**Phase-5**

**Smart public restroom**

**Planning**

1. **Simulate the Hardware:**

Build and connect the components in the Wokwi circuit simulator. You can use virtual components and wires to create a digital representation of your hardware design.

1. **Programming the Microcontroller:**

Write the code for the microcontroller to control the ventilation system. The code should include logic for monitoring air quality and adjusting the ventilation system based on sensor data. In Wokwi, you can use the Arduino IDE for coding and upload the program to your virtual microcontroller.

1. **Integration and Testing:**

Integrate your sensor data into the microcontroller program and set up the control logic to adjust ventilation as needed. Test your circuit and program to ensure they work together as expected.

1. **Visualization:**

Wokwi allows you to add visual elements to your simulation, so you can create visualizations of air quality parameters and how the ventilation system responds.

1. **Iterate and Refine:**

Simulate and test your system, and make any necessary adjustments to the hardware or code to achieve the desired ventilation control functionality.

1. **Record and Analyze Data:**

Collect and analyze data from the simulation to ensure that your ventilation control system is effectively maintaining air quality.

**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)

**working:**

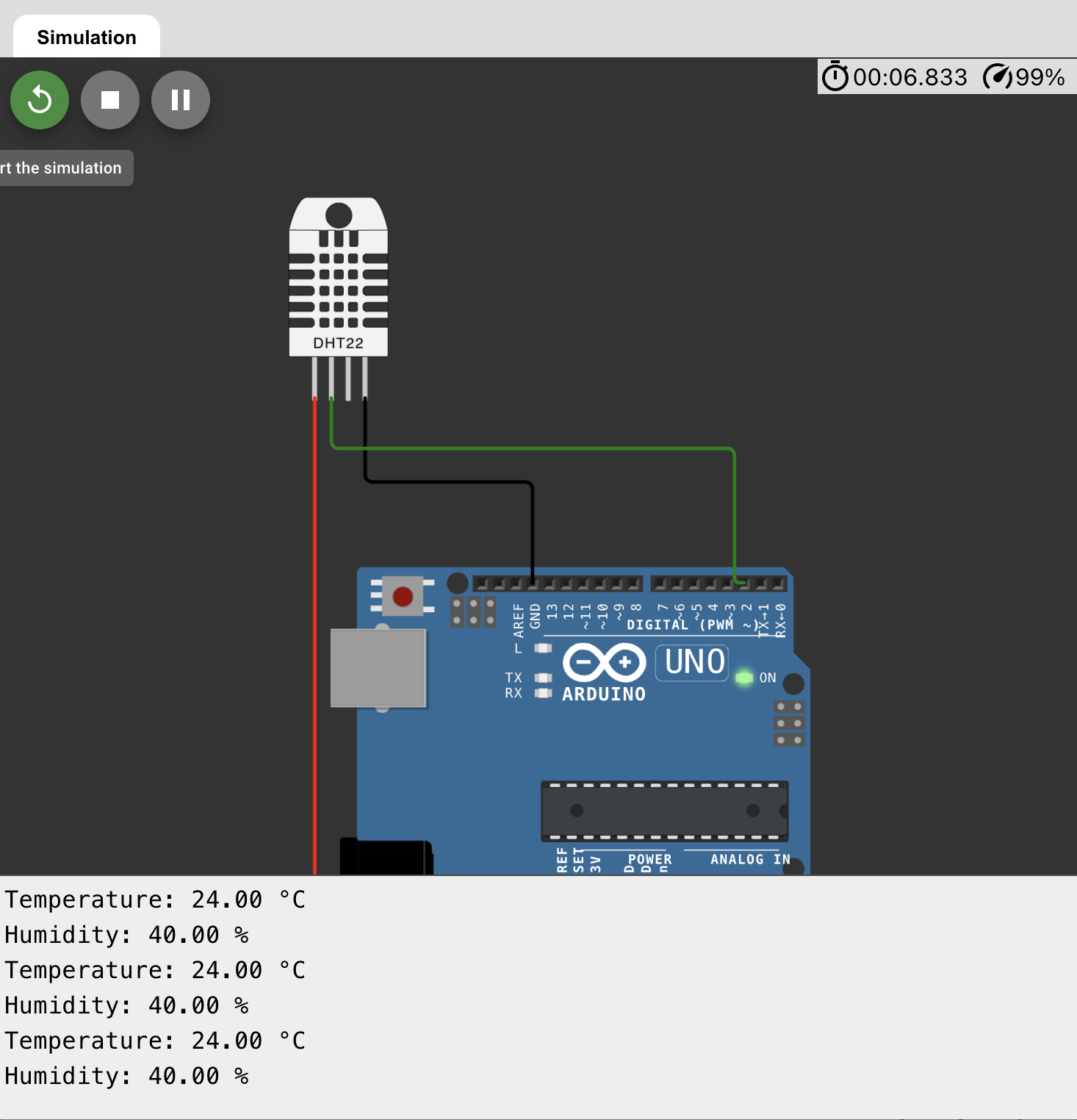
**Introduction**

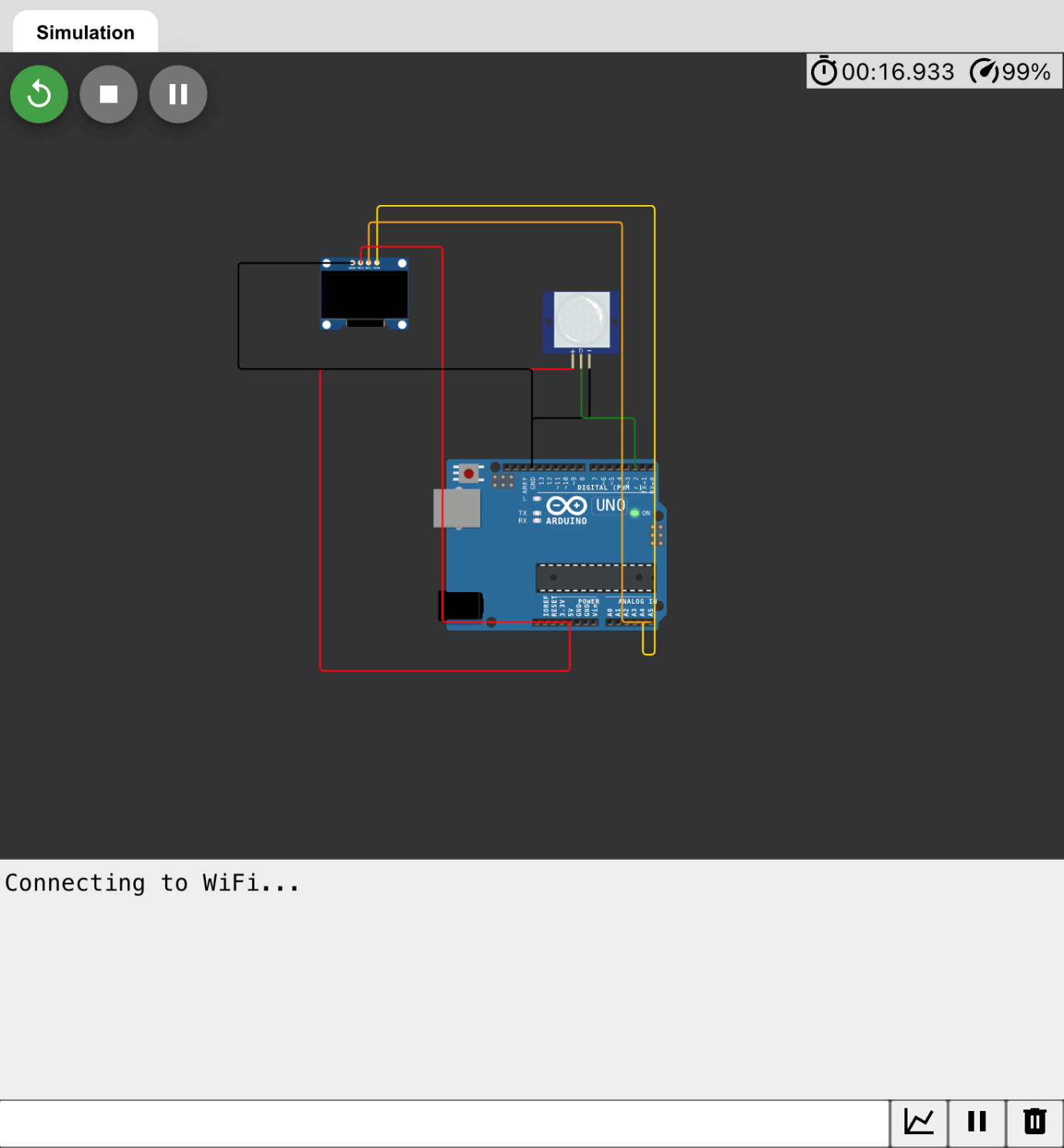
Welcome to the future of restroom experiences – the Smart Public Restroom. This cutting-edge facility combines IoT technology with a user-friendly app to redefine hygiene and accessibility. Say goodbye to restroom hassles – sensors ensure availability, cleanliness, and comfort. The app lets you find the nearest Smart Restroom, check real-time availability, and customize your preferences. Join us in creating a smarter, cleaner, and more enjoyable public restroom experience. Step into the future where innovation meets necessity, making every visit a seamless and connected encounter.

