Plan for School management project.

1. **Defining the structures (Structs)**
    - Creating a struct that contains the student ID, name, Array of subjects they study
    - Creating a struct that Contains teacher ID, name, and ID of the subject the teach.
    - **Subject:** Contains name of the subject and Subject ID

2. **Core Functionalities**
    - **Adding a student:**
        - Functionality to add a new student (Create a new instance of Student struct)
        - We add a new student using information provided in the command line (Allocate memory to store a new student ask user for the name of student using 'ScanF' and store it in the new instance of student created and we'll do the same for other fields like ID and the Array of subject then we'll be returning the newStudent instance in the function)

    - **Adding other fields:**
        - We'll repeat the same procedures for adding a student to match the requirements for adding teachers and subjects.

    - Adding a teacher:
        - Functionality to add a new teacher (Create a new instance of the Teacher struct).
        - Allocate memory to store a new teacher and prompt the user for the teacher's name using scanf. Store this in the new instance of the teacher.
        - Similarly, obtain and store the teacher's ID and the subject ID they teach.
        - Return the new teacher instance from the function.

    - Adding a subject:
        - Functionality to add a new subject (Create a new instance of the Subject struct).
        - Allocate memory to store a new subject and prompt the user for the subject's name using scanf. Store this in the new instance of the subject.
        - Obtain and store the subject's ID.
        - Return the new subject instance from the function.

3. **Assigning Subjects and Grades**

    - Assign Subjects to students:
        - Create a function that allows assigning subjects to a student.
        - The function should take the student's ID and the subject ID as inputs.
        - Update the student's record to include the new subject ID in their list of subjects.

    ```
    void assignSubjectToStudent(Student *student, int subjectId) {
      // Add subjectId to student's subjectIds array
    }
    ```

    - Assign Grades to students:
        - Implement a function to assign grades for a specific subject to a student.
        - The function should take the student's ID, subjects ID, and the grade as inputs.
        - Locate the student and the specific subject, then update the grade for that subject.

    ```
    void assignGradeToStudent(Student *student, int subjectId, float grade) {
      // Find the subjectId in the student's subjectIds array and assign the grade
    }
    ```

4. **Querying information**
    - Finding Students by subject:

        - Develop a function that given a Subject ID, returns a list of all students Studying that subject.
        - Iterate over the array of students and check their list of subjects to see if it includes the specified subject.
        - Print/Display: It prints or displays information about students who study a specific subject.

```
void findStudentsBySubject(Student *students, int numStudents, int subjectId) {
   for (int i = 0; i < numStudents; i++) {
      if (/* student[i] studies subjectId */) {
         // Display student information
      }
   }
}
```

    - Find which teacher teaches a specific subject:
        - Implement a function that given a subject ID, finds the teacher who teaches that subject.
        - Also Iterate over the array of teacher to find the one whose subject ID matches the given ID.

```
Teacher findTeacherForSubject(Teacher *teachers, int numTeachers, int subjectId)
 {
    //Iterating through the array of teachers to find the one whose subject ID matches the given ID
   for (int i = 0; i < numTeachers; i++) {
      //returns teachers[i] if teacher is found
   }
   // Return a default or a string saying teacher is not found
}
```

    - Find Grades of a student in a specific subject:
        - Create a function that takes a student's ID and a subject ID and returns the grade the student achieved in that subject.
        - It'll simply Locate the student and then search through their subjects to find the matching subject ID and its corresponding grade.

