Smart water

Objective :-

Develop a real-time water consumption monitoring system to help users track and manage their water usage.

Promote water conservation and sustainable practices by raising awareness of water consumption patterns.

Enable users to set water usage targets and receive alerts when they approach or exceed them.

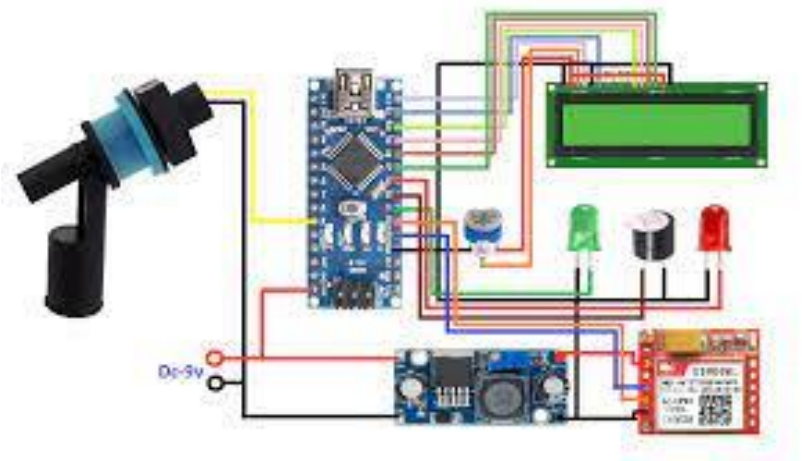
Collect and store data for historical analysis and reporting.

Create an intuitive mobile app for user interaction.

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.

Here's a simplified diagram of the IoT sensor setup:



Mobile app develop :-

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

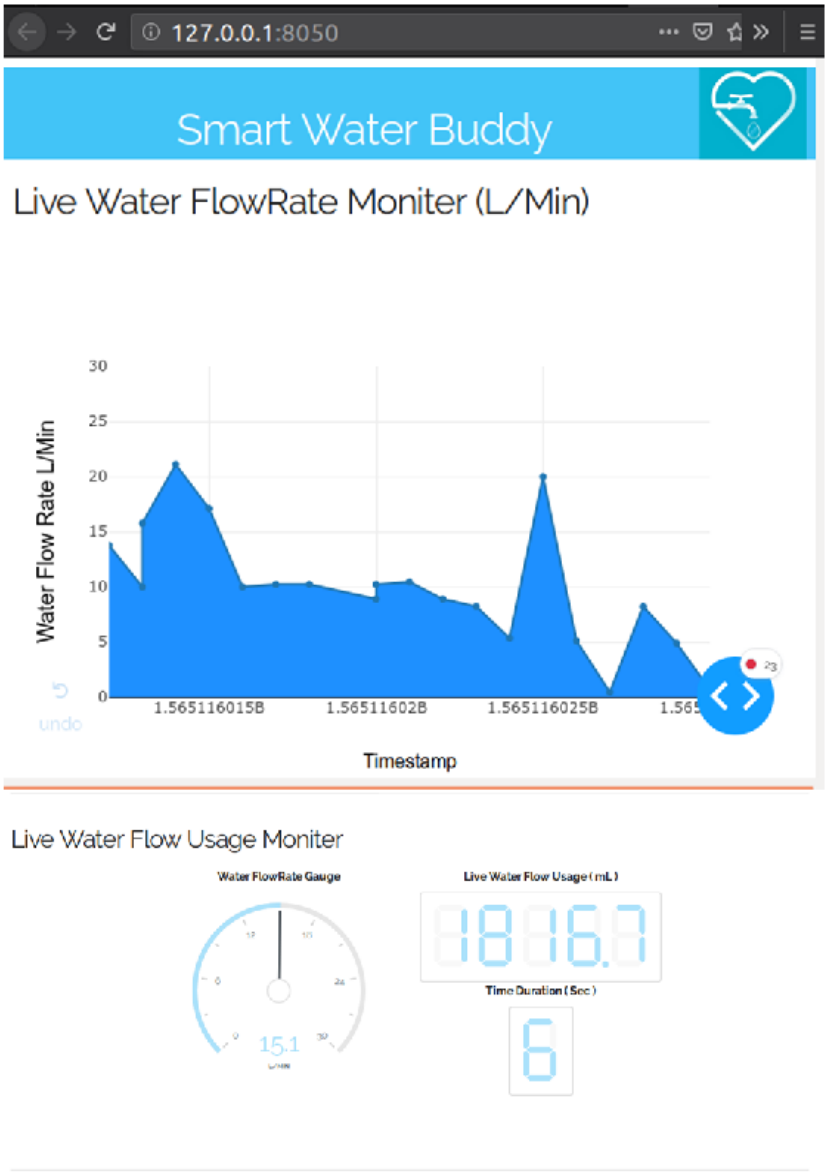
View real-time water consumption data.

