sTUDENT mANAGEMENT SYSTEM

IT161 PROJECT SEM1

 |

2024

**Indian Institute of Information Technology, Vadodara**



|  |  |
| --- | --- |
|  |  |

**IT161 PROJECT**

STUDENT MANAGEMENT SYSTEM

**Submitted by :**

Ishan Agrawal (202452321)

Sanskar Sharma (202451148)

Priyanshu Virani (202451169)

**Submitted to:**

Prof. Ravi Nahta

Prof. Swapnil Lokhande



**INDEX**

|  |  |
| --- | --- |
| S No. | TOPIC |
| 1 | NEED OF PROJECT |
| 2 | SOFTWARE AND HARDWARE REQUIREMENT |
| 3 | HOW TO RUN THE PROJECT |
| 4 | SOURCE CODE |
| 5 | OUTPUTS |
| 6 | A SMALL DISPLAY OF PROJECT FUNCTIONALITY |
| 7 | LIMITATIONS |
| 8 | CONCLUSION |

##### **NEED OF THE PROJECT**

1. **Student Data Management:** The system allows for the effective management of student data, including personal details (name, roll number, parents' names, room number of hostel), which is a fundamental requirement in any educational institution.
2. **Record Keeping:** The program enables the addition, retrieval of student records. This capability ensures that accurate and updated information is maintained, a critical aspect of educational administration.
3. **Report Card Generation:** The system facilitates the entry and management of student marks, ensuring that academic performance data can be uploaded and viewed as needed, supporting the creation of report cards or academic transcripts.
4. **User-Friendly Interface:** The interface design appears to guide users through various tasks, such as adding records, searching for student information and managing marks.
5. **Time Efficiency:** By automating these administrative tasks, the system aids in saving time that would otherwise be spent on manual data entry, ensuring efficiency in managing student-related information.

##### **SOFTWARE AND HARDWARE REQUIREMENTS**

≡ HARDWARE REQIUREMENTS

1. **Processor (CPU):** The system should ideally have a multi-core processor with a speed of at least 1.6 GHz or higher. A modern processor, such as an Intel Core i5 or AMD Ryzen 5 series, would handle the data processing tasks more efficiently. The number of cores is essential for multitasking and managing concurrent operations while running the system.
2. **Memory (RAM):** Sufficient memory is needed to run C and process data. The exact amount of RAM required depends on the volume of data being handled. As a guideline, a minimum of 4GB RAM is recommended, but larger datasets may require more RAM.
3. **Storage:** Adequate storage space is necessary to store the C code, CSV files, and any additional files or data associated with the system. Ensure enough disk space is available to accommodate data storage requirements.
4. **Input/Output Devices:** Standard input devices such as a keyboard and mouse, as well as an output device such as a monitor or display, are required for interacting with the system.

≡ SOFTWARE REQUIREMENTS

1. **Operating System:** The system should run on an operating system compatible with the programming language used (C in this case). Common choices include Windows, macOS, or Linux distributions.
2. **C:** Ensure C is installed on the system. The code provided seems to be written in C, so having a compatible version of C compiler installed is necessary.
3. **CSV Files:** The code reads and writes data from CSV files (details.csv and report.csv). Ensure these files are present in the specified paths or directories as indicated in the code.

**≡ HOW TO RUN THE PROGRAM**

1. **Install C:** If C is not installed on your system, download and install it from the [official C website](https://www.python.org/). Make sure to select the option to add C to PATH during installation.
2. **Download the Code and CSV Files:** Save the provided code to a directory on your computer. Ensure that the CSV files (**details.csv** and **report.csv**) are in the same directory as the C code.

**Running the Project:**

1. **Run the C Code:** Open a command prompt or terminal, navigate to the directory where the C code and CSV files are stored, and run the C script by entering:

Copy code

C filename.c

Replace **filename.c** with the name of the C file containing the project code.

1. **Interacting with the Project:** Once the code is running, follow the prompts displayed in the command line or terminal to interact with the Student Management System. The code provides options such as managing student details, uploading marks, displaying records.
2. **Performing Tasks:** Choose options as prompted by the system to perform various tasks like adding records, searching for student information, managing marks, based on the academic performance data.
3. **Exit the Program:** To exit the program, follow the provided prompts and choose the appropriate option (e.g., "Close Program," "Exit," or entering "0" when prompted to select an action).

