**IBM NAAN MUDHALVAN PROJECT**

**COURSE NAME :Internet of things(IOT)**

**TITLE :smart public restrooms**

**GROUP :07**

**PROJECT NUM :08**

**YEAR : 03**

**DEPARTMENT :electronics and communication engineering**

**PHASE 5:DOCUMENTATION AND SUBMISSION**

To implement the SMART PUBLIC RESTROOMS By using internet of things(IOT).

To maintain a disease free nation and to keep our times save in order in the form of using mobile applications.

PROBLEM DESCRIPTION:

A "smart public restroom" refers to a public restroom facility that incorporates advanced technologies and features to improve the user experience, enhance hygiene, and increase overall efficiency in managing and maintaining the facility. These restrooms leverage various innovations and digital solutions to offer a more convenient and pleasant experience for users while also benefiting facility managers and operators.

PROJECT OBEJECTIVE:

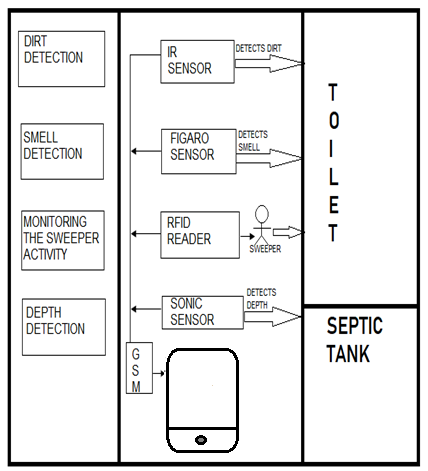
This project objective includes design thinking,IOT sensor design,real time transit information platform,integration approach,mobile App development,CODE implementation.

DESIGN THINKING:

1. Automated Entry and Exit Systems:
2. Occupancy Monitoring
3. Smart Cleaning Schedule
4. Air Quality Monitoring
5. Security and Privacy
6. Maintenance and Remote Monitoring
7. Analytics and Reporting
8. Emergency Alerts
9. Sustainability Measures
10. Public Wi-Fi and Charging Stations
11. Regular Maintenance and Upgrades

IOT SENSOR DESIGN:

**ARCHITECTURE OF THE PROPOSED SYSTEM:**



**DESCRIPTION OF ARCHITECTURE**

**HARDWARE REQUIREMENTS:**

 Microcontroller

 Power supply

 LCD display

 Buzzer

 Infrared sensor

 Sonicsensor

 Gassensor

 RFID

 GSMmodem

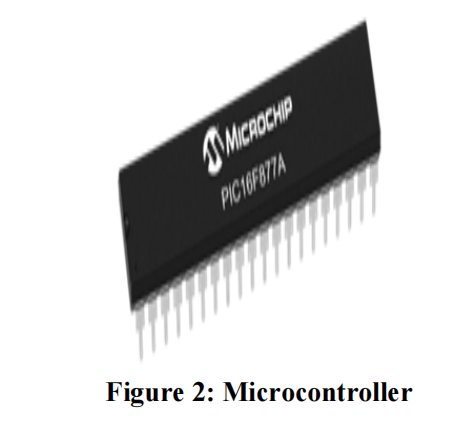
**SOFTWARE REQUIREMENTS :**

Embedded C

**2.2.1 MICROCONTROLLER**

A microcontroller is a small computer on a single combined circuit holding a processor core, memory and programmable input/output peripherals. Program memory in the form of Ferroelectric RAM, NOR flash or OTP ROM is also often included on chip, as well as

a typically small amount of RAM. Micro controllers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general-purpose applications.



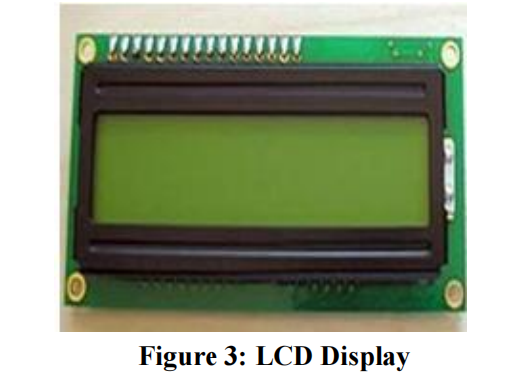
PIC 16F877 is one of the most advanced microcontroller from Microchip. This controller is

commonly used for experimental and modern applications because of its low price, wide range of requests, high quality, and ease of obtain-ability. It is ideal for applications such as machine control applications, measurement devices, study purpose, and so on. The PIC 16F877 features all the mechanisms which present micro-controllers usually have.

