KINGS ENGINEERING COLLEGE

PROJECT TITTLE: ENVIRONMENTAL MONITORING

BATCH MEMBERS: G.DEEPIKA, T.DHANALAKSHMI, D.DHARSHINI, D.S.GEETHA

DEPARTMENT: B.E-BIOMEDICAL ENGINEERING

MENTOR NAME: MARY LALITHA

Building the project by developing the environmental monitoring platform.

For an environmental monitoring platform, we have to consider a few key components:

Sensor Integration: Identify the types of environmental parameters you want to monitor (temperature, humidity, air quality, etc.). Choose appropriate sensors for each parameter. Integrate these sensors into your platform, ensuring they can communicate data effectively.

Data Collection: Implement a robust data collection system. This could involve setting up a central hub to aggregate data from different sensors.

Data Storage: Choose a database system to store the collected data. Consider factors like scalability, reliability, and ease of querying. Set up a structured database schema to organize and store different types of environmental data.

Real-time Monitoring: Implement real-time monitoring features to allow users to view current environmental conditions. Use visualizations like graphs or charts to represent data trends over time.

Alerting System: Develop an alerting system that notifies users when certain environmental parameters go beyond predefined thresholds. Alerts could be in the form of notifications through email, SMS, or a dedicated dashboard.

User Interface: Design a user-friendly interface for your platform. This could be a web-based dashboard or a mobile app. Include features for users to customize their monitoring preferences and view historical data.

Analytics: Implement analytical tools to derive insights from the collected data. Use machine learning algorithms to predict future environmental conditions or detect patterns.

Security: Prioritize security measures to protect the data and the platform. This includes encryption, secure APIs, and user authentication.

Scalability: Design your platform to scale with an increasing number of sensors and users. Consider cloud services for scalability and flexibility.

Documentation: Create thorough documentation for users, explaining how to set up sensors, interpret data, and use the platform effectively.

Integration with External Systems: Allow your platform to integrate with other systems, such as weather APIs or external databases, to enhance the breadth of information.

Maintenance and Updates: Plan for regular maintenance and updates to ensure the platform remains secure and up-to-date with the latest technologies

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

Let's break it down into steps. We'll use HTML for structure, CSS for styling, and JavaScript for real-time data updates. You might also need a backend for fetching and processing the environmental data

1.HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Environmental Data Platform</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Real-time Environmental Data</h1>
    </header>
    <main>
```

```
        <!-- Your data display goes here -->
    </main>
    <script src="app.js"></script>
</body>
</html>
```

2.CSS Styling (styles.css):

```
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 1em;
}

main {
    padding: 2em;
}
```

3.JavaScript for Real-time Updates (app.js):

```
// You'll need to replace this with actual code to fetch real-time data
function fetchData() {
    // Fetch environmental data from your backend or API
    // Update the DOM with the received data
}

// Update data every 5 seconds (adjust as needed)
setInterval(fetchData, 5000);

// Initial data fetch when the page loads
fetchData();
```

4.Backend:

We need a backend to handle data fetching. Here we use Node.js with Express, Django with Python, or any other backend technology.

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

let's assume a simple API that provides real-time temperature and humidity data.

1.HTML Structure:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Environmental Data Platform</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Real-time Temperature & Humidity Data</h1>
    </header>
    <main>
        <div id="temperature"></div>
        <div id="humidity"></div>
    </main>
    <script src="app.js"></script>
</body>
</html>
```

2.CSS Styling (styles.css):

```
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}

header {
    background-color: #333;
    color: #fff;
```

```
    text-align: center;
    padding: 1em;
}

main {
    display: flex;
    justify-content: space-around;
    align-items: center;
    height: 70vh;
}

#temperature,
#humidity {
    text-align: center;
    padding: 1em;
    border-radius: 8px;
    background-color: #fff;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
```

3. JavaScript for Real-time Updates (app.js):

```
 function fetchData() {
    // Assuming your API endpoint returns data in JSON format
    fetch('https://your-api-url/temperature-humidity')
        .then(response => response.json())
        .then(data => {
            const temperatureElement = document.getElementById('temperature');
```

```
        const humidityElement = document.getElementById('humidity');

        temperatureElement.textContent = `Temperature: ${data.temperature} °C`;
        humidityElement.textContent = `Humidity: ${data.humidity}%`;
    })
    .catch(error => console.error('Error fetching data:', error));
}

setInterval(fetchData, 5000);
fetchData(); // Initial fetch when the page loads
```

Here is a sample output ,