ATTENDANCE TRACKER

Python Case Study Report

Student Name: Rishi Thakker

Roll No: 150096725068

Cohort: Sam Altman

Course: B.Tech CSE

Subject: Python Programming

Project Title: Student Attendance Tracker with Visualization

---

1. Introduction

The Student Attendance Tracker with Visualization is a Python-based educational management system designed to record, store, analyze, and visualize student attendance data using dynamic user input. The objective of this project is to provide a structured and automated approach to attendance tracking while generating meaningful analytics such as attendance percentages, chronic absentee identification, and visual trends. This case study uses Object-Oriented Programming (OOP) concepts, CSV-based file handling, NumPy for data analysis, and Matplotlib for visualization, making it a practical real-world application of Python fundamentals.

2. System Architecture & Modules Used

The application is divided into multiple modules, each handling a specific responsibility.

Core Modules

- **student_class.py**
  - Contains Student and Class classes
  - Manages student details and class enrollment
- **attendance_recorder.py**
  - Records daily attendance into CSV files
  - Handles attendance date, status, and late arrivals
- **analytics_engine.py**
  - Analyzes attendance data using NumPy
  - Calculates attendance percentage
  - Identifies chronic absentees
  - Generates visual graphs
- **attendance_main.py**
  - Acts as the main controller
  - Integrates all modules
  - Generates reports and visual outputs

3. Data Storage & File Handling

The system uses CSV files for persistent data storage.

Data Folder

- **students.csv** – Stores student information
- **attendance_record.csv** – Stores daily attendance records

Reports Folder

- **attendance_report.txt** – Final attendance compliance report

Visuals Folder

- **attendance_percentage_bar.png** – Attendance percentage bar chart
- **daily_attendance_summary.png** – Daily attendance trend graph

This structure ensures clear separation between raw data, analysis output, and visual evidence.

---

4. Attendance Recording Process

Attendance is recorded using the AttendanceRecorder class based on user provided input at runtime. Each attendance entry stores:

- Student ID
- Date (auto-generated)

- Attendance status (Present/Absent)
- Late arrival flag

The attendance data is appended to the CSV file, maintaining a complete attendance history.

---

5. Attendance Analysis & Statistics

The AttendanceAnalyzer class processes attendance records using NumPy.

Key Analysis Performed

- Attendance Percentage Calculation
- Identification of Students Below Minimum Attendance Threshold
- Daily Attendance Summary
- Class-wise Attendance Evaluation

The minimum attendance threshold is set to 75%, and students falling below this limit are flagged as chronic absentees.

---

6. Data Visualization

The project uses Matplotlib to generate visual analytics.

Visual Outputs

- **Attendance Percentage Bar Chart**
    - Compares attendance percentage of each student
- **Daily Attendance Trend Graph**
    - Shows number of students present per day

---

7. Output Report

A text-based report is generated in attendance_report.txt which includes:

- Attendance percentage of each student
- List of students below the required attendance threshold
- Summary of attendance compliance

This report serves as a formal record for academic review.

---

8. Challenges Faced & Learning Outcomes

During the development of the Attendance Tracker, several challenges were encountered:

- Understanding CSV file handling and maintaining consistent headers
- Implementing NumPy-based calculations without using Pandas
- Structuring the project using OOP principles
- Managing module imports and file paths correctly

- Generating visual outputs programmatically using Matplotlib

These challenges helped strengthen practical understanding of Python programming and real-world application design.