**2.2.2 LCD**

LCD stands for Liquid Crystal Display. By using the LCD, all the outputs are displayed. LCD

doesn’t know about the content (data or commands) supplied to its data bus. It is the user who has to specify whether the content at its data pins are data or commands.



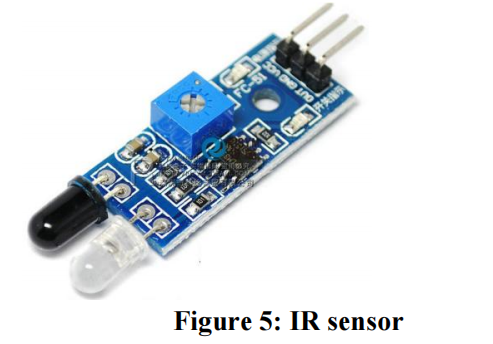
For this, if a command is inputted then a certainarrangement of 0s and 1s has to be applied to the Control lines so as to specify it is a command on the other hand if a data is inputted at the data lines thenan another combination of 0s and 1s has to be applied to the control lines to require it is Data.

**2.2.3 BUZZER**

Buzzer is also called as Beeper. It is a sound signalling mechanical device



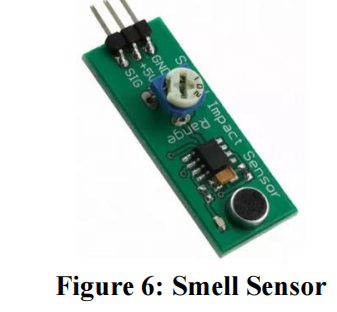
**2.2.4 INFRARED SENSOR**

The IR sensor is used to detect the dirt present in the toilet. Here we nourish the image models into the sensor. It can perceive the dirt by comparing the images we feed into it, after using the toilet. If it can detect the dirt, it raises the alarm, and the users may get embraced and they clean it. This system can create the responsiveness among the people. 

**2.2.5 SMELL SENSOR**

The Smell Sensor is used to detect the unwanted smell and gases in the toilet. For this

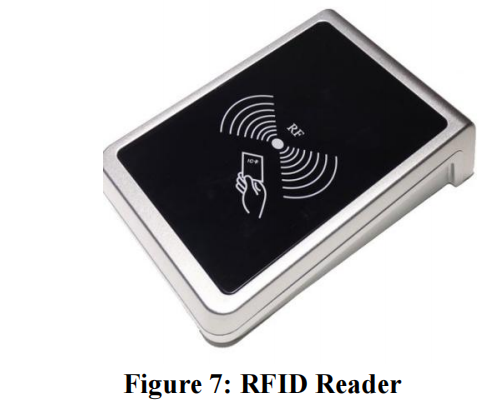
purpose, we are going to use the sensor called **Figaro** sensor.



**2.2.6 RFID READER**

The RFID stands for Radio Frequency Identification. It can be used for monitoring the

Sweeper. The Organization wishes to provide the identity tag for the Sweeper. The Sweeper desires to show the tag before the cleaning process is going to start and after it is finished.



Then the CR4 sensor can spot the presence of dirt. If it is present, it can blink the red light. If it is clean, it can blink the blue light. It assistances to understand the responsibilities of sweeper by his/her own. If Sweeper is not clean the toilets for period of time, his/her absence in cleaning the toilet also reported to the dependable organization. These all the details are

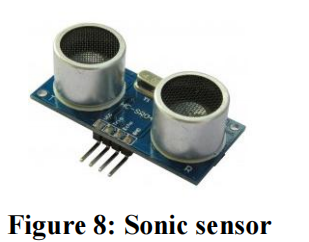
stored in the database.

**2.2.7 SONIC SENSOR**

The Sonic Sensor is used for computing the depth. Here it is used to measure the depth of the

septic tank. The Sonic Sensor is fixed into the Septic tank. Then the Septic tank get filled means, it can sends the communications to particular organization. Then they will allot persons to clean the septic tank. Then septic tank cleaners will clean the tank. After cleaning

it, the sensor can detect the level, and send messages to consistent organization.



