Smart Water Management

IoT with IBM GROUP 2

ABSTRACT:

The Smart Water Management System is a comprehensive, technology-driven solution designed to enhance the efficiency and sustainability of water resource management. This system integrates advanced sensors, Internet of Things (IoT) devices, data analytics, and user-friendly interfaces to monitor, analyze, and optimize water usage, distribution, and quality in a wide range of applications, including municipal water supplies, agriculture, industrial processes, and environmental conservation.

OBJECTIVES:

The primary objectives of a smart water management system are to optimize water distribution, reduce waste, ensure water quality and safety, minimize operational costs, and enhance sustainability through data-driven decision-making and efficient resource utilization.

1. **Efficient Distribution:** Optimize the supply and distribution of water resources, minimizing wastage and enhancing equitable access.
2. **Real-time Monitoring:** Continuously monitor water quality, quantity, and infrastructure health to ensure a reliable and safe water supply.
3. **Leak Detection:** Promptly identify and locate water leaks, reducing losses and preventing infrastructure damage.
4. **Water Quality Assurance:** Maintain water quality standards, complying with regulations and ensuring the delivery of safe drinking water.
5. **Non-Revenue Water Reduction:** Minimize non-revenue water, including losses and unbilled consumption, for improved financial sustainability.
6. **Emergency Response:** Enhance the system's ability to respond effectively to water-related emergencies, including disasters and supply disruptions.

IOT SENSOR SETUP:

The IoT sensor setup includes a water flow sensor connected to a microcontroller (e.g., Arduino or Raspberry Pi). This sensor is installed at a suitable point in the water supply line, typically near the water meter. The sensor measures the flow rate and sends data to the Raspberry Pi for processing. The setup could also include additional sensors for temperature and humidity to gather environmental data.

1. **Water Quality Sensors**:pH Sensors: Measure the acidity or alkalinity of water, important for assessing water quality.
2. **Ultrasonic Flow Sensors:** Use ultrasonic waves to measure the flow rate of water in pipes or channels.alkalinity of water, important for assessing water quality.
3. **Leak Detection Sensors:**Acoustic Sensors: Listen for the sounds of water leaks, making them useful for identifying underground leaks.
4. GPS Sensors:GPS sensors can be integrated to monitor the location and movement of water-related assets or monitoring stations.
5. Water Metering Sensors:IoT-enabled water meters track water consumption in real-time, enabling accurate billing and leak detection.

MOBILE APP DEVELOPEMENT :

The mobile app is the user interface for the system. It allows users to:

- View real-time water consumption data.
- Set daily, weekly, or monthly water usage targets.
- Receive notifications and alerts when approaching or exceeding these targets.
- Access historical water consumption data in the form of graphs and reports.
- The app can be developed for iOS and Android platforms using appropriate frameworks (e.g., React Native).

Rasberry pi integration :-

1.Define Integration Goals: Clearly outline how Raspberry Pi will be used in the smart water management system. Determine its role, such as data collection, processing, or control.

2.Raspberry Pi Selection: Selecting the right Raspberry Pi model for a smart water management system depends on your specific project requirements, including the number of sensors, data processing needs, and whether you need advanced features like wireless connectivity or GPIO pins for sensor integration. Such as compute power,number of sensors,operating system,environmental conditions.

3.Choose the appropriate Raspberry Pi model for your specific requirements. Consider factors like processing power, connectivity options, and available GPIO pins.

4.Sensors and Hardware: Select and connect the necessary sensors and hardware to the Raspberry Pi. This may include pH sensors, flow sensors, cameras, and more. Ensure compatibility and proper wiring.

5.Data Collection: Develop software to collect data from connected sensors. Implement appropriate interfaces, drivers, or protocols for data acquisition.

6.Data Processing:Use Raspberry Pi to process and analyze collected data. Implement algorithms for real-time analysis or data storage.

7.Data Storage: Store data locally on the Raspberry Pi or integrate with a cloud-based storage solution. Consider data retention policies and backup strategies.

8.Communication: Set up communication protocols to transmit data to a central server or database. Common options include Wi-Fi, Ethernet, or cellular connectivity.

