Project Report

# Project Title

AI for Environmental Monitoring and Conservation

# Introduction

This project is a Python-based Weather Data Recorder that allows users to log, store, and analyze weather information such as temperature and weather conditions (e.g., sunny, rainy). It helps simulate a basic environmental monitoring system that can be used for climate tracking, conservation planning, or smart city dashboards.

# Technologies Used

- Python  
- Pandas  
- DateTime Module

# Project Features

- Add daily weather entries (date, temperature, condition)  
- Prevent duplicate entries  
- View all stored weather data  
- Calculate average temperature  
- Export data to CSV file

# Source Code

import pandas as pd  
from datetime import datetime  
  
weather\_data = []  
dates\_seen = set()  
  
def add\_entry(date\_str, temperature, condition):  
 try:  
 datetime.strptime(date\_str, "%Y-%m-%d")  
 except ValueError:  
 print("Invalid date format. Use YYYY-MM-DD.")  
 return  
  
 if date\_str in dates\_seen:  
 print(f"Entry for {date\_str} already exists.")  
 return  
  
 entry = {"date": date\_str, "temperature": float(temperature), "condition": condition}  
 weather\_data.append(entry)  
 dates\_seen.add(date\_str)  
 print(f"Entry added for {date\_str}")  
  
def view\_data():  
 if not weather\_data:  
 print("No data available.")  
 else:  
 for entry in weather\_data:  
 print(entry)  
  
def export\_data(filename="weather\_data.csv"):  
 if not weather\_data:  
 print("No data to export.")  
 return  
 df = pd.DataFrame(weather\_data)  
 df.to\_csv(filename, index=False)  
 print(f"Data exported to {filename}")  
  
def calculate\_average():  
 if not weather\_data:  
 print("No data available.")  
 return  
 df = pd.DataFrame(weather\_data)  
 avg\_temp = df["temperature"].mean()  
 print(f"Average Temperature: {avg\_temp:.2f}°C")  
  
def main():  
 while True:  
 print("\nWeather Data Recorder Menu")  
 print("1. Add Entry")  
 print("2. View Entries")  
 print("3. Export to .csv")  
 print("4. Get Average Temperature")  
 print("5. Exit")  
 choice = input("Enter your choice: ")  
  
 if choice == "1":  
 date = input("Enter Date (YYYY-MM-DD): ")  
 temp = input("Enter Temperature: ")  
 condition = input("Enter Condition (e.g., Sunny/Rainy): ")  
 add\_entry(date, temp, condition)  
 elif choice == "2":  
 view\_data()  
 elif choice == "3":  
 export\_data()  
 elif choice == "4":  
 calculate\_average()  
 elif choice == "5":  
 print("Exiting the Weather Recorder.")  
 break  
 else:  
 print("Invalid choice. Try again.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

# Sample Output

Weather Data Recorder Menu  
1. Add Entry  
2. View Entries  
3. Export to .csv  
4. Get Average Temperature  
5. Exit  
Enter your choice: 1  
Enter Date (YYYY-MM-DD): 2025-07-01  
Enter Temperature: 32.5  
Enter Condition (e.g., Sunny/Rainy): Sunny  
✅ Entry added for 2025-07-01  
  
Enter your choice: 4  
🌡️ Average Temperature: 32.50°C  
  
Enter your choice: 5  
👋 Exiting the Weather Recorder. Stay weather-aware!

# Challenges Faced

- Ensuring valid date formats and preventing duplicate entries required use of `datetime` and `set`.  
- Exporting clean CSV data needed Pandas formatting.  
- Adding simple validation logic and user-friendly messages improved the overall experience.  
- Designing a menu-based interface that is beginner-friendly yet feature-rich took thoughtful planning.

Conclusion

This weather data recorder project demonstrates real-world Python programming skills like data validation, storage, analysis, and file export. It also simulates an environmental monitoring tool, which aligns with AI use cases in climate and conservation fields.

Submitted by

Chitteti Rohith.