PHASE 4 PROJECT TITLE : NOISE POLLUTION MONITORING

Name: S.GOKUL

Register number: 210521205017

Here we are creating a webpage for indicating the noise as sound as using html, css and javascript.

1. HTML, CSS, and JavaScript:

HTML: Create the structure of your web page, including headers, content areas, and placeholders for noise level data.

CSS: Style your web page for a visually appealing and user-friendly design. Consider responsive design to ensure it works on different devices.

JavaScript: Use JavaScript to fetch and display real-time noise level data. You may need to use technologies like WebSockets or AJAX for this purpose.

2. Real-Time Noise Data Integration:

To display real-time noise level data, you'll need to integrate with noise monitoring devices or APIs that provide this data.

Utilize JavaScript libraries or frameworks for data visualization (e.g., D3.js or Chart.js) to create charts or graphs to represent noise level data in real-time.

3. Data Fetching and Updates:

Implement a mechanism to fetch real-time data from your data source at regular intervals or in real-time (e.g., WebSocket for instant updates).

Use JavaScript to dynamically update the displayed noise level data in response to new data.

4. User Interface:

Design a user-friendly interface to present the real-time noise level data. This may include charts, gauges, or numeric displays.

Consider providing controls or settings for users to customize their view or set alerts for specific noise levels.

5. Notifications:

Implement notification mechanisms (e.g., browser notifications or in-app alerts) to inform users when noise levels exceed certain thresholds.

6. Location-Based Features:

If applicable, use geolocation services to show noise level data for the user's current location or allow users to select specific locations.

7. Data Storage and Logging:

Store historical noise data for analysis and reporting. Use technologies like local storage, session storage, or a backend database if necessary.

Code:

Index.html

```
!DOCTYPE html>

<html>

<head>

    <title>Real-Time Noise Monitor</title>

    <link rel="stylesheet" type="text/css" href="styles.css">

</head>

<body>

    <div class="container">
```

```
        <h1>Noise Level Monitor</h1>
        <div class="noise-display">
            <h2>Current Noise Level:</h2>
            <div id="noise-level">Loading...</div>
        </div>
        <div class="controls">
            <label for="threshold">Set Alert Threshold (dB):</label>
            <input type="number" id="threshold" placeholder="e.g., 70">
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Styles.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
}
.container {
    max-width: 800px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
```

```
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.2);
}
.noise-display {
    text-align: center;
}
.controls {
    margin-top: 20px;
}
input {
    width: 100px;
}
```

Script.js

```
document.addEventListener("DOMContentLoaded", function () {
    const noiseLevelElement = document.getElementById("noise-level");
    const thresholdInput = document.getElementById("threshold");
    // Simulate real-time noise data updates (replace with actual data source).
    const updateNoiseData = () => {
        const randomNoiseLevel = Math.floor(Math.random() * 101); // Random value between 0 and 100 dB.
        noiseLevelElement.textContent = randomNoiseLevel + " dB";
        // Check if the noise level exceeds the user-defined threshold.
        const threshold = parseFloat(thresholdInput.value);
```

```
        if (!isNaN(threshold) && randomNoiseLevel > threshold) {
            alert("Noise level exceeds the threshold!");
        }
    };
    // Update noise data every 5 seconds (adjust as needed).
    setInterval(updateNoiseData, 5000);
});
```