**Sensors**

1. **Occupancy Sensors**: These sensors are used to detect the presence of individuals within the restroom. They help in monitoring restroom usage in real-time, which is valuable for managing cleaning schedules and assessing occupancy levels. Occupancy data can also be used to display real-time availability and estimated wait times for users.
2. **Water Flow Sensors**: Water flow sensors are installed in water supply lines to monitor water usage. They help in promoting water efficiency by detecting leaks, monitoring water flow rates, and ensuring that water-saving fixtures are functioning correctly. This data can also be used to optimize water consumption.
3. **Air Quality Sensors**: Air quality sensors measure parameters such as humidity, temperature, and air quality (e.g., CO2 levels). These sensors help maintain a comfortable and healthy restroom environment by regulating temperature and humidity. Additionally, they ensure adequate ventilation and air quality for user comfort.
4. **Touchless Sensors**: Infrared or capacitive touchless sensors are used in faucets, soap dispensers, and flush systems. These sensors enable touchless operation, minimizing contact with surfaces and improving hygiene. They are crucial for reducing the spread of germs and ensuring user safety.
5. **Occupancy Indicator Sensors**: These sensors are often used outside the restroom to display occupancy status to users. They help users quickly determine if the restroom is available or occupied, reducing wait times and improving user experience.
6. **Security Cameras**: While not traditional sensors, security cameras are essential for maintaining security and ensuring user safety within the restroom. They deter vandalism and provide a record of activities inside the restroom, enhancing overall security.
7. **Light and Motion Sensors**: Light and motion sensors are used for energy efficiency. They ensure that lights are turned on only when needed, reducing energy consumption. Additionally, motion sensors can be used to trigger exhaust fans, ensuring proper ventilation.
8. **Rainwater Harvesting Sensors**: In restrooms equipped with rainwater harvesting systems for toilet flushing, sensors monitor the availability and level of collected rainwater. When rainwater is available, the system can automatically switch to using rainwater, reducing reliance on municipal water supply and promoting sustainability.
9. **Sound Sensors**: Sound sensors can be used to monitor sound levels within the restroom. They help in providing a more pleasant and private atmosphere by controlling ambient sounds, such as music or white noise.
10. **Temperature Sensors**: Temperature sensors help maintain a comfortable restroom environment. They ensure that heating and cooling systems are regulated effectively, ensuring user comfort.

