This example will be a simplified web application for real-time temperature and humidity data display. Below are the key steps involved:

1. Set Up the Development Environment:

   - Ensure you have a text editor and a web server installed. You can use simple Python's HTTP server for development.

2. Create the HTML Structure:

   - Start by creating an HTML file to structure the web page. You'll have elements for data display and user interaction.

```
html
<!DOCTYPE html>
<html>
<head>
    <title>Environmental Monitoring Platform</title>
</head>
<body>
    <div id="data-container">
        <h1>Real-time Environmental Data</h1>
        <p>Temperature: <span id="temperature">--</span> °C</p>
        <p>Humidity: <span id="humidity">--</span> %</p>
    </div>
</body>
</html>
```

3. Add CSS Styling:

   - Create a separate CSS file to style the web page.

```
css
/* style.css */
```

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

#data-container {
    border: 1px solid #ccc;
    padding: 20px;
    margin: 20px auto;
    max-width: 300px;
}
```

4. Implement Real-Time Data Display with JavaScript:

   - Use JavaScript to fetch and update real-time data from your IoT devices. In this example, we'll simulate real-time data updates.

```
html
<!-- Add this within the HTML file, below the </body> tag -->
<script src="script.js"></script>
```

```
javascript
// script.js
const temperatureElement = document.getElementById("temperature");
const humidityElement = document.getElementById("humidity");

function updateData() {
    // Simulate fetching real-time data from your IoT devices
    const temperature = getRandomValue(15, 30);
    const humidity = getRandomValue(40, 80);
    temperatureElement.textContent = temperature.toFixed(2);
    humidityElement.textContent = humidity.toFixed(2);

    setTimeout(updateData, 2000); // Update data every 2 seconds
}

function getRandomValue(min, max) {
    return Math.random() * (max - min) + min;
}

updateData(); // Start the real-time data update process
```

5. Serve the Web Application:

  - Start a web server (e.g., using Python or Node.js) to serve the HTML, CSS, and JavaScript files.

```
bash
python -m http.server
```

6. Access the Platform:

  - Open a web browser and navigate to http://localhost:8000 (or the address provided by your web server).

You now have a simple web application that displays real-time temperature and humidity data. In a real-world scenario, you would replace the simulated data with data fetched from your IoT devices. Additionally, consider using a framework like React, Vue, or Angular for more complex applications and improved user experience. Don't forget to secure the data transmission and user authentication for a production-ready platform.