**Smart Public Restrooms system**

**Step 1: Sensor Selection and Deployment**

***1. Identify Required Sensors:*** Determine the type of sensors you need for your Smart Public Restrooms system. In this case, you may want to deploy occupancy sensors and cleanliness sensors. Occupancy sensors can detect if a restroom stall is occupied, while cleanliness sensors can assess the cleanliness of the restroom.

***2. Select Appropriate Hardware:*** Choose the hardware for your IoT sensors. Depending on your budget and requirements, you can use off-the-shelf IoT development boards like Raspberry Pi, Arduino, or specialized IoT modules with built-in sensors.

***3. Sensor Deployment:*** Install and position the sensors in the public restrooms. For occupancy sensors, consider placing them near the entrance of each stall. Cleanliness sensors can be placed strategically to assess restroom conditions.

**Step 2: Sensor Data Collection**

***1. Connect Sensors to the IoT Platform:*** Interface the sensors with the IoT development board or module. You may need to write or configure drivers to read data from the sensors.

***2. Data Processing:*** Develop a Python script to read sensor data and process it. For example, occupancy sensors can detect movement and cleanliness sensors can provide cleanliness scores. Your script should gather this data and prepare it for transmission.

***3. Real-Time Data Collection:*** Continuously collect data from the sensors. In Python, you can use libraries like GPIO for sensor input or I2C/SPI communication for cleanliness sensors.

**Step 3: Data Transmission**

***1. Choose a Communication Protocol***: Select a communication protocol to send data to the restroom information platform. Common options include Wi-Fi, MQTT, HTTP, or LoRa for long-range communication.

***2. Develop the Python Script for Data Transmission:*** Write a Python script to transmit the collected data to the central platform. You can use libraries like Requests for HTTP communication or MQTT for publish-subscribe messaging.

***3. Data Encryption and Security:*** Ensure data security by encrypting the data before transmission. This is crucial, especially for cleanliness data, as it may be sensitive.

**Step 4: Restroom Information Platform**

***1. Set Up the Platform:*** Create a central platform or server that will receive and process data from the IoT sensors. This platform can be cloud-based or hosted locally.

***2. Data Reception and Processing:*** Develop server-side software to receive incoming data, validate it, and store it in a database.

***3. Data Analysis and Visualization:*** Implement data analysis and visualization tools to make sense of the collected data. You can use Python libraries like Pandas and Matplotlib for this purpose.

**Step 5: Monitoring and Notifications**

***1. Real-Time Monitoring:*** Set up real-time monitoring of restroom occupancy and cleanliness data. This can include dashboards for administrators and maintenance staff.

***2. Notifications:*** Implement notifications or alerts for situations like restroom overcrowding or low cleanliness scores. Notifications can be sent via email, SMS, or push notifications.

**Step 6: Testing and Optimization**

***1. Testing:*** Thoroughly test your system in real-world conditions to ensure accurate data collection and transmission.

***2. Optimization:*** Continuously optimize the system based on feedback and usage data.

**Here are the steps to create this Python script:**

***1. Set up the Hardware:***

Connect occupancy and cleanliness sensors to your IoT device (e.g., Raspberry Pi or Arduino). Make sure you have the required libraries for these sensors.

Ensure your IoT device has access to the internet, either through Wi-Fi or another network connection.

***2. Install Required Libraries:***

Depending on the sensor types you're using, you might need to install specific Python libraries for sensor data acquisition. For example, if you're using a PIR sensor for occupancy, you can use the "RPi.GPIO" library on a Raspberry Pi.

***3. Create Python Script:***

Write a Python script that reads data from your sensors at regular intervals. For example, you can read the occupancy sensor's state and the cleanliness sensor's data.

**CODING:-**

import time

import requests # For making HTTP requests to your platform

# Define the URLs for your platform's APIs

occupancy\_url = "https://your-platform.com/occupancy"

cleanliness\_url = "https://your-platform.com/cleanliness"

# Read data from sensors (replace these with actual sensor readings)

occupancy\_data = 1 # 1 for occupied, 0 for unoccupied

cleanliness\_data = 0.8 # A cleanliness score from 0 to 1

while True:

# Read sensor data

occupancy\_data = read\_occupancy\_sensor()

cleanliness\_data = read\_cleanliness\_sensor()

# Send data to the platform

try:

# Send occupancy data

response = requests.post(occupancy\_url, json={"occupancy": occupancy\_data})

if response.status\_code == 200:

print("Occupancy data sent successfully")

else:

print(f"Failed to send occupancy data. Status code: {response.status\_code}")

# Send cleanliness data

response = requests.post(cleanliness\_url, json={"cleanliness": cleanliness\_data})

if response.status\_code == 200:

print("Cleanliness data sent successfully")

else:

print(f"Failed to send cleanliness data. Status code: {response.status\_code}")

except Exception as e:

print(f"Error sending data: {str(e)}")

# Wait for a defined interval before sending data again

time.sleep(60) # Send data every minute

4. Replace the placeholder URLs with the actual endpoints provided by your restroom information platform.

***5. Set up Exception Handling:***

Implement error handling in case there are issues with data transmission.

***6. Run the Script:***

Run the Python script on your IoT device. It will continuously read sensor data and send it to your restroom information platform.

***7. Implement Data Processing on the Platform:***

On your restroom information platform, create endpoints to receive and process the data sent by the IoT device. Process the data as needed and make it available to users through a user interface.