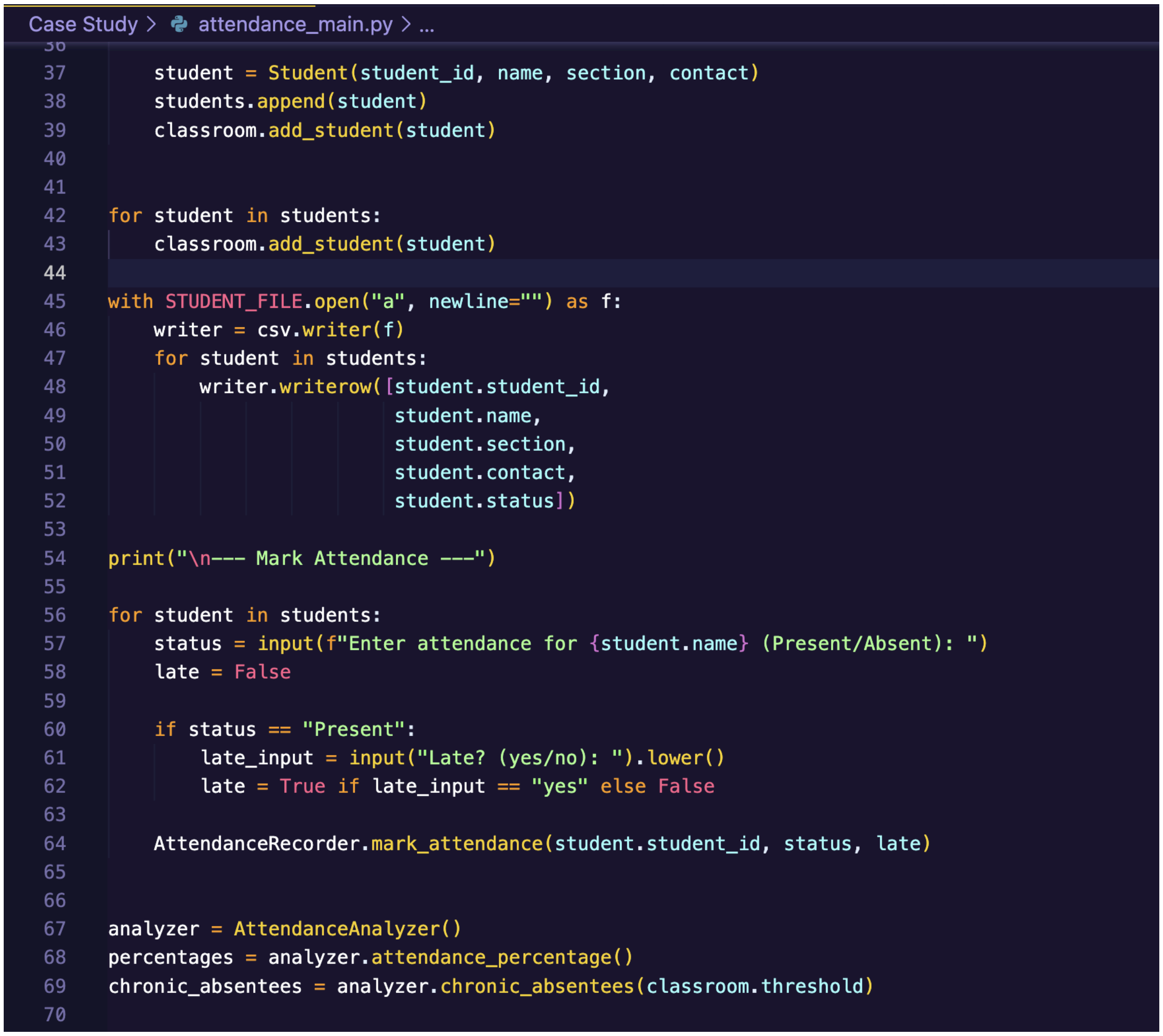
---

9. Conclusion

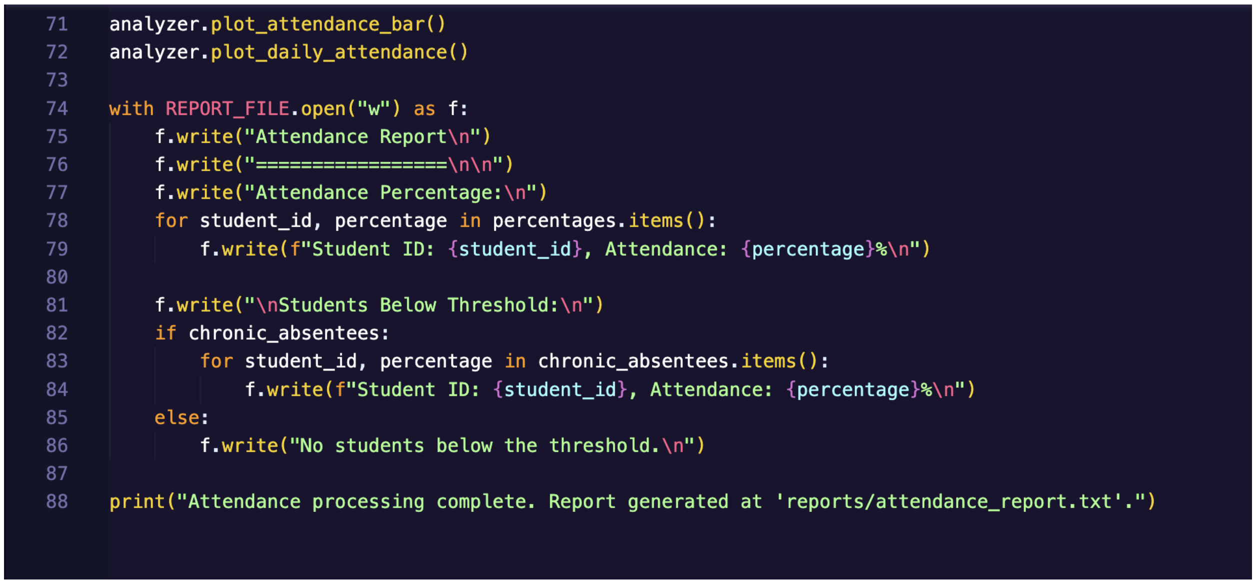
The Student Attendance Tracker successfully demonstrates the use of Python for managing academic attendance data. By combining file handling, object-oriented design, data analysis, and visualization, the project provides a complete and functional attendance management solution. The system improves accuracy, reduces manual effort, and offers clear insights through analytical reports and graphs. Overall, this project enhanced hands-on skills in Python programming and data-driven application development.

---

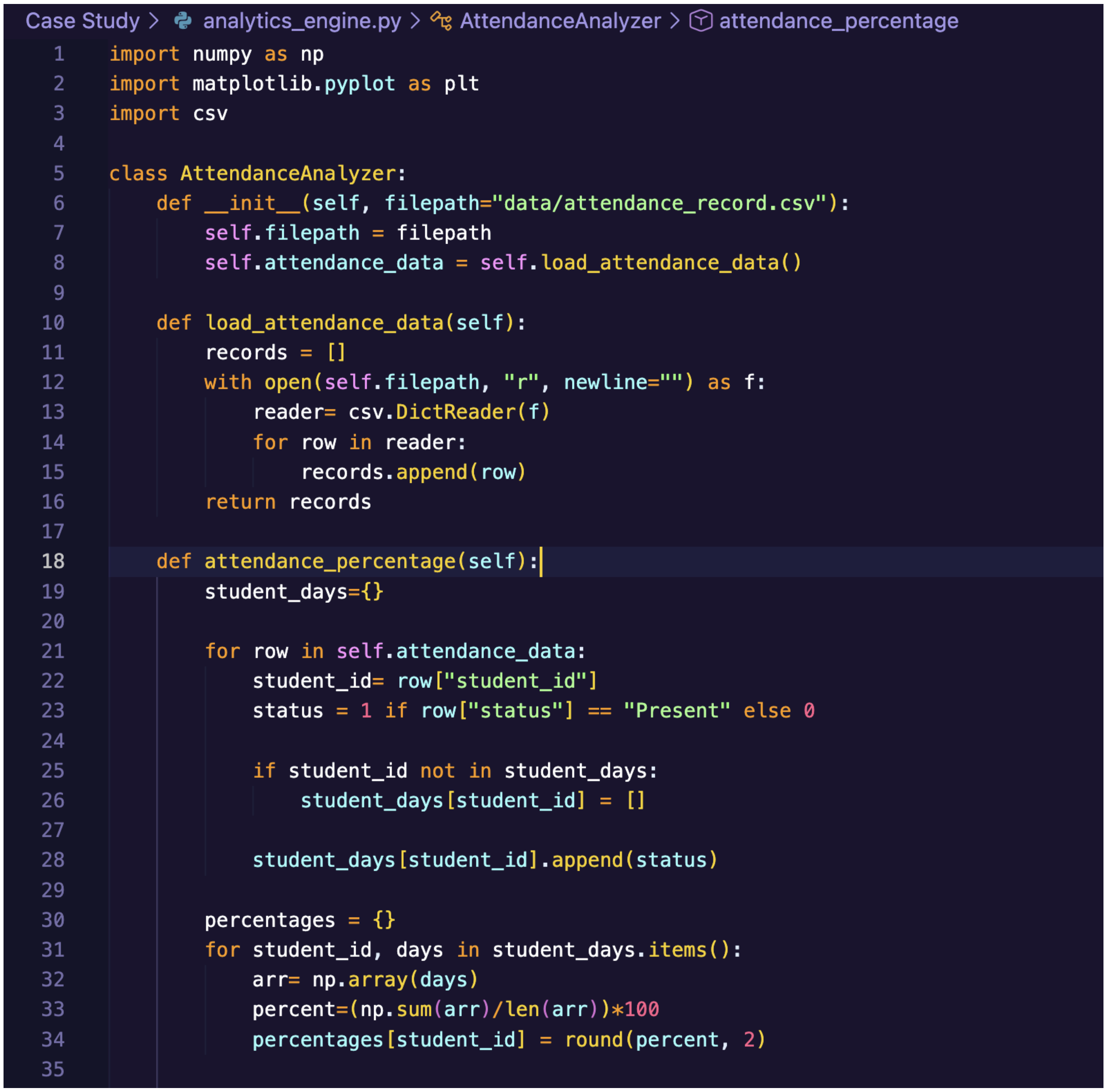
10. Screenshots:



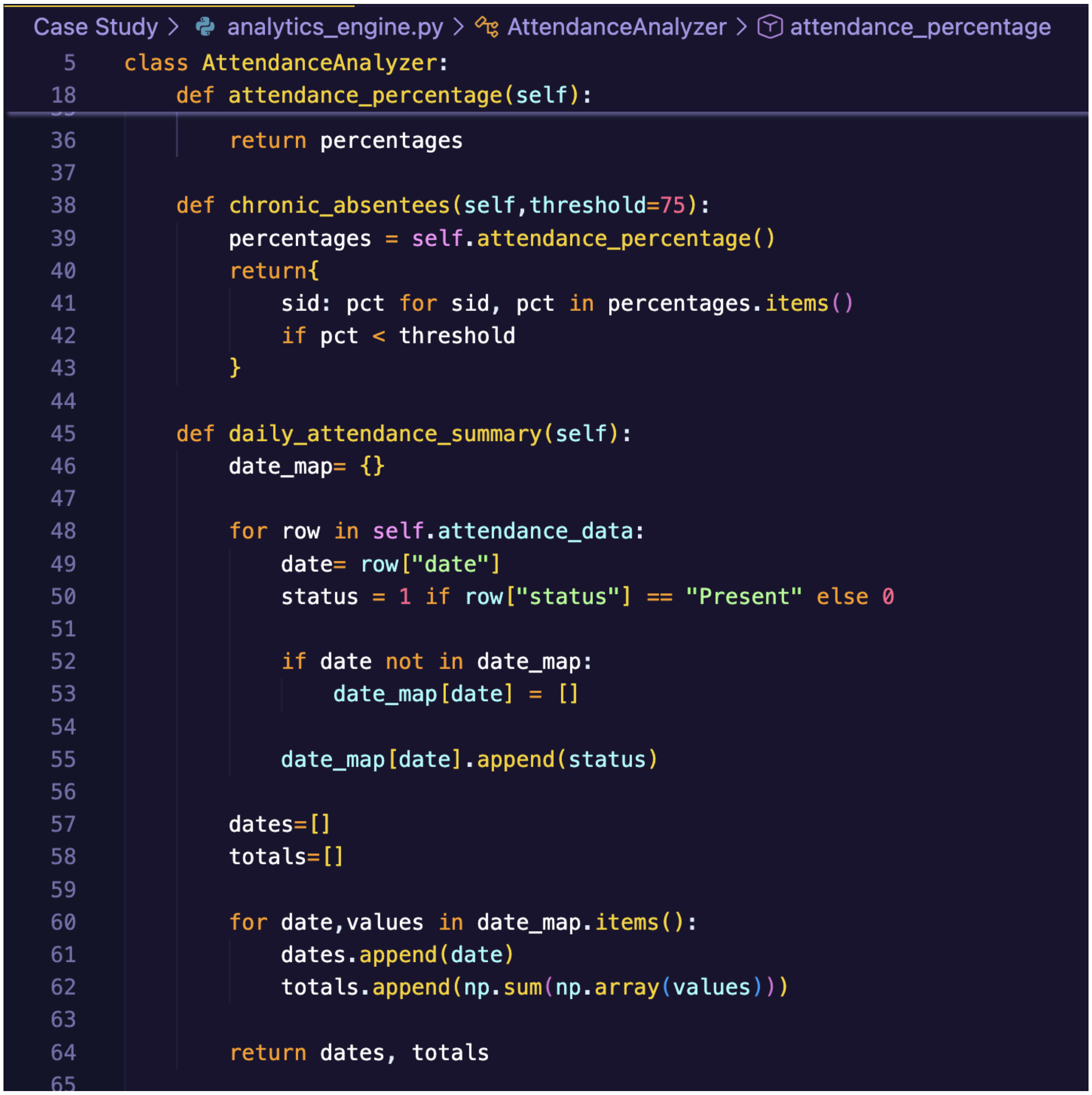
```python
    student = Student(student_id, name, section, contact)
    students.append(student)
    classroom.add_student(student)


for student in students:
    classroom.add_student(student)

with STUDENT_FILE.open("a", newline="") as f:
    writer = csv.writer(f)
    for student in students:
        writer.writerow([student.student_id,
                         student.name,
                         student.section,
                         student.contact,
                         student.status])

print("\n--- Mark Attendance ---")

for student in students:
    status = input(f"Enter attendance for {student.name} (Present/Absent): ")
    late = False

    if status == "Present":
        late_input = input("Late? (yes/no): ").lower()
        late = True if late_input == "yes" else False

    AttendanceRecorder.mark_attendance(student.student_id, status, late)


analyzer = AttendanceAnalyzer()
percentages = analyzer.attendance_percentage()
chronic_absentees = analyzer.chronic_absentees(classroom.threshold)