**SOFTWARE SETUP:**

Implementing a smart public restroom system using JavaScript for the Internet of Things (IoT) typically involves developing the server and user interface components. Below, I'll provide an outline of how you can structure the JavaScript code for both the server and a simple web-based user interface. Please note that this is a simplified example for illustration purposes.

### Server-Side (Node.js)

1. Set up a Node.js server to handle IoT device communication and data processing.

const express = require('express');

const bodyParser = require('body-parser');

const app = express();

const port = 3000;

app.use(bodyParser.json());

// Implement routes for IoT device data and control

app.post('/sensor-data', (req, res) => {

// Handle incoming sensor data, e.g., occupancy, temperature, water quality

const sensorData = req.body;

// Store or process the data as needed

console.log('Received sensor data:', sensorData);

res.status(200).send('Data received');

});

// Implement routes for IoT device control, e.g., faucet control, smart locks

app.post('/control-faucet', (req, res) => {

// Control faucet based on the request

const { faucetId, action } = req.body;

// Implement control logic here

res.status(200).send('Faucet control request processed');

});

app.listen(port, () => {

console.log(`Server is running on port ${port}`);

});

### User Interface (Web-Based)

Develop a web-based user interface for restroom users and administrators using HTML, CSS, and JavaScript. You can use popular front-end libraries and frameworks like React, Angular, or Vue.js.

Create a simple HTML structure and include JavaScript to interact with the server and display data.

<!DOCTYPE html>

<html>

<head>

<title>Smart Public Restroom</title>

</head>

<body>

<h1>Smart Public Restroom</h1>

<!-- Display restroom occupancy -->

<p>Restroom Status: <span id="restroomStatus">Loading...</span></p>

<!-- Control faucet -->

<button onclick="controlFaucet('faucet1', 'on')">Turn On Faucet</button>

<button onclick="controlFaucet('faucet1', 'off')">Turn Off Faucet</button>

<script>

// Function to fetch and update restroom occupancy

function updateRestroomStatus() {

// Fetch data from the server and update the UI

fetch('/get-restroom-status')

.then((response) => response.json())

.then((data) => {

document.getElementById('restroomStatus').textContent = data.status;

});

}

// Function to control a faucet

function controlFaucet(faucetId, action) {

// Send a request to control the faucet to the server

fetch('/control-faucet', {

method: 'POST',

headers: {

'Content-Type': 'application/json',

},

body: JSON.stringify({ faucetId, action }),

})

.then(() => {

// Handle the response if needed

console.log(`Faucet control request sent: ${action}`);

});

}

// Periodically update restroom occupancy

setInterval(updateRestroomStatus, 5000); // Update every 5 seconds

updateRestroomStatus(); // Initial update

</script>

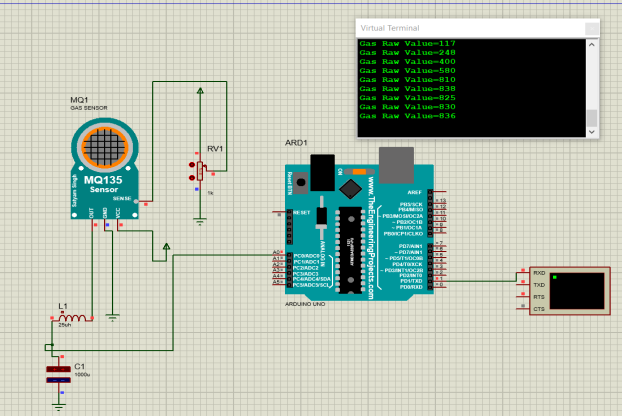
</body>

</html>

MODEL TRAINING:

*A. Simulation of Gas sensor*

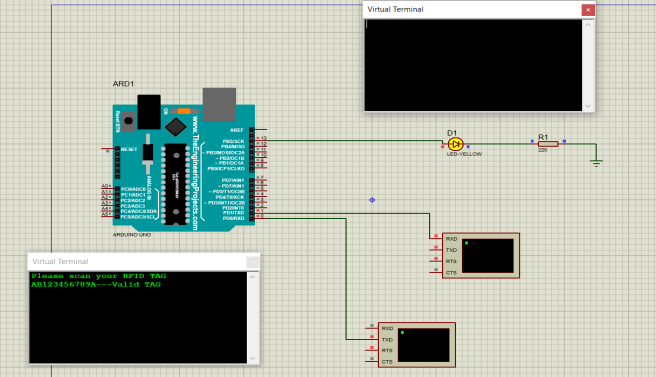
Within the toilet, the MQ-135 sensor detects smells. This MQ-135 sensor measures the amount of ammonia gas present in the toilet's air.