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, Flutter).



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:

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.

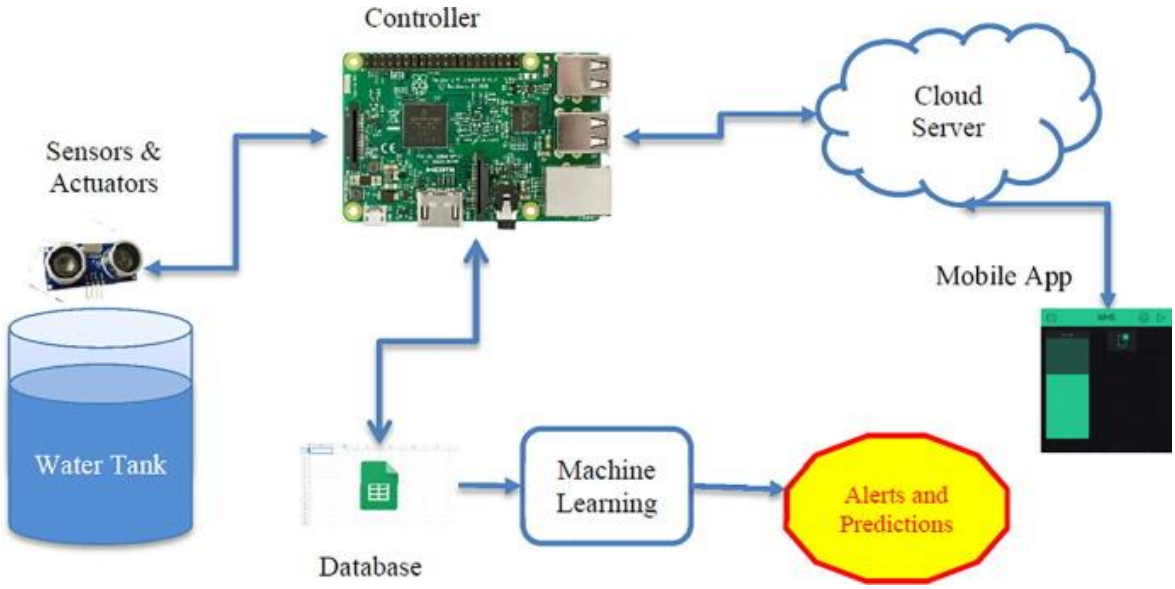
Alerts: The system notifies users when they approach or exceed their set limits, encouraging them to take immediate action to reduce water usage.

