VIT®
BHOPAL

VITYARTHI

COMPUTER SCIENCE AND ENGINEERING
Course Code : CSE1021

By:
Ms. Aishwarya Prasad
Regd. No: 25BCE11047

Submitted to:
Dr. Sandeep Monga

PYTHON ESSENTIALS

INTRODUCTION:

Academic Attendance Manager:

The new study sidekick, this one doesn't just care about numbers on a page. Forget those endless spreadsheets. This app keeps track of your attendance in real time and shows you exactly where you stand, no guesswork or last-minute panic. You won't have to stress over missing a class or dig through old records ever again. It handles the tracking and the charts, so you can actually enjoy college without getting lost in the math.

PROBLEM STATEMENT:

The Student Struggle:

Let's be real—the 75% attendance rule is both a blessing and a curse. Miss a few classes and suddenly, you're glued to your calculator, sweating over the numbers.

"Should I skip tomorrow for the cricket match?"

"Is one more nap going to wreck my internal marks?"

Trying to do the math yourself is just painful.

The Smart Solution:

Why bother? The Academic Attendance Manager handles it all. Just type in how many classes you've attended and the total so far, and boom:

"Chill—you can skip 2 more classes and you'll be fine."

"Yikes! Don't miss the next 5 or you'll be in trouble."

It crunches the numbers, predicts what you need, and even shows you charts. So now, you only have to decide which class to skip—not how many you still need.

FUNCTIONAL REQUIREMENTS:

1. DATA INPUT AND MANAGEMENT
    a. Handles all user inputs: entering subject names, total classes, attended classes.
    b. Manages and stores attendance data for each subject in Python dictionaries.
    c. Ensures accurate data collection for both single subject and multiple subjects workflows.

```python
def input_single_subject():
    subject = input("Enter subject name: ")
    total = int(input(f"Enter total number of classes for {subject}: "))
    attended = int(input(f"Enter total number of classes attended in {subject}: "))
    min_percentage = 75.0
    return {subject: {'total': total, 'attended': attended, 'min_percentage': min_percentage}}

def input_multiple_subjects():
    n = int(input("Enter total number of subjects: "))
    data = {}
    for _ in range(n):
        subject = input("Enter subject name: ")
        total = int(input(f"Enter total classes for {subject}: "))
        attended = int(input(f"Enter attended classes for {subject}: "))
        min_percentage = 75.0
        data[subject] = {'total': total, 'attended': attended, 'min_percentage': min_percentage}
    return data
```

2. ATTENDANCE CALCULATION, REPORTING & PREDICTION
    a. Calculates attendance percentages and determines SAFE or RISK status based on the required minimum (e.g., 75%).
    b. Generates detailed attendance reports per subject and overall.
    c. Predicts how many classes a user needs to attend (or can skip) to meet the minimum attendance requirement for each subject, providing actionable insights.

```python
info = data.get(subject)
if not info:
    print(f"No data found for {subject}.")
    return
total = info['total']
attended = info['attended']
perc = (attended / total) * 100 if total > 0 else 0
status = "SAFE" if perc >= info['min_percentage'] else "RISK"
print(f"\nAttendance Report for {subject}:")
print(f"Attended Classes: {attended}")
print(f"Total Classes: {total}")
print(f"Attendance Percentage: {perc:.2f}%")
print(f"Status: {status}")
```

```
if choice == '1':
    remaining = int(input("Enter number of remaining classes: "))
    target_ratio = min_perc / 100
    needed = 0
    while needed <= remaining:
        if (attended + needed) / (total + remaining) >= target_ratio:
            break
        needed += 1
    if needed > remaining:
        print("Not possible to reach minimum attendance even if all remaining classes are attended.")
    else:
        print(f"You need to attend at least {needed} out of {remaining} remaining classes.")
elif choice == '2':
    max_skips = 0
    while max_skips <= total - attended:
        if attended / (total - max_skips) < (min_perc / 100):
            break
        max_skips += 1
    max_skips -= 1
    print(f"You can skip up to {max_skips} classes and still maintain minimum attendance.")
else:
    print("Invalid choice.")
```

