# Phase 3
# IBM PROJECT
# SMART WATER MANAGEMENT PROJECT

**Algorithm for Smart Water Management Simulation**

Algorithm for Smart Water Management Simulation

1. **Initialization:**

- Create a class SmartWaterMeter to represent smart water meters.
- Initialize the meter's location and set initial water usage to 0.
- Implement a method measure_water_usage that simulates water usage data with random values.
- Create a class WaterQualitySensor to represent water quality sensors.
- Initialize the sensor's location and set water quality to "Good" by default.
- Implement a method measure_water_quality that simulates water quality data.
- Create a class CloudPlatform to represent the cloud platform for data storage.
- Initialize data structures to store water meter and water quality sensor data.

2. **Data Processing:**

- Define a function data_processing that takes the cloud platform's data as input.
- Inside the function, analyze and process the incoming data.
- Iterate through water meters' data:
- Check if water usage exceeds a threshold (e.g., 40 units).
- If usage is above the threshold, print an alert about a potential water leak at the meter's location.
- Iterate through water quality sensor data:
- Check if the water quality is not "Good."
- If water quality is not good, print an alert about a water quality issue at the sensor's location.

3. **Simulation:**

- Create instances of smart water meters and water quality sensors.
- Create an instance of the cloud platform.

4. **Data Collection and Storage:**

- In a continuous loop (simulation):
- Simulate water consumption and quality data for each smart meter and sensor.
- Store the simulated data in the cloud platform.

5. **Data Processing and Alerts:**

- Continuously process the stored data using the data_processing function.
- Detect potential water leaks and water quality issues.
- Print alerts as needed.

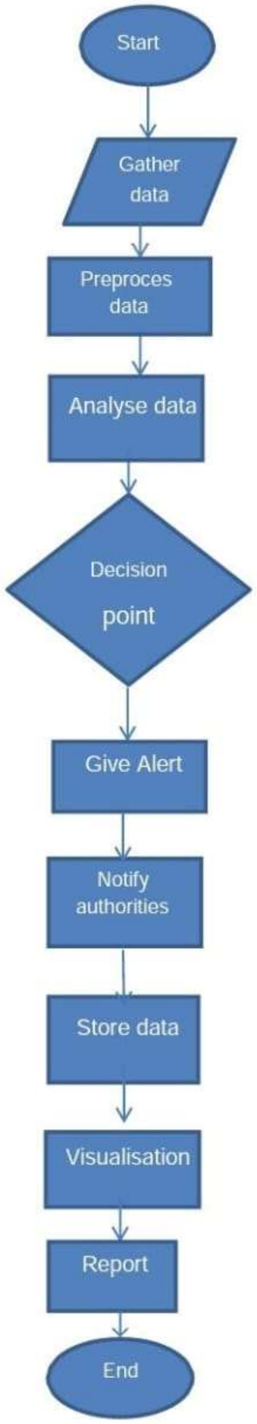
6. **Data Collection Interval:**

- Simulate data collection at regular intervals (e.g., every 5 minutes) using time.sleep.

7. **End of Algorithm:**

- The simulation and data processing will continue until the program is manually stopped.

