STUDENT MANAGEMENT SYSTEM

BY - MANNAT ARORA
SAP ID - 590028068
BATCH - 60
SUBMITTED TO - MR. PRASHANT TRIVEDI

OBJECTIVE

To create a C program that stores and manages student records efficiently.

FEATURES

- Add, view, search and delete records.
- Uses file handling for permanent storage.

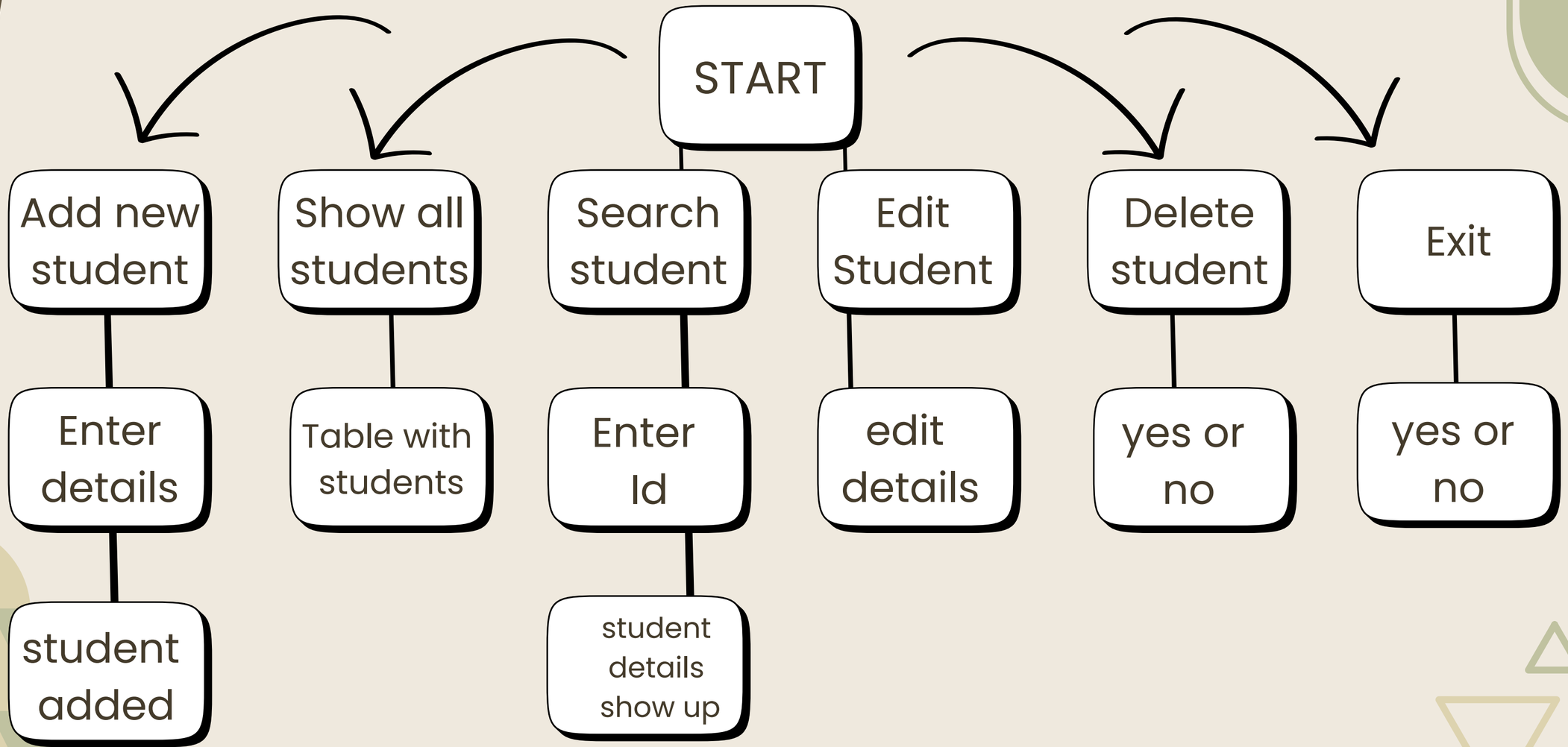
ABSTRACT

The Student Management System in C is a simple console-based program that stores and manages student details using structures and file handling. It allows users to add, view, search, update, and delete student records. The project demonstrates key C concepts such as structures, pointers, functions, and dynamic memory allocation, providing an efficient way to organize student information.

PROBLEM DEFINATION

Manual handling of student records is slow and error-prone. There is a need for a simple system that can store, search, update, and manage student information efficiently. The Student Management System in C solves this by providing an organized and fast way to handle student data.

