SMART RESTROOM

Sensors:

1. Occupancy Sensors: Install motion or infrared sensors to detect the presence of individuals in the restroom. These sensors can trigger various actions.

2. Toilet Stall Availability Sensors: Use sensors to indicate which toilet stalls are vacant or occupied.

3. Water Quality Sensors: Measure water quality, including factors like temperature, pH, or turbidity.

4. Consumable Level Sensors: Monitor the levels of essential supplies such as soap, toilet paper, or hand sanitizer.

Data Collection:

The installed sensors continuously collect data about restroom occupancy, water quality, and supply levels.

Data Transmission:

Data collected from the sensors are transmitted to a central server or cloud platform through wired or wireless communication protocols like Wi-Fi, LoRa, or Bluetooth.

Data Processing and Analytics:

The central server processes incoming data in real-time, including:

1. Identifying restroom occupancy trends to optimize cleaning schedules and maintenance.

2. Analysing water quality data to ensure hygienic conditions.

3. Managing supply levels and triggering restocking when necessary.

Control Mechanisms:

Based on the data analysis, the smart restroom system can perform various functions, such as:

1. Occupancy-Based Lighting: Adjust lighting based on occupancy to save energy.

2. Stall Availability Indicators: Display which stalls are available to users.

3. Water Quality Control: Ensure water quality by monitoring and adjusting water treatment systems.

4. Automatic Supply Restocking: Trigger orders for supplies when levels are low.

5. Occupancy-Based HVAC: Adjust heating, ventilation, and air conditioning based on occupancy.

6. Feedback and Alerts: Collect user feedback and generate alerts for maintenance or issues.

To adapt the provided code and description for a smart toilet with IoT capabilities, you can follow a similar structure, but with different sensors and actions. Here's an outline of how you can modify it:

\*\*Hardware Components:\*\*

1. Raspberry Pi or another IoT device for data processing and communication.

2. Sensors for toilet occupancy detection (e.g., motion sensors or pressure sensors).

3. Sensors for water quality monitoring.

4. Sensors for supply level monitoring (e.g., ultrasonic or capacitive sensors).

5. Control mechanisms for lighting, water quality, and supply management.

Software Components:

1. Operating System: Set up the Raspberry Pi with a suitable OS (e.g., Raspbian).

2. Python: Use Python for programming the IoT device.

Toilet Occupancy Detection:

1. Implement occupancy detection using the installed sensors.

2. Process sensor data to determine toilet availability.

Water Quality Monitoring:

1. Utilize sensors to measure water quality parameters such as temperature, pH, or turbidity.

2. Set thresholds and trigger actions based on water quality data.

Supply Level Monitoring:

1. Continuously monitor supply levels (e.g., soap, toilet paper, hand sanitizer).

2. Trigger restocking alerts or actions when supply levels are low.

Data Communication:

1. Establish a connection to a central server or cloud platform using MQTT, HTTP, or other IoT communication protocols.

2. Send data regarding toilet occupancy, water quality, and supply levels to the server.

Central Server:

1. Create a central server to collect and process data from IoT devices.

2. Use a web framework like Flask or Django to build the server application.

3. Implement logic for controlling restroom functions based on the data received.

**CODE**

import serial

import time

# Replace 'COMx' with your IoT device's serial port (e.g., COM3 on Windows, /dev/ttyUSB0 on Linux)

ser = serial.Serial('COM3', 9600, timeout=1)

time.sleep(2) # Allow time for the serial connection to establish

def set\_toilet\_actions(action):

ser.write(action.encode())

response = ser.read()

print(f"Toilet action set to {response.decode()}")

try:

while True:

set\_toilet\_actions('O') # Check occupancy

time.sleep(2)

set\_toilet\_actions('W') # Check water quality

time.sleep(2)

set\_toilet\_actions('S') # Check supply levels

time.sleep(2)

except KeyboardInterrupt:

ser.close()