3. VISUALIZATION & USER INTERACTION LOOP

a. Presents interactive menus to guide users through options and submenus, ensuring smooth navigation.
b. Uses matplotlib to generate visual feedback: pie charts for individual subjects and bar graphs for overall report.
c. Keeps users engaged with clear, repeatable operations until they choose to exit.

```
# Pie chart
plt.figure(figsize=(5,5))
labels = ['Attended', 'Missed']
sizes = [attended, total - attended]
colors = ['#4CAF50', '#FF6347']
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title(f"Attendance for {subject}")
plt.show()
```

```
import matplotlib.pyplot as plt
def view_attendance_report(subject, data):
    info = data.get(subject)
    if not info:
        print(f"No data found for {subject}.")
        return
    total = info['total']
    attended = info['attended']
    perc = (attended / total) * 100 if total > 0 else 0
    status = "SAFE" if perc >= info['min_percentage'] else "RISK"
    print(f"\nAttendance Report for {subject}:")
    print(f"Attended Classes: {attended}")
    print(f"Total Classes: {total}")
    print(f"Attendance Percentage: {perc:.2f}%")
    print(f"Status: {status}")
    # Pie chart
    plt.figure(figsize=(5,5))
    labels = ['Attended', 'Missed']
    sizes = [attended, total - attended]
    colors = ['#4CAF50', '#FF6347']
    plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
    plt.title(f"Attendance for {subject}")
    plt.show()
```

NON FUNCTIONAL REQUIREMENTS:

1. Usability:

   - ☐ Simple, menu-driven console interface.
   - ☐ Clear prompts and status messages help users understand available actions.
   - ☐ Visual feedback via graphs makes results easy to interpret.

2. Scalability:

   - ☐ Supports multiple subjects in overall report mode.
   - ☐ Structure allows future growth (adding more subjects or features without major redesign).

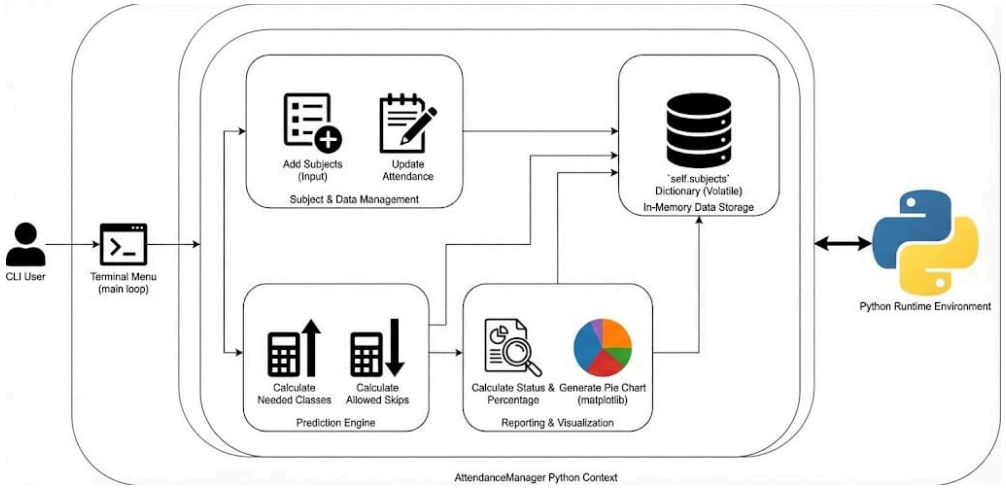
3. Performance:

   - ☐ Fast operations for small to medium data sets; uses in-memory storage and simple calculations.
   - ☐ Immediate feedback after menu selection without delay.

