IOT—PHASE 4

PROJECT NAME—AIR QUALITY MONITORING

Certainly, in Phase 4, you can use web development technologies to create a platform for displaying real-time air quality data from IoT devices. Here’s a high-level overview of the steps you can follow:

1. \*\*Front-end Development:\*\*

- Create a user interface using HTML for displaying air quality data.

- Style the interface using CSS to make it user-friendly and visually appealing.

- Use JavaScript to handle real-time data updates and user interactions.

2. \*\*Back-end Development:\*\*

- Set up a server to receive data from IoT devices. You can use technologies like Node.js or Python.

- Implement APIs to handle incoming data, store it in a database, and serve it to the front end.

3. \*\*Database Integration:\*\*

- Choose a database system (e.g., MySQL, MongoDB) to store air quality data.

- Design the database schema to efficiently store and retrieve data.

4. \*\*Real-Time Data Integration:\*\*

- Use web sockets (e.g., WebSocket API) to establish a real-time connection between IoT devices and the platform.

- When IoT devices send data, the server should push it to connected clients in real-time.

5. \*\*User Authentication (Optional):\*\*

- Implement user authentication if you want to restrict access to the platform.

- You can use libraries like Passport.js for authentication.

6. \*\*Testing:\*\*

- Thoroughly test the platform to ensure it can receive, process, and display real-time air quality data accurately.

7. \*\*Deployment:\*\*

- Deploy the platform to a web server or cloud platform for public or private access.

8. \*\*Scalability and Security:\*\*

- Plan for scalability, especially if the number of IoT devices and users grows.

- Implement security measures to protect the platform and data.

9. \*\*Documentation:\*\*

- Create documentation for the platform, including how to use it, the data format, and API endpoints.

10. \*\*Maintenance:\*\*

- Regularly update and maintain the platform to ensure it remains functional and secure.

Remember to adapt and refine your development process based on the specific requirements and technologies you’re using for your project. Good luck with your data-sharing platform development!

Creating a real-time air quality data platform involves several steps. Here’s a high-level overview using HTML, CSS, and JavaScript:

1. \*\*Data Source\*\*: Find a reliable source for real-time air quality data, such as an API from a weather or environmental agency.
2. \*\*HTML Structure\*\*: Create the basic HTML structure for your platform. This might include a header, content area, and a container for displaying the data.

```html

<!DOCTYPE html>

<html>

<head>

<title>Real-Time Air Quality</title>

</head>

<body>

<header>

<h1>Real-Time Air Quality Platform</h1>

</header>

<div id=”data-container”>

<!—Air quality data will be displayed here 🡪

</div>

<script src=”script.js”></script>

</body>

</html>

```

1. \*\*CSS Styling\*\*: Use CSS to style your platform. You can define the layout, fonts, colors, and overall design.

```css

Body {

Font-family: Arial, sans-serif;

Background-color: #f0f0f0;

Margin: 0;

Padding: 0;

}

Header {

Background-color: #007BFF;

Color: white;

Text-align: center;

Padding: 20px;

}

#data-container {

Margin: 20px;

Padding: 20px;

Background-color: white;

Border-radius: 5px;

Box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);

}

```

1. \*\*JavaScript\*\*: Use JavaScript to fetch real-time air quality data from the API and update the content on your platform.

```javascript

// Replace with the actual API endpoint

Const apiUrl = ‘https://api.example.com/air-quality’;

Async function fetchAirQualityData() {

Try {

Const response = await fetch(apiUrl);

Const data = await response.json();

// Update the data container with the fetched information

Document.getElementById(‘data-container’).innerHTML = `

<h2>Current Air Quality</h2>

<p>Location: ${data.location}</p>

<p>PM2.5: ${data.pm25} µg/m³</p>

<p>PM10: ${data.pm10} µg/m³</p>

<p>NO2: ${data.no2} µg/m³</p>

`;

} catch (error) {

Console.error(‘Error fetching air quality data’, error);

}

}

// Fetch data initially and then update it at regular intervals (e.g., every 5 minutes)

fetchAirQualityData();

setInterval(fetchAirQualityData, 300000);

```

1. \*\*Deployment\*\*: Host your platform on a web server. You can use a service like GitHub Pages, Netlify, or your own web hosting.
2. \*\*Testing and Optimization\*\*: Test your platform, ensure it works across different browsers, and optimize it for performance.
3. \*\*User Interface Enhancements\*\*: Consider adding interactive features, charts, or maps for a better user experience.
4. \*\*Documentation\*\*: Provide clear instructions on how to use your platform and the source of the air quality data.

Remember to replace the placeholder URLs and data with your actual data source and information. This is a simplified example, and real-world projects may require more robust error handling and security measures, especially when dealing with real-time data.

BY

R.SWETHA

21EC144

923321106044

ECE 3RD YEAR