|%s|%d|%c|%s|%.2f|%d|",
                s->id, s->name, s->age, s->gender,
                s->course, s->gpa, s->marks_count);

        for (int j=0; j<s->marks_count; j++) {
            fprintf(fp, "%d", s->marks[j]);
            if (j < s->marks_count -1) fprintf(fp, ",");
        }
        fprintf(fp, "\n");
    }

    fclose(fp);
    return OK;
}

/* ---------------- UTILITIES ---------------- */

int get_next_id(void) {
    int maxid = 0;
    for (int i=0; i<count; i++)
        if (arr[i].id > maxid) maxid = arr[i].id;
    return maxid + 1;
}

int find_index_by_id(int id) {
    for (int i=0; i<count; i++)
        if (arr[i].id == id) return i;
    return -1;
}

/* ---------------- PRINT ---------------- */

void print_student(const Student *s) {
    printf("ID:%d | Name:%s | Age:%d | Gender:%c | Course:%s | GPA:%.2f\n",
           s->id, s->name, s->age, s->gender, s->course, s->gpa);

    if (s->marks_count > 0) {
        printf(" Marks: ");
        for (int i=0; i<s->marks_count; i++) {
            printf("%d", s->marks[i]);
            if (i < s->marks_count-1) printf(", ");
        }
        printf("\n");
    }
}

/* ---------------- CORE FUNCTIONS ---------------- */

void add_student(void) {
    Student s;
    s.id = get_next_id();
    s.marks = NULL;
    s.marks_count = 0;

    printf("\n--- Add Student ---\n");

    read_line("Enter name: ", s.name, MAX_NAME);
    s.age = read_int("Enter age: ");
    s.gender = read_gender("Enter gender (M/F/O): ");
    read_line("Enter course: ", s.course, MAX_COURSE);
    s.gpa = read_float("Enter GPA: ");

    int mcount = read_int("Number of subjects: ");
    if (mcount > 0) {
        s.marks = malloc(sizeof(int)*mcount);
        s.marks_count = mcount;
        for (int i=0; i<mcount; i++) {
            char p[40];
            sprintf(p, "Marks for subject %d: ", i+1);
            s.marks[i] = read_int(p);
        }
    }

    ensure_capacity();
    arr[count++] = s;
    save_to_file();
}

void display_all(void) {
    printf("\n--- All Students (%d) ---\n", count);
    for (int i=0; i<count; i++)
        print_student(&arr[i]);
}

void search_student(void) {
    int id = read_int("Enter ID: ");
    int idx = find_index_by_id(id);
    if (idx == -1) printf("Not found.\n");
    else print_student(&arr[idx]);
}

void modify_student(void) {
    int id = read_int("Enter ID to modify: ");
    int idx = find_index_by_id(id);
    if (idx == -1) { printf("Not found.\n"); return; }

    Student *s = &arr[idx];
    print_student(s);

    char buf[128];
    read_line("New name (blank = keep): ", buf, sizeof(buf));
    if (strlen(buf)>0) strcpy(s->name, buf);

    read_line("New age (blank = keep): ", buf, sizeof(buf));
    if (strlen(buf)>0) s->age = atoi(buf);

    read_line("New gender (blank = keep): ", buf, sizeof(buf));
    if (strlen(buf)>0) s->gender = buf[0];

    read_line("New course (blank = keep): ", buf, sizeof(buf));
    if (strlen(buf)>0) strcpy(s->course, buf);

    read_line("New GPA (blank = keep): ", buf, sizeof(buf));
    if (strlen(buf)>0) s->gpa = atof(buf);

    read_line("Change marks? (y/N): ", buf, sizeof(buf));
    if (buf[0]=='y' || buf[0]=='Y') {
        free_student_marks(s);
        int m = read_int("New number of subjects: ");
        if (m > 0) {
            s->marks = malloc(sizeof(int)*m);
            s->marks_count = m;
            for (int i=0; i<m; i++) {
                char p[40];
                sprintf(p, "Marks %d: ", i+1);
                s->marks[i] = read_int(p);
            }
        }
    }

    save_to_file();
}

void delete_student(void) {
    int id = read_int("Enter ID to delete: ");
    int idx = find_index_by_id(id);
    if (idx == -1) { printf("Not found.\n"); return; }

    print_student(&arr[idx]);

    char b[8];
    read_line("Confirm (y/N): ", b, sizeof(b));
    if (b[0]!='y' && b[0]!='Y') return;

    free_student_marks(&arr[idx]);

    for (int i=idx; i<count-1; i++)
        arr[i] = arr[i+1];
    count--;

    save_to_file();
}

int ci_cmp(const char *a, const char *b) {
    while (*a && *b) {
        char ca = tolower(*a);
        char cb = tolower(*b);
        if (ca != cb) return ca - cb;
        a++; b++;
    }
    return tolower(*a) - tolower(*b);
```