**Simulator**

****

****

#include <Wire.h>

#include <Adafruit\_SSD1306.h>

#include <WiFi.h>

#include <ThingSpeak.h> // Include the ThingSpeak library

#define SCREEN\_WIDTH 128

#define SCREEN\_HEIGHT 64

#define OLED\_ADDR 0x3C

Adafruit\_SSD1306 display(SCREEN\_WIDTH, SCREEN\_HEIGHT, &**Wire**, OLED\_ADDR);

const int pirPin = 2;

int pirState = LOW;

const char\* ssid = "Wokwi-GUEST";

const char\* password = "";

const unsigned long channelID = 2310712;

const char\* writeAPIKey = "9PMHPSNFQEMZXR";

void setup() {

// Initialize Wi-Fi

**Serial**.begin(115200);

WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(10000);

**Serial**.println("Connecting to WiFi...");

}

**Serial**.println("Connected to WiFi");

pinMode(pirPin, INPUT);

if (!display.begin(SSD1306\_SWITCHCAPVCC, OLED\_ADDR)) {

for (;;);

}

display.clearDisplay();

display.setTextSize(1);

display.setTextColor(WHITE);

}

void loop() {

int pirValue = digitalRead(pirPin);

if (pirValue == HIGH) {

if (pirState == LOW) {

// Clear the display

display.clearDisplay();

display.setCursor(0, 0);

display.println("Occupied");

display.display();

// Send data to ThingSpeak

sendToThingSpeak(1); // Send 1 to indicate "Occupied"

delay(500);

pirState = HIGH;

}

} else {

display.clearDisplay();

display.setCursor(0, 0);

display.println("Free");

display.display();

// Send data to ThingSpeak

sendToThingSpeak(0); // Send 0 to indicate "Free"

delay(500);

pirState = LOW;

}

}

void sendToThingSpeak(int occupancyStatus) {

// ThingSpeak.begin(client); // You don't need to initialize the client

ThingSpeak.setField(1, occupancyStatus); // Set the value for field 1 (Occupancy Status)

int httpStatus = ThingSpeak.writeFields(channelID, writeAPIKey);