FLOWCHART


START
Add new student
Enter details
student added
Show all students
Table with students
Search student
Enter Id
student details show up
Edit Student
edit details
Delete student
yes or no
Exit
yes or no

IMPLEMENTATION

1. Menu

As the code is run, the first thing that appears is the menu with many options, you type the option number then press enter for further process.

```
void Menu()
{
printf("\n\n\t**Student Man
agement System Using C\n\n"
);
printf("\t\t\tMAIN MENU\n");
printf("\t\t======================\n");
printf("\t\t[1] Add A New Student.\n");
printf("\t\t[2] Show All Students.\n");
printf("\t\t[3] Search A student.\n");
printf("\t\t[4] Edit Student.\n");
printf("\t\t[5] Delete Student.\n");
printf("\t\t[6] Delete All Students.\n");
printf("\t\t[7] User Guideline.\n");
printf("\t\t[0] Exit the Program.\n");
printf("\t\t======================\n");
printf("\t\tEnter Your Choice: ");
printf("\t\tEnter The Choice: ");
}   // end menu
```

IMPLEMENTATION

2. Options -

2.a Add new student

The user is asked about the details of the new student, just enter them and press enter for the details to be saved.

```
void AddNewStudent()
{
    char StudentID[300];
    char Name[300];
    char Phone[300];
    char Email[300];
    int NumberOfCourses;
    char CourseCode[300];
    char CourseName[300];
```

2.b show all students

List of all students is shown in tabular form.

```
void ShowAllStudents()
{
    printf("|==========|
====================|
============================|
====================|=============|\n");
    printf("|    ID    |
        Name        |
            Email             |
       Phone        |  NO.Course  |\n");
    printf("|==========|
====================|
============================|
====================|=============|\n");
```

IMPLEMENTATION

2.c edit student

The user can edit details of the any student after selecting it.

2.d delete student

Delete the selected students details from the table.

```
void DeleteStudent(int StudentIndex)
{
    int d;
    int FirstCourseIndexs;
    struct StudentInfo ThisStudents;
    ThisStudents = Students[StudentIndex];
    for(d=0; d<TotalCourse; d++)
    {
        if(strcmp(ThisStudents.ID,Courses[d
].StudentID) == 0)
        {
            FirstCourseIndexs = d;
            break;
        }
    }
    for(d=1; d<=ThisStudents.NumberOfCourse
; d++)
    {
        DeleteCourseByIndex(
FirstCourseIndexs);
    }
    DeleteStudentByIndex(StudentIndex);
    printf(" Student Deleted Successfully.
\n\n");
    GoBackOrExit();
}
```

```
void EditStudent(int StudentFoundIndex)
{
    printf("\n\t\t **** Upda
te The New Student ****\n\n"
);

    char NewName[300];
    char NewPhone[300];
    char NewEmail[300];
    int NewNumberOfCourses;
    char StudentID[300];
    strcpy(StudentID, Students[
StudentFoundIndex].ID);
    int OldTotalNumberOfCourse = Students[
StudentFoundIndex].NumberOfCourse;

    int IsValidName = 0;
    while(!IsValidName)
    {
        printf(
" Enter The New Name(0 for skip): ");
        scanf(" %[^\n]s",&NewName);
        if(strlen(NewName) > 20)
        {
            printf(" Error: Name can
not be more than 20 characters.\n\n"
);
            IsValidName = 0;
        }
        else if(strlen(NewName) <= 0)
```

IMPLEMENTATION

2.e exit or go back

The user can go back to the program after doing any of the functions mentioned before or can even straight up exit the program any time.

```
void GoBackOrExit()
{
    getchar();
    char Option;
    printf(" Go back(b)? or Exit(0)?: ");
    scanf("%c",&Option);
    if(Option == '0')
    {
        ExitProject();
    }
    else
    {
        system("cls");
    }
}

void ExitProject()
{
    system("cls");
    int i;
    char ThankYou[100]     =
" ========= Thank You =========\n";
    char SeeYouSoon[100]   =
" ========= See You Soon ======\n";
    for(i=0; i<strlen(ThankYou); i++){
    printf("%c",ThankYou[i]);
        Sleep(40);
        }
    for(i=0; i<strlen(SeeYouSoon); i++){
            printf("%c",SeeYouSoon[i]);
    sleep(40);
    }
```

CONCLUSION

The Student Management System in C provides a simple and efficient way to store, manage, and retrieve student records. It reduces manual work, minimizes errors, and demonstrates key programming concepts like structures, file handling, and functions.

FUTURE WORK

The system can be improved by adding features like password protection, sorting of records, graphical interface, database integration (MySQL), and support for multiple classes or departments.

REFRENCES

1.Study material provided by the teacher.
2. Online C documentation.
(www.programiz.com
online C tutorials)

THANK YOU