```



```python
analyzer.plot_attendance_bar()
analyzer.plot_daily_attendance()

with REPORT_FILE.open("w") as f:
    f.write("Attendance Report\n")
    f.write("=================\n\n")
    f.write("Attendance Percentage:\n")
    for student_id, percentage in percentages.items():
        f.write(f"Student ID: {student_id}, Attendance: {percentage}%\n")

    f.write("\nStudents Below Threshold:\n")
    if chronic_absentees:
        for student_id, percentage in chronic_absentees.items():
            f.write(f"Student ID: {student_id}, Attendance: {percentage}%\n")
    else:
        f.write("No students below the threshold.\n")

print("Attendance processing complete. Report generated at 'reports/attendance_report.txt'.")
```

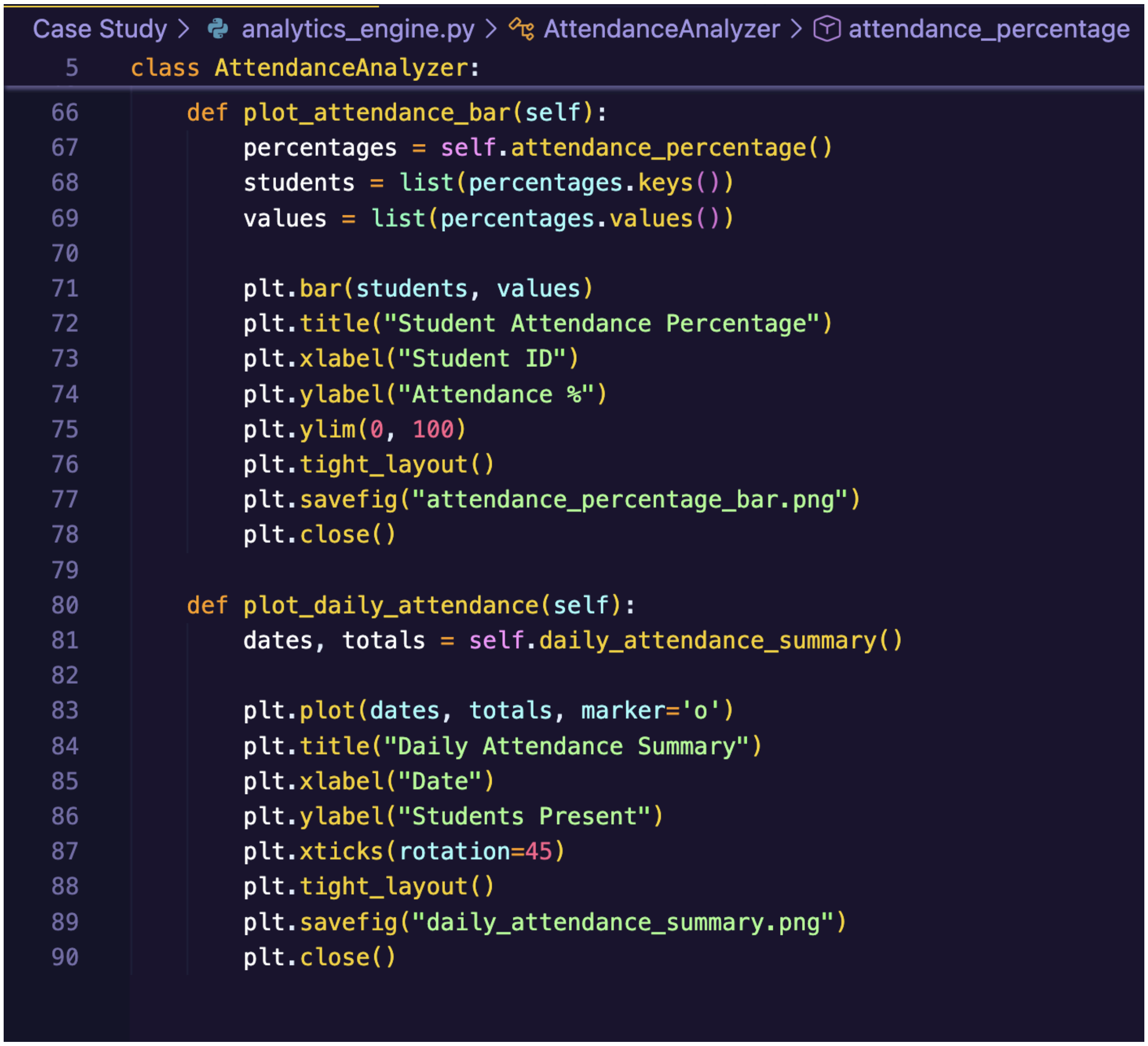

Case Study > analytics_engine.py > AttendanceAnalyzer > attendance_percentage

```python
import numpy as np
import matplotlib.pyplot as plt
import csv

class AttendanceAnalyzer:
    def __init__(self, filepath="data/attendance_record.csv"):
        self.filepath = filepath
        self.attendance_data = self.load_attendance_data()

    def load_attendance_data(self):
        records = []
        with open(self.filepath, "r", newline="") as f:
            reader= csv.DictReader(f)
            for row in reader:
                records.append(row)
        return records

    def attendance_percentage(self):
        student_days={}

        for row in self.attendance_data:
            student_id= row["student_id"]
            status = 1 if row["status"] == "Present" else 0

            if student_id not in student_days:
                student_days[student_id] = []

            student_days[student_id].append(status)

        percentages = {}
        for student_id, days in student_days.items():
            arr= np.array(days)
            percent=(np.sum(arr)/len(arr))*100
            percentages[student_id] = round(percent, 2)

