Smart Water System Management

Project Objectives:

1. **Real-time Water Consumption Monitoring:** The project's core objective is to implement IoT sensors for continuous and real-time monitoring of water consumption in public areas.

2. **Public Awareness:** To raise public awareness about water conservation, the project aims to develop a user-friendly data-sharing platform that provides easy access to water consumption data.

3. **Water Conservation:** The collected data will be utilized to identify water wastage and encourage responsible water use among the public.

4. **Sustainable Resource Management:** The project will contribute to sustainable water resource management by collecting and analyzing data to inform long-term strategies and policies.

IOT Sensor Design:

1. **Types of Sensors:** The project will use specific IoT sensors like flow meters and pressure sensors to ensure accurate measurement of water consumption.

2. **Deployment Locations:** Sensor deployment will be strategically planned in public places such as parks and gardens to capture relevant data.

3. **Connectivity:** Appropriate communication protocols (e.g., Wi-Fi, cellular, LoRa) will be chosen to enable sensors to transmit data to the central platform.

4. **Power Source:** Power options, including batteries, solar, or wired connections, will be considered to ensure uninterrupted sensor operation.

Real-Time Transit Information Platform (Mobile App):

1. **User Interface:** A user-friendly mobile app interface will be designed to display real-time water consumption data in a visually appealing and understandable manner.

2. **Features:** The mobile app will include features like historical data access, notifications, and water-saving tips to engage users and raise awareness.

3. **Data Visualization:** Effective data visualization tools, such as charts, graphs, and maps, will be integrated into the app for data presentation.

4. **User Engagement:** Strategies will be developed to encourage user participation and promote water conservation practices through the app.

Integration Approach:

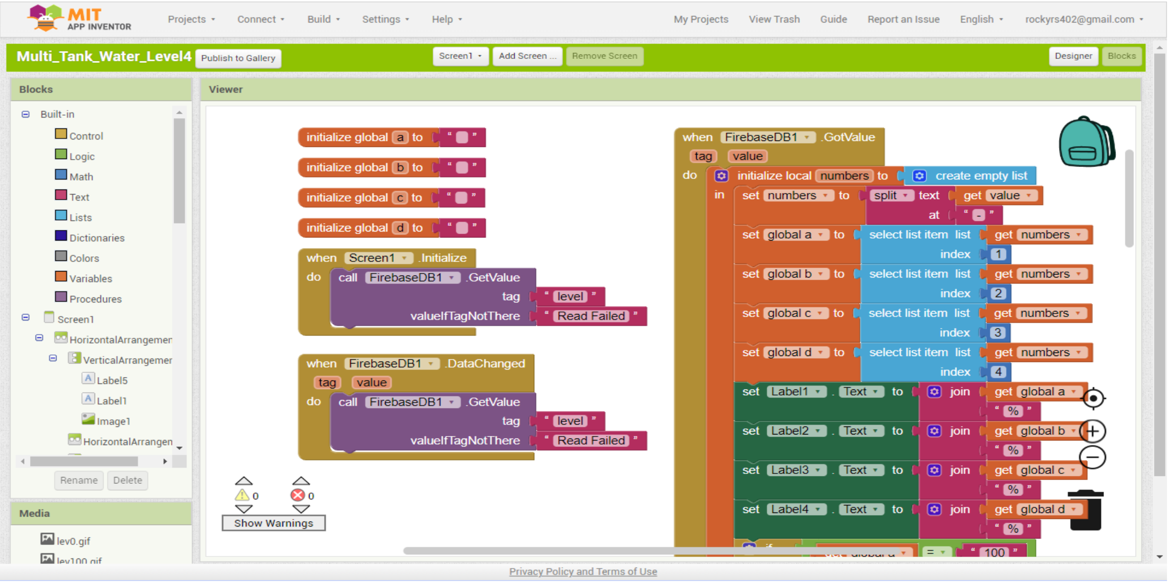
1. **Data Collection:** A clear process will be defined for IoT sensors to collect and transmit data to the data-sharing platform efficiently.

2. **Data Processing:** The project will outline how incoming data will be processed and analyzed to extract meaningful insights and identify water consumption patterns.

