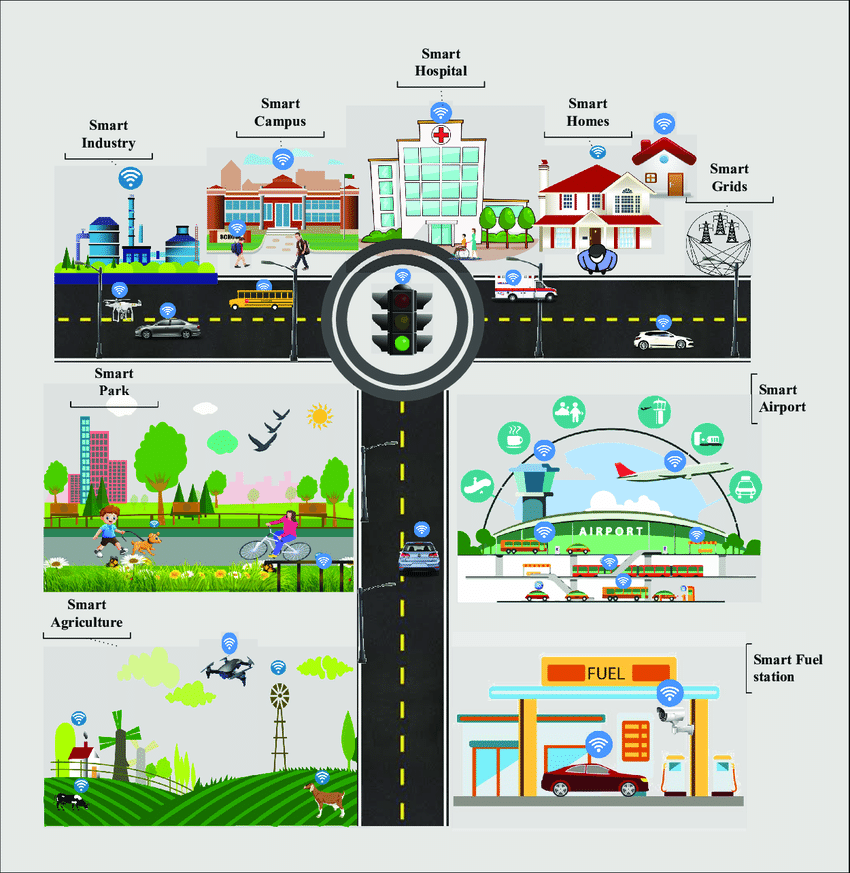
**Phase 3 : Development Part 1**

**Start building the IoT-enabled Environmental Monitoring in Parks system.**

**TEAM MEMBER**

**M. MAHASRINATH : 814821106020**



**In this part you will begin building your project**

The development phase of the IoT-enabled Environmental Monitoring in Parks system is a critical step towards creating a sustainable and efficient solution. This phase involves the implementation of key features and functionalities that will revolutionize park management and ensure the well-being of the environment and its visitors.

**Real-time Sensor Integration**

The IoT-enabled Environmental Monitoring in Parks system integrates real-time sensor data from various sources. By collecting and analyzing data from weather sensors, air quality monitors, and water level sensors, park authorities can make informed decisions to enhance visitor experiences and protect the park's natural resources.

To start building the IoT-enabled Environmental Monitoring in Parks system, follow these steps:

1. **Define Objectives and Requirements**

* Clearly outline the goals of your monitoring system (e.g., track temperature, humidity, air quality).
* Determine the number and types of sensors needed.

**2. Select IoT Devices**

* Choose appropriate sensors (temperature, humidity, air quality, etc.) based on your defined objectives.
* Ensure they are compatible with your chosen IoT platform.

**3. Choose IoT Platform**

* Select a reliable IoT platform or cloud service to manage data from the sensors (e.g., AWS IoT, Azure IoT, Google Cloud IoT, etc.).

**4. Hardware Setup**

* Install sensors in various locations within the parks.
* Ensure they are properly connected to power sources and have a stable internet connection (Wi-Fi, LoRa, NB-IoT, etc.).

