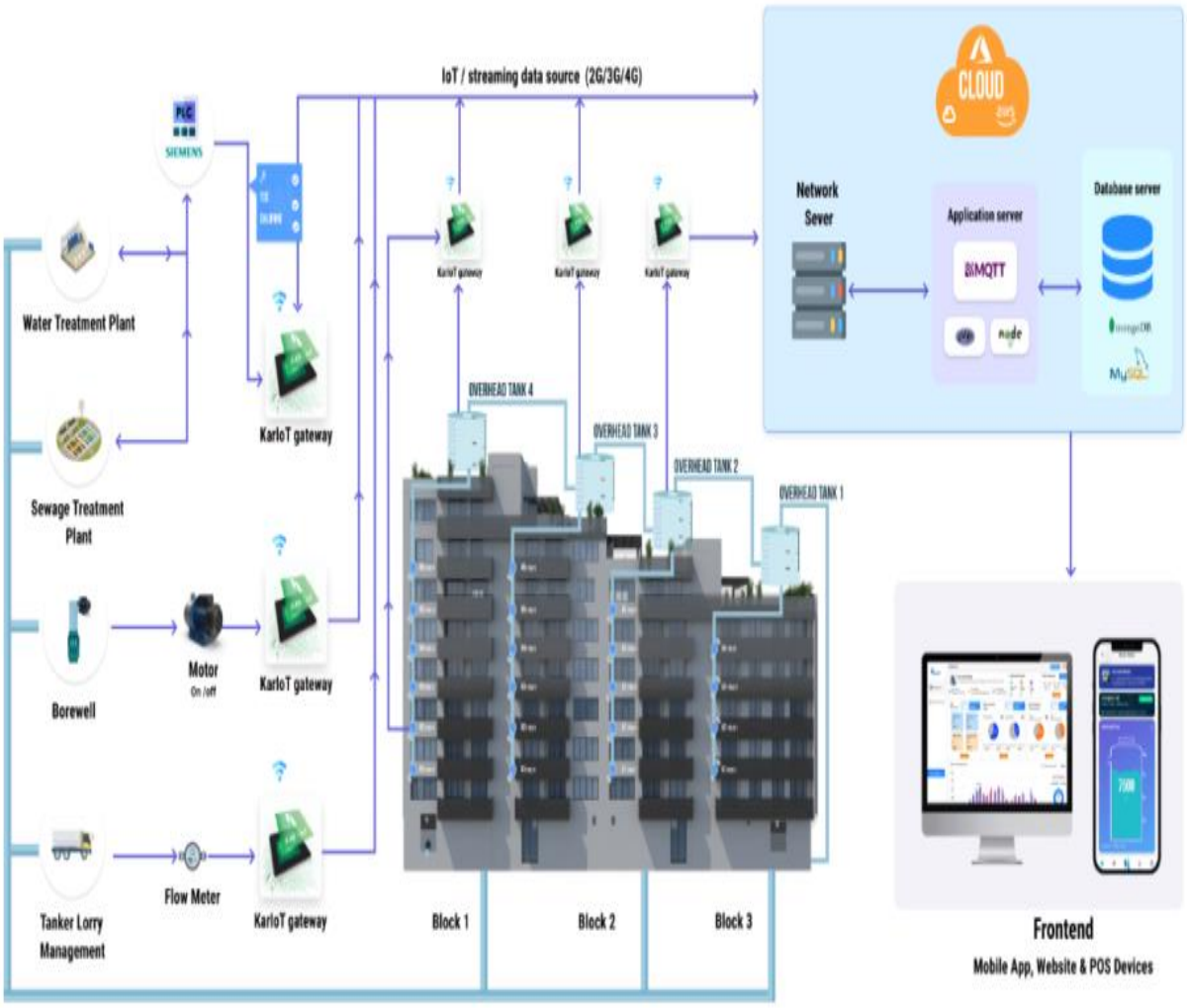
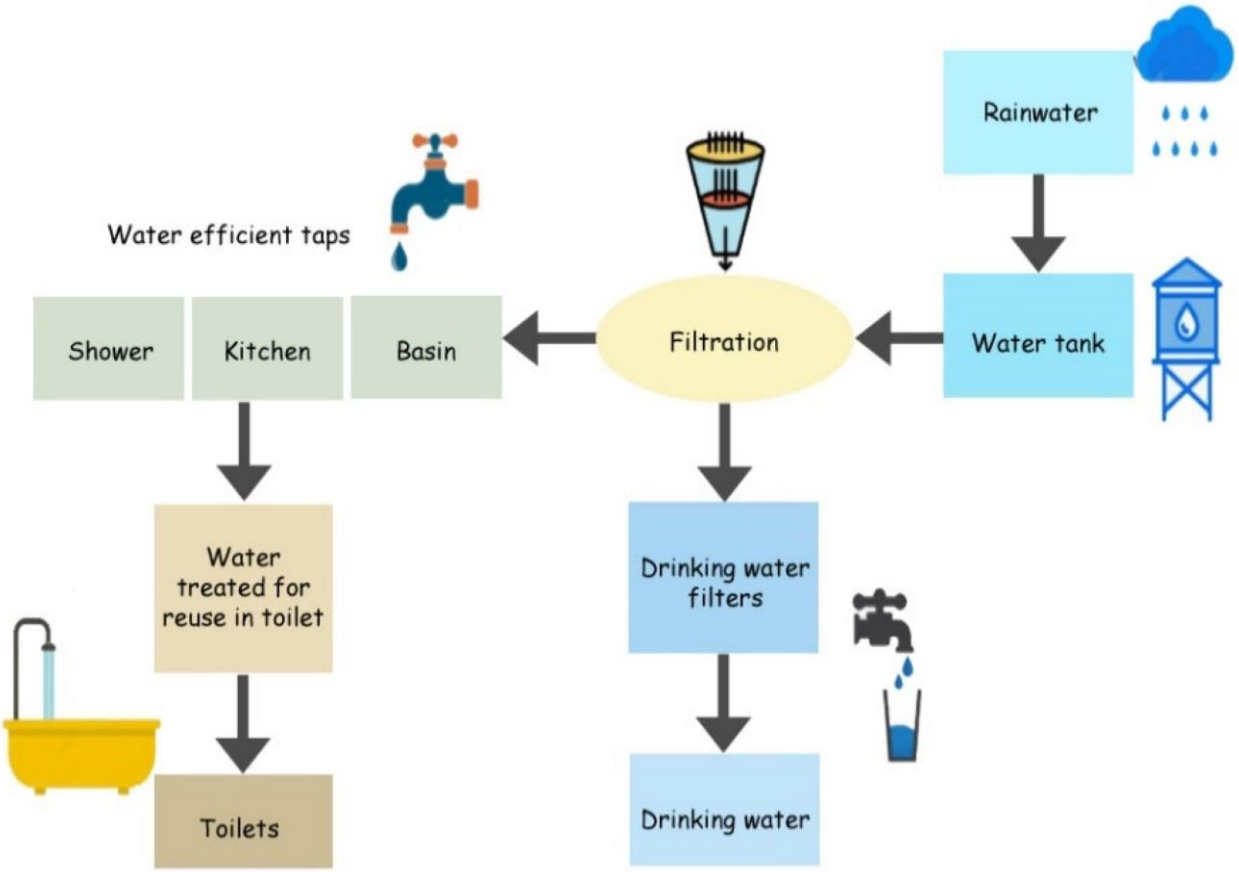
SMART WATER SYSTEM

