- Environmental Monitoring System (EMS)with OOP-Python concept

Project Members:

1. M HUZAIFA KHILJI (f24-002)
2. M MUNEED SHEIKH (BSDS 007)
3. UMAR ZIA (049)

The **objective** of this project is to apply Object Oriented Programming(OOP) approach for the Environmental Monitoring System

Environmental Monitoring System: *Project Description*

Overview This project aims to develop an environmental monitoring system that can collect and analyze data from various sensors to monitor environmental conditions like temperature, humidity, and air quality. The system will utilize a modular approach, employing object-oriented programming principles to create a flexible and extensible framework.

System Components

1. Sensor Classes:

Base Sensor Class: Provides a common interface for all sensor types, defining methods for reading values and setting units. Temperature Sensor: Inherits from the base sensor class. Stores the temperature value and unit (e.g., Celsius, Fahrenheit). Humidity Sensor: Inherits from the base sensor class. Stores the humidity value and unit (e.g., percentage). Air Quality Sensor: Inherits from the base sensor class. Stores the air quality value and unit (e.g., ppm, µg/m³).

2. Environment Monitor Class:

Manages a list of sensors. Provides a method to add new sensors to the system. Monitors the sensors periodically, reads their values, and prints the results.

3. Data Logging:

Implements a mechanism to store sensor readings in a file or database for later analysis. Can be configured to log data at specific intervals or based on certain thresholds.

4. User Interface-Project Scope:

Provides a simple command-line interface to control the system. Can be extended to include graphical user interfaces or web-based dashboards for more advanced visualization and control. System Functionality Sensor Reading: Each sensor periodically reads its value and stores it. Data Logging: Sensor readings are logged to a file or database for later analysis. Data Display: The system displays the current readings of all sensors. Alert System: The system can be configured to trigger alerts if certain thresholds are exceeded (e.g., high temperature, low humidity, poor air quality). Future Enhancements Real-time Data Visualization: Implement visualization tools to display sensor data in real-time. Remote Monitoring: Enable remote access to the system through a web interface or mobile app. Machine Learning: Use machine learning techniques to predict future trends and anomalies. Integration with IoT

Platforms: Connect the system to IoT platforms for seamless data sharing and integration with other devices. By following this approach, we can create a robust and flexible environmental monitoring system that can be easily customized and extended to meet specific needs.

```
import time
import random

class Sensor:
    def __init__(self, name):
        self.name = name
        self.value = None

    def read_value(self):
        self.value = 25 + (10 * (random.random() - 0.5))  # Random values between 15 and 35
        return self.value

class TemperatureSensor(Sensor):
    def __init__(self, name, unit="Celsius"):
        super().__init__(name)
        self.unit = unit

class HumiditySensor(Sensor):
    def __init__(self, name, unit="%"):
        super().__init__(name)
        self.unit = unit

class AirQualitySensor(Sensor):
    def __init__(self, name):
        super().__init__(name)
        self.unit = None  # Or a specific unit like "ppm"

    def set_unit(self, unit):
        self.unit = unit

class EnvironmentMonitor:
    def __init__(self):
        self.sensors = []

    def add_sensor(self, sensor):
        self.sensors.append(sensor)

    def monitor(self):
        for sensor in self.sensors:
            value = sensor.read_value()
            print(f"{sensor.name} reading: {value}{sensor.unit}")

# Create sensors
temperature_sensor = TemperatureSensor("Temperature")
humidity_sensor = HumiditySensor("Humidity")
air_quality_sensor = AirQualitySensor("Air Quality")
air_quality_sensor.set_unit("ppm")  # Set the unit for the air quality sensor

# Create monitor and add sensors
monitor = EnvironmentMonitor()
monitor.add_sensor(temperature_sensor)
monitor.add_sensor(humidity_sensor)
monitor.add_sensor(air_quality_sensor)
```

```
# Start monitoring
while True:
    monitor.monitor()
    time.sleep(5)

    # user input to stop the monitoring
    user_input = input("Enter 'stop' to exit OR alphanumeric for 'Continue'  ")
    if user_input.lower() == 'stop':
        break
```

```
Temperature reading: 20.919142597206317Celsius
Humidity reading: 24.08855554593184%
Air Quality reading: 24.95213663499342ppm
Enter 'stop' to exit: no
Temperature reading: 28.77130299692615Celsius
Humidity reading: 20.464566168307222%
Air Quality reading: 23.142468575536824ppm
Enter 'stop' to exit: /
Temperature reading: 21.792614744136838Celsius
Humidity reading: 24.91551367695476%
Air Quality reading: 29.01851594855977ppm
Enter 'stop' to exit: and
Temperature reading: 26.29435323954034Celsius
Humidity reading: 23.888091257644447%
Air Quality reading: 28.828401862655834ppm
Enter 'stop' to exit: stop
```