**IOT BASED AIR QUALITY MONITORING**

**Creating a real-time air quality data-sharing platform using web development technologies like HTML, CSS, and JavaScript is a multifaceted task. Here's a simplified outline of steps to get you started:**

1. **HTML Structure:**
   * Create the basic HTML structure for your platform. Include elements for data display, such as cards, tables, or graphs.
   * Set up placeholders for real-time data visualization.
2. **CSS Styling:**
   * Apply CSS to style your platform for a visually appealing and user-friendly interface.
   * Make sure it's responsive for various screen sizes.
3. **JavaScript for Real-Time Updates:**
   * Use JavaScript to establish a connection with the IoT devices. You may need WebSocket or AJAX for real-time data updates.
   * Create functions to handle incoming data and display it in your chosen format.
4. **Data Reception and Parsing:**
   * Implement a server-side component (e.g., Node.js or Python) to receive data from IoT devices.
   * Parse the data received and make it accessible to the frontend.
5. **Display Air Quality Data:**
   * Update your HTML elements with the incoming air quality data. You can display it in tables, charts, or any other suitable format.
6. **User Interaction:**
   * Add features for user interaction, like filters, search, or settings for customizing data display.
   * Consider implementing user authentication if this platform is meant for multiple users.
7. **Error Handling:**
   * Develop error handling mechanisms for data transmission issues or incorrect data formats.
8. **Testing and Debugging:**
   * Thoroughly test the platform to ensure that it updates in real time, displays accurate data, and handles different scenarios.
9. **Security:**
   * Implement security measures to protect data and the platform from potential vulnerabilities, like XSS or SQL injection.
10. **Documentation:**
    * Create clear documentation for how to use and maintain the platform. Include installation instructions, APIs, and any other relevant information.
11. **Deployment:**
    * Choose a suitable hosting platform and deploy your project so that it's accessible on the web.
12. **Scalability:**
    * Plan for scalability, especially if you expect a large number of IoT devices and users in the future.

**Remember that this is a simplified outline, and the actual development may require more in-depth work depending on your specific requirements and the technologies you choose. It's essential to keep user experience, data accuracy, and security in mind throughout the development process**.



**NOW,LETS SEE THE HTML, CSS, and JavaScript to design a simple platform for receiving and displaying air quality data sent by IoT sensors. This is a minimal representation and can be further enhanced and customized as needed.**

* **HTML (index.html):**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<link rel="stylesheet" href="styles.css">

<title>Air Quality Monitoring</title>

</head>

<body>

<div class="header">

<h1>Air Quality Monitoring Platform</h1>

</div>

<div class="data-container">

<h2>Real-time Air Quality Data</h2>

<table id="data-table">

<tr>

<th>Timestamp</th>

<th>Device ID</th>

<th>PM 2.5 (µg/m³)</th>

<th>PM 10 (µg/m³)</th>

</tr>

</table>

</div>

<script src="script.js"></script>

</body>

</html>

* **CSS (styles.css):**

body {

font-family: Arial, sans-serif;

}

.header {

background-color: #007BFF;

color: #fff;

padding: 20px;

text-align: center;

}

.data-container {

padding: 20px;

}

table {

border-collapse: collapse;

width: 100%;

}

table, th, td {

border: 1px solid #ddd;

}

th, td {

padding: 10px;

text-align: center;

}

th {

background-color: #f2f2f2;

}

* **JavaScript (script.js):**

// Simulated data from IoT sensors (replace this with actual data retrieval)

const airQualityData = [

{ timestamp: '2023-10-23 09:00', deviceId: 'Sensor1', pm25: 15, pm10: 20 },

{ timestamp: '2023-10-23 09:15', deviceId: 'Sensor2', pm25: 18, pm10: 23 },

// Add more data entries as needed

];

// Function to update the table with air quality data

function updateTable() {

const dataTable = document.getElementById('data-table');

dataTable.innerHTML = "<tr><th>Timestamp</th><th>Device ID</th><th>PM 2.5 (µg/m³)</th><th>PM 10 (µg/m³)</th></tr>";

airQualityData.forEach(entry => {

const row = dataTable.insertRow(-1);

const cell1 = row.insertCell(0);

const cell2 = row.insertCell(1);

const cell3 = row.insertCell(2);

const cell4 = row.insertCell(3);

cell1.innerHTML = entry.timestamp;

cell2.innerHTML = entry.deviceId;

cell3.innerHTML = entry.pm25;

cell4.innerHTML = entry.pm10;

});

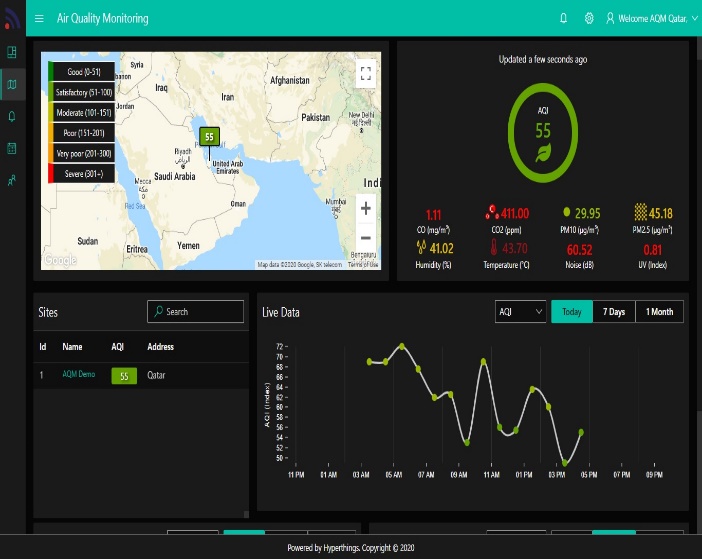
}

// Initial data update and refresh every 5 seconds (adjust as needed)

updateTable();

setInterval(updateTable, 5000);

* **The HTML file sets up the basic structure, including a header, a data container, and a table to display air quality data.**
* **The CSS file provides some basic styling for the elements.**
* **The JavaScript file simulates data from IoT sensors (you should replace this with actual data retrieval) and updates the table with the data in real-time.**



-PROJECT SUBMITTED BY,

NAME: M.Maheswari

NM ID: au713921106024

MAIL ID: maheswarimaheswari707@gmail.com

COLLEGE CODE: 7139