if (httpStatus == 200) {

**Serial**.println("Data sent to ThingSpeak successfully.");

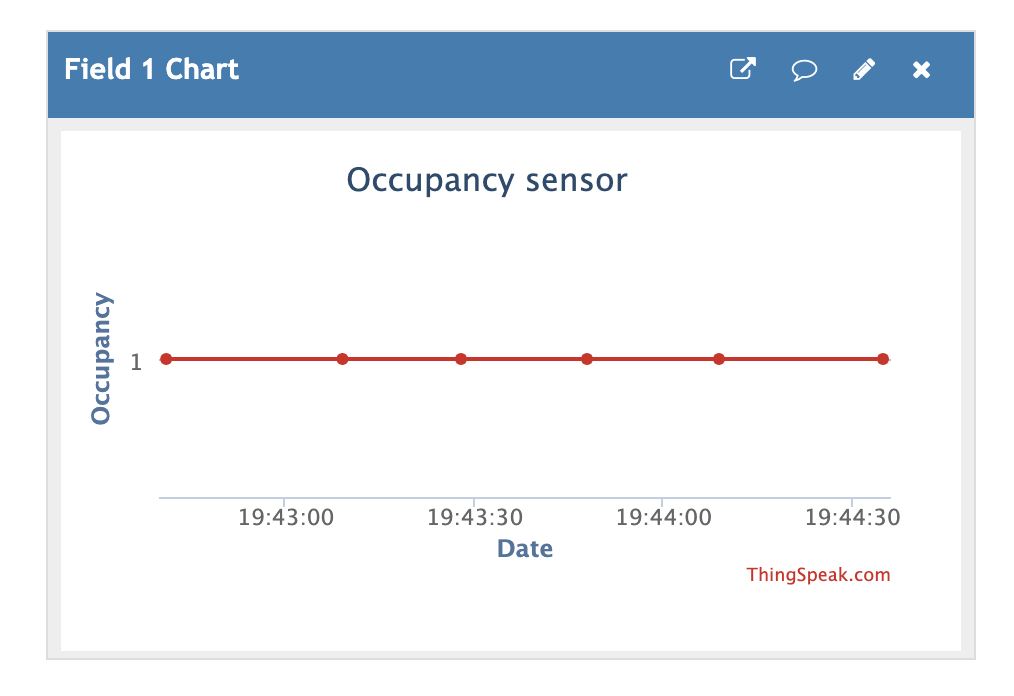
} else {

**Serial**.println("Data send to ThingSpeak failed. HTTP error code: " + String(httpStatus));

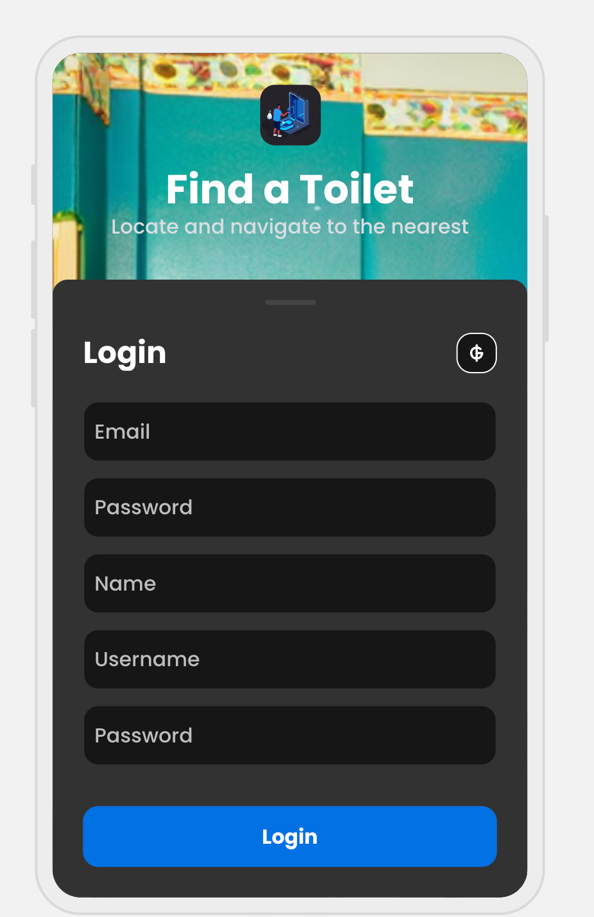
}

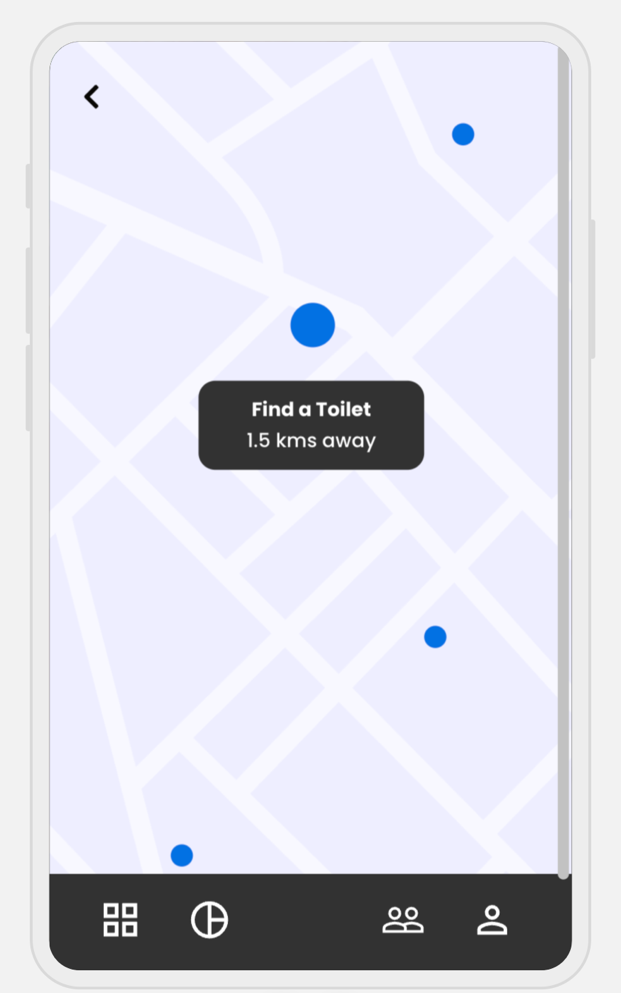
}

**Cloud:**

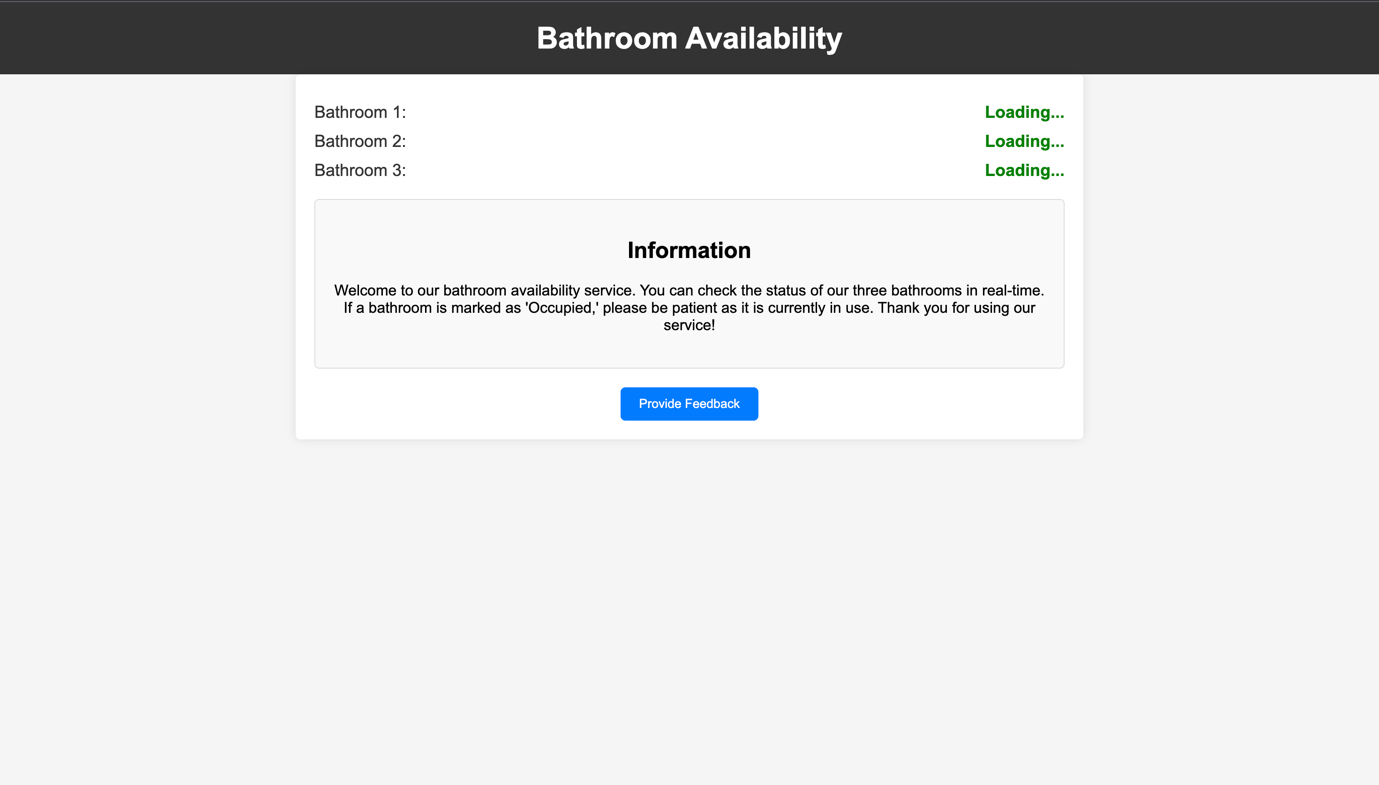
****

**APPLICATION**

****

****

**WEBSITE**

****