9.Control Mechanisms: Use Raspberry Pi to control water management processes, such as regulating flow, activating valves, or triggering alarms based on sensor data.

10.Remote Access:Enable remote access to the Raspberry Pi for monitoring, maintenance, and updates. Implement secure access controls to protect the system.

11.User Interface:Develop a user-friendly interface, which can be a web-based dashboard or a mobile app, to visualize data and control parameters.

12.Energy Management: Optimize power management to ensure Raspberry Pi operates efficiently, especially in remote or off-grid locations.

13.Security and Access Control: Implement security measures to safeguard the Raspberry Pi and the data it handles. Use encryption and access controls to prevent unauthorized access.

Code Implementation:

- The code for this project involves programming the microcontroller (Arduino or Raspberry Pi) to collect and transmit sensor data. It also includes the development of the mobile app and the code for the Raspberry Pi to process and manage data.
- Arduino/Raspberry Pi Code: Involves reading data from sensors, processing it, and transmitting it to the Raspberry Pi or a cloud service.
- Mobile App Code: Develop a user-friendly app that communicates with the Raspberry Pi or cloud service to fetch and display real-time and historical data, set targets, and receive alerts.

MQTT server code implementation:-

```
import paho.mqtt.client as mqtt

import time

import json
```

```
# Define the MQTT broker and topic
mqtt_broker = "afe474acb580460bb5c285743330aa90.s2.eu.hivemq.cloud"
mqtt_topic = "afe474acb580460bb5c285743330aa90.s2.eu.hivemq.cloud"
username="sathishkumar020"
password="sathish@106"
port=8883
# Simulated water usage data
water_data = {
    "location": "Sensor A",
    "flow_rate": 5.3,
    "pressure": 30,
    "timestamp": int(time.time())
}
# Callback when the client connects to the MQTT broker
def on_connect(client, userdata, flags, rc):
    print("Connected to MQTT broker with result code " + str(rc))
# Initialize the MQTT client
client = mqtt.Client()
client.on_connect = on_connect
# Connect to the MQTT broker
client.connect(mqtt_broker, 8883, 60)
while True:
    # Simulate collecting real-time water usage data
    # Replace this with actual data collection from water sensors
    water_data["flow_rate"] = water_data["flow_rate"] + 0.2
    water_data["timestamp"] = int(time.time())
# Publish the water usage data to the MQTT topic
```

```
  client.publish(mqtt_topic, json.dumps(water_data))
    print(f"Published: {water_data}")
  # Adjust the time interval for data updates as needed
    time.sleep(5)  # Update data every 10 seconds
# Keep the script running
client.loop_forever()
```

output :-

Published: {'location': 'Sensor A', 'flow_rate': 5.5, 'pressure': 30, 'timestamp': 1698759574}

Published: {'location': 'Sensor A', 'flow_rate': 5.7, 'pressure': 30, 'timestamp': 1698759579}

Published: {'location': 'Sensor A', 'flow_rate': 5.9, 'pressure': 30, 'timestamp': 1698759584}

Published: {'location': 'Sensor A', 'flow_rate': 6.1000000000000005, 'pressure': 30, 'timestamp': 1698759589}

Published: {'location': 'Sensor A', 'flow_rate': 6.300000000000001, 'pressure': 30, 'timestamp': 1698759594}

Published: {'location': 'Sensor A', 'flow_rate': 6.500000000000001, 'pressure': 30, 'timestamp': 1698759599}

Published: {'location': 'Sensor A', 'flow_rate': 6.700000000000001, 'pressure': 30, 'timestamp': 1698759604}

Promoting Water Conservation and Sustainable Practices:

This real-time water consumption monitoring system can promote water conservation and sustainable practices in several ways:

Awareness: By providing real-time data, users become more aware of their water consumption habits, encouraging them to make more conscious choices.

Setting Targets: Users can set water usage targets, helping them to establish goals for reducing consumption.

