Case Study 2: Smart Classroom Monitor

Step 1: Understanding the Problem

This case study requires us to design a Smart Classroom Monitor that simulates how modern classrooms are managed through digital systems.

In a real classroom, multiple tasks occur simultaneously, attendance is taken, the projector is used for lectures, the classroom temperature is monitored for comfort, and warnings or reminders may be necessary if something is amiss. The system we build must be able to handle these different responsibilities in one program.

The Smart Classroom Monitor should:

- Keep track of attendance by allowing students to be added or removed.
- Manage the projector status, toggling it between ON and OFF.
- Record and analyze temperature logs of the classroom environment.
- Track the topic being taught so that the session is documented.

Step 2: Inputs and outputs

Inputs:

Input	**Type**	**Unit**	**Description**
Student name	String	N/A	Add or remove students
Topic name	String	N/A	Setting the classroom topic
Temperature reading	Float	°C	Entered when recording the classroom temperature
Menu options	Integer	N/A	To select the options from the menu

Outputs:

Output	**Type**	**Unit**	**Description**
Projector status	String	N/A	ON/OFF message when toggled
Topic confirmation	String	N/A	Message when topic is set
Attendance update	String	N/A	Student added/removed with current count
Room full alert	String	N/A	Warning when attendance exceeds capacity
Temperature log confirmation	String	°C	Message when new temperature is added

Temperature statistics	Tuple	°C	Minimum, maximum, and average temperature
Temperature alert	String	°C	Warning if temperature <16°C or >28°C
Projector reminder	String	N/A	Reminder if topic is set but projector is OFF
Classroom report	String	N/A	Full summary of topic, projector, attendance, and temperatures

Step 3: Algorithm

The system's process can be Divided into the following algorithm:

1. Initialization: Create data structures for the classroom:
    - A dictionary for room details such as projector state, capacity, and topic.
    - A set for student attendance to avoid duplicates.
    - A list for temperature values.
2. Main loop: Continuously display a menu to the user until they choose "Exit."
3. Menu options:
    - Toggle projector: Change its state between ON and OFF.
    - Set topic: Record the classroom topic.
    - Add student: Add a student to the attendance set. If attendance exceeds capacity, display "ROOM FULL!"
    - Remove student: Remove a student if they exist in the set.
    - Add temperature: Append new temperature value to the list.
    - Show statistics: Compute minimum, maximum, and average of temperatures.
    - Classroom report: Display full report with alerts if necessary.
4. Exit: When the user selects exit, show final alerts before terminating the program.

This algorithm ensures that the program covers multiple operations and generates useful insights just like a real smart monitoring tool.