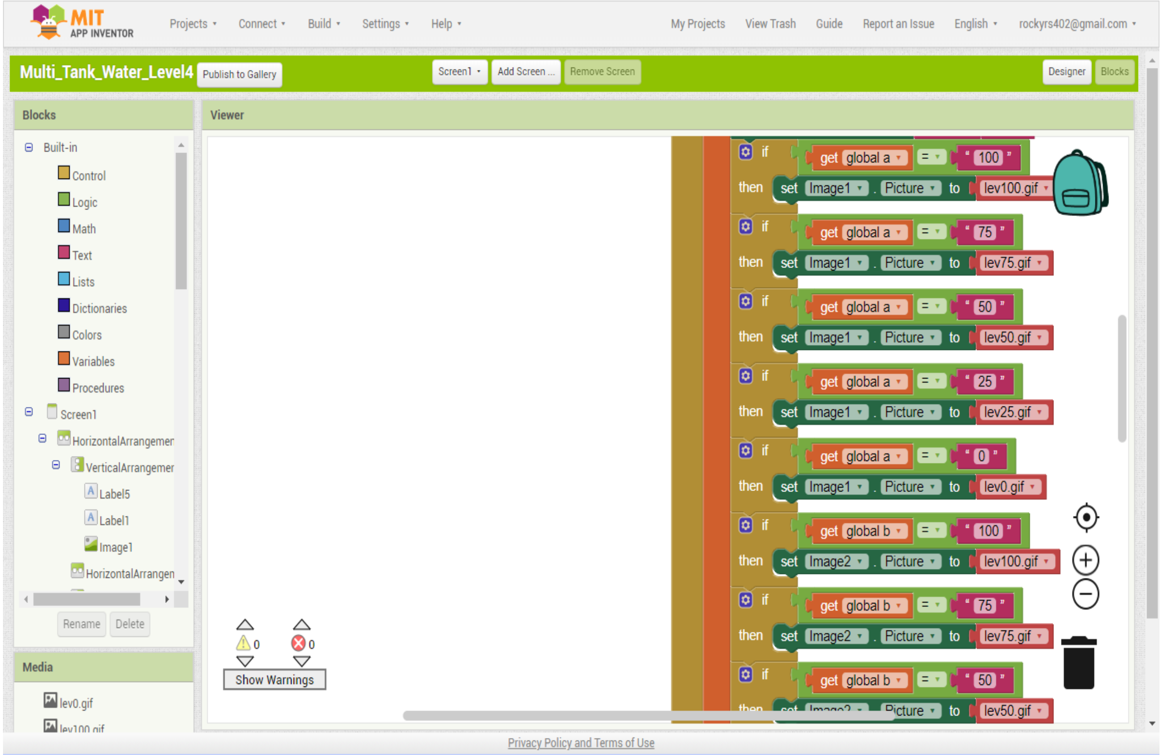
3. **Data-Sharing Platform:** The architecture of the data-sharing platform, including databases, APIs, and security measures, will be described and developed.

4. **IOT Technology and Python:** The project will utilize IoT technology and Python for sensor integration and data processing, ensuring efficient and reliable data transmission and analysis.

