# SMART PUBLIC RESTROOM

## Objective:

**1. User Experience Enhancement**

- **Real-time Information:** Provide users with real-time information regarding restroom availability, cleanliness status, and essential amenities.
- **Accessibility and Navigation:** Assist users in finding nearby smart restrooms and navigate to the nearest available one.
- **User Feedback and Ratings:** Enable users to give feedback and ratings on their restroom experience to maintain and improve service quality.

**2. Resource Management**

- **Water and Energy Efficiency:** Monitor and manage water and energy consumption to reduce waste and promote sustainability.
- **Supply Management:** Track inventory levels of essential supplies (toilet paper, soap, etc.) and automate orders for replenishment.
- **Occupancy Optimization:** Manage occupancy and usage patterns to improve operational efficiency.

**3. Maintenance and Hygiene**

- **Predictive Maintenance:** Implement predictive maintenance strategies to identify and address potential issues before they become problematic.
- **Hygiene Maintenance:** Ensure a clean and hygienic environment by monitoring cleanliness levels and triggering alerts for cleaning when needed.

**4. Data-driven Insights**

- **Usage Patterns and Analytics:** Collect and analyze data to understand usage patterns, peak hours, and trends to optimize operations.
- **Decision Support:** Use collected data to make informed decisions regarding maintenance, resource allocation, and service improvements.

**5. Sustainability and Cost Efficiency**

- **Reduced Environmental Impact:** Employ technology to minimize waste and energy consumption, contributing to a more sustainable environment.
- **Cost Savings:** Optimize resource usage to reduce operational costs and improve the overall financial efficiency of the facility.

**6. Smart Technology Integration**

- **IoT Sensor Implementation:** Install various sensors to gather data on occupancy, water usage, supplies, and cleanliness.
- **Mobile App Accessibility:** Develop a user-friendly mobile application for seamless access to restroom information and services.
- **Raspberry Pi Integration:** Utilize Raspberry Pi or similar devices as a central hub for processing data from IoT sensors and managing communication with the mobile app and backend systems.

# IOT sensor setup:

**1. Occupancy Sensors:**

- **Purpose:** To determine the presence of individuals within the restroom.
- **Sensor Type:** Motion sensors (infrared or ultrasonic) or door sensors.
- **Functionality:** Detects human presence and measures the occupancy status of the restroom to indicate availability.

**2. Water Usage Sensors:**

- **Purpose:** Monitor water consumption within the restroom.
- **Sensor Type:** Water flow sensors or water usage meters.
- **Functionality:** Measures the flow of water to track usage, helping in understanding consumption patterns and potential leaks.

**3. Supply Level Sensors:**

- **Purpose:** Monitor the availability of essential supplies.
- **Sensor Type:** Sensors integrated into toilet paper dispensers, soap dispensers, or other supply containers.
- **Functionality:** Tracks the level of supplies, generating alerts for restocking when levels are low, ensuring a continuous supply of necessities.

**4. Air Quality and Environmental Sensors:**

- **Purpose:** Maintain a comfortable and healthy environment.
- **Sensor Type:** Temperature, humidity, and air quality sensors.
- **Functionality:** Monitors temperature, humidity, and air quality to ensure a pleasant and healthy restroom environment for users.

**5. Waste Bin Sensors:**

- **Purpose:** Manage waste disposal efficiently.
- **Sensor Type:** Ultrasonic or weight sensors integrated into waste bins.
- **Functionality:** Indicates waste bin levels, triggering alerts for timely disposal and maintenance.

**6. Vandalism or Damage Sensors:**

- **Purpose:** Detect and prevent vandalism or damage.
- **Sensor Type:** Vandalism detection sensors or impact sensors.
- **Functionality:** Alerts facility management in case of vandalism, damage, or impact, ensuring immediate attention.

**Integration and Communication:**

- These sensors are integrated and connected to a central hub, which could be a microcontroller (e.g., Raspberry Pi) or a central server. They communicate via Wi-Fi, Bluetooth, or other network protocols to transmit data and receive instructions.

**Data Processing and Management:**

- The data collected by these sensors is processed and analyzed to derive meaningful insights, which can be used for decision-making, predictive maintenance, and resource optimization.