The ammonia inside the toilet produces a nasty odour. It has a strong odour that can be detected at concentrations more than 5 parts per million (parts per million).The ventilation fans are automatically switched on when the ammonia concentration in the toilet surpasses the predetermined threshold of 5 ppm; however, if the concentration falls below 5 ppm, no action is performed, as shown in Fig. 2.

1. *Simulation of RFID Sensor*

The RFID (Radio Frequency Identification) technology, is a key enabling technology of the IoT sensing layer. As shown in Fig. 3, when the cleaner enters the toilet and when he scans his ID in the RFID scanner, It displays his name with the valid tag, and in another case when he scans other than his ID it shows that the ID is invalid. This helps to monitor the cleaner’s attendance.



*C. Real time interface*

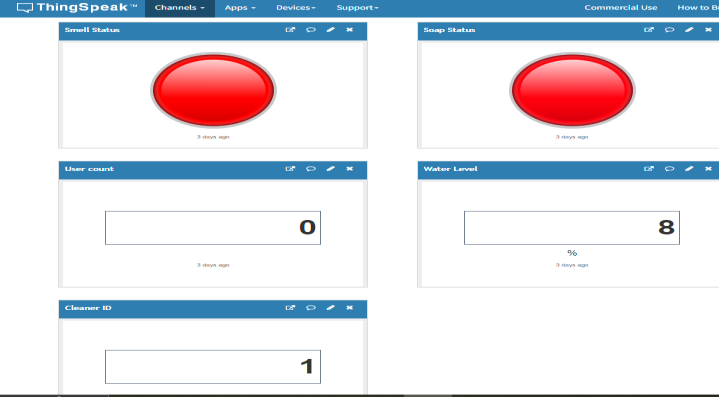
We employed five sensors in our project. The first is an RFID sensor, which is used to track the cleaner's attendance. The MQ-135 sensor was the second sensor we utilized to assess air quality. For soap detection, we employed an IR sensor as the third sensor. Ultrasonic sensor for tank water level measurement is the fourth sensor. The fifth sensor is an infrared sensor that counts how many people enter the

restroom. These sensors are all connected to Arduino, which collects all sensor data and sends it to the cloud via the ESP8266 Wi-Fi module. Here we have used Thingspeakopen-source website for storing the collected sensors data. All the sensors’ data is even displayed on LCD screen as well. Fig. 4 Shows the Thingspeak website where all the data are stored. It shows Smell status, Soap status, user count,

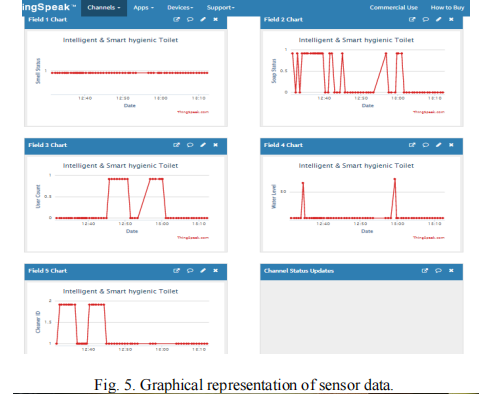
water level, cleaner id. Soap and smell status is determined

using red light as shown in Fig. 4. If the soap is present in the container then the red LED will turn on, similarly if there is smell in the toilet then the red LED for smell status will turn on. Water level is determined using percentages in water level block. It also counts number of people entering the

toilet in user count block.



Public view in Thingspeak software.



**MOBILE APP DEVELOPMENT:**

Developing a mobile app for smart public restrooms can provide a range of benefits for both restroom users and facility managers. These apps can offer features like finding nearby public restrooms, checking their availability, cleanliness, and even providing additional services such as automatic door unlocking, payment processing, and feedback collection.

Designing a mobile app UI for smart public restrooms involves creating an interface that provides users with easy access to restroom information and sensor data. Below, I'll outline a simple UI design for such an app. Keep in mind that this is a basic example, and you can expand and customize it based on your specific requirements.

