IOT Phase 4

Objective:

The objective of this technology project is to further advance the platform in accordance with project requirements. We will utilize web development technologies as necessary. Following the completion of the relevant tasks, a comprehensive document will be generated and shared for assessment.

Code:

//Creating a simple user interface for the restroom information platform involves designing a web-based dashboard.

HTML (index.html):

```
html
Copy code
<!DOCTYPE html>
<html>
<head>
    <title>Restroom Information Platform</title>
    <style>
        /* Basic styling for demonstration */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #333;
            color: #fff;
            text-align: center;
            padding: 10px 0;
        }
        .container {
```

```
            width: 80%;
            margin: 0 auto;
            padding: 20px 0;
        }
        .data-section {
            display: flex;
            justify-content: space-around;
        }
        .data-card {
            width: 45%;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Restroom Information Platform</h1>
    </header>
    <div class="container">
        <div class="data-section">
            <div class="data-card">
                <h2>Occupancy</h2>
                <p><strong>Status:</strong> <span
id="occupancyStatus">Fetching...</span></p>
            </div>
            <div class="data-card">
                <h2>Cleanliness</h2>
                <p><strong>Score:</strong> <span
id="cleanlinessScore">Fetching...</span></p>
            </div>
        </div>
    </div>

    <script>
```

```
    function updateData() {
        // Make an AJAX GET request to the Python server
        fetch('/get_data')
            .then(response => response.json())
            .then(data => {
                const { occupancy, cleanliness } = data;
                document.getElementById("occupancyStatus").innerText = (occupancy
=== 1) ? "Occupied" : "Vacant";
                document.getElementById("cleanlinessScore").innerText = cleanliness;
            })
            .catch(error => {
                console.error('Error fetching data:', error);
            });
    }

    setInterval(updateData, 5000);
</script>
</body>
</html>
```

Explanation:

This example creates a basic layout with two sections displaying the occupancy status and cleanliness score of the restroom. The data is updated every 5 seconds using simulated data for demonstration purposes.

For a production system, you would need to replace the simulated data with actual data retrieved from the IoT devices or a backend system. Additionally, you can enhance the UI further by adding more features, charts, or a more sophisticated design to suit your project's requirements.

Let's extend the Python script to generate a simple Flask-based web application. This application will act as the UI for the restroom information platform and will also act as the endpoint to receive data from the IoT sensors.

This example uses Flask, a Python web framework.

Code:

```
from flask import Flask, render_template, request

app = Flask(__name__)

# Data placeholders
occupancy_status = "Fetching..."
cleanliness_score = "Fetching..."

@app.route('/')
def index():
    return render_template('index.html', occupancy=occupancy_status,
cleanliness=cleanliness_score)

@app.route('/update_data', methods=['POST'])
def update_data():
    global occupancy_status, cleanliness_score
    occupancy_status = request.json.get('occupancy')
    cleanliness_score = request.json.get('cleanliness')
    return 'Data received', 200

if __name__ == '__main__':
    app.run(debug=True)
```

Explanation:

This script creates a basic Flask web application with two routes:

'/' route renders an HTML page (index.html) to display the occupancy and cleanliness data. It uses placeholders to display the current data (initially set to "Fetching...").

'/update_data' is a route that receives data updates from the IoT sensors via a POST request. It updates the occupancy_status and cleanliness_score variables with the new data received.

HTML Template for the data:

Create a new folder called templates in the same directory as your Python script, and within this folder, create a new HTML file named index.html.

index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Restroom Information Platform</title>
    <style>
        /* Basic styling for demonstration */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #333;
            color: #fff;
            text-align: center;
            padding: 10px 0;
        }
        .container {
            width: 80%;
            margin: 0 auto;
            padding: 20px 0;
        }
```

```
        .data-section {
            display: flex;
            justify-content: space-around;
        }
        .data-card {
            width: 45%;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
            margin-bottom: 20px;
        }
    </style>
</head>
<body>
    <header>
        <h1>Restroom Information Platform</h1>
    </header>
    <div class="container">
        <div class="data-section">
            <div class="data-card">
                <h2>Occupancy</h2>
                <p><strong>Status:</strong> <span id="occupancyStatus">{{
occupancy }}</span></p>
            </div>
            <div class="data-card">
                <h2>Cleanliness</h2>
                <p><strong>Score:</strong> <span id="cleanlinessScore">{{
cleanliness }}</span></p>
            </div>
        </div>
    </div>
</body>
</html>
```

This template uses Jinja2 templating to display the current values of occupancy and cleanliness. The Python script and HTML template together create a simple web application that displays the restroom data.

The Web application can be accessed at http://127.0.0.1:5000/. This will display the current status of occupancy and cleanliness as "Fetching..." initially, and it will update when new data is received via the /update_data route.

To simulate sending data to the web app, the Python script is used to send data to the Flask app's /update_data endpoint.