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

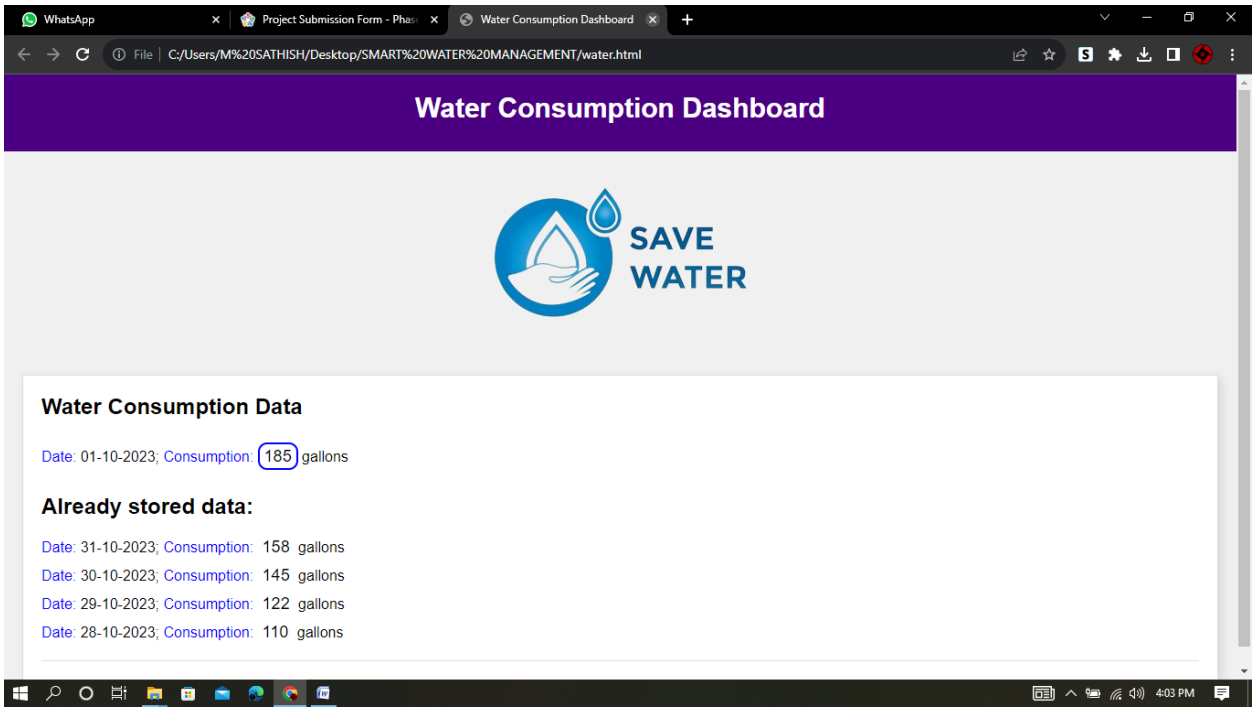
Data Sharing: Users can share their water usage data on social media, fostering a sense of competition and community around water conservation.

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 :-



Explanation :-

A real-time water consumption monitoring system plays a crucial role in promoting water conservation and sustainable practices by providing accurate and up-to-the-minute data on water usage. This technology leverages various sensors, meters, and data analytics to track water consumption at residential, commercial, industrial, or municipal levels. Here's how such a system can contribute to water conservation and sustainable practices:

1.Awareness and Transparency:

Real-time monitoring systems make water consumption data easily accessible to consumers, businesses, and utilities. This transparency helps individuals and organizations become more aware of their water usage and can lead to more responsible water management.

2.Leak Detection and Early Warning:

Real-time monitoring can quickly detect leaks, no matter how small. By alerting users to these issues in real-time, water waste can be minimized, and costly repairs can be prevented.

3.Behavior Modification:

When individuals have access to real-time water consumption data, they can adjust their behavior in response to this information. For instance, homeowners may reduce shower times or choose water-efficient appliances. In commercial and industrial settings, businesses can optimize processes to reduce water usage.

4.Incentives and Gamification:

Many real-time monitoring systems incorporate gamification and incentive features to encourage water conservation. Users may receive rewards or compete with others to reduce their water consumption, creating a sense of competition and motivation for water conservation.

5.Remote Control and Automation:

Some systems enable users to remotely control water usage by managing devices such as irrigation systems, faucets, or showers. This allows for more precise and efficient water use, especially in cases where users can turn off or adjust water usage from a mobile app.

6.Resource Allocation and Planning:

Water utilities and municipalities can benefit from real-time data to allocate resources more effectively. They can identify areas of high demand, address leaks, and plan infrastructure improvements based on real-time consumption patterns, ensuring a sustainable and efficient water supply.

7.Data Analytics and Predictive Maintenance:

Real-time monitoring systems often include data analytics that can identify consumption trends and anomalies. This data can be used for predictive maintenance of water infrastructure, optimizing water distribution, and reducing water loss.

8.Water Billing and Pricing:

Utilities can implement more sophisticated and equitable water billing based on real-time data. They may introduce tiered pricing to discourage excessive use and promote conservation.

9.Sustainable Practices:

By reducing water consumption, real-time monitoring systems contribute to sustainable water management practices, which are vital for preserving this finite resource. Sustainable practices can include the use of water-efficient appliances, rainwater harvesting, and wastewater recycling.

10.Environmental Impact:

Conserving water not only saves money but also has a positive impact on the environment. Reducing water usage decreases energy consumption (for water heating and distribution), lowers the strain on water sources, and minimizes wastewater treatment costs.

Real – time data display platform:-

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>
```

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.

c. Encourage users to share their achievements and water-saving tips through social media and community engagement.

Conclusion:-

The project involves multiple stages, including IoT sensor setup, Raspberry Pi integration, mobile app development, and Python-based data processing and visualization. These steps collectively create a comprehensive solution for monitoring, managing, and conserving water resources.