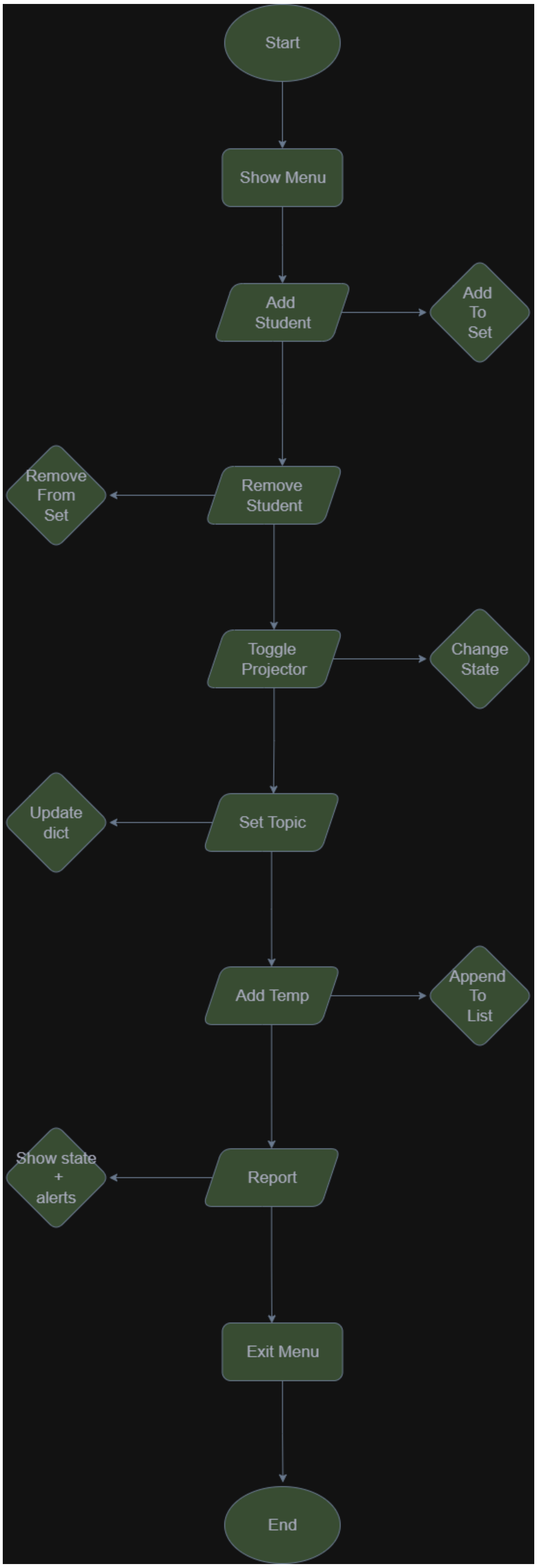
Step 4: Pseudocode

```
BEGIN
  SET room = {projector: False, topic: "", capacity: 5}
  SET attendance = {}
  SET temperatures = []

  WHILE choice != Exit:
    DISPLAY menu
    GET user input
    CASE choice:
        1 → Toggle projector
        2 → Input and set topic
        3 → Add student to attendance
            IF size of attendance > capacity → ALERT “ROOM FULL!”
        4 → Remove student from attendance
        5 → Input temperature → append to list
        6 → Calculate min, max, avg temperatures → DISPLAY
        7 → Generate classroom report and call alerts
        8 → Exit program
    DEFAULT → Invalid choice

  FOR temp IN temperatures:
    IF temp < 16 OR temp > 28 → ALERT “Temperature out of range”
END
```

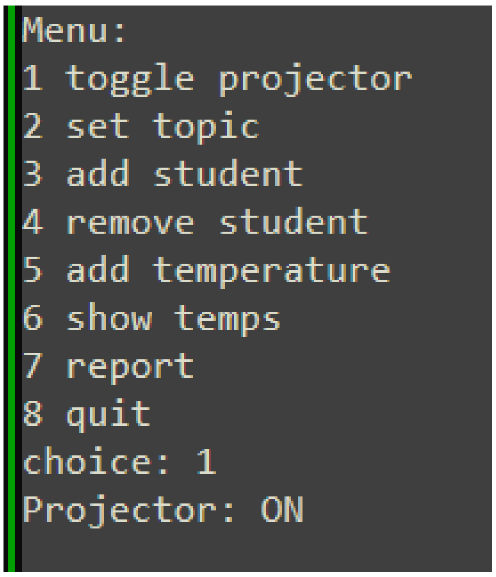
Flowchart:


Start
Show Menu
Add Student
Add To Set
Remove Student
Remove From Set
Toggle Projector
Change State
Set Topic
Update dict
Add Temp
Append To List
Report
Show state + alerts
Exit Menu
End

