PROJECT REPORT

STUDENT MANAGEMENT SYSTEM

Name: Siya Dubey

College: VIT Bhopal

Course: Python Project

Academic Year: 2025–2026

Registration no. – 25BCY10204

2. Introduction

The Student Management System is a small Python-based application created to store and manage basic student information. Managing student records manually becomes difficult and time-consuming on pen paper on some other way but it’s a good way, especially when the number of students increases. This project provides simple options such as adding, updating, deleting, and viewing student details through a menu-driven program.

The goal of this project is to understand basic data handling, menu-based programs, and simple logic building in Python.

3. Problem Statement

Maintaining student records by writing them down or storing them manually is not efficient. It can lead to errors, missing information, and difficulty in searching for a particular student’s data. Therefore, a simple system is required that helps store basic student details and makes it easy to perform operations like add, update, search, and delete.

4. Functional Requirements

The system must allow the user to:

Add a new student

View all students

Search a student by ID

Update existing student information

Delete a student record

Exit the program

All functions should work through simple text-based menu options.

5. Non-Functional Requirements

Usability: The system must be easy to use and require no technical background.

Reliability: Basic operations like add or delete should work correctly every time.

Performance: Since it is a small program, it should run quickly on any device.

Portability: The program should run on any system with Python installed.

Maintainability: The code should be simple to understand and modify.

6. System Architecture

The system follows a simple architecture:

User Interface: Text-based menu

Processing: Functions for add, update, delete, search, and view

Storage: Temporary list (in-memory storage)

Output: Display results

7. Design Diagrams

a) Use Case Diagram

User

Add Student

Update Student

Delete Student

Search Student

View All Students

b) Workflow Diagram

Display menu

User selects option

Corresponding function runs

Return to menu

c) Sequence Diagram

User → System: Select Option

System → Function: Execute task

Function → System: Return Output

System → User: Show Result

d) Class/Component Diagram

No classes used (simple functional program)

Components:

main_menu

add_student

update_student

delete_student

search_student

view_students

e) ER Diagram (if storage used)

Not used because storage is in simple Python list.

8. Design Decisions & Rationale

Python list used instead of database to keep project simple.

Functions used for modularity and easy understanding.

Menu-driven interface chosen because it is beginner-friendly.

No advanced libraries used so that the project stays easy to maintain.

9. Implementation Details

The program is implemented using Python. It uses:

Lists for storing student details

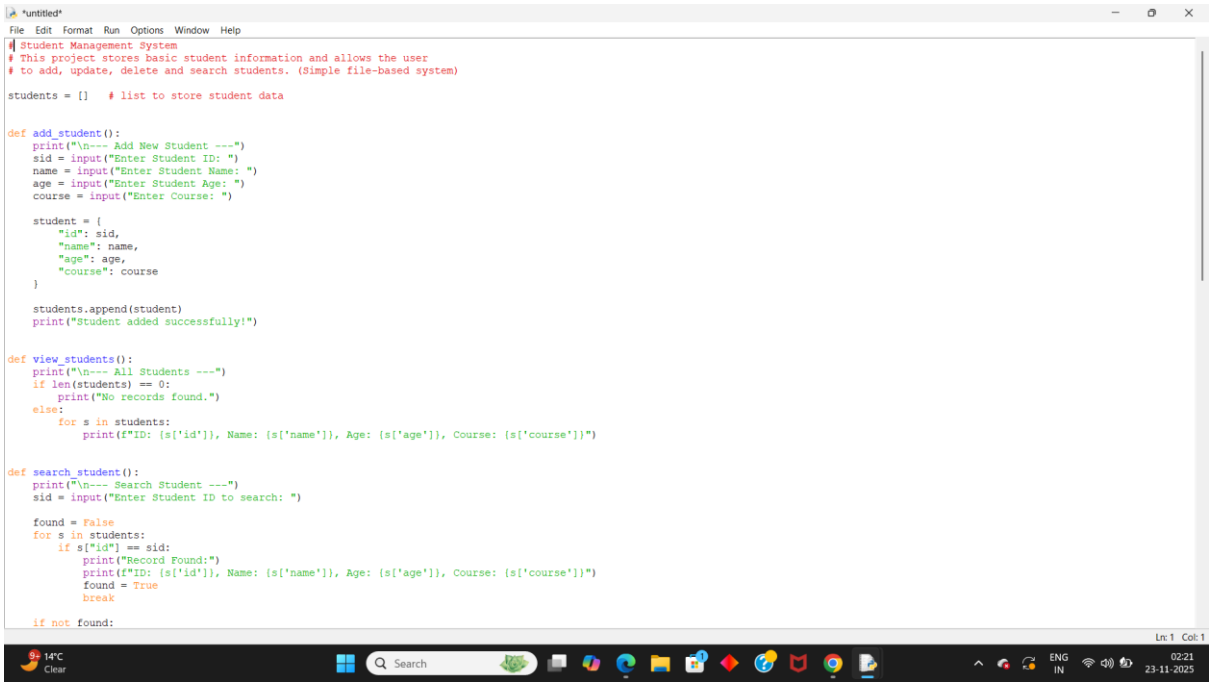
Dictionary for each student record

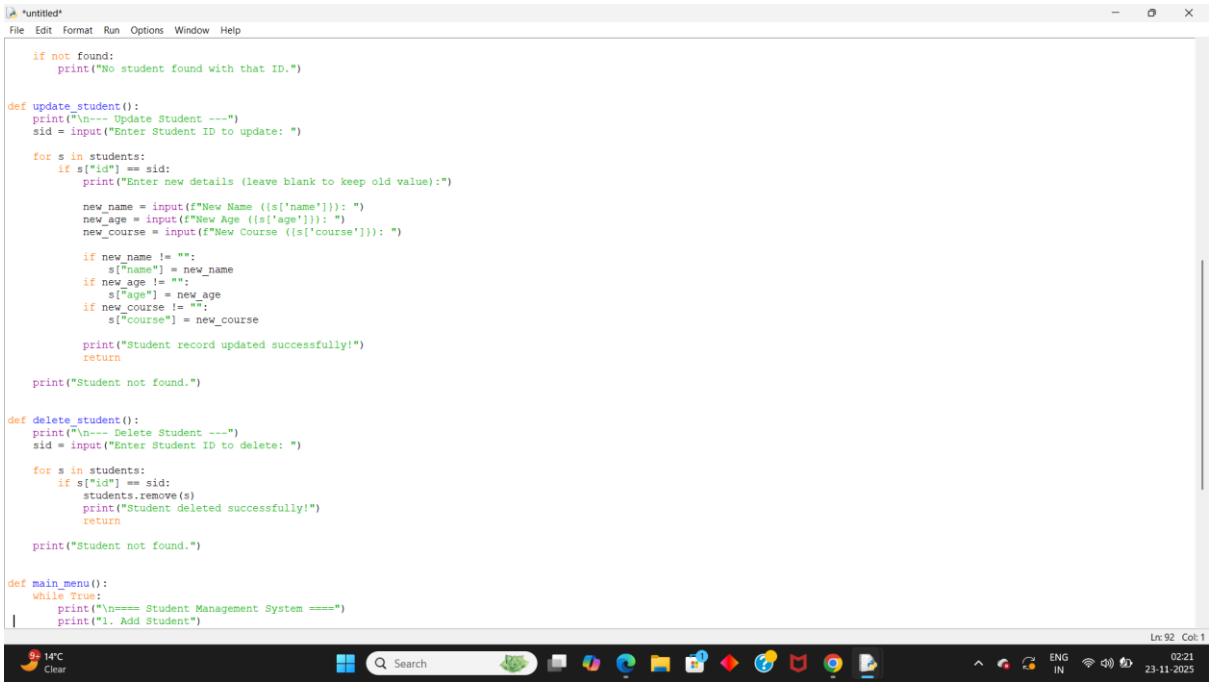
A loop for the main menu