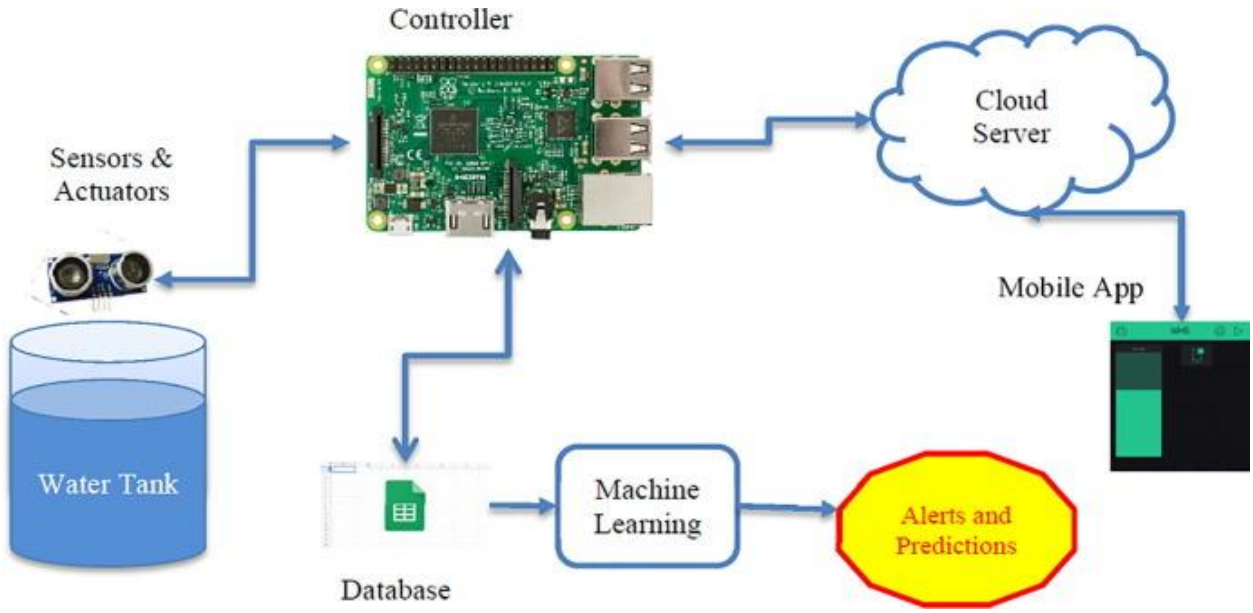
Alerts: The system notifies users when they approach or exceed their set limits, encouraging them to take immediate action to reduce water usage.

Historical Data: Historical data can be used to identify trends and patterns in water consumption, enabling users to make long-term adjustments.

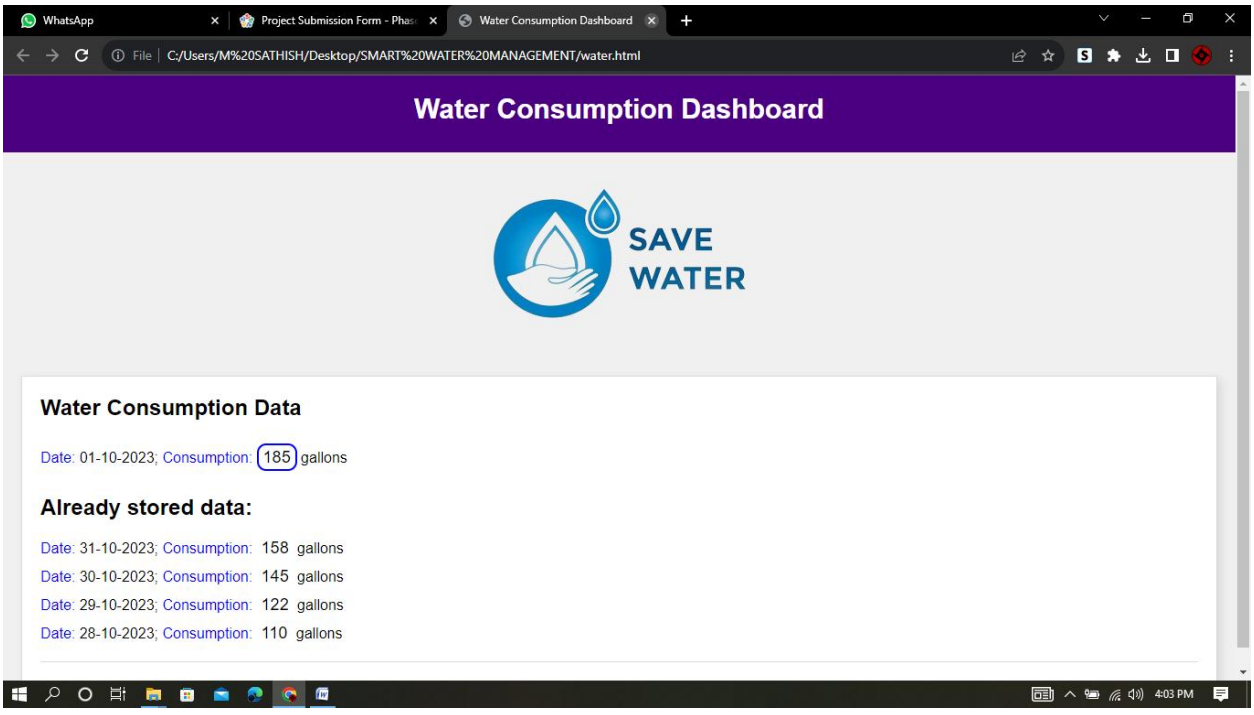
Data Sharing: Users can share their water usage data on social media, fostering a sense of competition and community around water conservation.