Step 6: Testing

Test Case 1: Toggle Projector

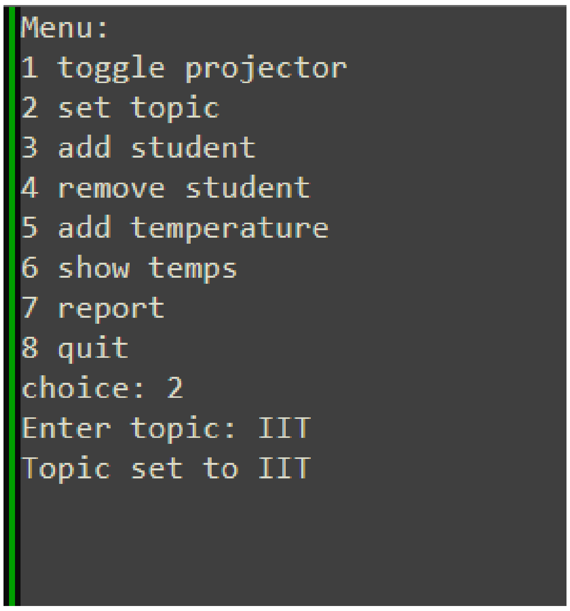
- Input: choice = 1
- Working: The projector is initially OFF. When option 1 is selected, the program toggles the projector state from OFF to ON.
- Output: *Projector: ON*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 1
Projector: ON
```

Test Case 2: Set Topic

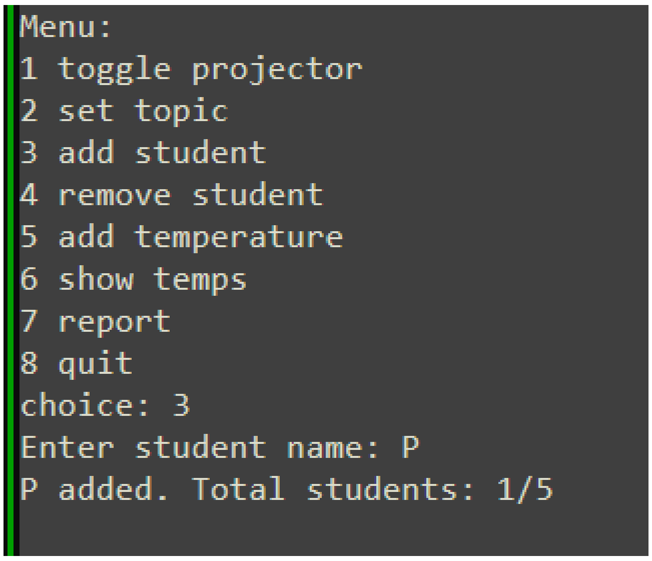
- Input: choice = 2, topic = “IIT”
- Working: When option 2 is selected, the program asks for a topic name. The entered topic “IIT” is stored in the room dictionary.
- Output: *Topic set to IIT*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 2
Enter topic: IIT
Topic set to IIT
```

Test Case 3: Add Student

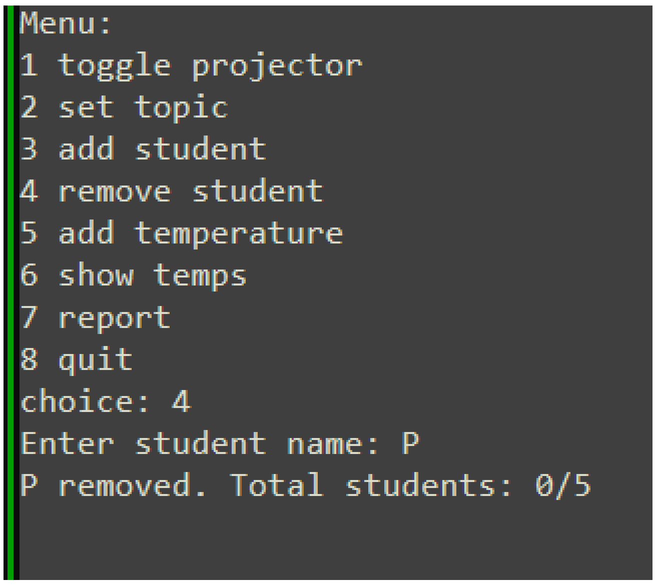
- Input: choice = 3, student = “P”
- Working: When option 3 is selected, the student “P” is added to the attendance set. The program updates the count of students.
- Output: *P added. Total students: 1/5*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 3
Enter student name: P
P added. Total students: 1/5
```

Test Case 4: Remove Student

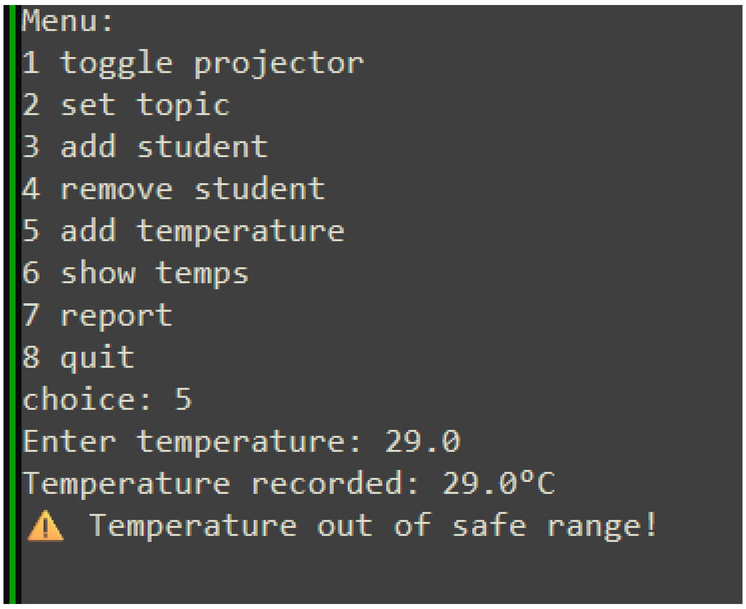
- Input: choice = 4, student = “P”
- Working: When option 4 is selected, the student “P” is removed from the attendance list.
- Output: P *removed. Total students: 0/5*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 4
Enter student name: P
P removed. Total students: 0/5
```