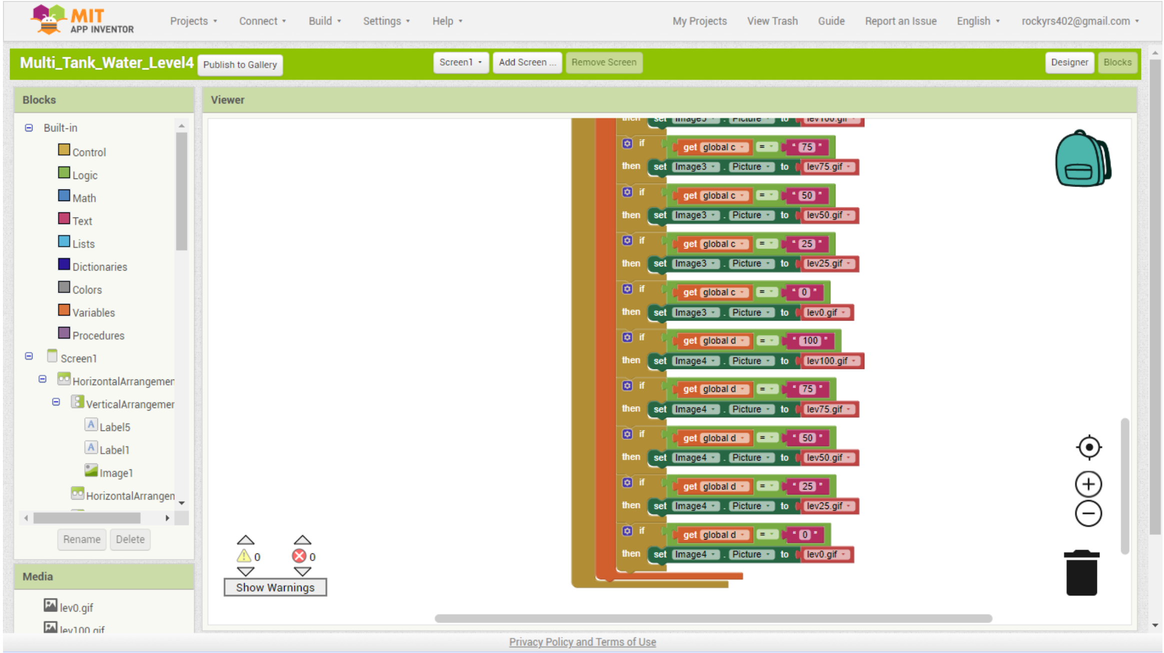
Mobile app development:



```
initialize global a to " "
initialize global b to " "
initialize global c to " "
initialize global d to " "

when Screen1 .Initialize
do  call FirebaseDB1 .GetValue
                         tag " level "
           valueIfTagNotThere " Read Failed "

when FirebaseDB1 .DataChanged
  tag  value
do  call FirebaseDB1 .GetValue
                         tag " level "
           valueIfTagNotThere " Read Failed "

when FirebaseDB1 .GotValue
  tag  value
do  initialize local numbers to  create empty list
    in  set numbers to  split text  get value
                                 at " - "
        set global a to  select list item  list  get numbers
                                          index  1
        set global b to  select list item  list  get numbers
                                          index  2
        set global c to  select list item  list  get numbers
                                          index  3
        set global d to  select list item  list  get numbers
                                          index  4
        set Label1 . Text to  join  get global a
                                    " % "
        set Label2 . Text to  join  get global b
                                    " % "
        set Label3 . Text to  join  get global c
                                    " % "
        set Label4 . Text to  join  get global d
                                    " % "
        if  get global a  =  " 100 "
```

