Project Definition: The project aims to enhance public restroom management by installing IoT sensors to monitor occupancy and maintenance needs. The goal is to provide real-time data on restroom availability and cleanliness to the public through a platform or mobile app. This project includes defining objectives, designing the IoT sensor system, developing the restroom information platform, and integrating them using IoT technology and Python.

Design Thinking:

1. **Project Objectives**

   The project objectives for a smart public restroom with IoT sensors for occupancy and cleanliness monitoring can be defined as follows:

   1. Real-Time Occupancy Monitoring:

   Implement IoT occupancy sensors in each restroom stall to provide real-time data on stall availability.
   Objectively measure and display the occupancy status of each stall to help users quickly find vacant facilities.

   2. Cleanliness Assessment:

   Deploy IoT sensors (e.g., ammonia, H2S, turbidity) to monitor cleanliness parameters within the restroom environment.
   Continuously assess cleanliness levels and provide users with data-driven information about restroom hygiene.

   3. Maintenance Needs Detection:

   Utilize cleanliness sensors to detect maintenance needs such as overflowing sinks, toilet malfunctions, or depleted soap dispensers.
   Automatically generate maintenance requests or alerts to restroom management for timely interventions.

   4. Data Analytics and Reporting:

   Collect and analyze historical data from IoT sensors to derive insights and trends related to restroom occupancy, cleanliness, and maintenance needs.
   Generate reports and visualizations to help facility managers make informed decisions for restroom management and maintenance.

   5. User Accessibility:

   Develop a user-friendly web-based platform or mobile app that allows the public to access real-time restroom information easily.

Ensure the platform is accessible to individuals with disabilities and provides multi-language support, if applicable.
6. Cost Efficiency:
- Optimize the project's cost-effectiveness by selecting appropriate sensors and technologies that balance functionality with budget constraints.

7. User Engagement:
- Develop features to engage users, such as gamification elements or rewards for providing feedback or maintaining cleanliness.

2. **IOT Sensor Design**
   For occupancy sensor
   Hardware Components:

   Occupancy Sensor: Choose a suitable occupancy sensor based on your requirements. Common choices include:

   Passive Infrared (PIR) Sensors: Detect changes in heat signatures (movement).
   Ultrasonic Sensors: Measure distance by emitting and receiving ultrasonic waves.
   Proximity Sensors: Use infrared or ultrasonic technology to detect nearby objects.
   Microcontroller: Select a microcontroller board to interface with the occupancy sensor and transmit data. Popular options include:

   Raspberry Pi (for more complex applications)
   Arduino (for simpler deployments)
   ESP8266 or ESP32 (for low-power and Wi-Fi connectivity)
   Power Supply: Depending on the sensor and microcontroller chosen, you may need a power supply source, which could be battery-powered or connected to a power outlet.

   Communication Module: To transmit data from the sensor to the central server or gateway, you'll need a communication module. Common options include:

   Wi-Fi (ESP8266/ESP32, Raspberry Pi)
   Bluetooth (for short-range applications)
   LoRa (for long-range, low-power applications)
   Software and Firmware:

Sensor Interface: Write firmware code for the microcontroller to interface with the occupancy sensor. This code will read data from the sensor (e.g., occupancy status) and prepare it for transmission.

Data Processing: Implement logic in the microcontroller firmware to process the occupancy data, handle false positives/negatives, and make decisions about whether a restroom stall is occupied or vacant.

Communication Protocol: Choose a communication protocol for transmitting data to the central server or gateway. Common protocols include MQTT, HTTP, or a custom API. Write code to package the data and send it securely to the designated server.

Data Encryption: Implement data encryption to ensure that the occupancy data is transmitted securely over the network.

Power Management: Implement power management features to conserve energy, especially if the sensor is battery-powered. This may involve putting the microcontroller to sleep between readings or using low-power modes.

Error Handling: Develop error-handling mechanisms to deal with communication failures, sensor malfunctions, or other unexpected issues.

OTA (Over-the-Air) Updates: If feasible, enable OTA updates for the sensor firmware to ensure easy maintenance and improvements.

Mounting and Installation:

Install the occupancy sensor in each restroom stall, ensuring it is properly positioned for accurate detection.
Connect the sensor to the microcontroller and power source securely.
Ensure that the sensor and microcontroller are well-protected within the stall to prevent tampering or damage.
Testing and Calibration:

Test the sensor system thoroughly to ensure accurate occupancy detection.
Calibrate the sensor if necessary to optimize its performance in the specific restroom environment.
Data Transmission:

Set up the sensor to transmit occupancy data at regular intervals or in real-time to the central server or gateway.