**5. Software Development**

* Develop the necessary firmware to collect data from the sensors and transmit it to the IoT platform.

**6. Data Processing and Storage**

* Configure the IoT platform to receive, process, and store the data from the sensors.
* Implement data aggregation, filtering, and analytics as needed.

**7. User Interface**

* Design a user interface (web or mobile app) for visualizing and accessing the environmental data.

**8. Data Visualization**

* Implement graphs, charts, and other visualizations to present the collected data in a user-friendly format.

**9. Alerting System**

* Set up alerts for abnormal environmental conditions (e.g., extreme temperatures, poor air quality) to notify relevant authorities or users.

**10. Testing and Validation**

* Thoroughly test the system to ensure sensors are working correctly, data is being transmitted accurately, and alerts are triggered as expected.

**11. Deployment**

* Install the complete system in the parks, making sure all sensors are securely placed and connected.

**12. Monitoring and Maintenance**

* Regularly monitor the system's performance and address any issues promptly.
* Replaceor upgrade sensors as needed to ensure accurate data collection**.**

**13. Data Analysis and Reporting**

* Analyze the collected data for trends, patterns, and anomalies.
* Generate reports or summaries for stakeholders or the public.

**14. Compliance and Regulations**

* Ensure that your system complies with local regulations and privacy policies.

**15. Documentation and Training**

* Create documentation for system setup, maintenance, and troubleshooting.
* Provide training for park staff or administrators who will be using the system.

**Development of Python Script**

**Simplicity**

* Python is easier to learn and use than other languages.

**Robustness**

* Python has great error-handling capabilities making it a reliable choice.

**Connectivity**

* Python can communicate well with IoT devices, making it an excellent choice for IoT projects.

The Basic Python script that you can adapt for sending real-time environmental data from your IoT devices to the monitoring platform. Please replace the placeholders

(e.g., `<SENSOR\_DATA>`, `<SENSOR\_ID>`, `<API\_ENDPOINT>`, `<API\_KEY>`) with your actual data and API details.

python

import requests

import random

import time

# Function to simulate sensor data

def generate\_sensor\_data():

temperature = random.uniform(10.0, 30.0)

humidity = random.uniform(30.0, 70.0)

return {"temperature": temperature, "humidity": humidity}

# Function to send data to the monitoring platform

def send\_data\_to\_platform(sensor\_id, sensor\_data):

api\_key = "<API\_KEY>"

api\_endpoint = "<API\_ENDPOINT>"

headers = {

"Content-Type": "application/json",

"Authorization": f"Bearer {api\_key}"

}

payload = {

"sensor\_id": sensor\_id,

"data": sensor\_data

}

response = requests.post(api\_endpoint, json=payload, headers=headers)

if response.status\_code == 200:

print(f"Data sent successfully for sensor {sensor\_id}")

else:

print(f"Failed to send data for sensor {sensor\_id}. Status code: {response.status\_code}")

# Main loop to continuously send data

while True:

sensor\_id = "<SENSOR\_ID>"

sensor\_data = generate\_sensor\_data()

send\_data\_to\_platform(sensor\_id, sensor\_data)

# Adjust the sleep time (in seconds) based on your desired data transmission frequency

time.sleep(60)

# Send data every 60 seconds

**Explanation:**

1. `generate\_sensor\_data()`: This function generates random temperature and humidity data for simulation purposes. In a real-world scenario, you would replace this with actual data retrieval from your sensors.

2. `send\_data\_to\_platform()`: This function sends the sensor data to the monitoring platform using an HTTP POST request. Replace `<API\_ENDPOINT>` and `<API\_KEY>` with your actual API endpoint and authentication details.

3. The `while` loop ensures that the script runs continuously. You may need to adjust the frequency of data transmission based on your specific requirements.

**Conclusion**

* Through the IoT-enabled Environmental Monitoring in Parks system, parks' managers can access real-time environmental data to monitor and improve environmental conditions while providing a better experience for park visitors.