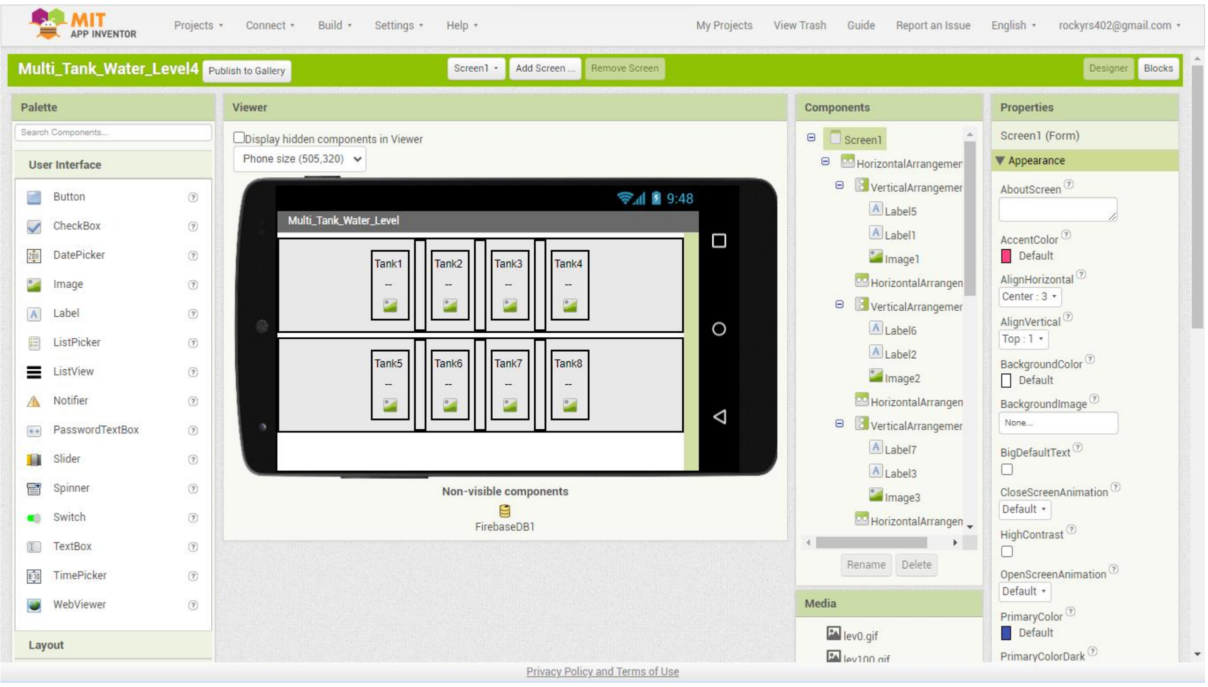


```
if   get global a = "100"
then set Image1 . Picture to lev100.gif
if   get global a = "75"
then set Image1 . Picture to lev75.gif
if   get global a = "50"
then set Image1 . Picture to lev50.gif
if   get global a = "25"
then set Image1 . Picture to lev25.gif
if   get global a = "0"
then set Image1 . Picture to lev0.gif
if   get global b = "100"
then set Image2 . Picture to lev100.gif
if   get global b = "75"
then set Image2 . Picture to lev75.gif
if   get global b = "50"
then set Image2 . Picture to lev50.gif
```



```
then set Image3 . Picture to lev100.gif
if   get global c = "75"
then set Image3 . Picture to lev75.gif
if   get global c = "50"
then set Image3 . Picture to lev50.gif
if   get global c = "25"
then set Image3 . Picture to lev25.gif
if   get global c = "0"
then set Image3 . Picture to lev0.gif
if   get global d = "100"
then set Image4 . Picture to lev100.gif
if   get global d = "75"
then set Image4 . Picture to lev75.gif
if   get global d = "50"
then set Image4 . Picture to lev50.gif
if   get global d = "25"
then set Image4 . Picture to lev25.gif
if   get global d = "0"
then set Image4 . Picture to lev0.gif
```

OUTPUT:



Multi_Tank_Water_Level4

□Display hidden components in Viewer

Phone size (505,320)

9:48

Multi_Tank_Water_Level

Tank1 -- Tank2 -- Tank3 -- Tank4 --

Tank5 -- Tank6 -- Tank7 -- Tank8 --

Non-visible components

FirebaseDB1

Screen1
HorizontalArrangemer
VerticalArrangemer
Label5
Label1
Image1
HorizontalArrangen
VerticalArrangemer
Label6
Label2
Image2
HorizontalArrangen
VerticalArrangemer
Label7
Label3
Image3
HorizontalArrangen

lev0.gif
lev100.gif

Screen1 (Form)

Appearance

AboutScreen

AccentColor: Default

AlignHorizontal: Center : 3

AlignVertical: Top : 1

BackgroundColor: Default

BackgroundImage: None...

