Air Quality Monitoring using IoT

Project Developments:

To create a data-sharing platform that displays real-time air quality data, you can use HTML, CSS, and JavaScript for the frontend, and you'll need to set up a backend to receive and process data from the IoT devices. Below is a basic example of how you might structure your project:

1. HTML Structure:

```
    html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Air Quality Data Platform</title>
    <!-- Add any necessary CSS and JavaScript files here -->
</head>
<body>
    <div id="dataDisplay">
        <!-- Data will be displayed here -->
    </div>
```

```
</body>
</html>
```

2. CSS Styling (you can add more styles as needed):

```
    css
#dataDisplay {
    width: 80%;
    margin: 0 auto;
    padding: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
    margin-top: 50px;
}
```

3. JavaScript Logic:

```
    javascript
// Assuming you are receiving data in the format { parameter: 'value' }
// You'll need to replace this with actual data from your IoT devices.
const airQualityData = {
    temperature: '25°C',
    humidity: '40%',
```

```
    pollutionLevel: 'Low',
    airPressure: '1012 hPa'
};
// Function to display the data on the webpage
function displayData() {
    const dataDisplay = document.getElementById('dataDisplay');
    for (let key in airQualityData) {
        if (airQualityData.hasOwnProperty(key)) {
            const para = document.createElement('p');
            para.textContent = `${key}: ${airQualityData[key]}`;
            dataDisplay.appendChild(para);
        }
    }
}
// Call the function to display the data when the page loads
window.onload = displayData;
```

This is a basic setup to display the air quality data on a webpage. However, to receive data from IoT devices, you will need a backend system that can handle data transmission. You might want to use a server-side language like Node.js, Python, or any other suitable backend technology to set up a server to receive and process data from IoT devices. You will also need to set up a database to store the received data if you want to analyze or visualize historical data.

Additionally, for real-time data, you may need to implement web sockets or another real-time communication method. You might consider using libraries like Socket.IO or other similar technologies to enable real-time data transfer between the IoT devices and the platform.

Make sure to consider security aspects, as handling sensitive environmental data requires robust security measures.

To create a data-sharing platform that can receive and display air quality data sent by IoT devices, you will need to implement a backend server to handle the data transmission. Below is a simplified example using Node.js and Express for the backend.

Make sure you have Node.js installed on your machine before proceeding. Here's a simple example of how you can set up the backend:

1. Create a new directory for your project and navigate into it.

2. Initialize a new Node.js project and install necessary dependencies.

npm init -y

npm install express

3. Create an `index.js` file and add the following code:
javascript

```
const express = require('express');

const app = express();

const PORT = 3000;

// Array to store air quality data

let airQualityData = [];

// Middleware to parse JSON bodies
```

```
app.use(express.json());

// Route to receive data from IoT devices
app.post('/data', (req, res) => {
    const data = req.body;
    airQualityData.push(data);
    console.log('Received data:', data);
    res.send('Data received successfully');
});

// Route to display the stored data
app.get('/data', (req, res) => {
    res.json(airQualityData);
});

// Start the server
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

4. Run the server by executing `node index.js` in the terminal.

With this basic setup, your server is ready to receive and store the data sent by the IoT devices. You can send POST requests to the `/data` endpoint with the air quality data, and it will be stored in the `airQualityData` array. The GET request to the same endpoint will return all the stored data.

On the frontend, you can use JavaScript to make requests to the server and display the data on the platform. You can use the previously provided HTML and CSS code and update the JavaScript to fetch data from the server.

Make sure to further enhance the backend to handle data securely and efficiently, considering authentication, validation, and error handling. Additionally, you may need to set up a database to persist the data for later analysis and visualization.