```
}

void sort_by_id(void) {
    for (int i=0; i<count-1; i++)
        for (int j=0; j<count-1-i; j++)
            if (arr[j].id > arr[j+1].id) {
                Student t = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = t;
            }
    display_all();
}

void sort_by_name(void) {
    for (int i=0; i<count-1; i++)
        for (int j=0; j<count-1-i; j++)
            if (ci_cmp(arr[j].name, arr[j+1].name) > 0) {
                Student t = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = t;
            }
    display_all();
}

/* ---------------- EXTRAS ---------------- */

long factorial(long n) {
    if (n <= 1) return 1;
    return n * factorial(n-1);
}

void pointer_demo(void) {
    int a=10, b=20;
    printf("Before: a=%d b=%d\n", a, b);
    int *pa = &a, *pb = &b;
    int tmp = *pa;
    *pa = *pb;
    *pb = tmp;
    printf("After: a=%d b=%d\n", a, b);
}

void extras_demo(void) {
    int n = read_int("Enter number: ");
    printf("Factorial = %ld\n", factorial(n));
    pointer_demo();
}

void marks_matrix_demo(void) {
    int rows = read_int("Students count: ");
    int cols = read_int("Subjects: ");
    if (rows<=0 || cols<=0) return;

    int *matrix = malloc(sizeof(int)*rows*cols);
    for (int i=0;i<rows;i++)
        for(int j=0;j<cols;j++){
            char p[40];
            sprintf(p,"S%d Sub%d: ",i+1,j+1);
            matrix[i*cols+j] = read_int(p);
        }

    printf("\nMatrix:\n");
    for (int i=0;i<rows;i++){
        printf("Student %d: ", i+1);
        for(int j=0;j<cols;j++)
            printf("%d ", matrix[i*cols+j]);
        printf("\n");
    }

    free(matrix);
}

/* ---------------- MENU ---------------- */

void menu(void) {
    while (1) {
        printf("\n=== Student Management System ===\n");
        printf("1. Add Student\n");
        printf("2. Display All\n");
        printf("3. Search\n");
        printf("4. Modify\n");
        printf("5. Delete\n");
        printf("6. Sort by ID\n");
        printf("7. Sort by Name\n");
        printf("8. Extras\n");
        printf("9. 2D Array Demo\n");
        printf("10. Exit\n");

        int c = read_int("Choice: ");

        if (c==1) add_student();
        else if (c==2) display_all();
        else if (c==3) search_student();
        else if (c==4) modify_student();
        else if (c==5) delete_student();
        else if (c==6) sort_by_id();
        else if (c==7) sort_by_name();
        else if (c==8) extras_demo();
        else if (c==9) marks_matrix_demo();
        else if (c==10) {
            for (int i=0;i<count;i++)
                free_student_marks(&arr[i]);
            free(arr);
            printf("Exiting...\n");
            break;
        }
        else printf("Invalid choice.\n");
    }
}

/* ---------------- MAIN ---------------- */

int main(void) {
    printf("Student Management System (Windows Fixed Code)\n");
    load_from_file();
    menu();
    return 0;
}
```

8. Conclusion

This project helped me understand how different concepts of C programming come together in a real working program. By creating the Student Management System, I got hands-on practice with structures, arrays, functions, pointers, dynamic memory and file handling. The program can store, update, search and display student data, and the use of a text file keeps the information saved even when the program is closed. Overall, the project gave me a good idea of how a small data-management system works and improved my confidence in writing modular C programs.

9. Future Enhancements

1) Adding a login or password system for better security.
2) Making the interface more user-friendly, maybe adding colors or a simple GUI.
3) Storing the data in a database or binary file for faster access.
4) Adding features like highest/lowest marks, average marks, ranking etc.
5) Giving an option to export student records to PDF or CSV files.