## Mobile app download:

**1. Real-Time Restroom Information:**

**Objective:** Display information regarding restroom availability, cleanliness status, and essential amenities.

**Coding Example (using Flutter and Dart):**

```
import 'package:flutter/material.dart';
class RestroomAvailability extends StatelessWidget {
  String availabilityStatus = "Available"; // Assume this data is retrieved from an API
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Restroom Availability'),
      ),
      body: Center(
        child: Text(
          'Restroom Status: $availabilityStatus',
          style: TextStyle(fontSize: 20),
        ),
      ),
    );
  }
}
```

**2. Navigation to Nearby Smart Restrooms:**

**Objective:** Assist users in finding nearby smart restrooms and navigate to the nearest available one.

**Coding Example (using Geolocation in Flutter):**

```
import 'package:geolocator/geolocator.dart';
Future<Position> getCurrentLocation() async {
  return await Geolocator.getCurrentPosition(desiredAccuracy: LocationAccuracy.high);
}
// Use the obtained location data for navigation within the app
```

**3. User Feedback and Ratings:**

**Objective:** Enable users to provide feedback and ratings on their restroom experience.

**Coding Example (using Flutter):**

```
// Implementation of a basic feedback form
class RestroomFeedbackForm extends StatefulWidget {
  @override
  _RestroomFeedbackFormState createState() => _RestroomFeedbackFormState();
}
class _RestroomFeedbackFormState extends State<RestroomFeedbackForm> {
  final TextEditingController _feedbackController = TextEditingController();
  int rating = 0;
```

```
    @override
  Widget build(BuildContext context) {
   return Scaffold(
    appBar: AppBar(
     title: Text('Restroom Feedback'),
 ),
    body: Column(
     children: [
      TextField(
       controller: _feedbackController,
       decoration: InputDecoration(labelText: 'Provide feedback'),
      ),
      // Implement rating stars or sliders for user ratings
      // ...
      ElevatedButton(
       onPressed: () {
        // Submit feedback and rating to the server/database
        submitFeedback(_feedbackController.text, rating);
       },
       child: Text('Submit'),
      ),
     ],
    ),
   );
  }
}
```

**4. Communication with Backend/API:**

**Objective:** Communicate with the backend to fetch real-time data and submit user feedback.

**Coding Example (using HTTP requests in Flutter):**

```
import 'package:http/http.dart' as http;
Future<String> fetchRestroomAvailability() async {
  final response = await http.get(Uri.parse('https://yourapi.com/restroom/status'));
  if (response.statusCode == 200) {
    return response.body;
  } else {
    throw Exception('Failed to fetch data');
  }
}
Future<void> submitFeedback(String feedback, int rating) async {
  // Implement code to send user feedback and rating to the server
  // ...
}
```

## Raspberry Pi integration and code implementation:

**1. Data Processing and Sensor Integration:**

**Objective:** Collect and process data from various sensors within the restroom environment.

**Coding Example (using Python on Raspberry Pi):**

```
from gpiozero import MotionSensor, DigitalInputDevice
# Example: Motion sensor for occupancy detection
pir = MotionSensor(4)  # GPIO pin 4 for motion sensor
# Function to check restroom occupancy
def check_occupancy():
    if pir.motion_detected:
        return "Restroom occupied"
    else:
        return "Restroom available"
# Example: Digital input for a supply level sensor
supply_sensor = DigitalInputDevice(17)  # Example GPIO pin 17
# Function to check supply level
def check_supply_level():
    if supply_sensor.value == 0:
        return "Low supply"
    else:
        return "Adequate supply"
```

**2. Communication with Backend or API:**

**Objective:** Transmit data from the Raspberry Pi to the backend server or receive instructions from the server.

**Coding Example (Python on Raspberry Pi - using Requests library):**

```
import requests
# Example function to send data to the backend
def send_data_to_backend(data):
    url = 'https://yourapi.com/data'
    payload = {'data': data}  # Adjust payload based on actual data format
    headers = {'Content-Type': 'application/json'}
    response = requests.post(url, json=payload, headers=headers)

    if response.status_code == 200:
        print("Data sent successfully")
    else:
        print("Failed to send data")
```

**3. Control Mechanisms (e.g., Actuators or Displays):**

**Objective:** Control various devices such as displays or actuators based on sensor data or commands.

**Coding Example (using Python GPIO on Raspberry Pi):**

```
from gpiozero import LED
# Example: LED to indicate restroom status
led = LED(18)  # Example GPIO pin 18
# Function to control the LED based on occupancy
def control_led(occupancy_status):
    if occupancy_status == "Restroom occupied":
        led.on()
    else:
        led.off()
```

