THEME : SMART WATER MANAGEMENT

SUBMITTED BY : **A.M.SURIYA**
E.GOPIKRISHNAN
A.THOLKAPPIYAN
P.VAITHEESHWARAN

Name of Institution : Ganesh college of Engineering

Address of Institution : Ganesh college of Engineering,
Attur main road,
Mettupatti (P.O), Valapady (T.K),
Salem (D.T), 636 111.

District : Salem State : Tamilnadu Pin : 636 111

Smart Water Management Phase-4

Certainly, when working on a student IoT Smart Water Management project, web technologies can play a crucial role in data visualization, remote monitoring, and user interfaces. Here's how you can incorporate web technologies into the project:

1. Data Collection and IoT Devices:

Set up IoT devices (sensors, flow meters) to monitor water parameters such as water level, quality, and flow rate.

Ensure these devices are capable of transmitting data to a central server.

2. Data Processing and Analysis:

Develop software for data processing, analytics, and decision-making algorithms on the server.

Use Python, Node.js, or other server-side languages and libraries to handle data efficiently.

3. Web Development:

Create a web-based platform to monitor and manage the Smart Water Management system.

4. User Interfaces:

Design user-friendly web interfaces for different stakeholders, including administrators, maintenance personnel, and end-users.

Implement responsive web design to ensure usability on various devices.

5. Data Visualization:

Use web-based data visualization libraries such as D3.js, Chart.js, or Plotly to display real-time and historical data through interactive charts and graphs.

Display water quality, consumption trends, and equipment health.

6. Alerts and Notifications:

Set up alert mechanisms to notify stakeholders via web notifications or email about critical water parameters, equipment issues, or consumption anomalies.

7. Remote Monitoring:

Implement remote monitoring of the water management system through a web dashboard, enabling stakeholders to check system status and make adjustments as needed.

8. User Authentication and Security:

Implement secure user authentication mechanisms to control access to system data.

Use HTTPS for secure data transmission and ensure that user data remains private.

9. Database Management:

Set up a database system (e.g., MySQL, MongoDB) to store historical data for analysis and reporting.

10. Mobile App Integration :

Develop mobile applications for Android and iOS platforms to enable on-the-go monitoring and control of the Smart Water Management system.

- Utilize web technologies, like React Native or Flutter, to build cross-platform mobile apps.

11. Testing and Quality Assurance:

Thoroughly test the web platform to ensure it functions correctly, is responsive, and is free from vulnerabilities.

Python Program for Connecting mobile app with Smart water management IOT project:

```
import 'package:flutter/material.dart';

import 'package:http/http.dart' as http;

import 'dart:convert';


void main() => runApp(MyApp());


class MyApp extends StatelessWidget {

 @override

 Widget build(BuildContext context) {

  return MaterialApp(

   home: WaterData(),

  );

 }

}

class WaterData extends StatefulWidget {
```

```
  @override
  _WaterDataState createState() => _WaterDataState();
}
class _WaterDataState extends State<WaterData> {
  String waterData = "";
  Future<void> fetchWaterData() async {
    final response = await http.get('http://your-python-server-url/api/water-data');
    if (response.statusCode == 200) {
      setState(() {
        waterData = json.decode(response.body).toString();
      });
    }
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Water Data Monitoring'),
      ),
      body: Center(
        child: Column(
          children: <Widget>[
            ElevatedButton(
              onPressed: fetchWaterData,
              child: Text('Refresh Water Data'),
            ),
            Text(waterData),
          ],
        ),
      ),
    );
  }
}
```

Testing and Debugging:

Thoroughly test the app to ensure that it correctly sends requests, handles responses, and updates the user interface.

Debug any issues with connectivity, data retrieval, or user interface updates.

Security Considerations:

Ensure that your server and app are secure. Use HTTPS to encrypt data transmission.

Implement authentication and authorization to restrict access to the app's features and data.

Deployment:

Deploy your mobile app to Android and iOS app stores or distribute it through enterprise channels, depending on your target audience.

Remember that real-world applications are typically more complex than this simplified example. They might involve multiple screens, data visualizations, and additional features. You should also consider scalability, error handling, and data synchronization for a production-ready app.

Connecting Mobile app with Smart water Management IOT Project:

To connect a mobile app to your Smart Water Management IoT project, you need to establish communication between the mobile app and the server where your IoT devices send data. This involves sending HTTP requests from the app to the server, retrieving the data, and displaying it in the app's user interface. Here's a step-by-step guide on how to achieve this:

1.Set Up a Server:

Ensure your Smart Water Management IoT project has a server that can receive and process data from IoT devices.

Implement APIs on the server to provide data to the mobile app. These APIs should handle incoming HTTP requests and return data in a format that the mobile app can understand (usually JSON).

2.Mobile App Development:

Use a mobile app development framework like Flutter or React Native to create your mobile app. In this guide, I'll use Flutter.

