

# Student Management System :

## Problem Statement:

A **Student Management System** is required to efficiently organize and manage student information in an academic environment. The system should support various operations such as adding, removing, searching, updating, and displaying student records. It should also provide analytical features to evaluate student performance based on **marks** and **attendance**. The system manages key student attributes including **Student ID, Name, Class, Marks, and Attendance**.

---

## Key Requirements:

- **Add Student:** The system should allow users to add new students with details such as Student ID, Name, Class, Marks, and Attendance.
  - **Remove Student:** Users should be able to delete a student record using the Student ID, ensuring proper data shifting and integrity.
  - **Search Student:** The system should provide a feature to search for a student using their unique Student ID and display complete details.
  - **Update Student Data:** Users must be able to update student information including Class, Marks, and Attendance to keep records accurate and up to date.
  - **Display Attendance Analysis:** The system should display students with **maximum and minimum attendance**, helping identify attendance trends.
  - **Display Marks Analysis:** The system should provide options to display **highest marks, lowest marks, and average marks** of students.
  - **Display All Students:** The system should allow users to view all student records in a structured tabular format.
- 

## Project Creation Guide (Step-by-Step)

### Step 1: Problem Statement

Design and develop a **Student Management System in C** that can store, manage, and analyze student records such as ID, name, class, marks, and attendance using a menu-driven approach.

---

## Step 2: System Requirements

### Hardware Requirements

- Computer / Laptop
- Minimum 2 GB RAM

### Software Requirements

- Operating System: Windows / Linux
  - Compiler: GCC / Turbo C
  - Text Editor / IDE: VS Code, Code::Blocks, Dev-C++
- 

## Step 3: Program Design

### a) Data Design

- Use `struct student` to group student attributes
- Use an array of structures to store records

### b) Functional Design

The project is divided into modules: - Add Student - Display Student - Update Student - Delete Student - Search Student - Attendance Analysis - Marks Analysis

---

## Step 4: Algorithm

### Add Student Algorithm

1. Read number of students
2. Loop through count
3. Accept student details
4. Store data in structure array
5. Increment index

### Delete Student Algorithm

1. Read student ID
2. Search for ID
3. Shift array elements
4. Reduce record count

### Update Student Algorithm

1. Read student ID
  2. Search student
  3. Update selected field
-

## Step 5: Flow of Execution

```
Start
↓
Display Menu
↓
User Choice
↓
Perform Operation
↓
Repeat Until Exit
```

## Step 6: Testing

Test Case	Input	Expected Output
Add Student	Valid data	Student added
Delete Student	Valid ID	Student deleted
Search Student	Invalid ID	Error message

## Step 7: Result

The project successfully performs all student record management operations using C programming concepts such as structures, arrays, and functions.

## Step 8: Learning Outcomes

- Practical understanding of structures
- Function modularization
- Menu-driven programming
- Real-world data handling

## Step 9: Applications

- School Management Systems
- College Record Systems
- Academic Projects

## Step 10: Conclusion

This project demonstrates how **C programming** can be used to build an efficient data management system using structured and modular programming techniques.

---

**End of Document**