PHASE 3: Development Part 1



WATER MANAGEMENT SYSTEM:



Hardware Components:

- Flow sensors to measure water flow rates.
- Microcontroller (e.g., Arduino, Raspberry Pi) to interface with the sensors.
- A solenoid valve to control the water supply.
- A display (LCD, LED) for showing the water consumption or alerts.

Software Components:

- Python for data processing and decision-making.
- Appropriate libraries for interfacing with the hardware (e.g., `gpiozero` for Raspberry Pi).
- A database or storage system to log water consumption data.

ALGORITHM:

1. **Initialization:**
    - Initialize system components and sensors.
    - Set threshold values for water consumption.
    - Initialize data storage for water usage records.
    - Connect to the user interface for real-time monitoring and control.
2. **Continuous Monitoring:**
    - Continuously monitor the flow sensors to measure water flow rates.
    - Update the total water consumption over a specific time period (e.g., per minute).
3. **Data Logging:**
    - Log the water consumption data, including timestamp and volume, to a database or storage system for historical analysis.
4. **Alert Generation:**
    - Compare the current water consumption with predefined thresholds.
    - If the water consumption exceeds a threshold, generate an alert. Possible alerts include:
        - Low water level alert.
        - Leak detection alert.
        - Excessive water usage alert.
5. **Decision-Making:**
    - Make decisions based on the collected data and generated alerts.
    - If a leak is detected, close the solenoid valve to prevent further water loss.
    - If water usage exceeds defined limits, send notifications to the user interface.

- If water supply is running low, send alerts for re-filling.

6. **User Interface Interaction:**
   - Allow users to monitor their water usage in real-time.
   - Enable users to remotely control the solenoid valve (e.g., turning off water supply when not needed).
7. **Data Analysis and Reporting:**
   - Periodically analyze historical water consumption data to identify patterns and anomalies.
   - Generate reports and insights for users to encourage water conservation.
8. **Maintenance and Error Handling:**
   - Implement error-handling mechanisms to handle sensor malfunctions or system errors.
   - Schedule regular maintenance checks for the sensors and components.
9. **Shutdown:**
   - Gracefully shut down the system when not in use or during scheduled maintenance.
10. **User Notifications:**
    - Notify users through the user interface, emails, or SMS when significant events occur (e.g., leak detection, alerts).

Python program:

```
import time

import RPi.GPIO as GPIO  # Use GPIO library for Raspberry Pi

from datetime import datetime


# Configuration

FLOW_SENSOR_PIN = 17

SOLENOID_PIN = 18

LOG_FILE = "water_usage.log"
```

```
# Initialize GPIO settings
GPIO.setmode(GPIO.BCM)
GPIO.setup(FLOW_SENSOR_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(SOLENOID_PIN, GPIO.OUT)

# Global variables
total_water_usage = 0.0

# Functions
def log_water_usage(timestamp, usage):
    with open(LOG_FILE, "a") as log:
        log.write(f"{timestamp}: {usage} liters\n")

def get_water_usage(flow_sensor_pin, pulse_frequency=450):
    global total_water_usage
    pulse_count = 0
    last_pulse_time = 0

    def count_pulse(channel):
        nonlocal pulse_count, last_pulse_time
        pulse_count += 1
        last_pulse_time = time.time()
```

```
    GPIO.add_event_detect(flow_sensor_pin, GPIO.FALLING, callback=count_pulse)

    while True:
        try:
            time_elapsed = time.time() - last_pulse_time
            if time_elapsed >= 1:
                flow_rate = pulse_count / pulse_frequency
                total_water_usage += flow_rate
                log_water_usage(datetime.now(), total_water_usage)
                pulse_count = 0
        except KeyboardInterrupt:
            GPIO.cleanup()
            break

def main():
    try:
        print("Smart Water System is running.")
        get_water_usage(FLOW_SENSOR_PIN)
    except KeyboardInterrupt:
        print("Smart Water System stopped.")

if __name__ == "__main__":
    main()
```