### Home Screen:

* **Title**: "Smart Restrooms"
* **Header/Navigation Bar**:
  + Logo or App Name
  + Icons for additional features (e.g., settings, information)
* **Current Status**:
  + Show whether the nearest public restroom is occupied or vacant.
  + Use icons or colors to represent occupancy status.
* **Quick Actions**:
  + Buttons for common actions (e.g., Find Nearest Restroom, Rate Cleanliness).
* **Map**:
  + A map with restroom locations (if applicable) and their occupancy status.

### Restroom Details Screen:

* **Header/Navigation Bar**:
  + Back button or icon
  + Restroom Name or Location
* **Restroom Information**:
  + Address
  + Cleanliness rating (user-generated or sensor-based)
* **Sensor Data**:
  + Real-time sensor data:
    - Occupancy status
    - Temperature
    - Air quality
  + Historical data (e.g., charts for temperature and air quality trends).
* **Actions**:
  + Buttons to send feedback or request cleaning service.
* **Additional Information**:
  + Opening hours
  + Accessibility features (e.g., baby changing station, handicap accessibility).

### Find Nearest Restroom Screen:

* **Header/Navigation Bar**:
  + Back button or icon
  + Search bar and filters for nearby restrooms.
* **List of Restrooms**:
  + Show nearby restrooms, their names, and occupancy status.
  + Allow users to sort by distance or cleanliness rating.
* **Map View**:
  + Display nearby restroom locations on a map.

### Settings Screen:

* **Header/Navigation Bar**:
  + Back button or icon
  + Title: "Settings"
* **User Preferences**:
  + Allow users to customize app settings:
    - Notification preferences (e.g., occupancy alerts)
    - Preferred units (e.g., Celsius/Fahrenheit)
* **About**:
  + Information about the app, version, and developers.

### Login/Registration (Optional):

* If the app requires user accounts for features like rating and feedback, create screens for registration and login.

SOURCE CODE :

#include <Servo.h> int ULT = 7; int PIR = 8; int BUZZ = 12; int GAS = A0; int BULB = 5; int JARAKU = 0; int KADARG = 0; int JARAKP = 0; int pin = 10;Servo SERV; long bacaULT(int pin) //fungsi penghitungan durasi pantulan

{

pinMode(pin, OUTPUT);

digitalWrite(pin, LOW); delayMicroseconds(2); digitalWrite(pin, HIGH); delayMicroseconds(10); digitalWrite(pin, LOW); pinMode(pin, INPUT); return pulseIn(pin, HIGH);

}

void setup()

{

pinMode(ULT,INPUT); pinMode(PIR,INPUT); pinMode(GAS,INPUT); pinMode(BUZZ,OUTPUT); pinMode(BULB,OUTPUT);

SERV.attach(pin);

Serial.begin(9600);//inisialisasi komunikasi serial 9600 bps

}

void loop()

{

JARAKU = 0.01723\*bacaULT(ULT);//membaca jarak objek

KADARG = analogRead(GAS);//membaca kadar gas

JARAKP = digitalRead(PIR);//membaca pergerakan objek

//Program Sensor Ultrasonik if (JARAKU<=120)

{

digitalWrite(BULB,HIGH); Serial.print("\nULT : Objek Dideteksi! ");

Serial.print(JARAKU,DEC);

Serial.print(" (Lampu Nyala)");

}

else

{

digitalWrite(BULB,LOW);

Serial.print("\nJarak: ");

Serial.print(JARAKU,DEC);

Serial.print(" (Lampu Mati)");

}

//Program Sensor Gas if (KADARG>=400)

{

tone(BUZZ,500);

Serial.print("\nGAS : Gas Dideteksi! ");

Serial.print(KADARG,DEC);

Serial.print(" (Buzzer Bunyi)");

}

else

{

noTone(BUZZ);

Serial.print("\nKadar: ");

Serial.print(KADARG,DEC);

Serial.print(" (Buzzer Berhenti)");

}

//Program Sensor PIR if (JARAKP==HIGH)

{

Serial.print("\nPIR : Objek Bergerak Dideteksi!");

Serial.print(" (Motor Nyala)\n"); SERV.write(180); delay(1000);

}

SUBMISSION:

Github repository link:

**PYTHON SOURCE CODE:**

Provide instructions on how to replicate the project, deploy IoT sensors, develop the transit information platform, and integrate them using Python.

# Import necessary libraries

import time

import random

# Mock IoT sensor data generation (replace with actual sensor data)

def generate\_sensor\_data():

occupancy = random.choice([0, 1])

temperature = random.uniform(15.0, 30.0)

air\_quality = random.uniform(0.0, 100.0)

return {

"occupancy": occupancy,

"temperature": temperature,

"air\_quality": air\_quality

}

# Mock transit information (replace with real-time data)

def get\_transit\_info():

return "Next bus departs in 10 minutes."