**THE SOURCE CODE**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define INPUT(a) scanf("%d",&a);

//adding a record

void addRecord(int roll){

    FILE \*fptr;

    fptr = fopen("details.csv", "a");

    char stuname[50], fname[50], mname[50];

    int hosno;

    long long int contact;

    printf("Enter name:");

    scanf("%s", &stuname);

    printf("Enter Father's name:");

    scanf("%s", &fname);

    printf("Enter Mother's name:");

    scanf("%s", &mname);

    printf("Enter contact number:");

    scanf("%lld", &contact);

    printf("Enter room number:");

    INPUT(hosno)

    fprintf(fptr, "\n%d,", roll);

    fprintf(fptr, "%s,", stuname);

    fprintf(fptr, "%s,", fname);

    fprintf(fptr, "%s,", mname);

    fprintf(fptr, "%lld,", contact);

    fprintf(fptr, "%d", hosno);

    fclose(fptr);

    return;

}

//showing all records

void showRecords(){

    FILE\* file= fopen("details.csv","r");

    if (file == NULL) {

        printf("No records insterted\n");

        return;

    }

    // Read and print each character until EOF

    char ch;

    while ((ch = fgetc(file)) != EOF) {

        if(ch==','){

            ch='\t';

        }

        putchar(ch);

    }

    // Close the file

    fclose(file);

    return;}

//Searching a record

void searchRecord(int searchno){

    int rollno, Roomno, found = 0;

    char line[256];  // Buffer to hold each line

    char name[50], Fname[50], Mname[50], Contactno[13];

    // Open file in read mode

    FILE \*fptr2 = fopen("details.csv", "r");

    if (fptr2 == NULL){

        printf("NO RECORD INSERTED.\n");

        return;

    }

    // Read lines one by one

    while (fgets(line, sizeof(line), fptr2)) {

        // Parse the line

        if (sscanf(line, "%d,%49[^,],%d,%49[^,],%49[^,],%12s",

                   &rollno, name, &Roomno, Fname, Mname, Contactno) == 6) {

            // Check if this is the record we are looking for

            if (rollno == searchno) {

                printf("Record Found:\n");

                printf("Roll No: %d\n", rollno);

                printf("Name: %s\n", name);

                printf("Room No: %d\n", Roomno);

                printf("Father's Name: %s\n", Fname);

                printf("Mother's Name: %s\n", Mname);

                printf("Contact No: %s\n", Contactno);

                found = 1;

                break;

            }

        } else {

            printf("Warning: Could not parse line: %s", line);

        }

    }

    if (!found) {

        printf("No record found with Roll No: %d\n", searchno);

    }

    // Close the file

    fclose(fptr2);

    return;

}

//adding marks

void addMarks(int roll){

    FILE \*fptr2;

    fptr2 = fopen("report.csv","a");

    int PH100Marks;

    int EC100Marks;

    int IT101Marks;

    int HS101Marks;

    int MA101Marks;

    printf("Enter IT101 marks:");

    scanf("%d", &IT101Marks);

    printf("Enter PH100 marks:");

    scanf("%d", &PH100Marks);

    printf("Enter HS101 marks:");

    scanf("%d", &HS101Marks);

    printf("Enter MA101 marks:");

    scanf("%d", &MA101Marks);

    printf("Enter EC100 marks:");

    scanf("%d", &EC100Marks);

    fprintf(fptr2, "\n%d,", roll);

    fprintf(fptr2, "%d,", IT101Marks);

    fprintf(fptr2, "%d,", PH100Marks);

    fprintf(fptr2, "%d,", HS101Marks);

    fprintf(fptr2, "%d,", MA101Marks);

    fprintf(fptr2, "%d", EC100Marks);

    fclose(fptr2);

    printf("Marks added successfully\n");

    return;

}

//showing all marks

void showMarks(){

    FILE\* file= fopen("report.csv","r");

    if (file == NULL) {

        printf("No records insterted\n");

        return;

