**INTERNET OF THINGS-GROUP 5**

**TEAM MEMBER**

**Project Title: Smart water system**

**Phase 3: Development 1**

IoT-based smart water system involves various components and technologies. The code using Python and MQTT (Message Queuing Telemetry Transport) as a communication protocol for IoT devices.

**Sensor Data Collection:**

**Coding:**

# Code to collect data from water quality and quantity sensors

import sensor\_library # Import the appropriate sensor library

def collect\_sensor\_data():

# Implement code to read data from sensors

return sensor\_data

Data Processing and Analytics:

**Coding:**

# Code for data processing and analytics

import pandas as pd # You may need to use a suitable data analysis library

def process\_data(sensor\_data):

# Implement data processing and analytics

processed\_data = data\_processing\_algorithm(sensor\_data)

return processed\_data

MQTT Communication:

**Coding**:

import paho.mqtt.client as mqtt # MQTT library for Python

# MQTT configuration

mqtt\_broker = "mqtt.server.com"

mqtt\_port = 1883

mqtt\_topic = "smart\_water\_system/data"

def publish\_data\_mqtt(data):

client = mqtt.Client("SmartWaterSystem")

client.connect(mqtt\_broker, mqtt\_port)

client.publish(mqtt\_topic, data)

client.disconnect()

**Remote Control Interface:**

You can develop a web-based or mobile app for remote control using frameworks like Flask (for web) or React Native (for mobile).

**Security**:

Implement security features like encryption, authentication, and authorization to protect your IoT system.

**Energy Efficiency:**

Code logic to optimize energy consumption, e.g., scheduling sensor readings and device operations.

**User Interface:**

Develop the user interface to visualize data, control devices, and display analytics. This can be done using HTML, CSS, and JavaScript for web or relevant technologies for mobile.

**Integration**:

Ensure that your system can integrate with other components of a smart city infrastructure through appropriate APIs and protocols.

**Testing and Validation:**

Rigorously test the entire system, including unit testing, integration testing, and user acceptance testing.

**Documentation:**

Document code, APIs, and configurations for future reference and maintenance.

this is a high-level , and to design and structure based on specific hardware and software requirements.