4. Visualization:

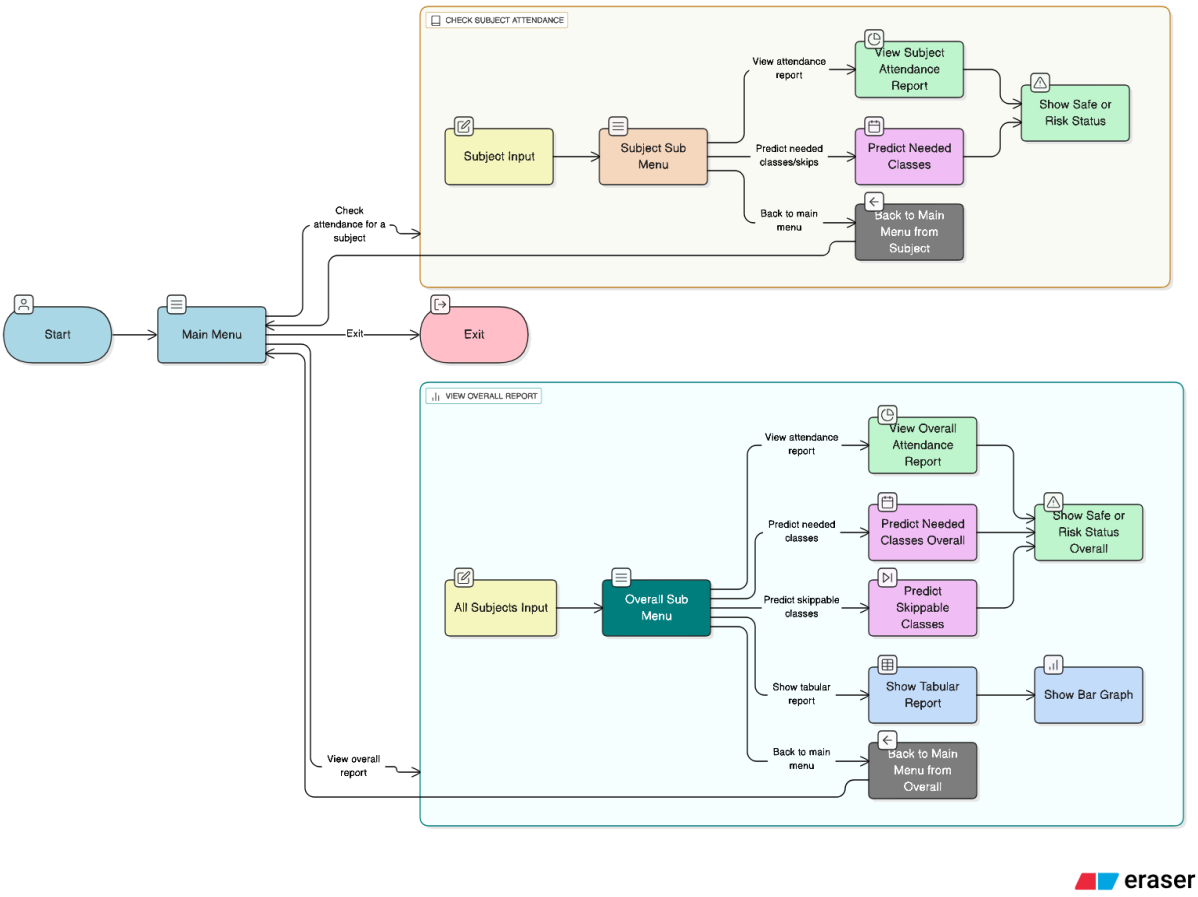
   - ☐ Uses matplotlib for pie and bar charts, enhancing comprehension and appeal.
   - ☐ Graphical elements automatically update with new user data.

SYSTEM ARCHITECTURE:

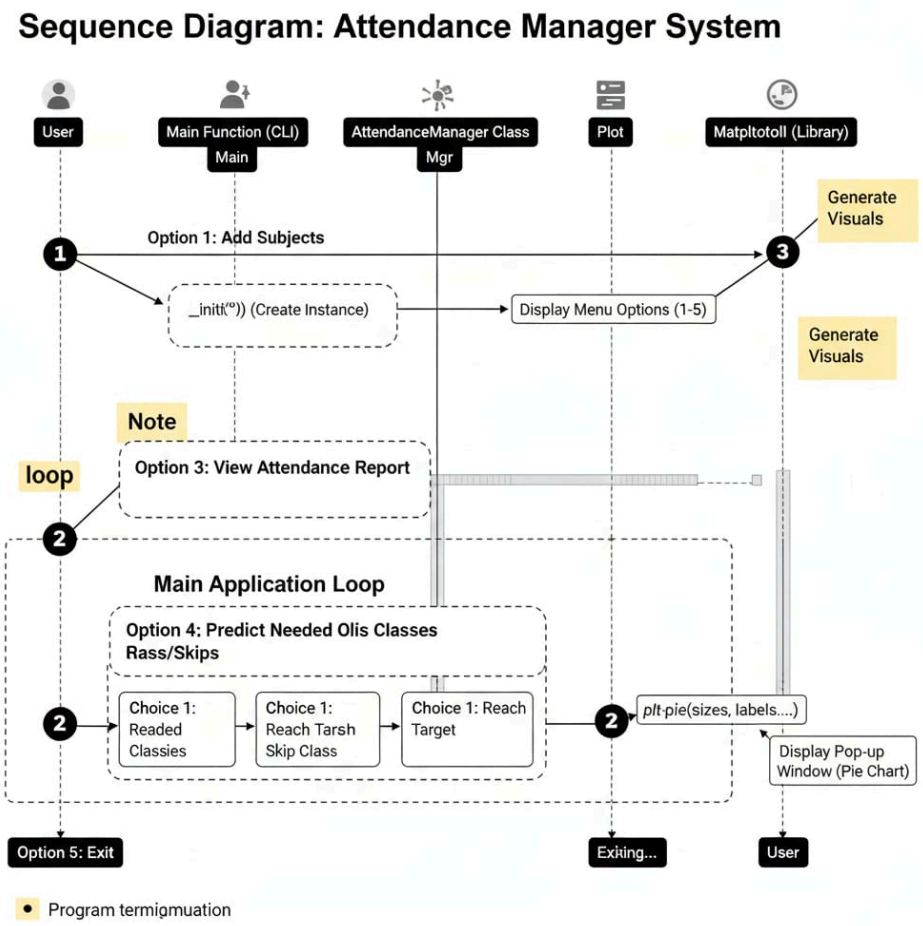


DESIGN DIAGRAMS :

a. WORKFLOW DIAGRAM:



b. SEQUENCE DIAGRAM:



DESIGN DECISIONS & RATIONALE:

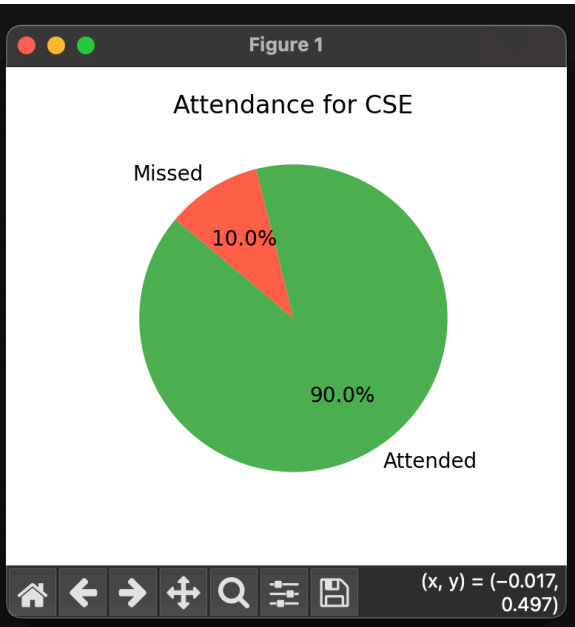
- ☐ Choose Python for simplicity and readability.
- ☐ Used dictionaries to allow dynamic and flexible management of subject data.
- ☐ Modular functions promote reuse and maintainability.
- ☐ matplotlib for effective and familiar data visualization.