By combining IoT technology, mobile app development, and data analysis, this system empowers individuals to take a more active role in reducing water waste and contributing to a more sustainable future.

Diagram :-



Screen shot :-



Explain how the real-time water consumption monitoring system can promote water conservation and sustainable practices:

A real-time water consumption monitoring system can significantly promote water conservation and sustainable practices by providing users and water management authorities with the data and tools they need to make informed decisions and take proactive measures. Here's how it achieves this:

1. Real-time Data Access: Users and authorities can access up-to-the-minute data on water consumption, allowing them to track usage patterns and identify anomalies in real time. This immediate feedback enables timely intervention in case of excessive or wasteful water usage.

2. Leak Detection: The system can quickly detect leaks and abnormal water usage, enabling users to take prompt action to fix leaks and reduce water losses. This minimizes both water waste and the associated repair costs.

3. Usage Awareness: By providing consumers with real-time information about their water consumption, the system raises awareness and encourages responsible water usage. Users are more likely to reduce their water consumption when they have visibility into their usage patterns.

4. Alerts and Notifications: Users can set up alerts and notifications for unusual water usage, which can help identify leaks or inefficient practices. These alerts prompt users to investigate and address the issues promptly.

5. Behavioral Change:Real-time data and alerts can motivate individuals and organizations to adopt more water-efficient behaviors, such as fixing running faucets, using water-saving appliances, or adjusting irrigation schedules.

6. Resource Allocation: Water utilities and management authorities can use real-time consumption data to allocate resources more efficiently. They can redirect water to areas with higher demand or implement water rationing when necessary.

7. Conservation Programs: Water utilities can develop and implement conservation programs based on real-time data, offering incentives and guidelines to encourage water-saving practices among consumers and industries.

8. Load Balancing: Real-time data allows water utilities to balance the load on distribution systems, reducing the risk of system overloads and improving overall efficiency.

9. Environmental Impact: Sustainable water management practices can have positive environmental impacts. By reducing water consumption, there is less strain on water sources and ecosystems, contributing to ecological conservation.

10. Cost Savings: Efficient water usage and reduced losses translate to cost savings for both consumers and water utilities. Fewer resources are wasted, resulting in lower bills and operational expenses.

11. Sustainability Reporting: The system provides accurate and real-time data for sustainability reporting, helping organizations and communities measure their progress in conserving water resources.

12. Compliance with Regulations: Real-time monitoring ensures compliance with water conservation and environmental regulations by promptly identifying and addressing excessive water use or water quality issues.

13. Scalability: The real-time monitoring system can scale to accommodate growing water management needs and changing environmental conditions, ensuring long-term sustainability.