```



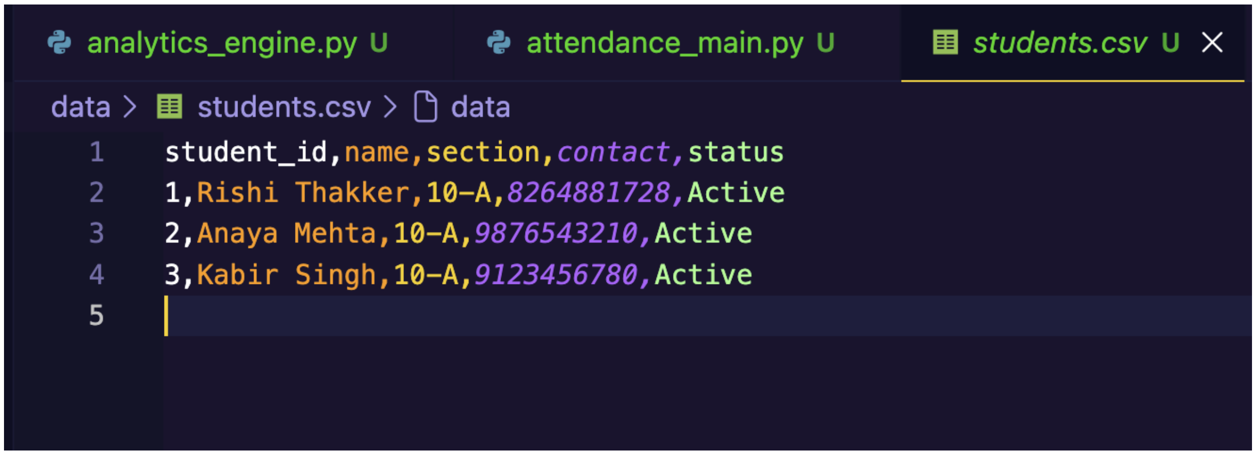
Case Study > analytics_engine.py > AttendanceAnalyzer > attendance_percentage

```python
class AttendanceAnalyzer:
    def attendance_percentage(self):
        return percentages

    def chronic_absentees(self,threshold=75):
        percentages = self.attendance_percentage()
        return{
            sid: pct for sid, pct in percentages.items()
            if pct < threshold
        }

    def daily_attendance_summary(self):
        date_map= {}

        for row in self.attendance_data:
            date= row["date"]
            status = 1 if row["status"] == "Present" else 0

            if date not in date_map:
                date_map[date] = []

            date_map[date].append(status)

        dates=[]
        totals=[]

        for date,values in date_map.items():
            dates.append(date)
            totals.append(np.sum(np.array(values)))

        return dates, totals
```

