**PHASE -4**

**PROBLEM AND INNOVATION PROCEDURE**

|  |  |
| --- | --- |
| **Date** | 26/10/2023 |
| **Team ID** | Proj\_223986\_Team |
| **Project Name** | Environmental Monitoring in Parks |
| **Students with ID** | **Vimal K**  Nirmalkumar K  Parthiban G  Shreepal S  Sabari Nathan C |
| **Thingspeak link** | [**https://thingspeak.com/channels/2307358/private\_show**](https://thingspeak.com/channels/2307358/private_show) |
| **Wokwi link** | [**https://wokwi.com/projects/378812252192158721**](https://wokwi.com/projects/378812252192158721) |

**Report: IoT Environmental Monitoring Website**

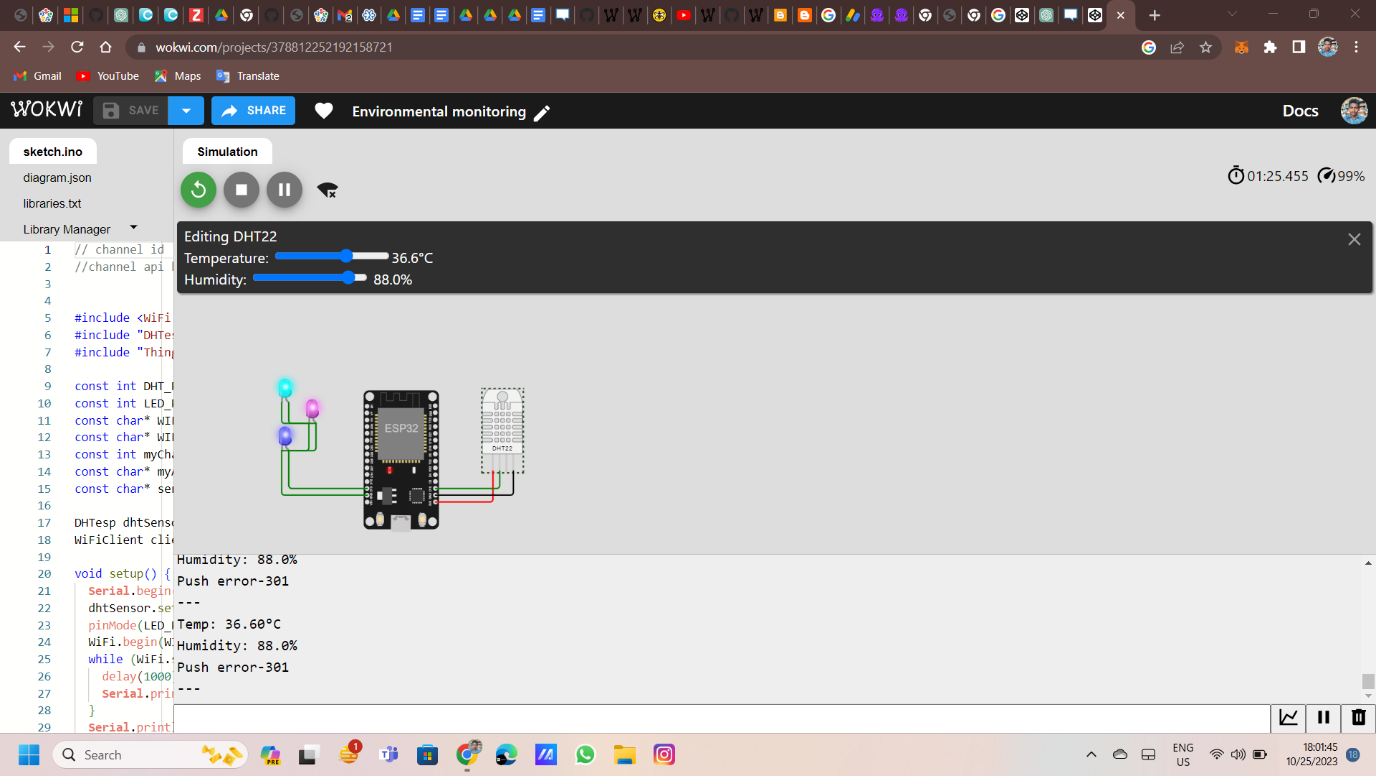
**Introduction**

The purpose of this report is to detail the development of a website that connects to an IoT environmental monitoring system. The website utilizes HTML, CSS, and JavaScript to provide a user-friendly interface for real-time data visualization, historical data display, and basic data analysis.