    }

    // Read and print each character until EOF

    char ch;

    while ((ch = fgetc(file)) != EOF) {

        if(ch==','){

            ch='\t';

        }

        putchar(ch);

    }

    // Close the file

    fclose(file);

    return;

}

//Searching marks of a student

void searchMarks(int roll){

    typedef struct {

    int rollno;

    int IT101;

    int PH100;

    int HS101;

    int MA101;

    int EC100;

    } StudentMarks;

    StudentMarks marks;  // Instance of the structure

    FILE \*file = fopen("report.csv", "a");  // Open the file for reading

    if (file==NULL) {

        printf("Error: Could not open marks file.\n");

        return;

    }

    int found = 0;

    // Read data using fscanf in a loop

    while (fscanf(file, "%d,%d,%d,%d,%d,%d",

                  &marks.rollno, &marks.IT101, &marks.PH100,

                  &marks.HS101, &marks.MA101, &marks.EC100) == 6) {

        if (marks.rollno == roll) {

            printf("\nMarks Found:\n");

            printf("Roll No: %d\n", marks.rollno);

            printf("IT101: %d\n", marks.IT101);

            printf("PH100: %d\n", marks.PH100);

            printf("HS101: %d\n", marks.HS101);

            printf("MA101: %d\n", marks.MA101);

            printf("EC100: %d\n", marks.EC100);

            found = 1;

            break;

        }

    }

    if (!found) {

        printf("No marks found for Roll No: %d\n", roll);

    }

    fclose(file);  // Close the file

    return;

}

//main interface

int main(){

    while(1){

        while(1){//taking task

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

            printf("1.Student details.\n");

            printf("2.Student report.\n");

            printf("3.Close program\n");

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

            int ch;

            printf("Which platform you want to use:");

            scanf("%d",&ch);

            //Student details section

            if(ch==1){

                printf("\n\t\t=============================\n");

                printf("\t\tWelcome to student details section\n");

                printf("\n\t\t=============================\n");

                while(1){

                        //Open student details section

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

                        printf("1. Add Record\n");

                        printf("2. Show all Record\n");

                        printf("3. Search Record\n");

                        printf("0. Exit\n");

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

                        //taking task

                        int ch;

                        printf("Enter task to perform:");

                        scanf("%d",&ch);

                        //adding record

                        if(ch==1){

                            int rollno;

                            printf("Enter roll number:");

                            scanf("%d",&rollno);

                            addRecord(rollno);

                        }

                        //showing records

                        else if(ch==2){

                            printf("\n\n");

                            showRecords();

                        }

                        //searching record

                        else if(ch==3){

                            int rollno;

                            printf("Enter roll no for search:");

                            scanf("%d", &rollno);

                            searchRecord(rollno);

                        }

                        //Exiting

                        else if(ch==0)

                            break;

                        else{

                            printf("\n\nWrong choice entered\n");

                            continue;

                            }

                }

            }

            //Student report section

            else if(ch==2){

                printf("\n\t\t========================================\n");

                printf("\t\tWelcome to student report section\n");

                printf("\t\t========================================\n");

                while(1){

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

                    printf("1. Add  marks of a student\n");

                    printf("2. Show marks of all students \n");

                    printf("3. Search marks of a studnet\n");

                    printf("0. Exit\n");

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

                    //taking task

                    int ch;

                    printf("Enter task to perform:");

                    scanf("%d",&ch);

                    if(ch==1){

                        int rollno;

                        printf("Enter roll number:");

                        INPUT(rollno);

                        addMarks(rollno);

                    }

                    //showing records

                    else if(ch==2){

                        printf("\n\n");

                        showMarks();

                    }

                    //searching record

                    else if(ch==3){

                        int rollno;

                        printf("Enter roll no for search:");

                        INPUT(rollno);

                        searchMarks(rollno);

                    }

                    //Exiting

                    else if(ch==0){

                        break;

                    }

                    else{

                        printf("\n\nWrong choice entered\n");

                        continue;

                    }

                }

            }

            //exiting the program

            else if(ch==3){

                break;