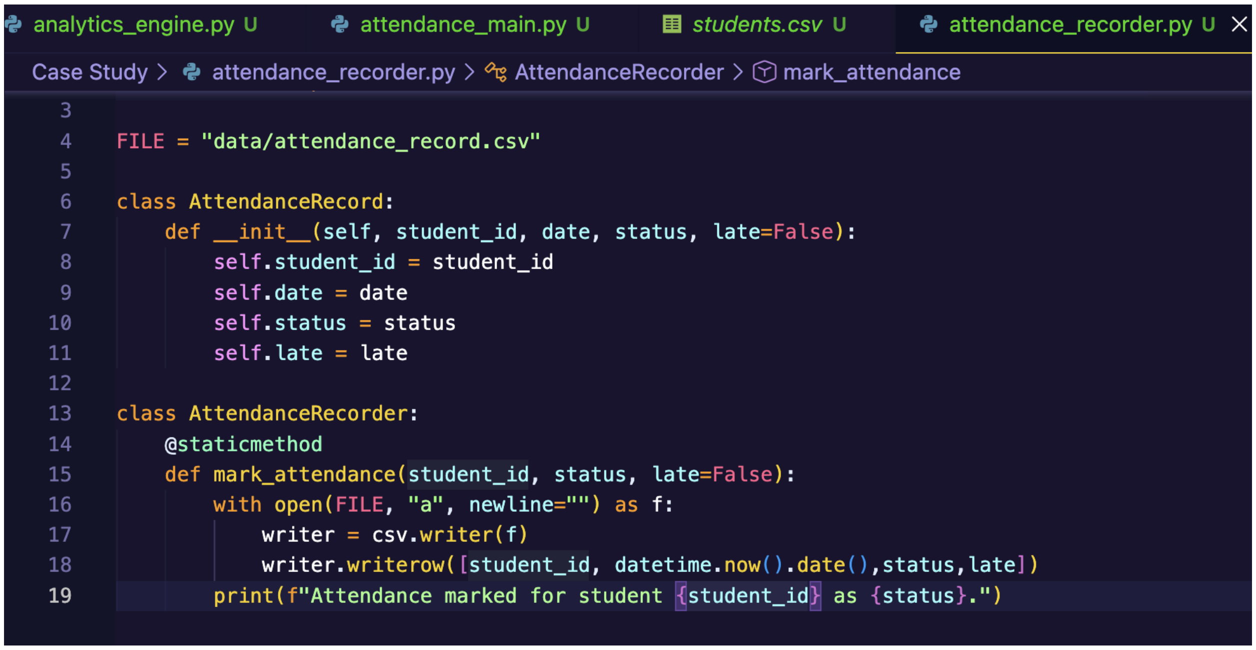


Case Study > analytics_engine.py > AttendanceAnalyzer > attendance_percentage

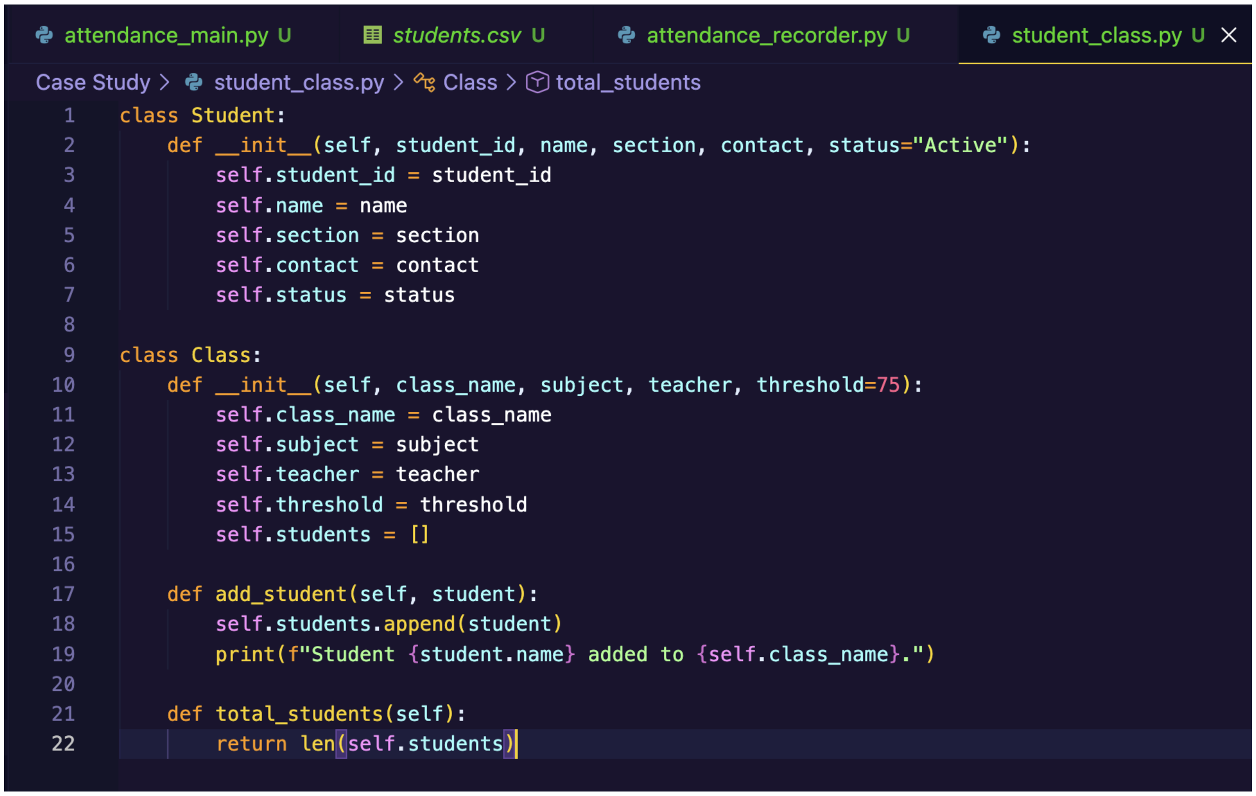
```python
class AttendanceAnalyzer:
    def plot_attendance_bar(self):
        percentages = self.attendance_percentage()
        students = list(percentages.keys())
        values = list(percentages.values())

        plt.bar(students, values)
        plt.title("Student Attendance Percentage")
        plt.xlabel("Student ID")
        plt.ylabel("Attendance %")
        plt.ylim(0, 100)
        plt.tight_layout()
        plt.savefig("attendance_percentage_bar.png")
        plt.close()

    def plot_daily_attendance(self):
        dates, totals = self.daily_attendance_summary()

        plt.plot(dates, totals, marker='o')
        plt.title("Daily Attendance Summary")
        plt.xlabel("Date")
        plt.ylabel("Students Present")
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.savefig("daily_attendance_summary.png")
        plt.close()
```



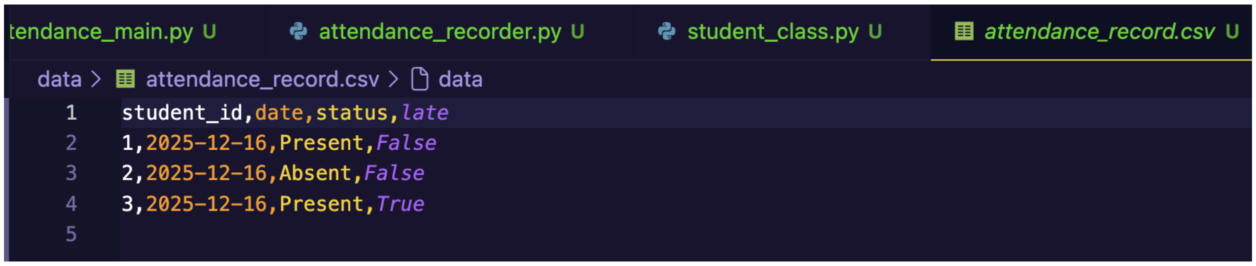
```
student_id,name,section,contact,status
1,Rishi Thakker,10-A,8264881728,Active
2,Anaya Mehta,10-A,9876543210,Active
3,Kabir Singh,10-A,9123456780,Active
```



