Creating a real-time air quality monitoring platform using web development technologies (HTML, CSS, and JavaScript) requires both a frontend for displaying the data and a backend for receiving and processing data from IoT devices. A basic example is provided below of how to create such a platform:

**Frontend (HTML, CSS, and JavaScript):**

1. Create the HTML structure for the user interface.
2. Design the layout and elements to display air quality data.
3. Use JavaScript to fetch and display real-time data from the backend.
4. Update the UI periodically to reflect the latest air quality information.
5. Style the platform using CSS for a user-friendly interface.

Here's a simple example of a frontend using HTML, JavaScript, and CSS:

<!DOCTYPE html>

<html>

<head>

<title>Air Quality Monitor</title>

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

</head>

<body>

<h1>Air Quality Monitor</h1>

<div id="airQualityData">

<p>Air Quality Index: <span id="aqiValue">Loading...</span></p>

<p>Location: <span id="location">Loading...</span></p>

<p>Last Updated: <span id="lastUpdated">Loading...</span></p>

</div>

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

</body>

</html>

**Backend (Server for Receiving Data):**

1. Set up a backend server to receive and process data from IoT devices.
2. Use a database (e.g., MongoDB or MySQL) to store historical air quality data.
3. Define API endpoints to receive and provide air quality data.
4. Store incoming data from IoT devices in the database.
5. Implement a mechanism to send real-time data to the frontend.

Here's a simplified example of a Node.js backend:

const express = require('express');

const app = express();

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/air\_quality\_db', { useNewUrlParser: true, useUnifiedTopology: true });

const AirQualityData = mongoose.model('AirQualityData', {

aqi: Number,

location: String,

timestamp: Date

});

app.use(bodyParser.json());

// API endpoint to receive data from IoT devices

app.post('/api/air-quality-data', (req, res) => {

const { aqi, location } = req.body;

const airQualityData = new AirQualityData({

aqi,

location,

timestamp: new Date()

});

airQualityData.save()

.then(() => {

// Send the newly received data to connected clients

// You can use WebSockets or another real-time communication method here

})

.catch(err => {

console.error(err);

res.status(500).send('Error saving data');

});

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

The style.css file used in Frontend:

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

}

h1 {

color: #333;

}

#airQualityData {

border: 1px solid #999;

padding: 10px;

background-color: #fff;

}

#aqiValue {

font-size: 24px;

color: #0074cc;

}

/\* Add more CSS rules for your specific styling needs \*/