IOT PHASE_4

Technolog

TO BUILD AN AIR QUALITY MONITORING PROJECT USING WEB DEVELOPMENT TECHNOLOGIES, YOU CAN FOLLOW THESE STEPS:

01.

Define project requirements: Determine the specific features and functionalities you want in your air quality monitoring platform. Consider aspects like real-time data collection, data visualization, user authentication, and data analysis.

02.

Choose web development technologies: Select the appropriate web development technologies for your project. For the frontend, consider using HTML, CSS, and JavaScript, and utilize frontend frameworks like React, Angular, or Vue.js for efficient development. For the backend, you can choose from server-side languages such as Python (using frameworks like Django or Flask), Node.js (using frameworks like Express.js), or Java (using frameworks like Spring Boot).

03.

Design the database: Decide on the database technology to store the air quality data collected. MySQL, PostgreSQL, or MongoDB are popular choices. Design the database schema to efficiently store and retrieve the required information.

04.

Implement data collection: Set up mechanisms to collect air quality data from various sources, such as sensors or APIs. Develop integration points to capture data and store it in your database.

05.

Real-time monitoring: Implement real-time monitoring capabilities by using technologies like WebSockets or server-sent events. These technologies enable you to push live data updates to the client-side for real-time visualization.

06.

Data visualization: Build a user-friendly interface to display air quality data in a visually appealing format. Utilize charting libraries like D3.js, Chart.js, or Plotly.js to create interactive and informative visualizations.

07.

User authentication and access control: Implement user authentication mechanisms to ensure secure access to the platform. Users should be able to register, log in, and manage their own data. Consider using technologies such as JSON Web Tokens (JWT) for authentication

08.

Data analysis: Utilize data analysis or machine learning techniques to derive insights from the collected air quality data. This could involve identifying trends, correlations, or anomalies in the data.

09.

Test and deploy: Thoroughly test all aspects of the platform, ensuring its functionality, performance, and security. Once tested, deploy the platform on a reliable web hosting service or cloud platform.

CONCLUSION:

Remember to document your development process and share it for assessment as mentioned in the project requirements. Good luck with your air quality monitoring project!

Certainly! Here's an example of a Python script for air quality monitoring in an IoT project using the BME680 sensor:

```
import time
import board
import adafruit_bme680
import requests

# Initialize the BME680 sensor
i2c = board.I2C()  # Create the I2C bus object
bme680 = adafruit_bme680.Adafruit_BME680_I2C(i2c)

# Air quality thresholds
CO2_THRESHOLD = 1000  # ppm
TVOC_THRESHOLD = 50  # ppb

# API endpoint for sending air quality data
API_ENDPOINT = 'https://example.com/api/air_quality'

# Main program loop
while True:
    # Read sensor data
    temperature = bme680.temperature
    humidity = bme680.humidity
    gas = bme680.gas
    air_quality_index = bme680.air_quality_score
```

```
# Print sensor data
print(f'Temperature: {temperature:.2f} °C')
print(f'Humidity: {humidity:.2f} %')
print(f'Gas Resistance: {gas:.2f} ohms')
print(f'Air Quality Index: {air_quality_index}')

# Prepare JSON data
data = {
    'temperature': temperature,
    'humidity': humidity,
    'gas': gas,
    'air_quality_index': air_quality_index
}

# Send data to the API endpoint
try:
    response = requests.post(API_ENDPOINT, json=data)
    response.raise_for_status()
    print('Data sent successfully!')
except requests.exceptions.RequestException as e:
    print(f'Error occurred while sending data: {e}')

# Check air quality levels
if gas > CO2_THRESHOLD or air_quality_index > TVOC_THRESHOLD:
    print('High levels of CO2 or TVOC detected!')
    # Implement further actions (e.g., trigger alerts, activate ventilation systems)

# Delay between readings
time.sleep(10)
```

CONCLUSION

In this example, the script uses the adafruit_bme680 library to read the temperature, humidity, gas resistance, and air quality index from the BME680 sensor. It then sends the data to a specified API endpoint using the requests library.

You can customize the script by replacing API_ENDPOINT with the actual endpoint URL where you want to send the data. Additionally, you can modify or add actions within the conditional block to respond to high levels of CO2 or TVOC.

Remember to install the necessary libraries (adafruit_bme680, requests) before running the script.