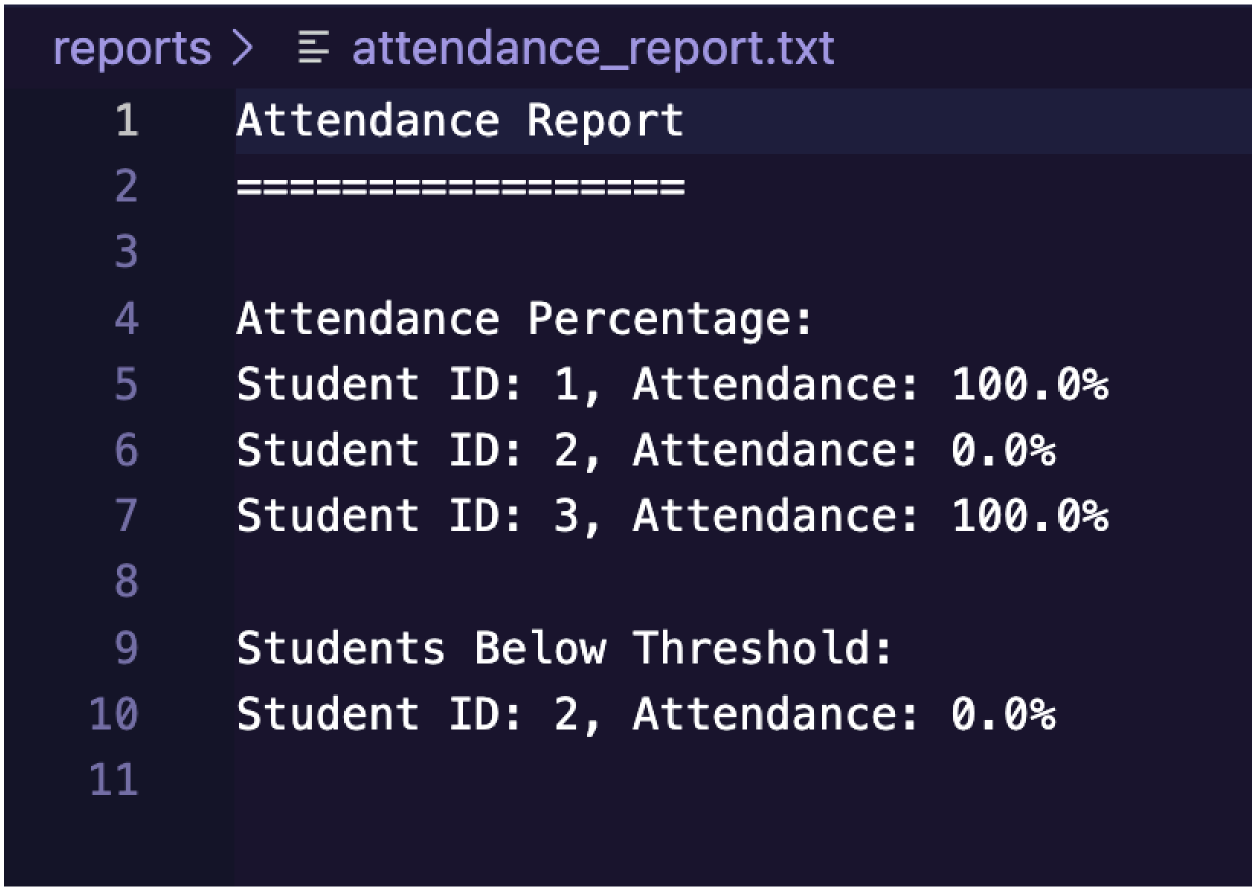
```python

FILE = "data/attendance_record.csv"

class AttendanceRecord:
    def __init__(self, student_id, date, status, late=False):
        self.student_id = student_id
        self.date = date
        self.status = status
        self.late = late

class AttendanceRecorder:
    @staticmethod
    def mark_attendance(student_id, status, late=False):
        with open(FILE, "a", newline="") as f:
            writer = csv.writer(f)
            writer.writerow([student_id, datetime.now().date(),status,late])
        print(f"Attendance marked for student {student_id} as {status}.")
```



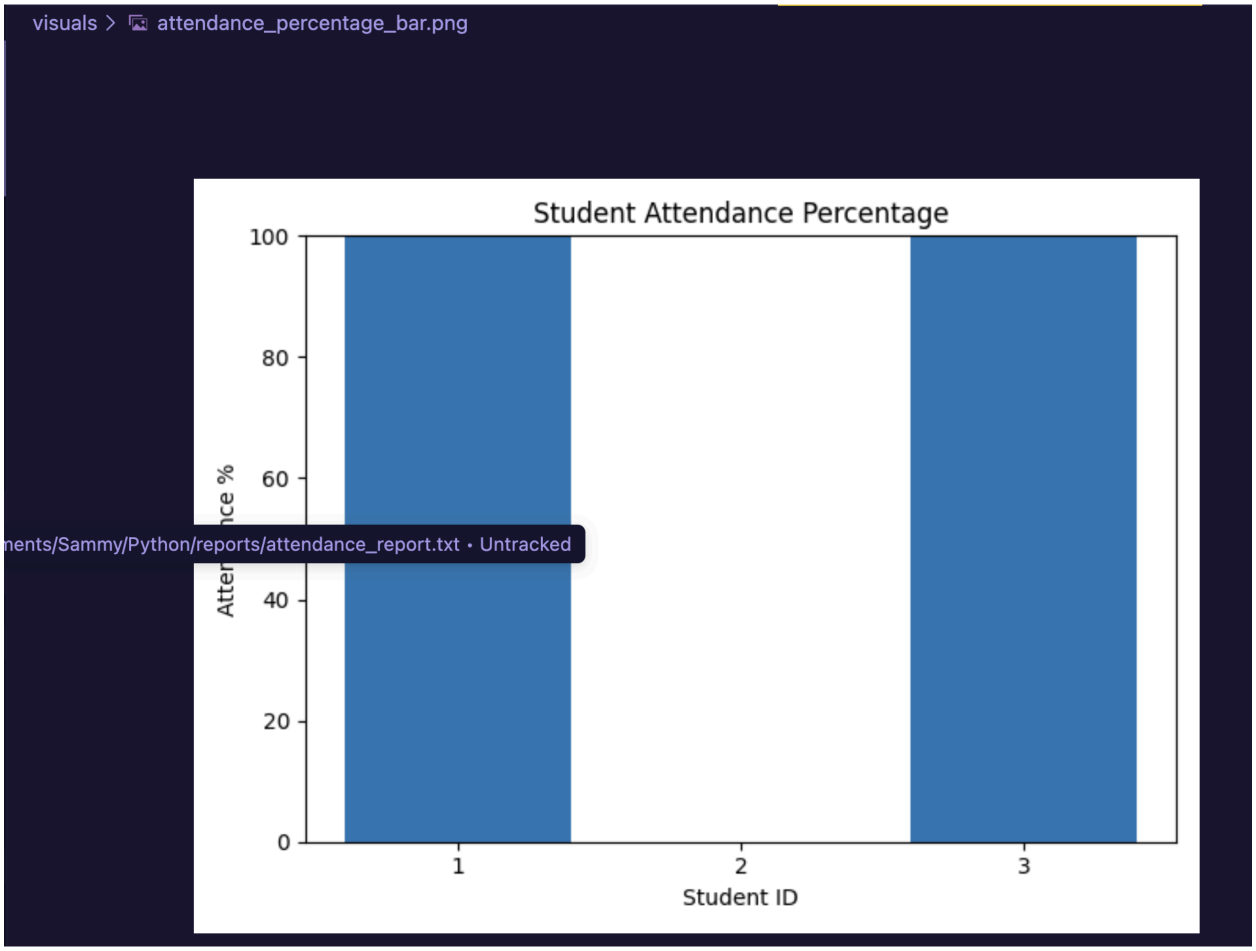
```python
class Student:
    def __init__(self, student_id, name, section, contact, status="Active"):
        self.student_id = student_id
        self.name = name
        self.section = section
        self.contact = contact
        self.status = status

class Class:
    def __init__(self, class_name, subject, teacher, threshold=75):
        self.class_name = class_name
        self.subject = subject
        self.teacher = teacher
        self.threshold = threshold
        self.students = []

    def add_student(self, student):
        self.students.append(student)
        print(f"Student {student.name} added to {self.class_name}.")

    def total_students(self):
        return len(self.students)
```



