**Phase 4**

**Development Part 2**

**Environmental System**

In this part you will continue building your project.

Continue building the project by developing the environmental monitoring platform.

Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time environmental data.

Design the platform to receive and display real-time temperature and humidity data from IoT devices.

**Html Code : (index.html)**

<!DOCTYPE html>

<html>

<head>

    <meta charset="UTF-8">

    <title>Environmental Monitoring Platform</title>

    <link rel="stylesheet" type="text/css" href="styles.css"> <!-- Link to your CSS file -->

</head>

<body>

    <header>

        <h1>Environmental Monitoring Platform</h1>

    </header>

    <nav>

        <ul>

            <li><a href="#real-time-data">Real-Time Data</a></li>

            <li><a href="#data-visualization">Data Visualization</a></li>

            <li><a href="#about">About Us</a></li>

        </ul>

    </nav>

    <p align=" center">Environmental monitoring refers to the process of systematically observing, measuring, <br>

        and assessing various environmental factors and conditions to understand and manage the state of the environment. <br>

        This practice is essential for tracking changes in the environment, identifying potential issues, and making informed decisions

        to protect and sustain the natural world. <br>

        Environmental monitoring encompasses a wide range of parameters and areas, including:</p>

    <section id="updates">

        <h2>Latest Updates</h2>

        <ul>

            <li><strong>October 2023:</strong> New sensor data added for air quality monitoring.</li>

            <li><strong>September 2023:</strong> Improved data visualization features.</li>

        </ul>

    </section>

    <main>

        <section class="sensor-data">

            <h2>Real-Time Data</h2>

            <div class="data-display">

                <div class="data-item">

                    <h3>Temperature</h3>

                    <p id="temperature">Loading...</p> <!-- Temperature value will be updated via JavaScript -->

                </div>

                <div class="data-item">

                    <h3>Humidity</h3>

                    <p id="humidity">Loading...</p> <!-- Humidity value will be updated via JavaScript -->

                </div>

            </div>

        </section>

        <section class="data-visualization">

            <h2>Data Visualization</h2>

            <canvas id="chart"></canvas> <!-- Add data visualization elements here, e.g., charts or graphs -->

        </section>

        <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

        <canvas id="lineChart"></canvas>

    </main>

    <section id="about">

        <h2>About Us</h2>

        <p>Welcome to the Environmental Monitoring Platform, where we provide real-time data and visualizations to help you monitor environmental conditions. Our mission is to promote sustainability and environmental awareness.</p>

    </section>

    <section id="contact">

        <h2>Contact Us</h2>

        <p>If you have any questions or feedback, please feel free to contact us at <a href="mailto:contact@example.com">contact@example.com</a>.</p>

    </section>

    <section id="disclaimer">

        <h2>Disclaimer</h2>

        <p>Information provided on this platform is for informational purposes only. Please consult with experts for critical decisions related to the environment.</p>

    </section>

    <footer>

        <p>&copy; 2023 Environmental Monitoring Platform</p>

    </footer>

    <script src="node.js"></script> <!-- Link to your JavaScript file for real-time updates -->