# Data processing and analysis

def process\_sensor\_data(data):

# Add data processing logic here

pass

# User interface (a simple text-based UI)

def display\_ui(sensor\_data, transit\_info):

print("Smart Public Restroom System")

print("----------------------------")

print(f"Occupancy: {'Occupied' if sensor\_data['occupancy'] else 'Vacant'}")

print(f"Temperature: {sensor\_data['temperature']}°C")

print(f"Air Quality: {sensor\_data['air\_quality']} ppm")

print("\nTransit Information:")

print(transit\_info)

# Main loop for data collection and display

while True:

# Generate mock sensor data

sensor\_data = generate\_sensor\_data()

# Mock data processing and analysis

process\_sensor\_data(sensor\_data)

# Mock transit information retrieval

transit\_info = get\_transit\_info()

# Display the user interface

display\_ui(sensor\_data, transit\_info)

# Sleep for a predefined interval (simulating real-time data)

time.sleep(30) # Sleep for 30 seconds

Output:

Smart Public Restroom System

----------------------------

Occupancy: Vacant

Temperature: 28.90909849160353°C

Air Quality: 3.4615787222680883 ppm

Transit Information:

Next bus departs in 10 minutes.

Smart Public Restroom System

----------------------------

Occupancy: Vacant

Temperature: 20.31489978196703°C

Air Quality: 18.97944652511603 ppm

Transit Information:

Next bus departs in 10 minutes.

***RASPBERRY PI DATA TRANSMISSION:***

To collect data from various sensors in smart public restrooms using a Raspberry Pi, you'll need to interface with these sensors and read data. Raspberry Pi is a versatile platform for IoT projects, and you can use it to gather information from various types of sensors. Below is a basic example of how to read data from a few common types of sensors using Python on a Raspberry Pi:

### Requirements:

* Raspberry Pi (any model with GPIO pins)
* Appropriate sensors (e.g., motion sensor, temperature sensor, humidity sensor)
* Appropriate libraries for interfacing with the sensors (e.g., RPi.GPIO, Adafruit DHT library)

import RPi.GPIO as GPIO

import Adafruit\_DHT # For DHT temperature and humidity sensor

import time

# Set up GPIO mode

GPIO.setmode(GPIO.BCM)

# Define GPIO pins for sensors

motion\_sensor\_pin = 18

dht\_sensor\_pin = 4 # This pin number may vary

# Initialize the DHT sensor

dht\_sensor = Adafruit\_DHT.DHT22

try:

while True:

# Read data from motion sensor

GPIO.setup(motion\_sensor\_pin, GPIO.IN)

motion\_detected = GPIO.input(motion\_sensor\_pin)

# Read data from DHT sensor

humidity, temperature = Adafruit\_DHT.read\_retry(dht\_sensor, dht\_sensor\_pin)

# Display the sensor data

print("Motion Detected: " + ("Yes" if motion\_detected else "No"))

if humidity is not None and temperature is not None:

print(f"Temperature: {temperature:.2f}°C")

print(f"Humidity: {humidity:.2f}%")

else:

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

# Sleep for a predefined interval (e.g., 30 seconds)

time.sleep(30)

except KeyboardInterrupt:

print("Terminating the data collection process.")

finally:

GPIO.cleanup()

This code reads data from a motion sensor and a DHT22 temperature and humidity sensor connected to the Raspberry Pi. You'll need to connect the sensors to the appropriate GPIO pins on the Raspberry Pi and ensure you have the necessary libraries installed.

**CONCLUSION:**

The initiative met its goal of creating an economical, user-friendly interface between the cleaning company and public restrooms, allowing for more effective staffing. This program's installation is straightforward. Time series forecasts can be utilized to conduct simultaneous toilet research. If this toilet condition is utilized in the toilet, it assists in keeping the toilet clean before it becomes

unclean. The mobile app is considerably easier to use now that the data display has been updated. The Internet of Things device is both inexpensive and portable. In future, this study might be enhanced by employing sensitive and modern sensors to generate more precise data. By learning from sample data and increasing the amount of test data, machine learning increases prediction accuracy. When this approach is used on a large scale, better storage systems and cloud servers can be used.