Create the app's user interface and layout, including buttons and widgets to display data.

3.HTTP Requests from Mobile App:

Use the http package (or equivalent) in Flutter to send HTTP requests to the server. For example, you can use the http.get method to request data.

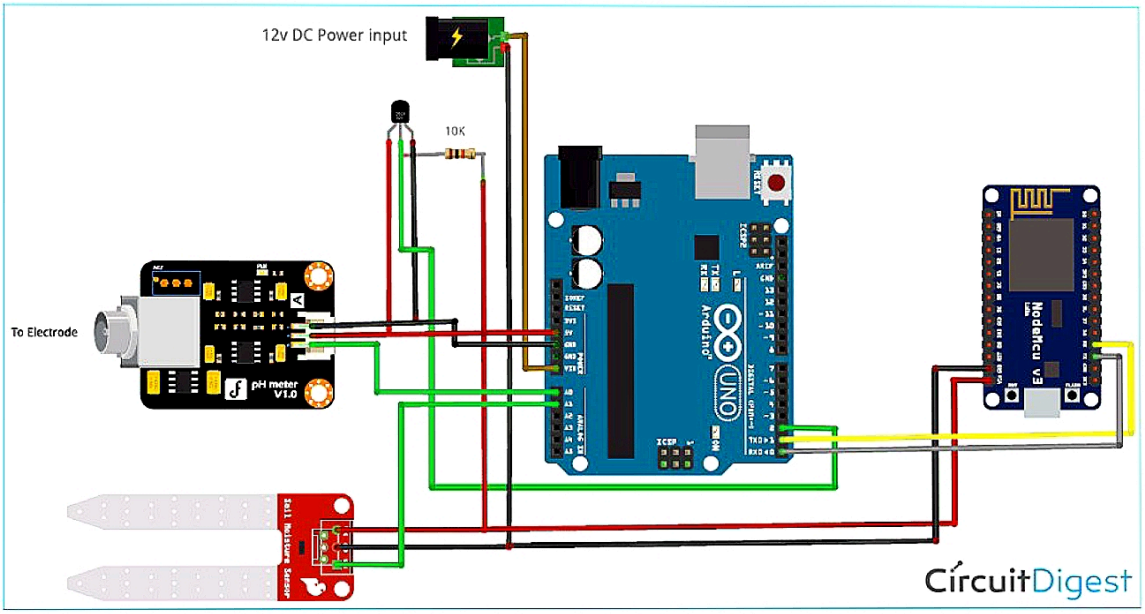
In the app, when a user interacts with a button or refresh action, send an HTTP request to the server.

4.Handling Server Responses:

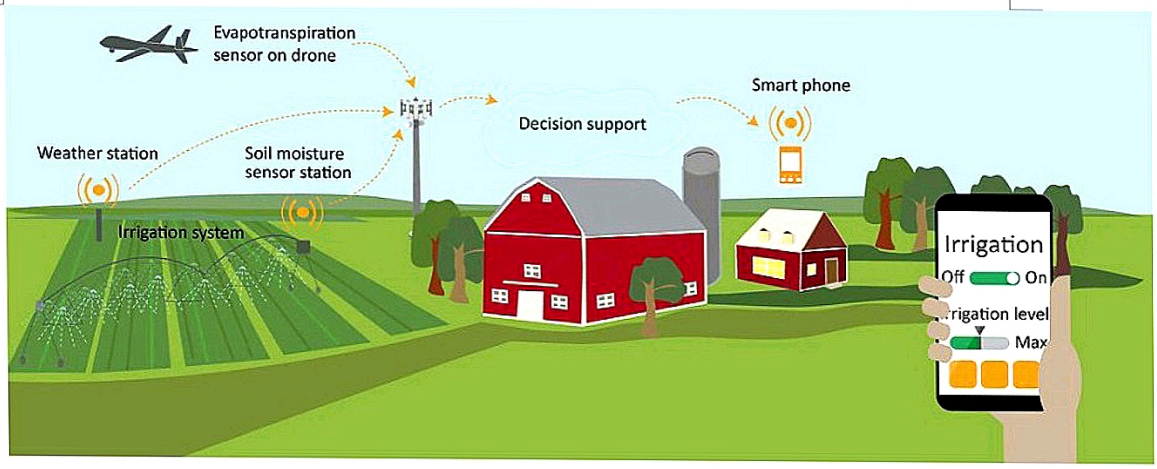
Once the server responds to the HTTP request, parse the JSON data received from the server. You can use the dart:convert library to decode JSON responses.

Update the app's user interface with the data received from the server.

Circiut diagram for Smart water management:



3-D Representation for Smart Water Management:


Evapotranspiration
sensor on drone
Smart phone
Decision support
Weather station
Soil moisture
sensor station
Irrigation system
Irrigation
Off
On
Max