**4. Data Analysis and Decision Making:**

**Objective:** Analyze data and trigger actions or alerts based on specific conditions.

**Coding Example (Python for Data Analysis):**

```
# Example: Analyzing water flow data
def analyze_water_flow(water_flow):
    if water_flow > 100:  # Example threshold for excessive flow
        return "Possible leak detected"
    else:
        return "Water flow within normal range"
```

## Restroom information platform:

**1. User-Facing Application:**

- **Objective:** Provide users with real-time information and services.
- **Features:**
    - **Restroom Availability:** Displays the current occupancy status (occupied or available) of the restroom.
    - **Cleanliness Ratings:** Allows users to provide ratings or feedback on the cleanliness of the restroom.
    - **Location and Navigation:** Helps users find nearby smart restrooms and navigate to the nearest available one.

**2. Backend Management System:**

- **Objective:** Collect and manage data from sensors, and handle user interactions.
- **Features:**
    - **Data Collection:** Gathers information from IoT sensors within the restroom (occupancy, water usage, supply levels, etc.).
    - **User Feedback Management:** Stores and processes user feedback and ratings.
    - **APIs for Mobile App:** Provides endpoints for the mobile app to fetch real-time data.

**3. IoT Sensor Integration:**

- **Objective:** Integrates with various sensors to gather data.
- **Features:**
    - **Occupancy Sensors:** Monitor restroom occupancy and availability.
    - **Water Usage Sensors:** Track water consumption for efficiency and leakage detection.
    - **Supply Level Sensors:** Manage inventory levels for supplies.
    - **Environmental Sensors:** Monitor temperature, humidity, and air quality.

**4. Data Analysis and Decision Support:**

- **Objective:** Analyze collected data for insights and decision-making.
- **Features:**
    - **Usage Patterns Analysis:** Identifies peak usage times and patterns.
    - **Predictive Maintenance:** Forecasts maintenance requirements based on sensor data.
    - **Resource Optimization:** Helps in managing resources effectively.

**5. Notification and Alerting System:**

- **Objective:** Alerts users and maintenance staff about relevant information.
- **Features:**
    - **Restroom Availability Alerts:** Notifies users about the availability of restrooms nearby.
    - **Maintenance Alerts:** Alerts staff about cleanliness, supply levels, or technical issues.

**6. Management Dashboard:**

- **Objective:** Provides administrators with insights and controls for maintenance and operations.
- **Features:**
    - **Real-time Data Visualization:** Graphical representation of restroom usage and status.
    - **Control Mechanisms:** Allows remote control or adjustment of certain systems if necessary.
    - **Reporting and Analytics:** Generates reports based on data collected for analysis and decision-making.

## Mobille app interface:

**1. Restroom Availability Status:**

- **Display:** A prominent section showing the current availability status of the restroom (occupied or available).
- **Icon/Color Codes:** Utilizes visual indicators like colors or icons for quick recognition of availability.

**2. Nearby Restroom Locator:**

- **Map Integration:** Displays a map with nearby smart restrooms and their availability status.

- **GPS Integration:** Allows users to locate the nearest available restroom based on their current position.

**3. Cleanliness and Ratings:**

- **User Feedback Section:** Enables users to provide ratings and feedback on the cleanliness and overall experience.
- **Review Section:** Shows user reviews or comments related to the restroom's condition.

**4. Essential Supplies Information:**

- **Supply Status:** Indicates the availability of essential supplies like toilet paper, soap, and hand sanitizer.
- **Alerts for Low Supplies:** Provides alerts or notifications when supplies are running low.

**5. Notifications and Alerts:**

- **Restroom Status Alerts:** Notifies users about changes in the availability or cleanliness status of nearby restrooms.
- **Maintenance Alerts:** Informs users about scheduled maintenance or cleaning activities.

**6. User Profile and Preferences:**

- **Personalized Experience:** Allows users to set preferences, view their history, and save favorite/rest frequently used locations.
- **Profile Management:** Provides an interface for users to manage their profiles and settings.

**7. Navigation and UI Components:**

- **Intuitive Design:** Simple, user-friendly design for easy navigation and information retrieval.
- **Search Functionality:** Enables users to search for specific locations or amenities within the restroom.