            }

            else{

                printf("\n\n\*\*\*\*\*\*Wrong choice entered\*\*\*\*\*\n\n");

                continue;

            }

            }

            break;

        }

    return 0;

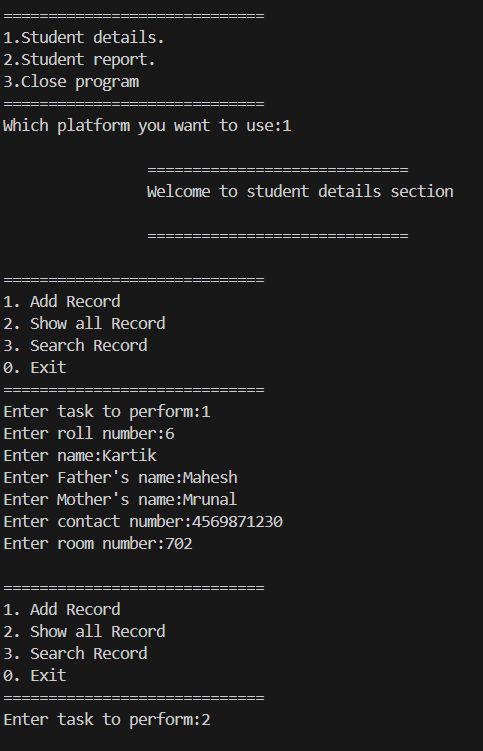
}

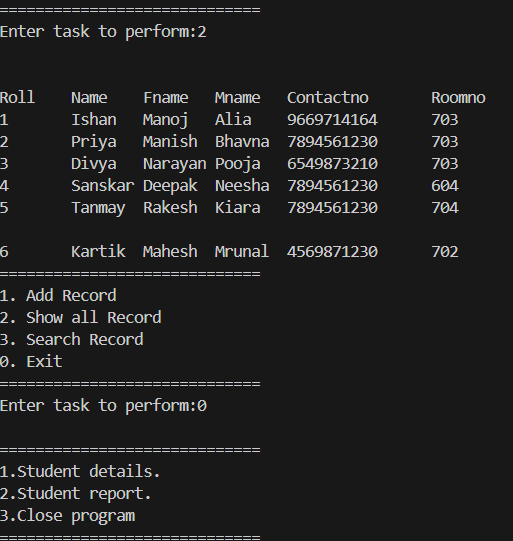
**OUTPUTS :-**

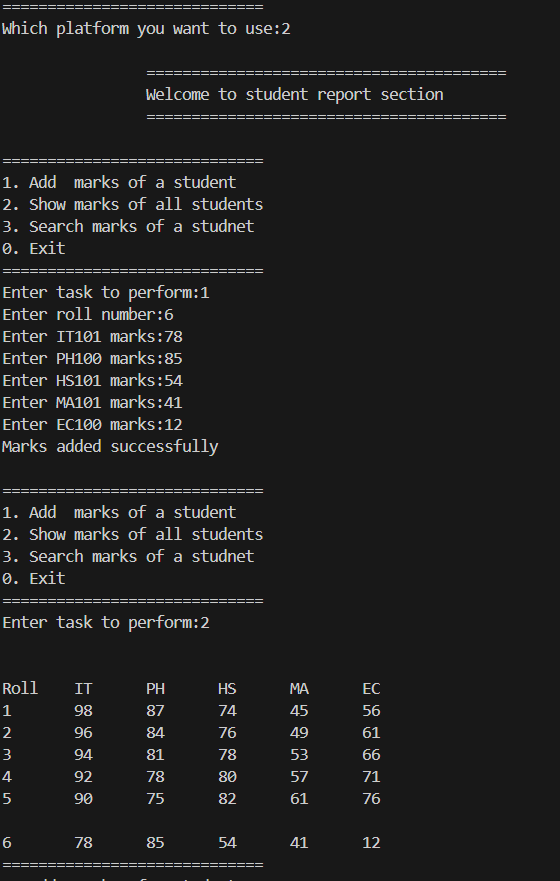
The outputs or results you can expect from running the Student Management System code provided earlier will depend on the functionalities implemented within the code. Here are the potential outputs based on the actions you perform within the program:

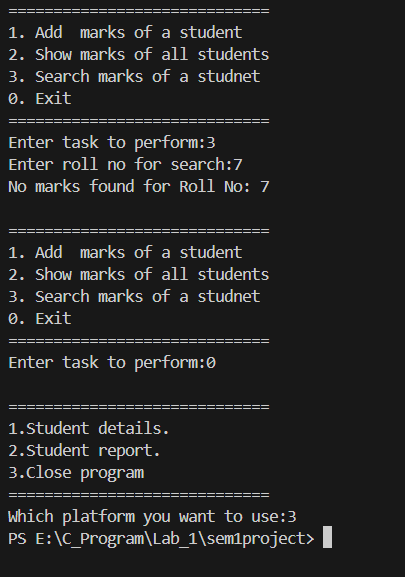
1. **Displaying Student Details:**
   * Output: Displayed student information including Roll Number, Name, Father's Name, Mother's Name, and Room number. This output could be shown on the command line or terminal.
2. **Adding New Records:**
   * Output: Confirmation messages indicating the successful addition of a new student record to the system. The updated student details will be written to the CSV file (**details.csv**).
3. **Viewing All Records:**
   * Output: Displaying all student records available in the CSV file (**Details.csv**). This output will show all existing student details stored in the system.
4. **Searching for Specific Records:**
   * Output: Upon searching by Roll Number the program will display the specific student record(s) matching the search criteria.
5. **Uploading Marks:**
   * Output: For uploading marks for a specific subject and student, the program will confirm the successful addition of marks to the **Report.csv** file.
6. **Showing Marks:**
   * Output: Displaying the marks obtained by students in various subjects as stored in the **Report.csv** file.

**A small display of project:**

****

****

****

****

**LIMITATIONS**

Limitations of the Student Management System implemented with CSV files:

1. **Limited Storage Capacity:** CSV files may become slow or inefficient when handling a large number of student records. They might not handle extensive data well.
2. **Security Risks:** Storing sensitive student information in plain text (CSV) files can pose security risks, as there are no built-in security measures like encryption or access controls.
3. **Difficulty in Data Integrity:** Ensuring data consistency and preventing errors or duplication can be more challenging in CSV files compared to using a structured database.
4. **Details.csv** and **Report.csv** must exist in your device

**CONCLUSION :-**

key points can be summarized as the conclusion of this project:

1. **Functionality:** The project successfully demonstrates functionalities such as adding, student records, managing academic marks, displaying information, and generating basic graphical reports.
2. **Limitations:** The project's reliance on CSV files as a data storage method imposes limitations in terms of scalability, security, and concurrent access. It lacks advanced database features and a user-friendly interface.
3. **Improvement Areas:** To enhance the system's effectiveness, future iterations or improvements might focus on migrating to a database system for better scalability, implementing security measures to safeguard student data, developing a more user-friendly interface (potentially through a graphical user interface), and addressing issues related to concurrent access and data integrity.
4. **Learning Outcomes:** This project serves as a valuable learning experience for understanding data handling in C, and creating a basic command-line interface for managing student information.
5. **Future Development:** Further development could involve expanding functionalities, incorporating additional modules for tasks like fee management or attendance tracking, and ensuring the system's compatibility with larger datasets and multiple users.

In conclusion, while the current project showcases fundamental functionalities for managing student records, there is room for enhancement and refinement to create a more robust, secure, and user-friendly Student Management System suitable for broader educational institution usage**.**

Top of Form

**SOME LINKS:-**

1. Link of google drive containg code in .c format and related files:

<https://drive.google.com/drive/folders/1NbYHtg6HD6pm5B3ZWOUZ_RvPKNYEuVaM?usp=sharing>

1. Our Email id’s:

Ishan Agrawal: [202452321@iiitvadodara.ac.in](mailto:202452321@iiitvadodara.ac.in)

Priyanshu Virani: [202451169@iiitvadodara.ac.in](mailto:202451169@iiitvadodara.ac.in)

Sanskar Sharma: [202451148@iiitvadodara.ac.in](mailto:202451148@iiitvadodara.ac.in)