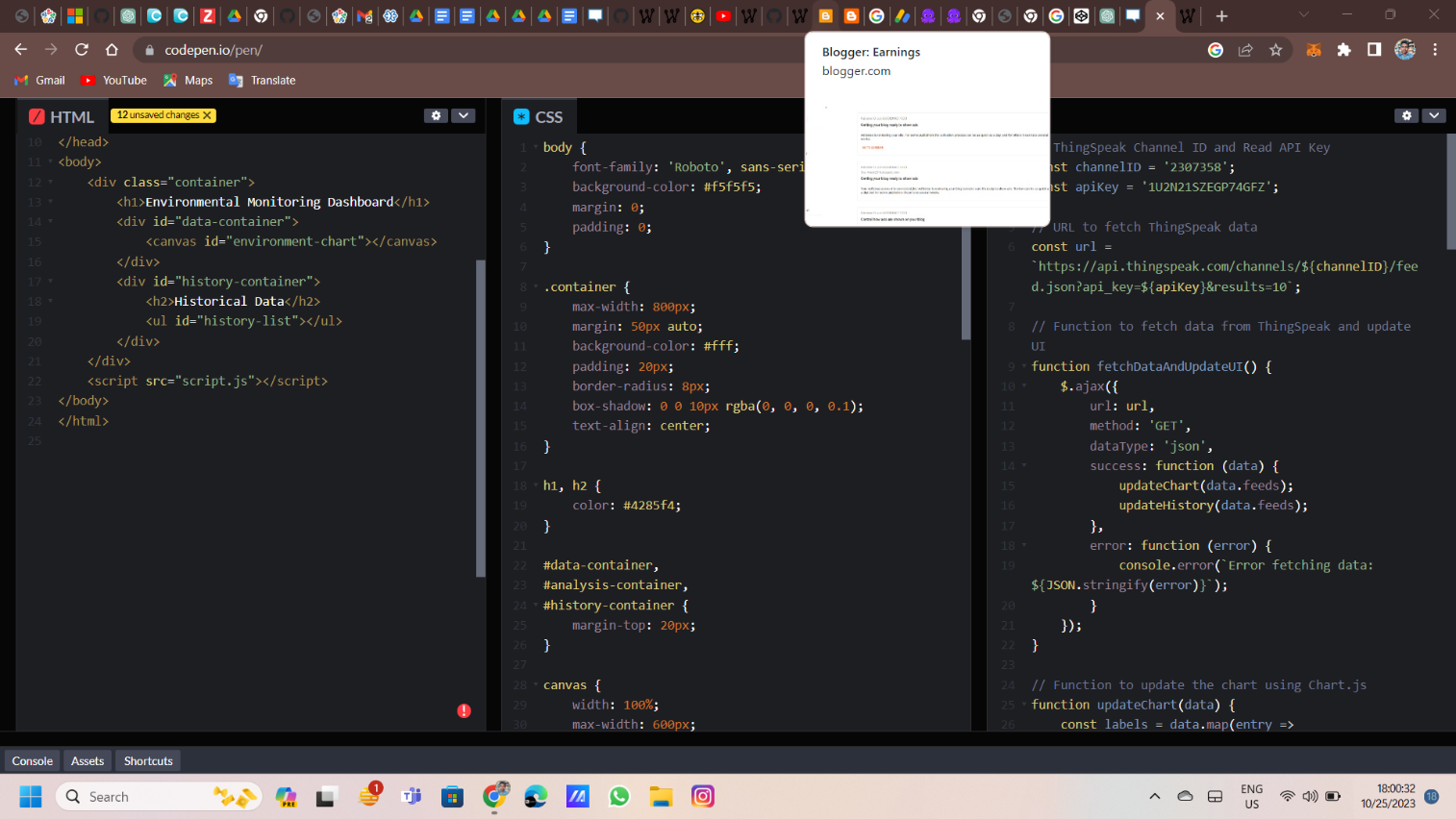
**Objectives**

Real-time Data Display: Develop a webpage that dynamically displays real-time environmental data fetched from an IoT device.

Historical Data Presentation: Include a historical data section to showcase past environmental conditions.