Test Case 5: Add Temperature

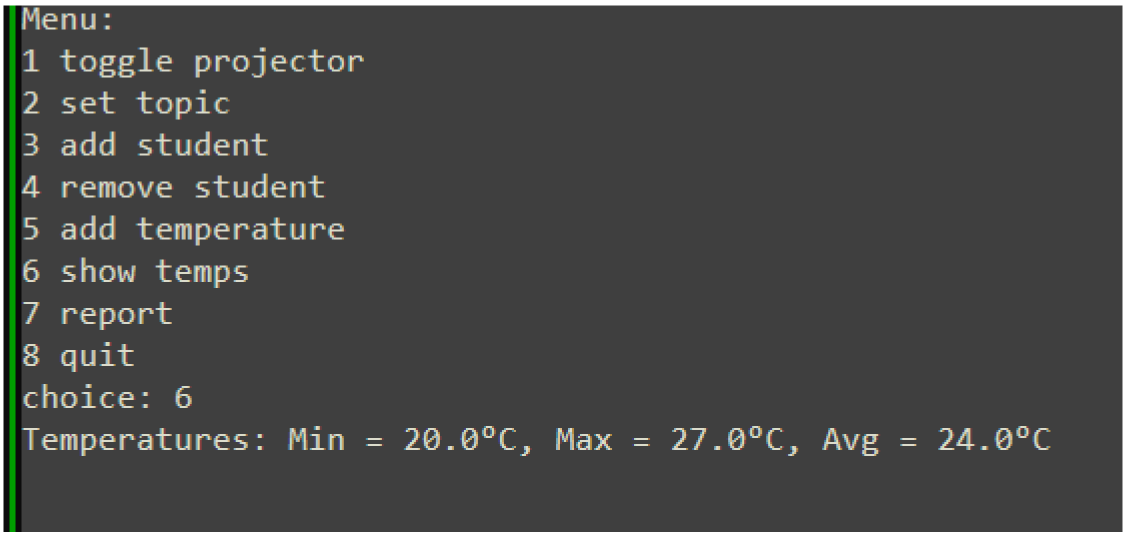
- Input: choice = 5, temperature = 29.0
- Working: When option 5 is selected, the program records the temperature reading. Since 29.0°C is above the safe range, it should trigger an alert.
- Output: *Temperature recorded: 29.0°C*; ⚠ *Temperature out of safe range!*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 5
Enter temperature: 29.0
Temperature recorded: 29.0°C
⚠ Temperature out of safe range!
```

Test Case 6: Show Temperature Statistics

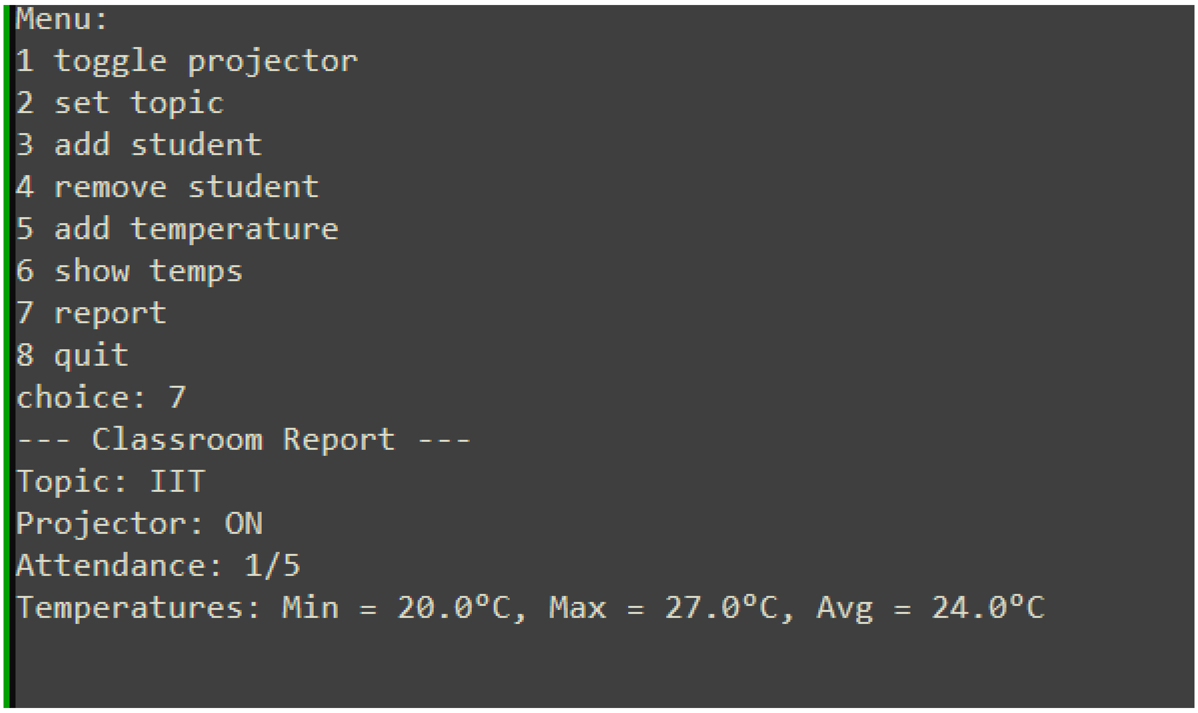
- Input: choice = 6, temperatures = [20.0, 25.0, 27.0]
- Working: The program calculates the minimum, maximum, and average temperature from the list.
- Output: *Min: 20.0°C, Max: 27.0°C, Avg: 24.0°C*
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 6
Temperatures: Min = 20.0°C, Max = 27.0°C, Avg = 24.0°C
```