BigDefaultText: □

CloseScreenAnimation: Default

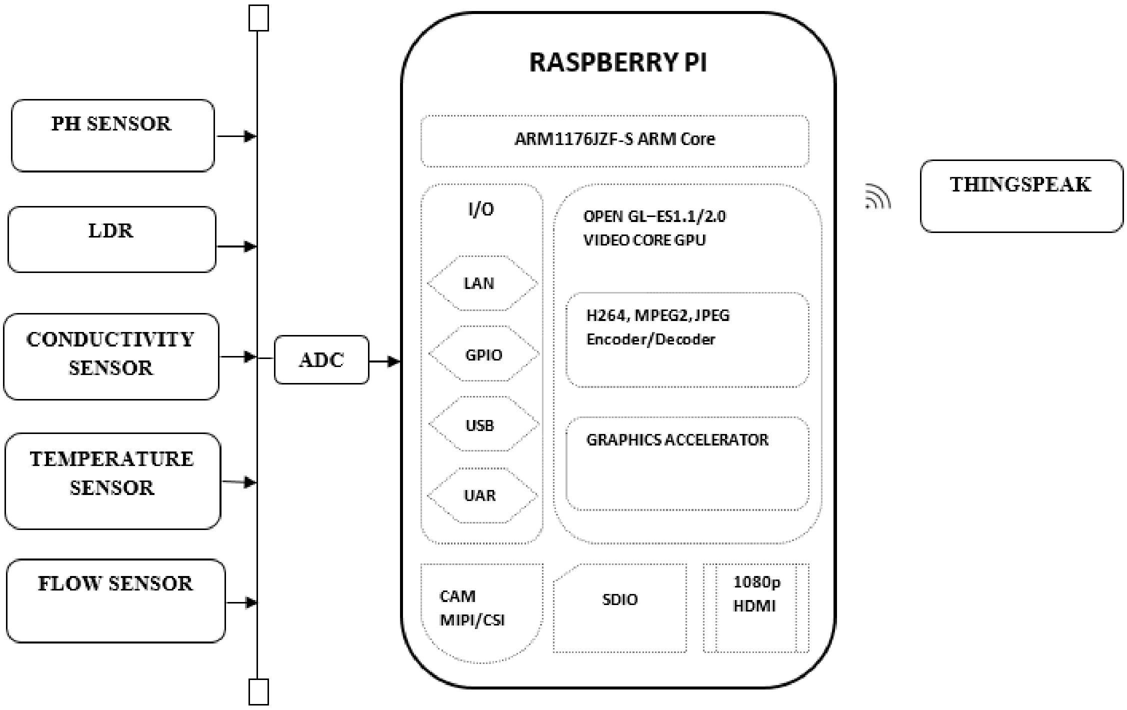
HighContrast: □

OpenScreenAnimation: Default

PrimaryColor: Default

PrimaryColorDark

Circuit diagram:



PROGRAM:

```
import RPi.GPIO as GPIO
import time
import requests

# Configure GPIO pin for the flow meter
FLOW_METER_PIN = 18
GPIO.setmode(GPIO.BCM)
GPIO.setup(FLOW_METER_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)

# Variables to keep track of water consumption
flow_rate = 0.0
total_flow = 0.0

# Data-sharing platform endpoint
DATA_PLATFORM_URL = "https://your-data-platform-url.com"

while True:
    try:
        pulse = GPIO.wait_for_edge(FLOW_METER_PIN, GPIO.FALLING, timeout=10)
        if pulse is not None:
            flow_rate += 1
            total_flow += 1

            # Send real-time data to the data-sharing platform
            data = {
                "flow_rate": flow_rate,
                "total_flow": total_flow
            }
            response = requests.post(DATA_PLATFORM_URL, json=data)

            # Handle the response if needed
            if response.status_code == 200:
                print("Data sent successfully")


    except KeyboardInterrupt:
        GPIO.cleanup()
        Break
```

WEB APPLICATION:

1. Head Section:

- The head section contains metadata and links to external resources.
    - Two **<title>** tags specify the page titles. You should have only one **<title>** tag.
    - **<link>** tags link to an external CSS file (style.css) and an icon (img1.img).
    - Bootstrap CSS and JavaScript are linked from a CDN for styling and interactivity.

2. JavaScript:

- The JavaScript section defines a function **updateData()** that simulates the update of sensor data (pH, conductivity, temperature) at regular intervals. This is for demonstration purposes and should be replaced with real data retrieval logic.
- The **setInterval** function calls **updateData()** every 5 seconds.
- There's also a function for displaying notifications, but it's not fully defined in the provided code.

3. CSS Styles:

- There are multiple CSS styles defined within **<style>** tags. These styles are used for styling various elements on the page. It's good practice to consolidate your CSS into one section.

4. Navbar:

- The **<nav>** element represents the navigation bar with links to different sections of the page (HOME, NOTIFICATION, REAL TIME DATA, USAGE HISTORY). The navigation bar is responsive, making it collapsible on smaller screens.

5. Header:

- The **<header>** contains the title "Smart Water System."

6. Main Content:

- The main content section contains three subsections:

- **Sensor Data:** Displays water level, temperature, and pH level.
- **Control Panel:** Provides buttons to start and stop the pump.
- **Notification and Alert Example:** Allows users to trigger and view different types of notifications (success, warning, error).

7. Real-Time Data Updates:

- Displays real-time sensor data for pH, conductivity, and temperature. However, the **id** attributes should be unique on the page, and it's recommended to rename them to avoid conflicts.