**Data Analysis Tool:** Implement a basic data analysis tool to compare environmental data over a specified time range.

**Technologies Used**

**HTML: For structuring the webpage.**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Environmental Monitoring Dashboard</title>

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

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

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

</head>

<body>

<div class="container">

<h1>Environmental Monitoring Dashboard</h1>

<div id="data-container">

<canvas id="environment-chart"></canvas>

</div>

<div id="analysis-container">

<h2>Data Analysis Tool</h2>

<label for="startDate">Start Date:</label>

<input type="date" id="startDate">

<label for="endDate">End Date:</label>

<input type="date" id="endDate">

<button onclick="compareData()">Compare</button>

<div id="comparison-chart-container">

<canvas id="comparison-chart"></canvas>

</div>

</div>

<div id="history-container">

<h2>Historical Data</h2>

<ul id="history-list"></ul>

</div>

</div>

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

</body>

</html>

**CSS: For styling and layout.**

body {

font-family: 'Roboto', sans-serif;

background-color: #f5f5f5;

margin: 0;

padding: 0;

}

.container {

max-width: 800px;

margin: 50px auto;

background-color: #fff;

padding: 20px;

border-radius: 8px;

box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);

text-align: center;

}

h1, h2 {

color: #4285f4;

}

#data-container,

#analysis-container,

#history-container {

margin-top: 20px;

}

canvas {

width: 100%;

max-width: 600px;

margin: 20px auto;

}

#comparison-chart-container {

margin-top: 20px;

}

#analysis-container label,

#analysis-container button {

margin: 10px;

}

#comparison-chart-container canvas {

width: 100%;

max-width: 600px;

margin-top: 10px;

}

#history-container ul {

list-style: none;

padding: 0;

margin: 0;

}

li {

margin-bottom: 10px;

color: #555;

}

**JavaScript: For dynamic content, real-time updates, and data analysis.**

**Chart.js: A JavaScript library for creating interactive charts.**

// ThingSpeak 2307358and Read API Key

const channelID = '2307358';

const apiKey = '1U2N21SZEGP74GFZ';

// URL to fetch ThingSpeak data

const url = `https://api.thingspeak.com/channels/${channelID}/feed.json?api\_key=${apiKey}&results=10`;

// Function to fetch data from ThingSpeak and update UI

function fetchDataAndUpdateUI() {

$.ajax({

url: url,

method: 'GET',

dataType: 'json',

success: function (data) {

updateChart(data.feeds);

updateHistory(data.feeds);

},

error: function (error) {

console.error(`Error fetching data: ${JSON.stringify(error)}`);

}

});

}

// Function to update the chart using Chart.js

function updateChart(data) {

const labels = data.map(entry => entry.created\_at);

const temperatureData = data.map(entry => entry.field1);

const humidityData = data.map(entry => entry.field2);

const ctx = document.getElementById('environment-chart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Temperature (°C)',

borderColor: '#e44d26',

data: temperatureData,

},

{

label: 'Humidity (%)',

borderColor: '#4285f4',

data: humidityData,

},

],

},

});

}

// Function to update the historical data list

function updateHistory(data) {

const historyList = $('#history-list');

historyList.empty();

data.forEach(entry => {

const listItem = `<li>${entry.created\_at} - Temperature: ${entry.field1}°C, Humidity: ${entry.field2}%</li>`;

historyList.append(listItem);

});

}

// Function to compare data over time

function compareData() {

const startDate = $('#startDate').val();

const endDate = $('#endDate').val();

if (!startDate || !endDate) {

alert('Please select both start and end dates.');

return;

}

$.ajax({

url: `${url}&start=${startDate}&end=${endDate}`,

method: 'GET',

dataType: 'json',

success: function (data) {

updateComparisonChart(data.feeds);

},

error: function (error) {

console.error(`Error fetching data for comparison: ${JSON.stringify(error)}`);

}

});

}

// Function to update the comparison chart using Chart.js

function updateComparisonChart(data) {

const labels = data.map(entry => entry.created\_at);

const temperatureData = data.map(entry => entry.field1);

const humidityData = data.map(entry => entry.field2);

const ctx = document.getElementById('comparison-chart').getContext('2d');

new Chart(ctx, {

type: 'line',

data: {

labels: labels,

datasets: [

{

label: 'Temperature (°C)',

borderColor: '#e44d26',

data: temperatureData,

},

{

label: 'Humidity (%)',

borderColor: '#4285f4',

data: humidityData,

},

],

},

});