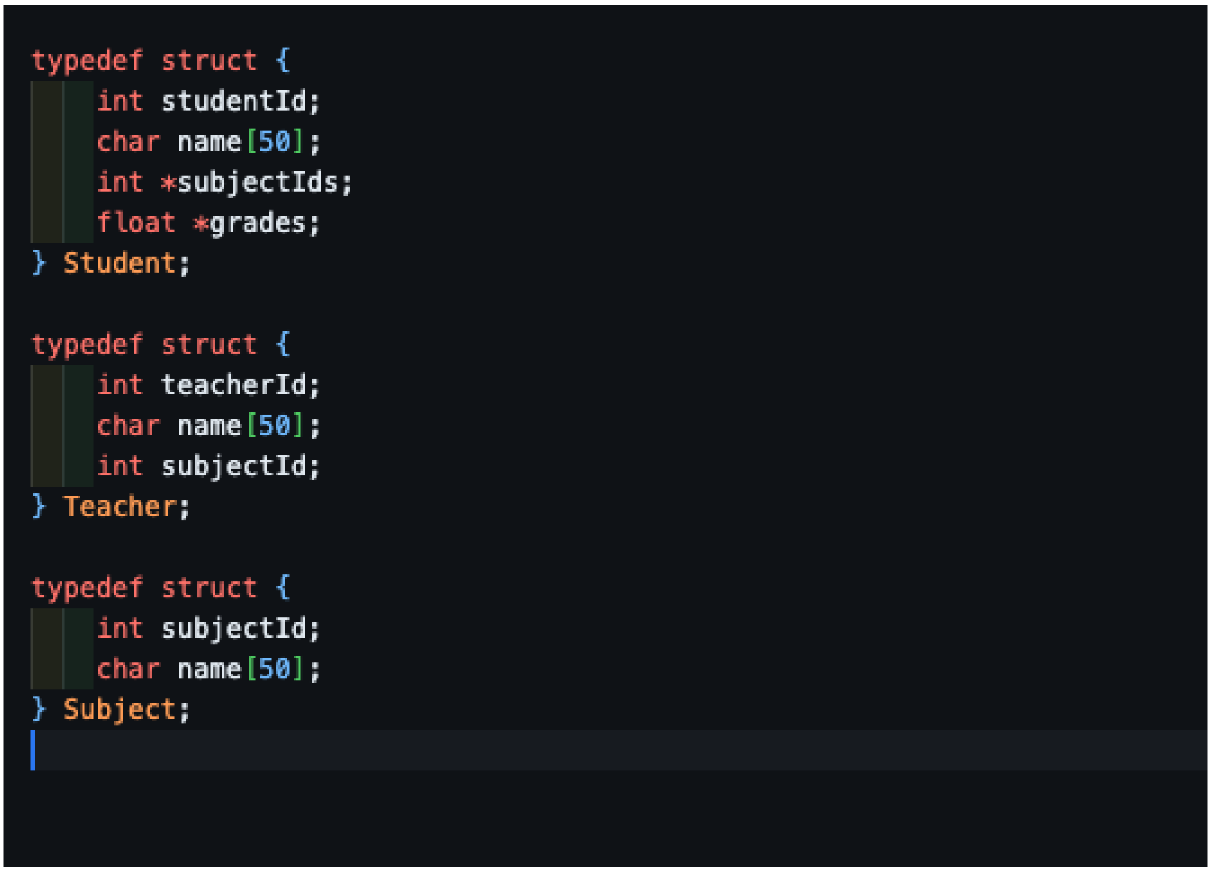
```
float findStudentGradeInSubject(Student student, int subjectId) {
   // Search for subjectId in student's subjectIds array and return the corresponding grade
}
```

5. User Interface(inputs and output)
    - Implementing the Command Line Interface (CLI):
        - Developing a text-based interface that takes input from the user through the command line.
        - Implement a menu with options to add students, teachers, subjects, assign subjects, assign grades, and query information.
        - Using scanf for input and printf for displaying information.

6. The main Function
    - Initializing any required data structure
    - Entering a loop that displays the main menu to the user and handles the user's input by invoking the relevant functions based on their choice and input .
    - Ensuring that the program can exit gracefully and freeing up any allocated memory.
7. Documentation

- Providing clear documentation by commenting Code and explaining all of my written functionalities and logic.
- Optionally writing a readme file explaining how to use the program

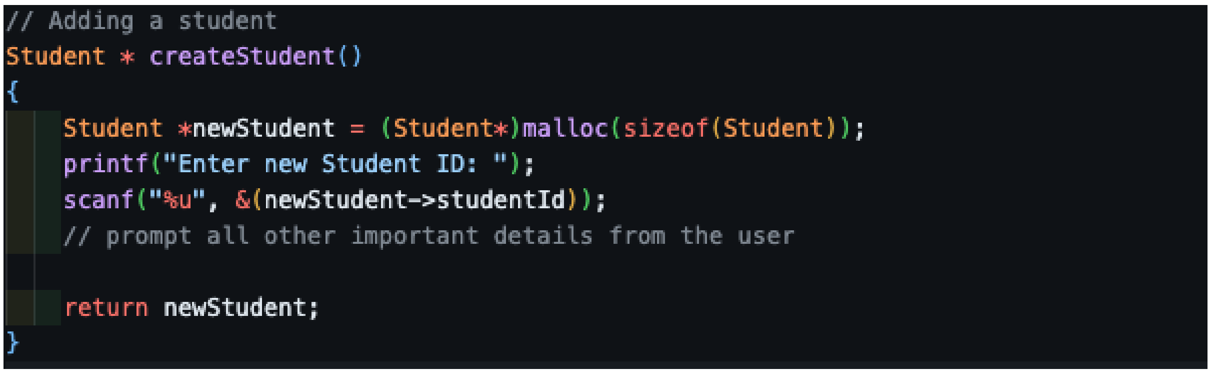
Relevant Screen Shots of code snippets



```c
typedef struct {
    int studentId;
    char name[50];
    int *subjectIds;
    float *grades;
} Student;

typedef struct {
    int teacherId;
    char name[50];
    int subjectId;
} Teacher;

typedef struct {
    int subjectId;
    char name[50];
} Subject;
```



```c
// Adding a student
Student * createStudent()
{
    Student *newStudent = (Student*)malloc(sizeof(Student));
    printf("Enter new Student ID: ");
    scanf("%u", &(newStudent->studentId));
    // prompt all other important details from the user

    return newStudent;
}
```