8. Usage History:

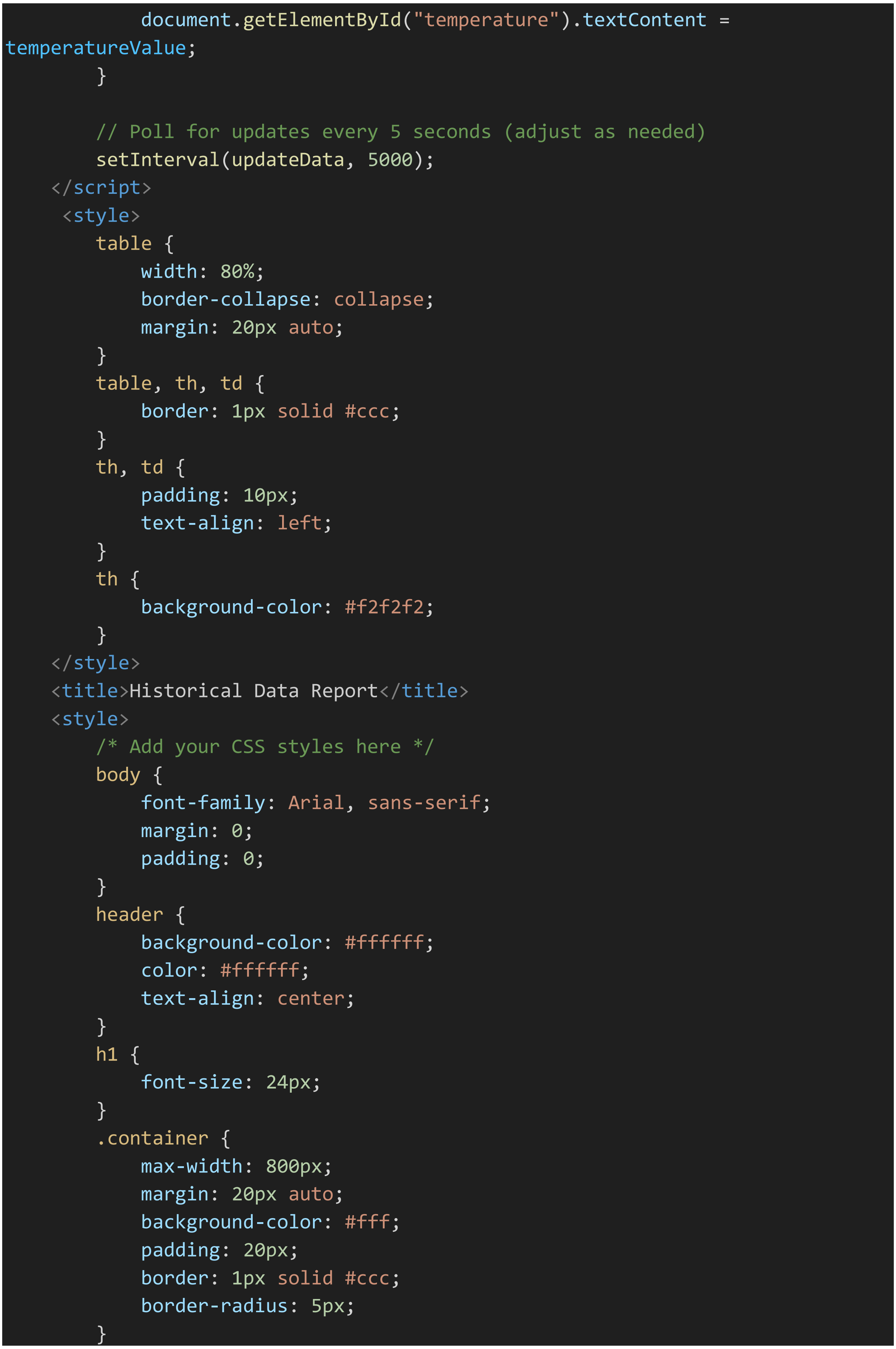
- Displays a table with historical water usage data, including date and usage in gallons.