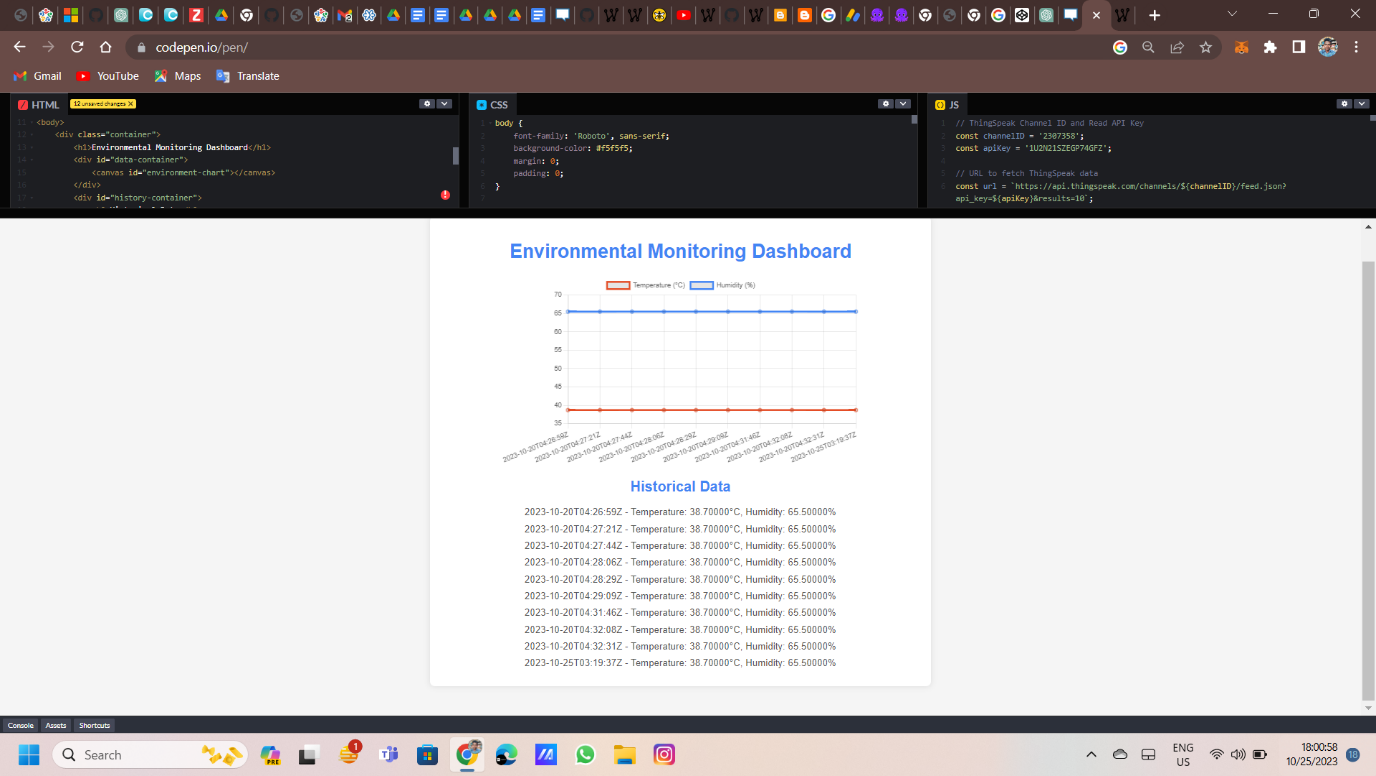
}

// Update data every 10 seconds

setInterval(fetchDataAndUpdateUI, 10000);

// Initial data fetch on page load

fetchDataAndUpdateUI();

**Implementation**

**Real-time Data Display**

The real-time data display is achieved by fetching live data from the ThingSpeak API, which is connected to the IoT environmental monitoring system. The data is updated at regular intervals to provide users with the most recent environmental conditions. Chart.js is employed to visualize temperature and humidity trends.