</body>

</html>

1. **<!DOCTYPE html>**: This declaration specifies that the document follows HTML5 standards.
2. **<html>**: The root element of an HTML document.
3. **<head>**: This section contains metadata and links to external resources. In this case, it links to an external CSS file and sets the document's character encoding.
4. **<meta charset="UTF-8">**: Specifies the character encoding of the document as UTF-8, which is a widely used encoding for handling text in different languages.
5. **<title>Environmental Monitoring Platform</title>**: Sets the title of the web page, which appears in the browser's title bar or tab.
6. **<link rel="stylesheet" type="text/css" href="styles.css">**: Links an external CSS file named "styles.css" to style the web page.
7. **<body>**: The main content of the web page is contained within the **<body>** element.
8. **<header>**: The header section typically contains the title or logo of the website. In this case, it displays the title "Environmental Monitoring Platform."
9. **<nav>**: This section contains navigation links. It's structured as an unordered list **<ul>** with list items **<li>**, each of which contains an anchor **<a>** element for navigation.
10. **<p align="center">**: This paragraph element aligns its text to the center. However, the **align** attribute is deprecated in HTML5, and it's recommended to use CSS for text alignment.
11. The **<p>** element provides information about environmental monitoring, its purpose, and what it encompasses. It's a description of the environmental monitoring concept.
12. **<section id="updates">**: This section is labeled "Latest Updates" and contains an unordered list **<ul>** of updates. Each update is presented as a list item **<li>** with a date and a description.
13. **<main>**: The main content of the web page, which contains various sections related to real-time data, data visualization, and information about the platform.
14. **<section class="sensor-data">**: This section is dedicated to displaying real-time sensor data. It contains two data items, "Temperature" and "Humidity," each with a heading **<h3>** and a placeholder paragraph **<p>** for data to be loaded via JavaScript.
15. **<section class="data-visualization">**: This section is for data visualization. It currently includes a canvas element **<canvas>** with the ID "chart" where data visualization elements like charts or graphs can be added. Additionally, it includes a script to include the Chart.js library for creating charts and another canvas element with the ID "lineChart."
16. **<section id="about">**: This section provides information about the platform, its mission, and what it offers.
17. **<section id="contact">**: This section offers contact information, including an email address where users can reach out with questions or feedback.
18. **<section id="disclaimer">**: This section includes a disclaimer regarding the informational nature of the platform and advises users to consult with experts for critical decisions related to the environment.
19. **<footer>**: The footer section displays a copyright notice for the year 2023, indicating ownership of the content.
20. **<script src="node.js"></script>**: This script element links to an external JavaScript file named "node.js." This file is used for real-time updates and functionality related to the platform.

CSS Code : (Styles.css)

/\* Reset some default styles to ensure consistency across browsers \*/

html, body, h1, h2, h3, p {

    margin: 0;

    padding: 0;

}

body {

    font-family: Arial, sans-serif; /\* Set the default font for the entire page \*/

}

header {

    background-color: #333; /\* Set a dark background color for the header \*/

    color: #fff; /\* Set text color to white \*/

    padding: 10px;

    text-align: center;

}

h1 {

    font-size: 24px;

}

nav ul {

    list-style: none;

    text-align: center;

}

nav ul li {

    display: inline;

    margin-right: 20px;

}

nav a {

    text-decoration: none;

    color: #007bff;

}

nav a:hover {

    text-decoration: underline;

}

#updates {

    background-color: #f0f0f0; /\* Set a light background color for the "Latest Updates" section \*/

    padding: 20px;

}

#updates h2 {

    font-size: 22px;

}

.main-content {

    padding: 20px;

}

.sensor-data {

    border: 1px solid #ccc;

    padding: 20px;

}

.data-item {

    margin: 10px 0;

}

.data-item h3 {

    font-size: 18px;

}

.data-visualization {

    margin-top: 20px;

}

canvas#lineChart {

    width: 100%; /\* Make the chart fill the available space \*/

    height: 300px; /\* Set a fixed height for the chart \*/

}

h2 {

    font-size: 22px;

    margin-top: 20px;

}

ul {

    list-style-type: disc;

    margin-left: 20px;

}

footer {

    background-color: #333;

    color: #fff;

    text-align: center;

    padding: 10px;

}

/\* Add specific styles for other sections like "About Us," "Contact Us," and "Disclaimer" if needed \*/

Java Script : (node.js)

// Function to update the temperature and humidity values

function updateSensorData() {

    // Simulate data retrieval (you would replace this with actual data)

    const temperatureValue = Math.random() \* 30 + 10; // Random temperature between 10 and 40

    const humidityValue = Math.random() \* 60 + 40; // Random humidity between 40 and 100

    // Update the HTML elements with the new values

    document.getElementById("temperature").textContent = temperatureValue.toFixed(2) + "°C";

    document.getElementById("humidity").textContent = humidityValue.toFixed(2) + "%";