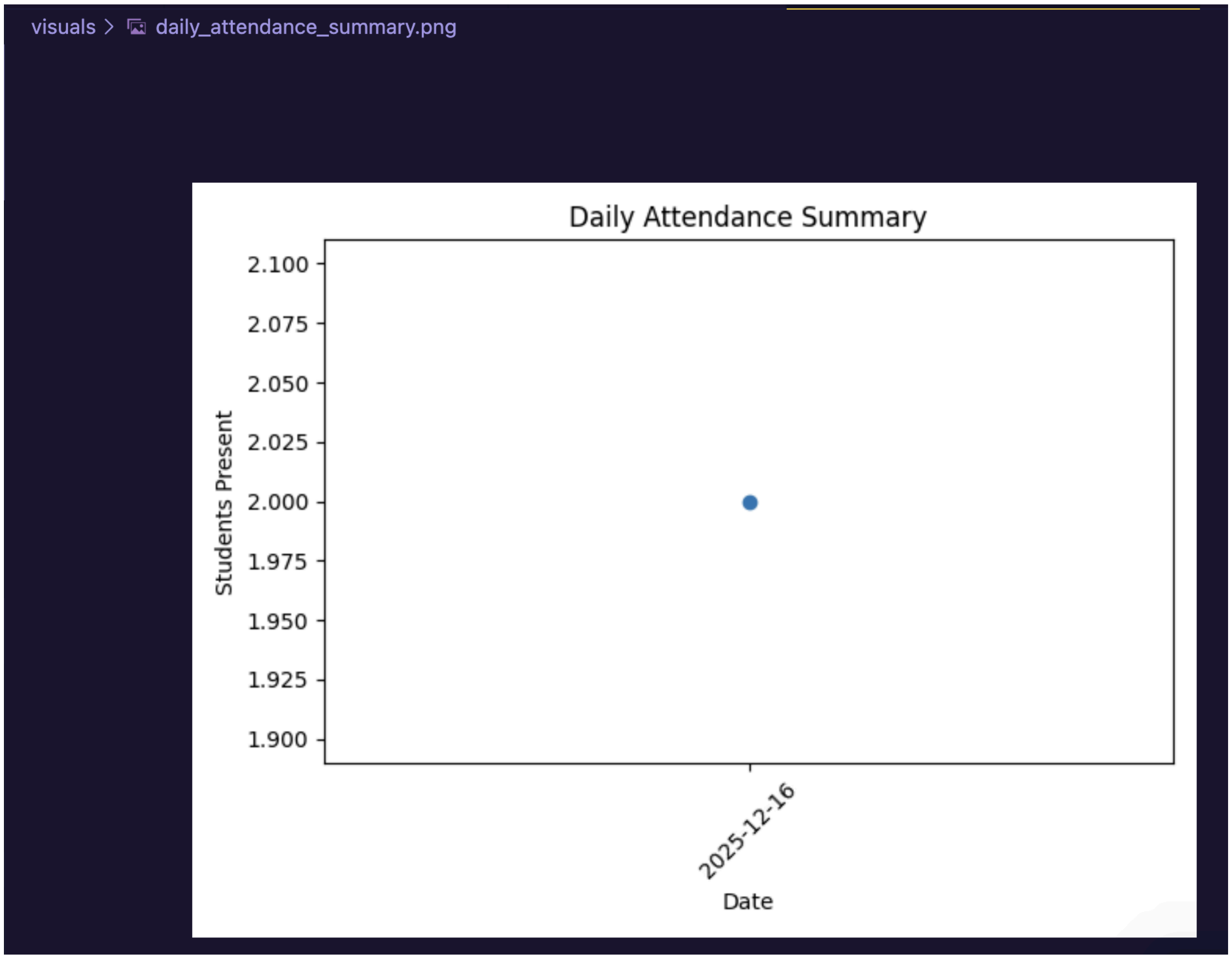
```
student_id,date,status,late
1,2025-12-16,Present,False
2,2025-12-16,Absent,False
3,2025-12-16,Present,True
```



```
Attendance Report
=================

Attendance Percentage:
Student ID: 1, Attendance: 100.0%
Student ID: 2, Attendance: 0.0%
Student ID: 3, Attendance: 100.0%

Students Below Threshold:
Student ID: 2, Attendance: 0.0%

```


visuals > attendance_percentage_bar.png

ments/Sammy/Python/reports/attendance_report.txt • Untracked


visuals > daily_attendance_summary.png
Daily Attendance Summary
Students Present
2.100
2.075
2.050
2.025
2.000
1.975
1.950
1.925
1.900
2025-12-16
Date