Test Case 7: Generate Report

- Input: choice = 7
- Working: The program compiles all current classroom information into a single report, including projector state, topic, attendance, and temperature statistics.
- Output:
  Topic: IIT
  Projector: ON
  Attendance: 1/5
  Temperatures: Min 20.0°C, Max 27.0°C, Avg 24.0°C
- Result: Pass



```
Menu:
1 toggle projector
2 set topic
3 add student
4 remove student
5 add temperature
6 show temps
7 report
8 quit
choice: 7
--- Classroom Report ---
Topic: IIT
Projector: ON
Attendance: 1/5
Temperatures: Min = 20.0°C, Max = 27.0°C, Avg = 24.0°C
```

Step 7: Refinement using AI

After creating the initial version of the program, refinement was done using Generative AI (GenAI) tools to make the solution more robust, user-friendly, and professional.

- **Prompts I've Used:**
  Queries were made such as *"How to validate numeric input in Python?"* and *"How to round averages in Python to one decimal place?"* Additionally, advice was sought on improving alert clarity and formatting for better readability.
- **Changes Implemented:**
  - **Input Validation:** AI suggested ensuring that all numeric inputs, such as menu options and temperature readings, are validated. Invalid entries are now caught, and appropriate error messages are displayed.
  - **Rounding for Averages:** The average temperature calculation was refined to round values to one decimal place, making the output cleaner and easier to interpret.
  - **Improved Alerts:** Alerts were made more noticeable and structured, for example: *"⚠ Temperature out of safe range: 29.0°C."* The use of symbols improves readability and ensures alerts capture the user's attention.
  - **Report Formatting:** The classroom report was reformatted to be more structured, with headings and clear divisions between sections such as topic, attendance, and temperature data.
- **Justification:**
  These refinements not only improve the technical robustness of the program but also enhance the user experience by producing outputs that are easier to interpret. They also align the solution more closely with the assignment's goal of creating a professional and polished monitoring system. By leveraging AI for suggestions, the program was transformed from a functional prototype into a more complete and polished system.