STUDENT MANAGEMENT SYSTEM

Submitted by

Name: Yedla Yashwanth

Reg.No: 25MIP10073

Course Name: Introduction to Problem solving and programming

Course code: CSE1021

Slot: A11+A12+A13

Faculty Name: Dr. A.V.R Mayuri

Academic Year: 2025-2026

1. Introduction

The Student Management System is a Python-based command-line application designed to store and manage student information such as name, roll number, class, and marks. Since manually maintaining records can be time-consuming, error-prone, and disorganized, this project provides a simple, structured digital alternative. The system functions completely in-memory and uses a menu-driven interface to perform operations like adding, viewing, updating, searching, and deleting student records. It demonstrates efficient handling of data using Python's built-in data structures and modular programming.

2. Problem Statement

Manual student record management using registers or spreadsheets becomes inefficient when handling larger datasets. Updating, searching, or modifying records is slow and often leads to errors.

There is a need for a simple, lightweight program that can manage essential student information quickly and consistently without relying on external databases or complex tools.

3. Functional Requirements

The system must support the following functionalities:

1. **Add Student** – Enter name, roll number, class, and marks.
2. **Display All Students** – Show all records in a formatted table.
3. **Search Student** – Retrieve details using roll number.
4. **Update Student** – Modify existing student information.
5. **Delete Student** – Remove a record with confirmation.
6. **Exit** – Terminate the program (data is cleared on exit).

4. Non-Functional Requirements

1. **Usability:** The interface must be simple, clear, and menu-driven.
2. **Performance:** All operations should execute instantly since data is stored in memory.
3. **Reliability:** The system should validate input (e.g., marks range, duplicate roll numbers).
4. **Maintainability:** The program must be modular with separate functions for each operation.

5. **Portability:** Should run on any system with Python installed.

5. System Architecture

The system follows a straightforward flow:

- User interacts with the menu.
- The selected option calls a specific function.
- Data is stored in a Python dictionary during runtime.
- The program continues until the user chooses to exit.

6. Design Decisions & Rationale

- **In-memory dictionary storage:** Selected for fast access and simple implementation.
- **Menu-based CLI:** Ideal for clean navigation without needing a GUI.
- **Separate functions:** Allows better readability and easier debugging.
- **No external libraries:** Keeps the project lightweight and universally runnable.
- **Input validation:** Ensures stability and prevents unexpected errors.

7. Implementation Details

The system is implemented using Python. The core components include:

- **Dictionary (students)** to store records with roll number as key.
- **Functions:**
    - add_student()
    - display_all()
    - search_student()
    - update_student()
    - delete_student()
    - main() to control workflow
- **Loop-based menu** for repeated actions.
- **Validation checks** for empty fields, duplicate roll numbers, and marks range.

Reference code: *Student management system.py*

8. Screenshots / Results

(You can insert screenshots from your terminal here showing menu, add, display, update, search, delete, and exit.)

9. Testing Approach

Manual Testing

- Each function tested separately for correctness.

Invalid Input Testing

- Empty name or roll number
- Duplicate roll number
- Marks outside 0–100
- Non-numeric inputs

Integration Testing

- Full workflow tested from adding → updating → deletion.

10. Challenges Faced

- Handling edge cases (invalid marks, duplicate roll numbers).
- Ensuring menu flow remains clean and user-friendly.
- Formatting table output without using external libraries.
- Keeping the entire logic stable without crashing.

11. Learnings & Key Takeaways

- Gained practical experience using Python dictionaries for data management.

- Learned how to design a functional and modular CLI application.
- Improved understanding of user-input handling and validation.
- Understood how to write clean, maintainable code and project documentation.

12. Future Enhancements

- Add JSON/CSV storage so data is saved after exiting.
- Develop a GUI version using Tkinter or PyQt.
- Add sorting features (by marks, roll number, name).
- Generate performance or attendance reports.
- Add admin login for secure access.