## Flow chart of Algorithm



### Code in Python:

```
import random
import time

class SmartWaterMeter:
    def __init__(self, location, pin):
        self.location = location
        self.water_usage = 0
        self.pin = pin  # Pin used for simulating activity
(optional)

    def measure_water_usage(self):
        # Simulate water usage data
        simulated_data = random.randint(1, 50)
        self.water_usage += simulated_data

        # Simulate water meter activity using a pin (optional)
        # digitalWrite(self.pin, HIGH)  # Simulate the meter is
active
        # Simulate the time taken to measure water usage
(optional)
        # delay(100)
        # digitalWrite(self.pin, LOW)  # Simulate the meter is
inactive (optional)

        return self.water_usage

class WaterQualitySensor:
    def __init__(self, location, pin):
        self.location = location
        self.water_quality = "Good"
        self.pin = pin  # Pin used for simulating activity
(optional)

    def measure_water_quality(self):
        # Simulate water quality data
        simulated_data = "Good"

        # Simulate sensor activity using a pin (optional)
        # digitalWrite(self.pin, HIGH)  # Simulate the sensor is
active
        # Simulate the time taken to measure water quality
(optional)
        # delay(100)
        # digitalWrite(self.pin, LOW)  # Simulate the sensor is
inactive (optional)

        return simulated_data

class CloudPlatform:
    def __init__(self):
        self.water_meters = {}
        self.water_quality_sensors = {}

    def store_water_meter_data(self, meter_location, data):
        self.water_meters[meter_location] = data

    def store_water_quality_data(self, sensor_location, data):
        self.water_quality_sensors[sensor_location] = data

def data_processing(data):
    # Analyze and process incoming data here
    for location, usage in data.water_meters.items():
        if usage > 40:
            print(f"Potential water leak detected at {location}.
Alert sent.")
    for location, quality in data.water_quality_sensors.items():
        if quality != "Good":
            print(f"Water quality issue detected at {location}.
Alert sent.")

# Specify the pins for each component
meter1 = SmartWaterMeter("Residential", 2)  # Pin 2 used for
simulating activity (optional)
meter2 = SmartWaterMeter("Commercial", 3)   # Pin 3 used for
simulating activity (optional)
quality_sensor1 = WaterQualitySensor("Reservoir", 4)  # Pin 4
used for simulating activity (optional)
quality_sensor2 = WaterQualitySensor("Treatment Plant", 5)  #
Pin 5 used for simulating activity (optional)
```

```
# Create a cloud platform instance
cloud_platform = CloudPlatform()

while True:
    # Simulate water consumption and quality data
    meter1_data = meter1.measure_water_usage()
    meter2_data = meter2.measure_water_usage()
    quality_sensor1_data =
quality_sensor1.measure_water_quality()
    quality_sensor2_data =
quality_sensor2.measure_water_quality()

    # Store data in the cloud platform
    cloud_platform.store_water_meter_data(meter1.location,
meter1_data)
    cloud_platform.store_water_meter_data(meter2.location,
meter2_data)

cloud_platform.store_water_quality_data(quality_sensor1.location
, quality_sensor1_data)

cloud_platform.store_water_quality_data(quality_sensor2.location
, quality_sensor2_data)

    # Process the stored data (simplified)
    data_processing(cloud_platform)

    # Simulate data collection at regular intervals (e.g., every
5 minutes)
    time.sleep(300)  # Sleep for 5 minutes
```

**Code in Arduino:**

```
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // Define RX and TX pins for
communication with a serial terminal

class SmartWaterMeter {
  public:
    SmartWaterMeter(String location, int pin) {
      this->location = location;
      this->pin = pin;
      water_usage = 0;
      pinMode(pin, OUTPUT);
    }

    int measure_water_usage() {
      int simulatedUsage = random(1, 51); // Simulate water
usage data
      water_usage += simulatedUsage;
      digitalWrite(pin, HIGH); // Simulate water meter activity
using an LED (optional)
      delay(100); // Simulate the time taken to measure water
usage
      digitalWrite(pin, LOW);
      return water_usage;
    }

  private:
    String location;
    int water_usage;
    int pin;
};

class WaterQualitySensor {
  public:
    WaterQualitySensor(String location, int pin) {
      this->location = location;
      this->pin = pin;
      water_quality = "Good";
      pinMode(pin, OUTPUT);
    }

    String measure_water_quality() {
      // Simulate water quality data
      digitalWrite(pin, HIGH); // Simulate sensor activity using
an LED (optional)
      delay(100); // Simulate the time taken to measure water
quality
      digitalWrite(pin, LOW);
      return water_quality;
    }

  private:
    String location;
    String water_quality;
    int pin;
};

SmartWaterMeter meter1("Residential", 2); // Define the pin for
the first water meter (e.g., pin 2)
SmartWaterMeter meter2("Commercial", 3);  // Define the pin for
the second water meter (e.g., pin 3)
WaterQualitySensor sensor1("Reservoir", 4); // Define the pin
for the first water quality sensor (e.g., pin 4)
WaterQualitySensor sensor2("Treatment Plant", 5); // Define the
pin for the second water quality sensor (e.g., pin 5)

void setup() {
  mySerial.begin(9600);
}

void loop() {
  // Simulate water consumption and quality data
  int meter1_data = meter1.measure_water_usage();
  int meter2_data = meter2.measure_water_usage();
  String sensor1_data = sensor1.measure_water_quality();
  String sensor2_data = sensor2.measure_water_quality();

  // Print data to the serial terminal
  mySerial.print("Meter 1 - Location: ");
  mySerial.print(meter1_data);
  mySerial.print(" - Water Usage: ");
  mySerial.println(meter1_data);

  mySerial.print("Meter 2 - Location: ");
  mySerial.print(meter2_data);
  mySerial.print(" - Water Usage: ");
  mySerial.println(meter2_data);

  mySerial.print("Sensor 1 - Location: ");
  mySerial.print(sensor1_data);
  mySerial.print(" - Water Quality: ");
  mySerial.println(sensor1_data);

  mySerial.print("Sensor 2 - Location: ");
  mySerial.print(sensor2_data);
  mySerial.print(" - Water Quality: ");
  mySerial.println(sensor2_data);

  delay(300000); // Sleep for 5 minutes (300,000 milliseconds)
}
```