**SMART PUBLIC RESTROOM USING IOT**

**Building the smart public restroom by using IOT by developing the data sharing platform**

**1.Sensors and IoT Devices:** Install various sensors in the restroom to collect data. For instance, occupancy sensors to determine if stalls are in use, air quality sensors, and water usage sensors.

**2.Connectivity:** Ensure that these sensors are connected to the internet through a secure network. This may involve using Wi-Fi, Bluetooth, or other IoT communication protocols.

**3.Data Collection:** Collect data from the sensors in real-time. This data can include information on restroom occupancy, temperature, humidity, and more.

**4.Data Processing:** Process the collected data to extract meaningful information. For instance, you can use algorithms to identify peak restroom usage times or detect issues like water leaks.

**5.Data Sharing Platform:** Develop a platform to store and share this data. Consider using cloud services for data storage and APIs for data sharing.

**6.Maintenance and Monitoring:** Regularly monitor the system for sensor malfunctions or data inaccuracies. Implement a maintenance protocol for quick response to issues.

**Use web technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time restroom availability and cleanliness data.**

**1.HTML Structure:**

Create an HTML file to define the structure of your web page. This will include headers, navigation, and the main content area.

**2.CSS Styling:**

Use CSS to style your web page, making it visually appealing and responsive. You can create a clean and user-friendly design for displaying restroom data.

**3.Real-Time Data Integration:**

Implement JavaScript to fetch and display real-time data. You can use technologies like WebSockets or AJAX to continuously update restroom availability and cleanliness information.

**4.Display Restroom Data:**

Use HTML elements to present the restroom data, such as tables, cards, or lists. Update this data in real-time based on the information you receive from your data source.

**To create a web interface for Smart public restroom with IoT:**

**Front-end (HTML, CSS, JavaScript):**

**1. HTML:**

- Create a basic HTML structure for your platform.

- Include elements for restroom availability and cleanliness data display.

<!DOCTYPE html>

<html>

<head>

<title>Restroom Status</title>

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

</head>

<body>

<h1>Restroom Status</h1>

<div id="availability">Availability: Loading...</div>

<div id="cleanliness">Cleanliness: Loading...</div>

</body>

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

</html>

**2. CSS:**

- Style the HTML elements to make your platform look presentable.

/\* styles.css \*/

body {

font-family: Arial, sans-serif;

text-align: center;

background-color: #f5f5f5;

}

h1 {

color: #333;

}

#availability, #cleanliness {

margin: 20px;

font-size: 18px;

}

**3. JavaScript:**

- Use JavaScript to fetch and display real-time data (you'll need a server and API for this).

// script.js

document.addEventListener("DOMContentLoaded", () => {

// Fetch and display real-time data from your server

fetch("your\_api\_endpoint")

.then(response => response.json())

.then(data => {

document.getElementById("availability").textContent = `Availability: ${data.availability}`;

document.getElementById("cleanliness").textContent = `Cleanliness: ${data.cleanliness}`;

})

.catch(error => console.error("Error fetching data:", error));

});

**Back-end (Server and Database):**

For a simple version, you can skip a database and serve mock data directly from your server. You can use Node.js as the server technology:

**4. Node.js Server:**

- Create a basic Node.js server that serves your HTML, CSS, and JavaScript files.

// server.js

const express = require('express');

const app = express();

const port = 3000;

app.use(express.static('public'));

app.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

**5. API Endpoint:**

- Create a simple API endpoint that provides mock data.

// server.js (continuation)

app.get('/api/restroom-data', (req, res) => {

const data = {

availability: 'Available',

cleanliness: 'Clean'

};

res.json(data);

});

To run this simple platform, you would need Node.js installed. Run your server script (server.js), and access the platform in your web browser at `http://localhost:3000`.

Remember that this is a basic example, and in a real-world scenario, you'd connect to a more robust back-end system for retrieving real-time data from IoT devices or sensors in restrooms.