AIR QUALITY MONITORING

Team Members:

Mukesh M (810021106051)

Sriram S (810021106081)

Vengatesan M (810021106091)

Vengatesan T (810021106092)

Sushmitha K (810021106083)

INTRODUCTION:

In this project, we see about how to create a data sharing platform to monitor air quality using iot. We use web development technologies such as HTML, CSS , JavaScript to gather and display specific values of temperature, air quality measure, humidity.

USING WEB DEVELOPMENT CREATE A PLATFORM THAT DISPLAYS REAL-TIME AIR QUALITY DATA:

HTML code for webpage:

```
<!DOCTYPE html>
<html>
<head>
   <title>Real-Time Air Quality Monitor</title>
   <link rel="stylesheet" type="text/css" href="styles.css">
</head>
```

```
<body>
  <img  src="https://www.ppsthane.com/wp-content/uploads/2023/02/IoT-based-Air-Pollution-Monitoring-System.png" alt="image" height="340" width="400">


    <h1>Real-Time Air Quality Monitor</h1>
    <div id="data-container">
      <div class="data">
        <h2>Temperature</h2>
        <p id="temperature">Loading...Getting data from Wifi module</p>
      </div>
      <div class="data">
        <h2>Humidity</h2>
        <p id="humidity">Loading...Getting data from Wifi module</p>
      </div>
      <div class="data">
        <h2>Particulate Matter Level</h2>
        <p id="pm-level">Loading...Getting data from Wifi module</p>
      </div>
      <div class="data">
        <h2>Air Quality Level</h2>
        <p id="aqi-level">Loading...Getting data from Wifi module</p>
      </div>
    </div>
    <div id="refresh-button">
      <button onclick="fetchAirQualityData()">Refresh Data</button>
```

```
    </div>
    <script src="script.js"></script>
</body>
</html>
```

CSS code:

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

h1 {
    color: lightsalmon;
}

.data {
    display: inline-block;
    margin: 20px;
    padding: 20px;
    background-color: lightblue;
    border: 1px solid #ccc;
    border-radius: 8px;
}

#refresh-button {
    margin-top: 20px;
```

```
}

button {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #0073e6;
  color: white;
  border: 1px solid #ccc;
  cursor: pointer;
}
```

JavaScript code:

```
function fetchAirQualityData() {
  fetch('http://your-esp8266-ip-address/data') // Replace with your ESP8266 IP
    .then(response => response.json())
    .then(data => {
      document.getElementById("temperature").textContent = `Temperature: ${data.temperature}°C`;
      document.getElementById("humidity").textContent = `Humidity: ${data.humidity}%`;
      document.getElementById("pm-level").textContent = `Particulate Matter Level: ${data.pmLevel} µg/m³`;
      document.getElementById("aqi-level").textContent = `Air Quality Level: ${data.aqiLevel}`;
    })
    .catch(error => console.error('Error fetching data:', error));
}
```

```
// Initial data fetch on page load
fetchAirQualityData();

// Set up an interval to periodically refresh the data (e.g., every 5 minutes)
setInterval(fetchAirQualityData, 300000); // 300000ms = 5 minutes
```

Code to connect esp8266 wifi module with webpage:

```
#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>

const char* ssid = "YourNetworkSSID";
const char* password = "YourNetworkPassword";

ESP8266WebServer server(80);

// Simulated air quality data (replace with actual data)
float temperature = 25.0;
float humidity = 50.0;
float particulateMatter = 15.0;
String airQuality = "Good";

void setup() {
  // Connect to Wi-Fi
```

```
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");

  // Set up web server routes
  server.on("/data", HTTP_GET, handleData);

  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}

void handleData() {
  // Replace with code to read and send sensor data
  String data = "{";
  data += "\"temperature\":\"" + String(temperature) + "\",";
  data += "\"humidity\":\"" + String(humidity) + "\",";
  data += "\"particulateMatter\":\"" + String(particulateMatter) + "\",";
```

```
    data += "\"airQuality\":\"" + airQuality + "\"";
    data += "}";
    server.send(200, "application/json", data);
}
```

DESIGN THE PLATFORM TO RECEIVE AND DISPLAY AIR QUALITY DATA SENT BY THE IOT DEVICES:

Real-Time Air Quality Monitor

Temperature

Temperature: 24Â°C

Humidity

Humidity: 52%

Particulate Matter Level

Particulate Matter Level: 30 Âµg/mÂ³

Air Quality Level

Air Quality Level: Moderate

Refresh Data

1. IoT Devices:

- Deploy air quality sensors and IoT devices at various locations to collect data.

- Ensure these devices are capable of measuring parameters such as particulate matter (PM2.5, PM10), volatile organic compounds (VOCs), carbon monoxide (CO), and more.

2. Data Transmission:

- IoT devices should transmit data securely to a central server. Use MQTT, HTTP, or other IoT communication protocols.

- Implement encryption and authentication for secure data transmission.

3. Data Ingestion:

- Develop a data ingestion system to receive and process data from IoT devices.

- This system should validate and preprocess the incoming data.

4. Database:

- Set up a database to store the received air quality data.

- Choose a suitable database technology like InfluxDB, MongoDB, or a time-series database for efficient storage and retrieval.

5.Data Processing:

- Implement data processing pipelines to clean, aggregate, and compute relevant metrics (e.g., AQI) from the raw sensor data.

6. API:

- Create a RESTful API or GraphQL API to provide access to the processed air quality data.

- The API should support queries for historical and real-time data.

7. Real-Time Updates:

- Implement real-time updates using technologies like WebSockets or server-sent events to push new air quality data to the user interface.

8. User Interface:

- Develop a web-based or mobile user interface for users to access the air quality data.

- Include a map-based display to show air quality in different locations.

9. Data Visualization:

- Utilize data visualization libraries to present air quality data in a user-friendly format, including charts, graphs, and maps.

10.User Authentication and Authorization:

- Implement user accounts and access control to ensure the privacy and security of air quality data.

11.Security:

- Ensure the platform is secure by implementing encryption, firewall rules, and regular security audits.

12.Monitoring:

- Implement logging and monitoring systems to track system performance and diagnose issues.

13.Data Backup and Recovery:

- Regularly back up the database and create a disaster recovery plan to prevent data loss.

14. Testing and Quality Assurance:

- Thoroughly test the platform, including data accuracy, performance, and security.

15. Maintenance and Updates:

- Plan for ongoing maintenance, bug fixes, and feature updates to keep the platform current.

CONCLUSION:

In conclusion, the implementation of an IoT data sharing platform for air quality monitoring holds great promise for addressing environmental challenges. By enabling real-time data collection and sharing, it empowers communities, researchers, and policymakers to make informed decisions, mitigating air pollution and safeguarding public health. This technology has the potential to drive positive change by fostering collaboration and transparency, ultimately leading to cleaner and healthier environments.