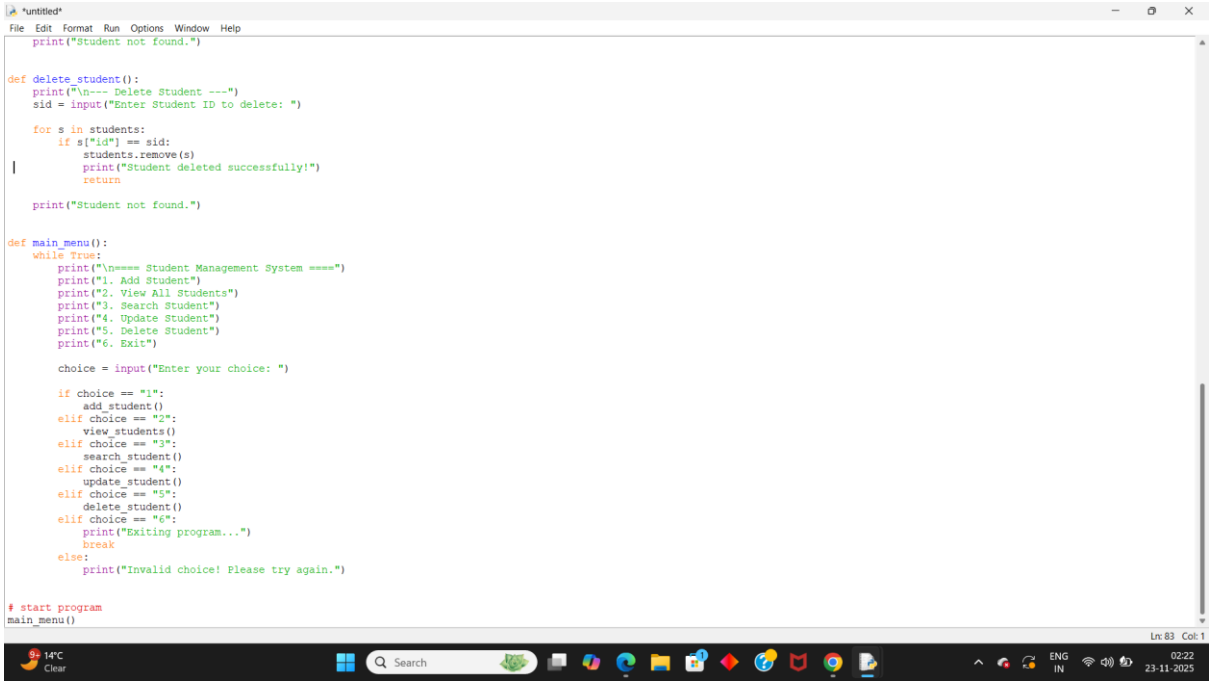
Simple input/output operations

Each feature is written as a separate function to keep the code organized and readable.