Programming code:-

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Water Consumption Dashboard</title>
  <style>
    * {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
  margin: 0;
}
```

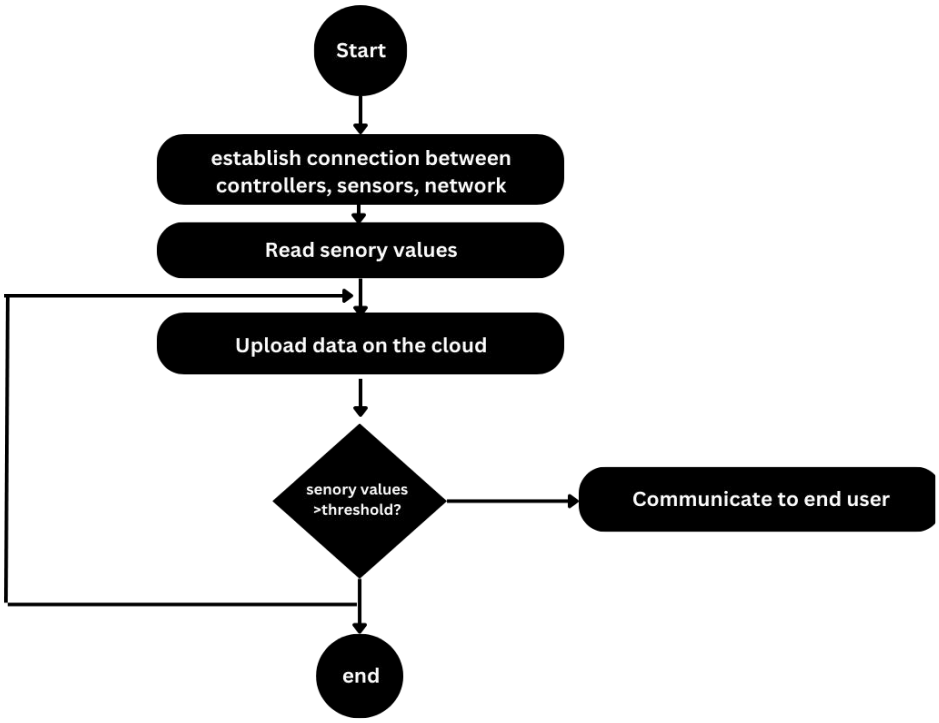
```
/* Style the header section with 3D effect */
header {
  /* background: linear-gradient(135deg, #004e92, #0077b6); */
  background-color:indigo;
  color: white;
  text-align: center;
  padding: 20px;
  position: relative;
  /* transform: perspective(50px) rotateX(2deg); */
}
/* Style the data and promotion sections with 3D effect */
#data, #promotion {
  margin: 20px;
  padding: 20px;
  background-color: #ffffff;
  border: 1px solid #ddd;
  box-shadow: 4px 4px 10px rgba(0, 0, 0, 0.1);
  /* transform: perspective(50px) rotateX(2deg); */
}
/* Style the h1 and h2 headings */
h1, h2 {
  font-size: 24px;
  margin-bottom: 10px;
}
/* Style the paragraphs within the sections */
p {
  font-size: 16px;
```

```
  margin: 10px 0;
}
/* Style the simulated data entries */
#data div {
  border-bottom: 1px solid #ddd;
  padding: 10px 0;
}
/* Style the list items in the promotion section */
#promotion li {
  list-style-type: disc;
  margin-left: 20px;
  font-size: 16px;
}
/* Add some spacing between section elements */
section {
  margin-top: 20px;
}
  </style>
</head>
<body>
  <header>
    <h1 style="font-size: 30px;" >Water Consumption Dashboard</h1>
  </header>
  <div style="display: flex; justify-content: center; ">
    <div>
       <img src="./water.png" alt="">
    </div>
```

```
</div>
<section id="data">
  <!-- Display water consumption data here -->
</section>
<!-- <section id="promotion">
  <h2 >Water Conservation Efforts</h2>
  <ul>
    <li>Fix any water leaks in your home promptly.</li>
    <li>Use low-flow faucets and showerheads to reduce water usage.</li>
    <li>Water your garden during the cooler parts of the day to minimize evaporation.</li>
  </ul>
</section> -->
<script>
  function getRandomNumber(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
  }
    function updateWaterData() {
    const waterData = [
      { date: '01-10-2023', consumption: getRandomNumber(150, 200) },
      // Add more data here
    ];
  const dataSection = document.getElementById('data');
    dataSection.innerHTML = '<h2>Water Consumption Data</h2>';
 for (const entry of waterData) {
      const div = document.createElement('div');
      div.innerHTML = `<p  ><span style="color:blue;">Date:</span> ${entry.date};
<span style="color:blue;">Consumption:</span> <span style="color:black;
```