}

// Function to create a random chart for data visualization (you can replace this with a real chart library)

function createRandomChart() {

    const ctx = document.getElementById("chart").getContext("2d");

    const data = {

        labels: ["Jan", "Feb", "Mar", "Apr", "May"],

        datasets: [

            {

                label: "Temperature",

                data: [Math.random() \* 10 + 20, Math.random() \* 10 + 20, Math.random() \* 10 + 20, Math.random() \* 10 + 20, Math.random() \* 10 + 20],

                borderColor: "#FF5733",

                borderWidth: 2,

            },

            {

                label: "Humidity",

                data: [Math.random() \* 20 + 80, Math.random() \* 20 + 80, Math.random() \* 20 + 80, Math.random() \* 20 + 80, Math.random() \* 20 + 80],

                borderColor: "#33FF57",

                borderWidth: 2,

            },

        ],

    };

    const config = {

        type: "line",

        data: data,

    };

    new Chart(ctx, config);

}

// Update sensor data and chart every 5 seconds (you can adjust the interval)

setInterval(() => {

    updateSensorData();

    createRandomChart();

}, 5000);

// Call the update functions on page load

updateSensorData();

createRandomChart();

// Function to create a line chart using Chart.js

function createLineChart() {

    const ctx = document.getElementById("lineChart").getContext("2d");

    // Sample data (replace with your actual data)

    const data = {

        labels: ["Jan", "Feb", "Mar", "Apr", "May"],

        datasets: [

            {

                label: "Temperature (°C)",

                data: [15, 18, 20, 22, 25],

                borderColor: "#FF5733",

                borderWidth: 2,

                fill: false,

            },

            {

                label: "Humidity (%)",

                data: [45, 42, 40, 38, 35],

                borderColor: "#33FF57",

                borderWidth: 2,

                fill: false,

            },

        ],

    };

    const options = {

        scales: {

            x: {

                type: "category",

                labels: data.labels,

            },

            y: {

                beginAtZero: true,

                max: 30, // Set the maximum value for the y-axis

            },

        },

    };

    const config = {

        type: "line",

        data: data,

        options: options,

    };

    new Chart(ctx, config);

}

// Call the createLineChart function when the document is ready

document.addEventListener("DOMContentLoaded", createLineChart);

The provided JavaScript code contains functions to update sensor data, create a random chart for data visualization, and create a line chart using the Chart.js library. Let's break down the code and provide an explanation:

1. **updateSensorData Function**:
   * This function simulates the update of temperature and humidity values. In a real application, you would replace the random data with actual sensor data.
   * It generates random values for temperature and humidity within specified ranges.
   * It updates the HTML elements with the new values, specifically the elements with IDs "temperature" and "humidity."
2. **createRandomChart Function**:
   * This function creates a random chart for data visualization. Please note that in a real application, you would use a real charting library and actual data.
   * It utilizes the Chart.js library to create a line chart.
   * The chart includes two datasets (Temperature and Humidity) with random data points.
   * The chart's configuration includes labels, colors, and other properties.
   * The chart is drawn on the canvas element with the ID "chart."
3. **Updating Data and Chart on an Interval**:
   * **setInterval** is used to repeatedly update sensor data and create a new random chart every 5 seconds (5000 milliseconds). You can adjust the interval to suit your needs.
   * The **updateSensorData** function and **createRandomChart** function are called within the interval to provide updated data and charts.
4. **Calling Update Functions on Page Load**:
   * To ensure that the sensor data and initial chart are displayed when the page loads, the **updateSensorData** and **createRandomChart** functions are called immediately after defining them.
5. **createLineChart Function**:
   * This function is responsible for creating a line chart using the Chart.js library. It is distinct from the **createRandomChart** function.
   * It defines the sample data for the line chart (temperature and humidity values for different months). In a real application, you would replace this with your actual data.
   * The function sets various chart properties, including labels, colors, and scale configurations.
6. **Calling createLineChart on Page Load**:
   * To ensure that the line chart is displayed when the document is ready, the **createLineChart** function is called when the "DOMContentLoaded" event is triggered.

**Result :**