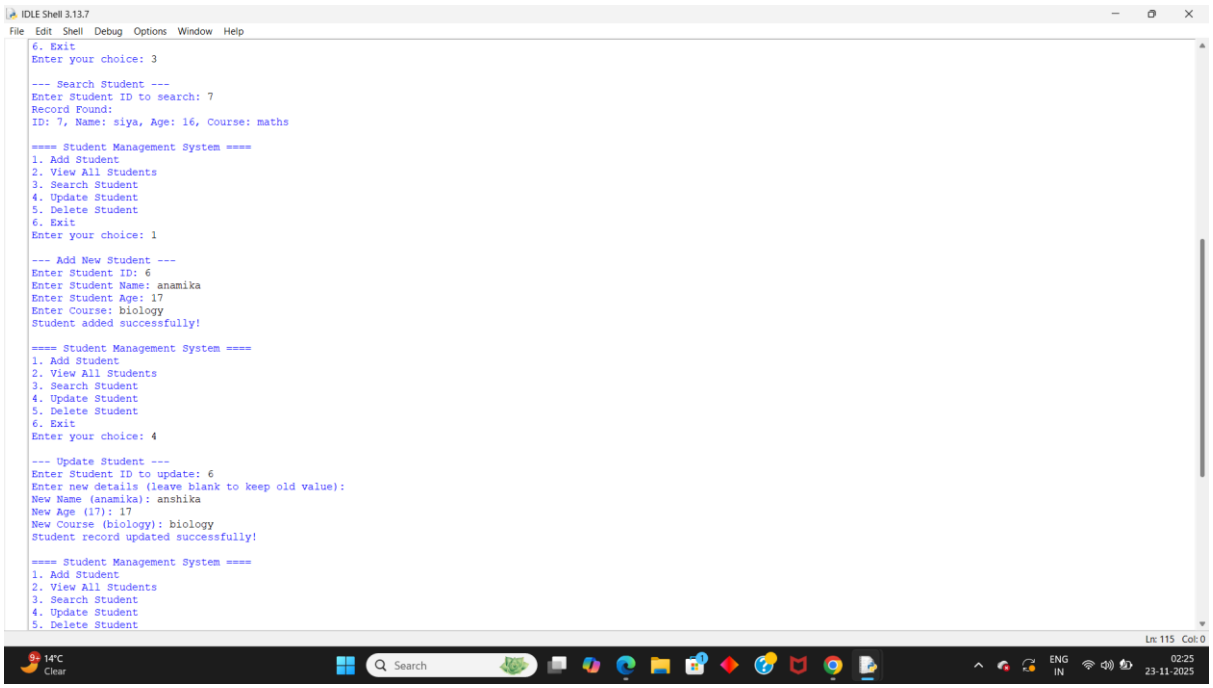
10. Screenshots / Results

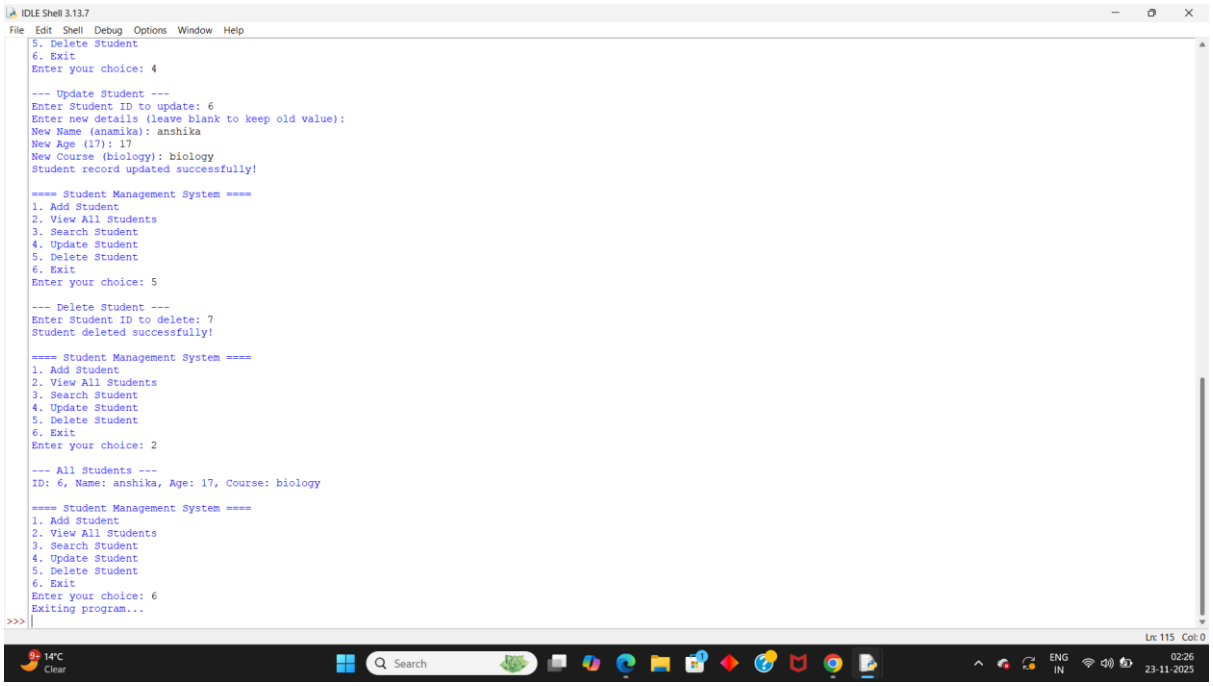



untitled
File Edit Format Run Options Window Help
if not found:
print("No student found with that ID.")
def update_student():
print("\n--- Update Student ---")
sid = input("Enter Student ID to update: ")
for s in students:
if s["id"] == sid:
print("Enter new details (leave blank to keep old value):")
new_name = input(f"New Name ({s['name']}): ")
new_age = input(f"New Age ({s['age']}): ")
new_course = input(f"New Course ({s['course']}): ")
if new_name != "":
s["name"] = new_name
if new_age != "":
s["age"] = new_age
if new_course != "":
s["course"] = new_course
print("Student record updated successfully!")
return
print("Student not found.")
def delete_student():
print("\n--- Delete Student ---")
sid = input("Enter Student ID to delete: ")
for s in students:
if s["id"] == sid:
students.remove(s)
print("Student deleted successfully!")
return
print("Student not found.")
def main_menu():
while True:
print("\n==== Student Management System ====")
print("1. Add Student")
Ln: 92 Col: 1
14°C Clear
Search
ENG IN
02:21 23-11-2025


untitled
File Edit Format Run Options Window Help
print("Student not found.")
def delete_student():
print("\n--- Delete Student ---")
sid = input("Enter Student ID to delete: ")
for s in students:
if s["id"] == sid:
students.remove(s)
print("Student deleted successfully!")
return
print("Student not found.")
def main_menu():
while True:
print("\n==== Student Management System ====")
print("1. Add Student")
print("2. View All Students")
print("3. Search Student")
print("4. Update Student")
print("5. Delete Student")
print("6. Exit")
choice = input("Enter your choice: ")
if choice == "1":
add_student()
elif choice == "2":
view_students()
elif choice == "3":
search_student()
elif choice == "4":
update_student()
elif choice == "5":
delete_student()
elif choice == "6":
print("Exiting program...")
break
else:
print("Invalid choice! Please try again.")
# start program
main_menu()
Ln: 83 Col: 1
14°C Clear
Search
ENG IN
02:22 23-11-2025


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
Python 3.13.7 (tags/v3.13.7:bcee1c3, Aug 14 2025, 14:06:58) [MSC v.1944 32 bit (Intel)] on win32
Enter "help" below or click "Help" above for more information.
>>>
========== RESTART: C:/Users/Siya/OneDrive/Desktop/code/project 25 nov.py ==========
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice: 1