```
font-size: large;padding: 3px 5px 3px; background-color: white; border: 2px blue solid;
border-radius: 10px;">${entry.consumption}</span> gallons</p>
  <h2 style="margin-top: 30px; margin-bottom: 20px">Already stored data:</h2>

      <p  ><span style="color:blue;">Date:</span> 31-10-2023; <span
style="color:blue;">Consumption:</span> <span style="color:black; font-size: large;padding:
3px 5px 3px; background-color: white; ">158</span> gallons</p>

      <p  ><span style="color:blue;">Date:</span> 30-10-2023; <span
style="color:blue;">Consumption:</span> <span style="color:black; font-size: large;padding:
3px 5px 3px; background-color: white; ">145</span> gallons</p>

      <p  ><span style="color:blue;">Date:</span> 29-10-2023; <span
style="color:blue;">Consumption:</span> <span style="color:black; font-size: large;padding:
3px 5px 3px; background-color: white; ">122</span> gallons</p>

      <p  ><span style="color:blue;">Date:</span> 28-10-2023; <span
style="color:blue;">Consumption:</span> <span style="color:black; font-size: large;padding:
3px 5px 3px; background-color: white; ">110</span> gallons</p> `;
      dataSection.appendChild(div);
    }
  }
// Update the data every 3 seconds
  updateWaterData(); // Initial data
  setInterval(updateWaterData, 3000); // Update every 3 seconds
 </script>
</body>
</html>
```

FLOW CHART:



Instructions:-

1. IoT Sensor Setup:

a. Choose the appropriate IoT water flow sensors, microcontrollers (e.g., Arduino, Raspberry Pi), and any additional environmental sensors (e.g., temperature, humidity).

b. Connect the water flow sensor and other sensors to the microcontroller. Ensure that you have power and ground connections for each sensor.

c. Write firmware for the microcontroller to read sensor data and transmit it to your Raspberry Pi. Use Python libraries or the Arduino IDE if you're using Arduino.

d. Install the necessary libraries for your sensors. For Raspberry Pi, you can use Python libraries like RPi.GPIO to interface with the GPIO pins.

2. Raspberry Pi Integration:

a. Set up your Raspberry Pi with Raspbian or a suitable operating system.

b. Write Python code to receive data from the IoT sensors connected to the microcontroller. Use serial communication (e.g., UART) or another suitable protocol to transfer data to the Raspberry Pi.

c. Process and store the received data on the Raspberry Pi. You may want to use a database like SQLite or set up a cloud service for long-term data storage.

3. Develop the Transit Information Platform:

a. Create a Python application or a web-based platform to serve as the transit information platform. This platform will display real-time and historical water consumption data.

b. Design the platform to allow users to set water usage targets, receive notifications, and access their data through a user-friendly interface.

c. Implement any features for data visualization, such as graphs and charts, to help users understand their water usage patterns.

4. Integration with Python:

a. Integrate the IoT sensor data with the Transit Information Platform using Python. You can use Python libraries like Flask or Django to create a web service that receives data from the Raspberry Pi.

b. Ensure that the platform can request and display real-time data as well as historical usage data from the Raspberry Pi.

c. Implement functionality for setting water usage targets and generating alerts when targets are met or exceeded.

5. Deployment:

a. Deploy the Raspberry Pi and the IoT sensor setup at a suitable location near the water supply.

b. Host the Transit Information Platform on a web server or a cloud platform. Ensure that it's accessible from mobile devices.

c. Secure the system with appropriate access controls and authentication mechanisms to protect user data.

6. Testing and Monitoring:

a. Thoroughly test the entire system to ensure data accuracy, real-time monitoring, and the functionality of the Transit Information Platform.

b. Implement monitoring and error handling to detect and address issues in real-time.

7. Promote Water Conservation:

a. Educate users on the benefits of the system and how to use it effectively for water conservation.

b. Provide incentives and rewards for users who meet or exceed their water usage targets.