# Student Management System - Technical Documentation

## Overview

The **Student Management System** is a Python-based desktop application developed using the **Tkinter** GUI framework. It enables users to **manage student records and module grades** effectively. All data is persistently stored using **CSV files**, ensuring simplicity and accessibility.

## Project Structure

```
student_management_system/
│
├── students.csv        # Stores student personal data
├── modules.csv         # Stores student modules and grades
└── main.py             # Main Python script (code provided above)
```

## Module Breakdown

### 1. Database Class

Handles all data-related operations, including storage, retrieval, and manipulation.

#### Initialization

```
def __init__(self, students_file="students.csv", modules_file="modules.csv")
```

- Initializes the system with two CSV files.
- Loads students and modules into in-memory dictionaries.

#### load_data()

- Reads from `students.csv` and `modules.csv`.
- Populates:
  - `self.students = {student_id: [name, age, course, phone]}`
  - `self.modules = {student_id: [(module_name, grade), ...]}`

#### save_data()

- Writes current data from memory to CSV files.

**add_student()**

- Adds a student to the dictionary and initializes an empty module list.

**get_students()**

- Returns all student entries for display.

**add_module()**

- Adds a module and grade for a student.

**get_modules()**

- Retrieves all modules of a given student.

**delete_module()**

- Removes a specific module for a student.

**delete_student()**

- Completely removes a student and their module records.

**update_module_grade()**

- Updates the grade of a specific module for a student.

**calculate_gpa()**

- Computes GPA on a 4.0 scale using standard grading criteria.

## 2. StudentManagementApp Class

Handles the **Graphical User Interface (GUI)** using Tkinter.

### Initialization

```
def __init__(self, master)
```

- Sets up the main window and connects it to the `Database` class.

**create_dashboard()**

- Home screen with options:
  - Add Student
  - View Students

**add_student_window()**

- Collects student details and module inputs via form fields.
- Includes options to:
  - Add/Remove modules before saving
  - Save the student with modules

**add_temp_module()**

- Temporarily stores module and grade until the student is saved.

**clear_window()**

- Clears widgets from the current screen to navigate smoothly between views.

*Note: More UI functions follow similar patterns (not included in your snippet's visible section).*

## Data Format

### students.csv

| student_id | name | age | course | phone |
|---|---|---|---|---|
| 001 | Alice | 20 | CS | 0987654321 |

### modules.csv

| student_id | module_name | grade |
|---|---|---|
| 001 | Python | 85 |

## Features

- Add/Delete student
- Add/Remove modules with grades
- Update grades
- Auto GPA calculation

- Persistent CSV-based storage
- Simple and clean GUI with styled buttons and layout

## Error Handling

- Wrapped file operations in try-except blocks to catch:
    - File not found
    - CSV format issues
    - Type conversion errors
- GUI alerts (`messagebox.showerror`) notify the user in case of issues.

## Dependencies

- Python 3.x
- Tkinter (built-in)
- `csv`, `os` (built-in)

## Conclusion

This system provides a lightweight, user-friendly, and persistent student record management solution. It can be enhanced further with:

- Search functionality
- Sorting/filtering views
- CSV export/import
- Authentication