**8. Contact and Support:**

- **Support Channels:** Offers access to support services or contact information for facility management in case of issues or queries.
- **FAQ Section:** Contains frequently asked questions about the restroom facility.

**9. Emergency and Special Needs Assistance:**

- **Emergency Call/Assistance:** Includes a button or feature for emergency help in case of urgent situations.
- **Special Needs Accessibility Information:** Provides information on accessibility features for users with special needs.

**Coding Aspects:**

- **Frontend Development:** Involves the coding for the visual elements, navigation, and user interaction using programming languages like Dart (for Flutter) or Java/Kotlin (for Android) or Swift (for iOS).
- **Backend Integration:** Establishes communication with the server or IoT devices, retrieves data, and manages user feedback and reviews.

## User experience and restroom management:

**Enhancing User Experience:**

1. **Improved Accessibility:** Real-time information about restroom availability enables users to quickly locate an available restroom, reducing wait times and frustration.
2. **Enhanced Hygiene:** Users can access cleanliness ratings and information, allowing them to make informed decisions about using the restroom, promoting a cleaner and more hygienic experience.
3. **Efficient Resource Utilization:** Users are aware of essential supplies' availability, ensuring they don't encounter empty dispensers for items like toilet paper or soap.
4. **Personalized Experience:** Customizable settings within the app, such as preferred restroom locations or special needs assistance, contribute to a more personalized experience for users.

**Optimizing Restroom Management:**

1. **Predictive Maintenance:** Real-time data collected from sensors enables predictive maintenance. For instance, alerts on low supplies or potential restroom maintenance requirements are sent to management, enabling proactive action.
2. **Resource Efficiency:** Monitoring water and energy usage allows for the identification of inefficiencies or leaks, leading to better resource management and reduced operational costs.
3. **Data-Driven Decision Making:** Analysis of usage patterns, peak hours, and user feedback provides insights for better decision-making, allowing for more effective resource allocation and operational improvements.
4. **Timely Maintenance:** Immediate alerts on issues such as restroom overcrowding, cleanliness issues, or low supply levels prompt quick action, maintaining a positive user experience and avoiding potential problems.
5. **Optimized Staff Allocation:** Efficient monitoring of restroom usage patterns allows management to allocate staff more effectively, especially during peak hours, ensuring a better overall user experience.

**Overall Benefits:**

- **User Satisfaction:** Quick and reliable access to real-time information contributes to improved user satisfaction and positive reviews.
- **Cost Savings:** Efficient resource usage and proactive maintenance strategies can lead to reduced operational costs over time.
- **Sustainability:** Better resource management contributes to a more sustainable environment by reducing waste and energy consumption.

## Hardware Setup:

**Occupancy Sensor Setup:**

```
# Code for occupancy sensor using a PIR sensor with Raspberry Pi
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt
import time
```

```
sensor_pin = 17
mqtt_broker = "broker_address"
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN)
def on_message(client, userdata, message):
   print("Message received: " + str(message.payload.decode("utf-8")))
client = mqtt.Client("OccupancySensor")
client.connect(mqtt_broker)
client.on_message = on_message
while True:
   if GPIO.input(sensor_pin):
      client.publish("occupancy", "Occupied")
   else:
      client.publish("occupancy", "Vacant")
   time.sleep(1)
```

**Water Flow Sensor Setup:**

```
# Code for water flow sensor with Raspberry Pi
import RPi.GPIO as GPIO
import time
import paho.mqtt.client as mqtt
flow_sensor_pin = 18
mqtt_broker = "broker_address"
GPIO.setmode(GPIO.BCM)
GPIO.setup(flow_sensor_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def on_message(client, userdata, message):
   print("Message received: " + str(message.payload.decode("utf-8")))
client = mqtt.Client("WaterFlowSensor")
client.connect(mqtt_broker)
client.on_message = on_message
pulse_count = 0
def count_pulse(channel):
   global pulse_count
   pulse_count += 1
GPIO.add_event_detect(flow_sensor_pin, GPIO.FALLING, callback=count_pulse)
while True:
   try:
      pulse_count = 0
      time.sleep(5)
      flow_rate = pulse_count / 7.5  # Replace 7.5 with your sensor's pulses per liter value
      client.publish("water_flow", str(flow_rate))
   except KeyboardInterrupt:
      GPIO.cleanup()
```

