**Phase-5**

**Smart public restroom**

**Connection**

* **Install Database Driver/Connector:**

Depending on the programming language you are using, you need to install a suitable database driver or connector. For example, if you're using Python with MySQL, you might install the mysql-connector or pymysql library.

pip install mysql-connector-python

* **Import the Database Connector:**

import mysql.connector

* **Set Up Connection Parameters:**

Define the connection parameters such as host, user, password, and database name. These parameters may vary based on your specific database setup.

db\_config = { 'host': 'your\_database\_host', 'user': 'your\_username', 'password': 'your\_password', 'database': 'your\_database\_name', 'port': 'your\_database\_port', # Optional, depending on the database }

* **Establish Connection:**

Use the connection parameters to establish a connection to the database.

try:

connection = mysql.connector.connect(\*\*db\_config)

if connection.is\_connected():

print("Connected to the database!")

except Exception as e:

print(f"Error: {e}")

* **Create a Cursor:**

After establishing a connection, create a cursor object. The cursor allows you to execute SQL queries.

cursor = connection.cursor()

* **Execute SQL Queries:**

You can now use the cursor to execute SQL queries.

query = "SELECT \* FROM your\_table;"

cursor.execute(query)

result = cursor.fetchall()

* **Close Connection:**

It's essential to close the cursor and the database connection when you're done.

cursor.close()

connection.close()

* **Error Handling:**

Implement error handling to manage exceptions that may occur during the connection process or query execution.

except mysql.connector.Error as err:

print(f"Error: {err}")

finally:

if connection.is\_connected():

cursor.close()

connection.close()

print("Connection closed.")

**CODING**

import requests

import time

import random

import Adafruit\_DHT # Make sure to install the Adafruit DHT library

channel\_id = "YOUR\_CHANNEL\_ID"

write\_api\_key = "YOUR\_WRITE\_API\_KEY"

thing\_speak\_url = f"https://api.thingspeak.com/update?api\_key={write\_api\_key}"

# Function to send data to ThingSpeak

def send\_data\_to\_thingspeak(data):

try:

# Create a dictionary with the field number and data value

payload = {

'field1': data # Replace 'field1' with the appropriate field on your ThingSpeak channel

}

# Send an HTTP POST request to ThingSpeak

response = requests.post(thing\_speak\_url, data=payload)

if response.status\_code == 200:

print(f"Data sent to ThingSpeak: {data}")

else:

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

except Exception as e:

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

# Simulated motion detection function

def motion\_detected():

return random.randint(0, 1) # Simulate motion detection (0 for no motion, 1 for motion)

# Function to read DHT sensor data

def read\_dht\_sensor(pin):

try:

sensor = Adafruit\_DHT.DHT22 # Use DHT11 or DHT22 based on your sensor type

humidity, temperature = Adafruit\_DHT.read\_retry(sensor, pin)

if humidity is not None and temperature is not None:

return temperature, humidity

else:

print("Failed to read DHT sensor data.")

except Exception as e:

print(f"Error reading DHT sensor: {str(e)}")

while True:

if motion\_detected():

temperature, humidity = read\_dht\_sensor(4) # Replace '4' with the GPIO pin connected to your DHT sensor

if temperature is not None and humidity is not None:

# Send temperature and humidity data to ThingSpeak

send\_data\_to\_thingspeak(temperature) # Sending temperature data to 'field1'

send\_data\_to\_thingspeak(humidity) # Sending humidity data to 'field2'

time.sleep(60) # Check for motion and read DHT sensor data every 60 seconds (adjust as needed)