IMPLEMENTATION DETAILS:

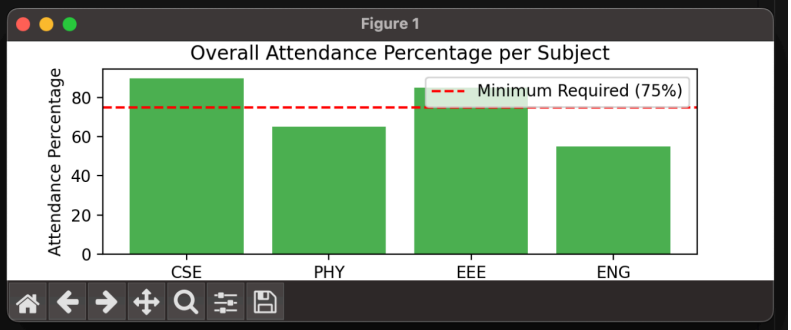
- ☐ Each operation is handled in its own function for clarity.
- ☐ Data is collected via console input.
- ☐ Attendance calculations done using basic arithmetic.
- ☐ Pie and bar charts generated with matplotlib.

SCREENSHOT/RESULTS:

```
Main Menu:
1. Check attendance for a particular subject
2. View overall attendance report
3. Exit
Enter your choice: 1
Enter subject name: CSE
Enter total number of classes for CSE: 20
Enter total number of classes attended in CSE: 18

Subject Menu:
a. View Attendance Report
b. Predict Needed Classes to Attend
c. Back to Main Menu
Enter choice (a/b/c): a

Attendance Report for CSE:
Attended Classes: 18
Total Classes: 20
Attendance Percentage: 90.00%
Status: SAFE

Subject Menu:
a. View Attendance Report
b. Predict Needed Classes to Attend
c. Back to Main Menu
Enter choice (a/b/c): b

Prediction Options:
1. Classes needed to attend to meet requirement
2. Classes can skip and still meet requirement
Enter choice (1 or 2): 1
Enter number of remaining classes: 5
You need to attend at least 1 out of 5 remaining classes.

Subject Menu:
a. View Attendance Report
b. Predict Needed Classes to Attend
c. Back to Main Menu
Enter choice (a/b/c): b

Prediction Options:
1. Classes needed to attend to meet requirement
2. Classes can skip and still meet requirement
Enter choice (1 or 2): 2
You can skip up to 2 classes and still maintain minimum attendance.
```




```
Main Menu:
1. Check attendance for a particular subject
2. View overall attendance report
3. Exit
Enter your choice: 2
Enter total number of subjects: 4
Enter subject name: CSE
Enter total classes for CSE: 20
Enter attended classes for CSE: 18
Enter subject name: PHY
Enter total classes for PHY: 20
Enter attended classes for PHY: 13
Enter subject name: EEE
Enter total classes for EEE: 20
Enter attended classes for EEE: 17
Enter subject name: ENG
Enter total classes for ENG: 20
Enter attended classes for ENG: 11

Overall Report Menu:
a. View attendance report for a subject
b. Show full report with bar graph
c. Predict classes for a subject
d. Back to Main Menu
Enter choice (a/b/c/d): b

Full Attendance Report:
Subject            Attended  Total     Percentage
CSE                18        20        90.00
PHY                13        20        65.00
EEE                17        20        85.00
ENG                11        20        55.00
```



TESTING APPROACH:

- ☐ Manual testing for user flows: entering subjects, updating data, viewing/predicting attendance.
- ☐ Input edge cases (zero, invalid, etc.) checked for appropriate error handling.

CHALLENGES FACED:

- ☐ Handling edge cases in calculation (division by zero).
- ☐ Designing a menu structure that's user-friendly and non-repetitive.
- ☐ Ensuring modular code for future extension.

LEARNING AND KEY TAKEAWAYS:

- ☐ Importance of clear modular design in interactive applications.
- ☐ User experience improves with persistent submenus and visual feedback.
- ☐ Combining logic and visualization within a console app is feasible and effective.

FUTURE ENHANCEMENTS:

- ☐ Add persistent storage (file/database) for saving data.
- ☐ Build graphical (GUI) version.
- ☐ Integrate login/roles for multi-user management.
- ☐ Automated reminders and email notifications.

REFERENCES:

- ☐ Python Official Documentation
- ☐ matplotlib Documentation
- ☐ Requirements from BuildYourOwnProject.pdf
- ☐ General software engineering best practices