9. Historical Data Report:

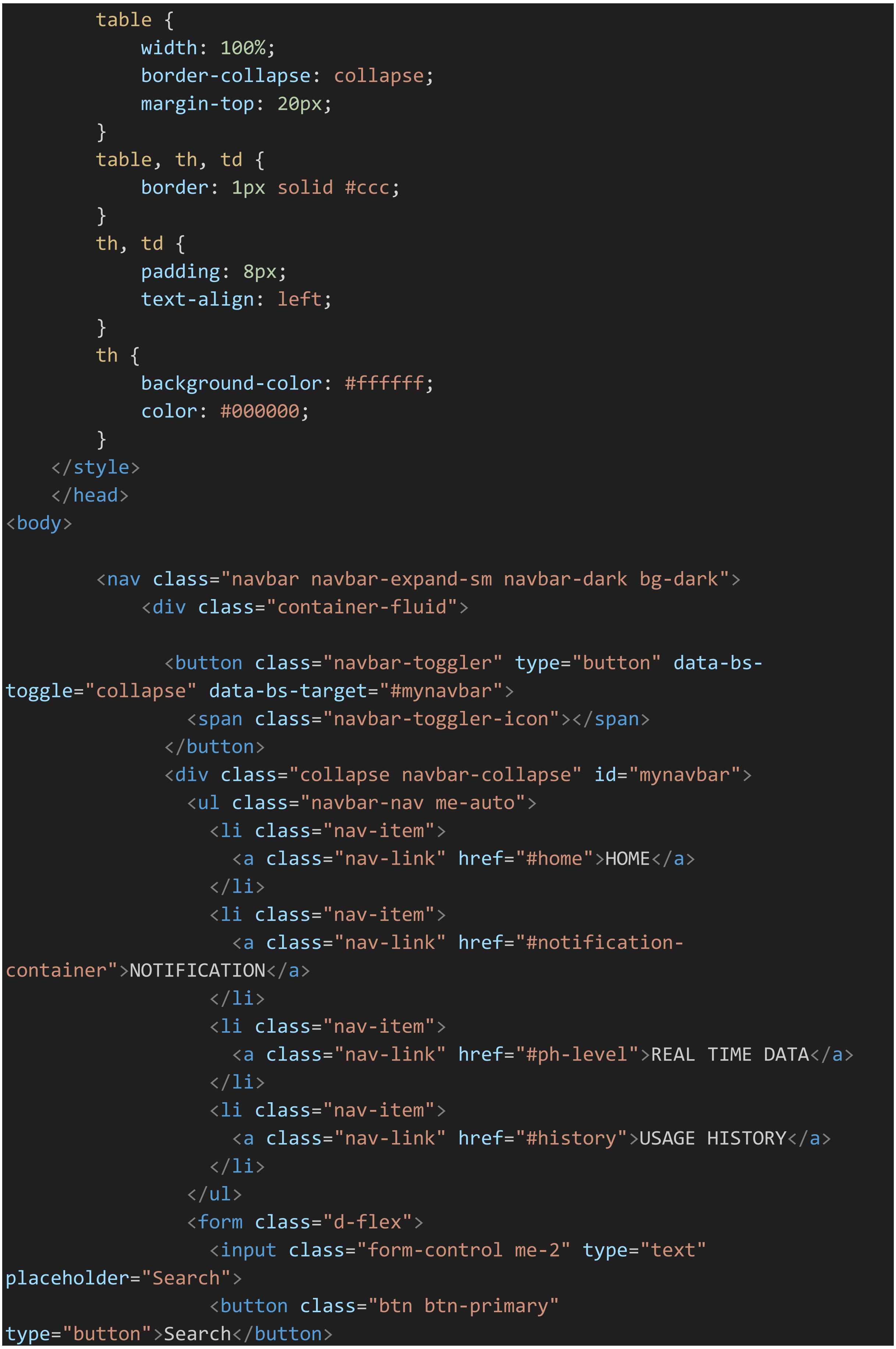
- Contains historical data presented in a table. The table includes time, water flow rate (GPM), and water quality (pH) data for a specific date.