**Software Development:**

**Transmission Platform:**

```
# MQTT broker setup (e.g., using Mosquitto)
# Install Mosquitto: sudo apt-get install mosquitto mosquitto-clients
# Start Mosquitto: mosquitto –v
```

**Cloud Integration:**

For cloud integration, you'd typically need a cloud service provider (e.g., AWS IoT Core, Google Cloud IoT, or Azure IoT Hub). The following code is an example of how to publish data to an MQTT broker (in this case, the cloud service).

```
import paho.mqtt.client as mqtt
client = mqtt.Client("DataPublisher")
client.connect("cloud_broker_address")
# Assuming the data is already collected and stored in variables (occupancy_data,
water_flow_data)
client.publish("restroom/occupancy", occupancy_data)
client.publish("restroom/water_flow", water_flow_data)
```

**Integration using Python:**

1. **Data Processing:**

Once the data is received from the MQTT broker in the cloud, you might process it using Python for analytics or other purposes.

2. **Dashboard Creation:**

Here's an example of using Dash to create a simple dashboard to visualize the sensor data:

```
import dash
from dash import dcc, html
import paho.mqtt.client as mqtt
app = dash.Dash(__name__)
# Function to update sensor data
def update_sensor_data():
   # Implement MQTT subscription and data update logic here
   pass
app.layout = html.Div([
   dcc.Interval(id='interval-component', interval=1000, n_intervals=0),
   html.Div(id='output-data')
])
@app.callback(dash.dependencies.Output('output-data', 'children'),
          [dash.dependencies.Input('interval-component', 'n_intervals')])
def update_output(n):
   update_sensor_data()
return f"Occupancy: {occupancy_data}, Water Flow: {water_flow_data}"

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

**Raspberry Pi Data Transmission (Python script):**

This example demonstrates a Raspberry Pi sending mock sensor data via MQTT:

```
import paho.mqtt.client as mqtt
import time
import random
broker_address = "broker.example.com"  # Replace with your MQTT broker address
client = mqtt.Client("RaspberryPi")
client.connect(broker_address)
while True:
    # Simulating sensor data (occupancy and water flow)
    occupancy_data = random.randint(0, 1)
    water_flow_data = round(random.uniform(0.0, 10.0), 2)
    client.publish("restroom/occupancy", occupancy_data)
    client.publish("restroom/water_flow", water_flow_data)
    time.sleep(5)  # Sending data every 5 seconds (adjust as needed)
```

**Mobile App UI (using Kivy):**

Here's an example of a simple mobile app UI that could display sensor data. This code creates a basic Kivy app with placeholders for receiving and displaying MQTT data:

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
import paho.mqtt.client as mqtt
broker_address = "broker.example.com"  # Replace with your MQTT broker address
class RestroomApp(BoxLayout):
    def __init__(self, **kwargs):
        super(RestroomApp, self).__init__(**kwargs)
        self.orientation = 'vertical'
        self.occupancy_label = Label(text="Occupancy: --")
        self.add_widget(self.occupancy_label)
        self.water_flow_label = Label(text="Water Flow: --")
        self.add_widget(self.water_flow_label)
        self.client = mqtt.Client("MobileApp")
        self.client.on_connect = self.on_connect
        self.client.on_message = self.on_message
        self.client.connect(broker_address)
        self.client.subscribe("restroom/#")  # Subscribe to relevant topics
        def on_connect(self, client, userdata, flags, rc):
        print("Connected with result code "+str(rc))

    def on_message(self, client, userdata, msg):
        topic = msg.topic
        message = msg.payload.decode("utf-8")
```

```
        if topic == "restroom/occupancy":
            self.occupancy_label.text = f"Occupancy: {message}"
        elif topic == "restroom/water_flow":
            self.water_flow_label.text = f"Water Flow: {message}"
class RestroomMainApp(App):
    def build(self):
        return RestroomApp()
if __name__ == '__main__':
    RestroomMainApp().run()
```

Overall, the project involves a multi-tier system where IoT sensors collect data, Raspberry Pi serves as a hub for processing and transmitting data, and mobile applications provide a user-friendly interface for real-time information and interaction. The code implementation involves writing programs for sensor communication, data analysis, and app interfacing to ensure a seamless and smart restroom experience.