**Historical Data Presentation**

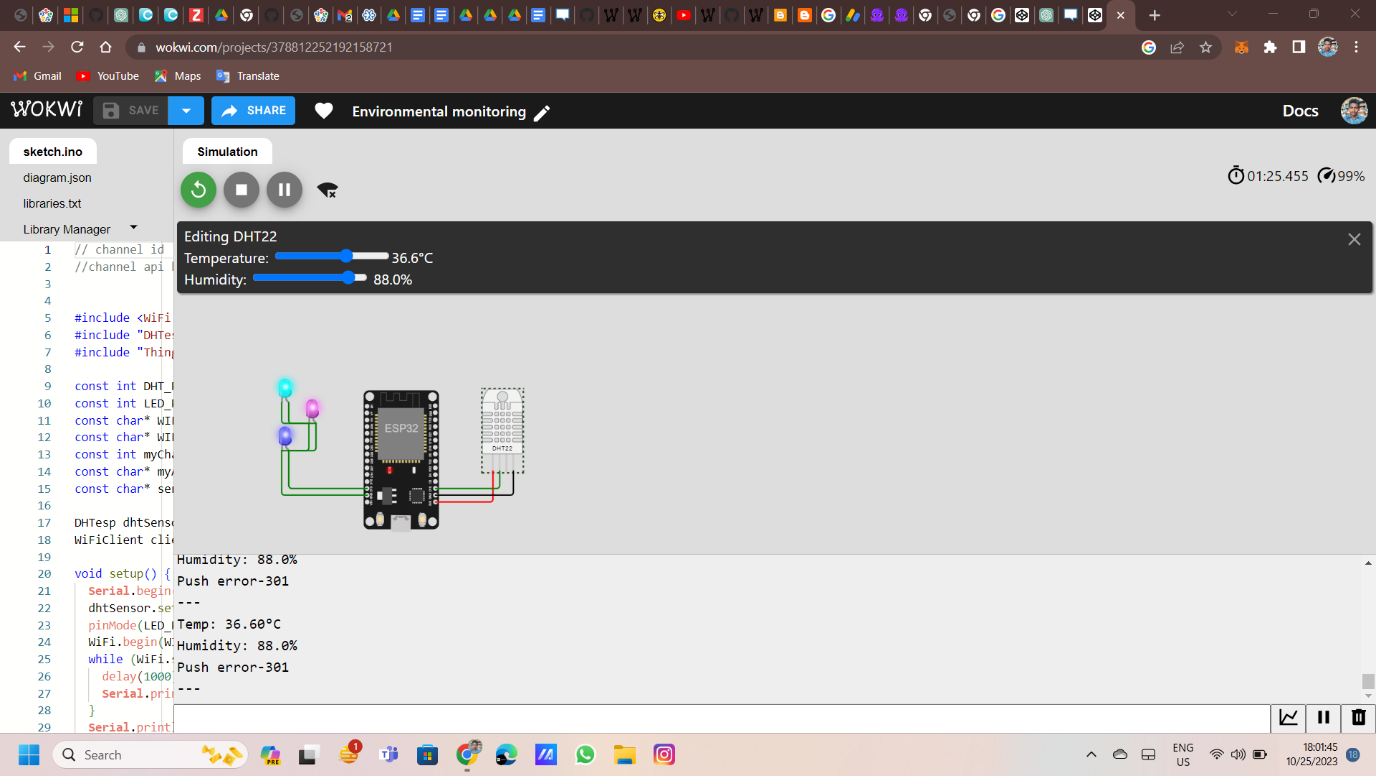
A historical data section is incorporated to showcase past environmental conditions. The website fetches historical data from ThingSpeak, presenting it in a user-friendly list format. Users can review historical data to identify patterns or trends over time.

**Data Analysis Tool using codepen**

A simple data analysis tool is integrated into the website, allowing users to compare environmental data over a specified time range. Users can input start and end dates, and the website fetches and displays the corresponding data in a separate chart for detailed analysis.

**User Experience**

The website is designed to be visually appealing and user-friendly. Responsive design ensures compatibility with various devices. The color scheme and layout are chosen to enhance readability and engagement. The inclusion of charts aids in the intuitive understanding of data trends.



**Conclusion**

The development of the IoT environmental monitoring website has successfully achieved its objectives. Users can now access real-time data, review historical trends, and perform basic data analysis. The use of HTML, CSS, and JavaScript, along with the Chart.js library, has allowed for the creation of a responsive and interactive platform.

**Future Enhancements**

**Advanced Analytics**: Implement more sophisticated data analysis tools for in-depth insights.

User Authentication: Introduce user accounts for personalized data tracking.

Notifications: Enable notifications for significant environmental changes.

Localization: Provide multi-language support for a broader user base.