For cleanliness sensor:
Hardware Components:

Ammonia Sensor: Select a suitable ammonia gas sensor capable of detecting ammonia levels in the restroom air. These sensors are typically based on electrochemical or gas-sensitive semiconductor technologies.

Hydrogen Sulfide (H2S) Sensor: Choose an H2S gas sensor capable of detecting hydrogen sulfide concentrations in the air. Similar to the ammonia sensor, these sensors are typically based on electrochemical or gas-sensitive semiconductor technologies.

Turbidity Sensor: Use a turbidity sensor to measure the cloudiness or haziness of water in restroom fixtures such as sinks, urinals, or toilet bowls. Turbidity sensors often utilize optical or light-scattering principles.

Microcontroller: Select a microcontroller or single-board computer (e.g., Raspberry Pi) to interface with the sensors, process data, and handle communication.

Power Supply: Depending on the chosen hardware, you may need a power supply source, which could be battery-powered or connected to a power outlet.

Communication Module: Incorporate a communication module (e.g., Wi-Fi or Ethernet) to transmit data from the sensors to the central server or gateway.

Software and Firmware:

Sensor Interface: Write firmware code for the microcontroller to interface with the sensors. Different types of sensors will require specific code to read and interpret data accurately.

Data Processing: Implement logic in the microcontroller firmware to process the data from each sensor, including ammonia concentration, H2S concentration, and turbidity levels.

Communication Protocol: Choose a communication protocol (e.g., MQTT or HTTP) to transmit the data securely to the central server or gateway. Develop code to package and send the data.

Data Storage and Analysis: Set up a database on the central server to store incoming data. Implement data analysis algorithms to assess cleanliness based on the sensor readings. You can define cleanliness thresholds for each parameter.

User Interface: Create a user-friendly web-based dashboard or mobile app that displays cleanliness status in real-time. Visualize the data from the sensors and provide alerts or notifications when cleanliness thresholds are exceeded.

Power Management: Implement power-saving features in the microcontroller firmware to optimize energy usage, especially if the sensors are battery-powered.

Mounting and Installation:

Install the sensors strategically within the restroom, considering airflow and accessibility for sensor placement.
Ensure that the sensors are securely attached and positioned appropriately to provide accurate cleanliness data.
Testing and Calibration:

Test the sensor system extensively to ensure accurate readings.
Calibrate the sensors as needed to account for environmental variations and ensure precision.
Data Transmission:

Configure the sensors to transmit data at regular intervals to the central server or gateway, allowing continuous monitoring of cleanliness parameters.

3. **Real-Time Transit Information Platform**
   Cloud Platform:
   Platform Selection: AWS, Azure, Google Cloud, etc.
   Data Storage: Securely store sensor data and historical records.
   Data Analytics: Implement data analysis algorithms to assess cleanliness and occupancy patterns.
   APIs: Develop APIs for data communication between IoT devices and the cloud platform.

User Interface:
Mobile App: Provides users with real-time restroom occupancy and cleanliness data.
Web Dashboard: Allows facility managers to monitor and manage restroom data, schedule maintenance, and receive alerts.

Maintenance and Alerts:
Alerting System: Configure automated email or SMS alerts when cleanliness thresholds aren't met or maintenance tasks are due.
Predictive Maintenance: Use data analysis to schedule maintenance tasks efficiently.

User Feedback Mechanism:
In-App Feedback System: Implement a feedback system within the mobile app for user reports and feedback on restroom conditions.

4. **Integration Approach**
   Data Collection: IoT sensors continuously collect occupancy and hygiene data.
   Data Transmission: Microcontrollers process and transmit data to the cloud via Wi-Fi.
   Data Storage and Analysis: Cloud servers receive and store data, analyze it for cleanliness issues, and predict maintenance needs.
   User Access: Users access real-time data through the mobile app or web dashboard.
   Maintenance Alerts: Automated alerts are sent when cleanliness thresholds aren't met or maintenance tasks are due.
   Predictive Maintenance: Data analysis helps schedule maintenance tasks efficiently.
   User Feedback: Users provide feedback through the mobile app for continuous improvement.