# Student Management System Documentation

## Introduction

The Student Management System is a C-based program designed to efficiently manage student information and academic records. It provides a user-friendly interface for administrators to add, update, delete, and retrieve student details. The program also calculates percentages, generates performance reports, and ensures data security and persistence with file-based storage and backups.

## Table of Contents

1. Features  
2. System Requirements  
3. File Structure  
4. Functionalities  
 - User Authentication  
 - Adding a Student  
 - Updating Student Records  
 - Deleting a Student  
 - Calculating Percentages  
 - Generating Reports  
 - File Handling (Save, Load, Backup)  
5. Implementation Details  
6. Flow of Execution  
7. How to Compile and Run  
8. Sample Input/Output  
9. Future Enhancements

## Features

- User Authentication: Secure login to restrict access to authorized personnel.  
- CRUD Operations: Add, update, delete, and view student records.  
- Academic Management: Input grades for six predefined subjects, calculate percentages, and check pass/fail status.  
- Reporting: Generate detailed reports for individual students or all students.  
- Data Persistence: Save and load student records from files, with backup functionality.  
- Search Functionality: Quickly locate students by name.

## System Requirements

- Operating System: Windows, Linux, or macOS  
- Compiler: GCC or any C compiler  
- RAM: Minimum 512 MB  
- Disk Space: At least 1 MB for storing files

## File Structure

- Source Code File: `student\_management.c`  
- Data File: `students\_data.txt` (stores student information)  
- Backup File: `backup\_students\_data.txt` (used for data recovery)

## Functionalities

### User Authentication

Prompts the admin for a username and password. Predefined credentials (`admin/adminpass`) restrict access. Exits the program on failed login attempts.

### Adding a Student

Inputs a student’s name, age, and gender. Automatically assigns a unique ID. Stores data in memory until saved to a file.

### Updating Student Records

Allows modification of student name, age, and gender. Ensures input validation for gender (`M` or `F`).

### Deleting a Student

Deletes a student by ID. Automatically shifts remaining records to maintain consistency.

### Input Grades

Accepts grades for six predefined subjects. Validates marks to be within the 0-100 range.

### Calculating Percentages

Computes the percentage based on the total marks across six subjects.

### Generating Reports

- Detailed Report: Displays all details of a single student, including grades and percentage.  
- Print All Reports: Prints reports for all students in the system.

### Search Student

Locates a student by name. Displays a detailed report if the student is found.

### File Handling

- Save Data: Writes all student records to `students\_data.txt`.  
- Load Data: Reads student records from `students\_data.txt`.  
- Backup Data: Creates a copy of the data file for recovery purposes.

## Implementation Details

- Data Structures:  
 - `struct Student`: Stores student details, grades, and percentage.  
 - `students[MAX\_STUDENTS]`: Array to hold student records.  
- Validation:  
 - Gender validation (`M` or `F`).  
 - Marks range validation (0-100).  
- File I/O:  
 - Uses `fopen`, `fprintf`, and `fscanf` for reading and writing files.  
 - Backups performed with `fread` and `fwrite`.

## Flow of Execution

1. Program starts and loads saved data from `students\_data.txt`.  
2. User is prompted for login credentials.  
3. Upon successful login, the main menu is displayed.  
4. User selects a functionality:  
 - Add, update, delete student records.  
 - Calculate percentage or input grades.  
 - Save or load data.  
5. Actions are executed based on the chosen option.  
6. Program terminates upon choosing the exit option, saving unsaved data.

## How to Compile and Run

\*\*Compilation:\*\*  
```bash  
gcc -o student\_management student\_management.c  
```

\*\*Execution:\*\*  
```bash  
./student\_management  
```

## Sample Input/Output

### Login

Enter username: admin  
Enter password: adminpass  
Login successful as Admin.

### Add Student

Enter student name: John Doe  
Enter age: 20  
Enter gender (M/F): M  
Student added successfully.

## Future Enhancements

- User Roles: Add support for multiple user roles (e.g., teachers, parents).  
- GUI Integration: Develop a graphical interface for easier navigation.  
- Enhanced Search: Allow search by multiple criteria like ID or age.  
- Grade Analysis: Provide insights such as average performance and top scorers.

## Conclusion

The Student Management System is a reliable tool for managing academic records, ensuring efficiency and accuracy in educational institutions. Its modular design and robust features make it easy to use and extend.