--- Add New Student ---
Enter Student ID: 7
Enter Student Name: siya
Enter Student Age: 16
Enter Course: maths
Student added successfully!
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice: 2
--- All Students ---
ID: 7, Name: siya, Age: 16, Course: maths
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice: 3
--- Search Student ---
Enter Student ID to search: 7
Record Found:
ID: 7, Name: siya, Age: 16, Course: maths
Ln: 115 Col: 0
14°C Clear
Search
ENG IN
02:25 23-11-2025


IDLE Shell 3.13.7
File Edit Shell Debug Options Window Help
6. Exit
Enter your choice: 3
--- Search Student ---
Enter Student ID to search: 7
Record Found:
ID: 7, Name: siya, Age: 16, Course: maths
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice: 1
--- Add New Student ---
Enter Student ID: 6
Enter Student Name: anamika
Enter Student Age: 17
Enter Course: biology
Student added successfully!
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
Enter your choice: 4
--- Update Student ---
Enter Student ID to update: 6
Enter new details (leave blank to keep old value):
New Name (anamika): anshika
New Age (17): 17
New Course (biology): biology
Student record updated successfully!
==== Student Management System ====
1. Add Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
Ln: 115 Col: 0
14°C Clear
Search
ENG IN
02:25 23-11-2025



11. Testing Approach

I tested the program by:

Adding multiple students

Checking if they appear in the list

Updating their details

Searching them by ID

Deleting a student and verifying removal

Trying invalid inputs to see if the system handles errors

12. Challenges Faced

Deciding how to store data without using a database

Handling invalid user inputs

Making sure update and delete functions work correctly

Keeping the code simple but functional

13. Learnings & Key Takeaways

Learned how to use functions effectively in Python

Improved understanding of lists and dictionaries

Understood menu-driven programming

Learned how small systems are structured and implemented

Gained confidence in writing simple applications

14. Future Enhancements

In the future, this system can be improved by:;

Adding file storage so data is saved permanently

Using a database like MySQL

Adding a login system

Creating a GUI

Adding and exploring features

15. References

Python official documentation

Classroom note

W3Schools Python tutorials

GeeksforGeeks