Html Code : (title.html)

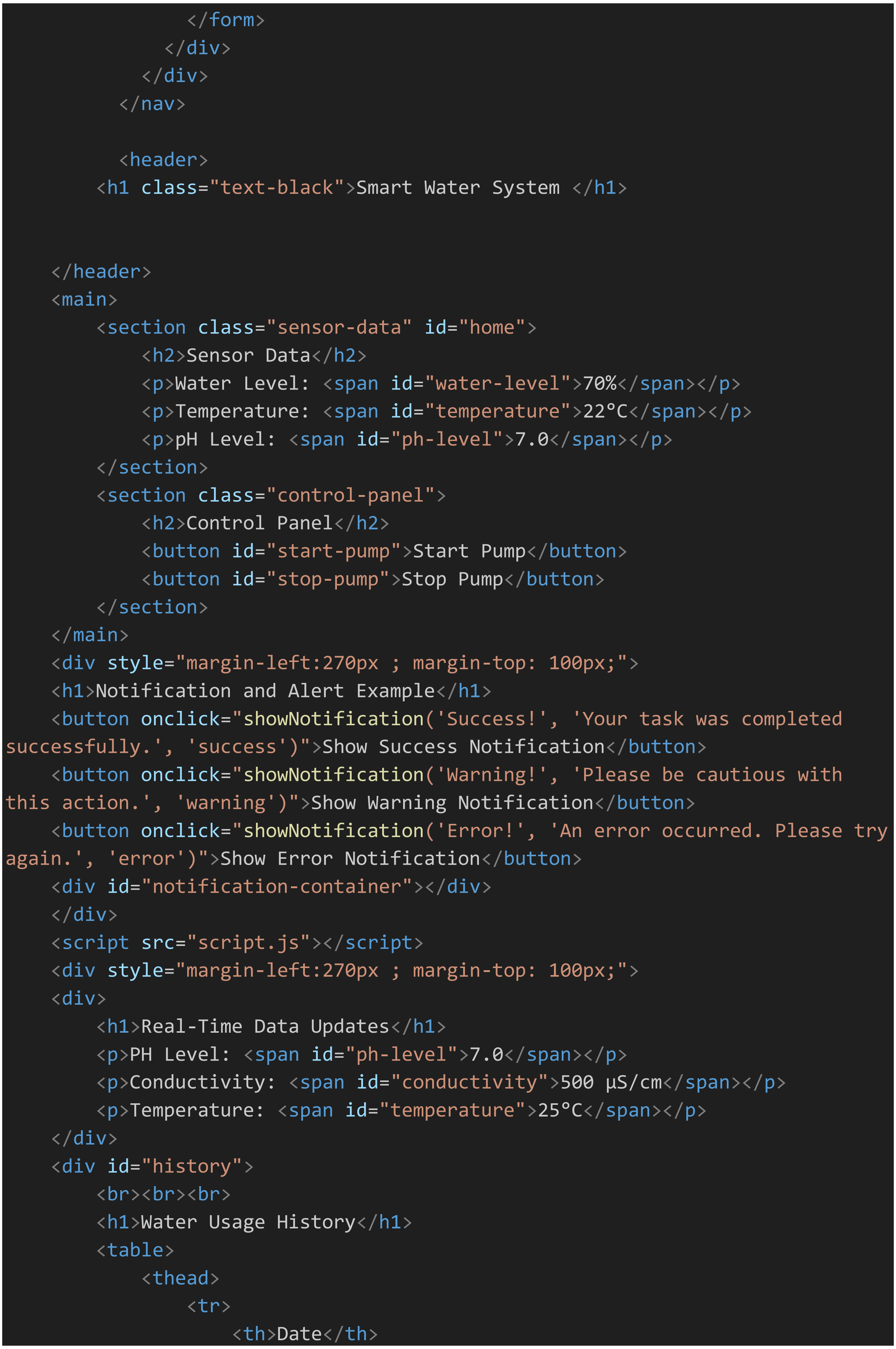
```
<!DOCTYPE html>
<html>
<head>
    <title>Smart Water System </title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <link rel="icon" href="./img1.img">
    <title>Notification and Alert Example</title>
    <link rel="stylesheet" type="text/css" href="style.css">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"
rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min
.js"></script>
    <script>
        // Function to update the data
        function updateData() {
            // Simulated data (replace with actual data retrieval logic)
            const phValue = (Math.random() * 14).toFixed(2);
            const conductivityValue = (Math.random() * 1000).toFixed(2);
            const temperatureValue = (Math.random() * 30).toFixed(2);

            // Update the HTML elements with the new data
            document.getElementById("ph-level").textContent = phValue;
            document.getElementById("conductivity").textContent =
conductivityValue;
```



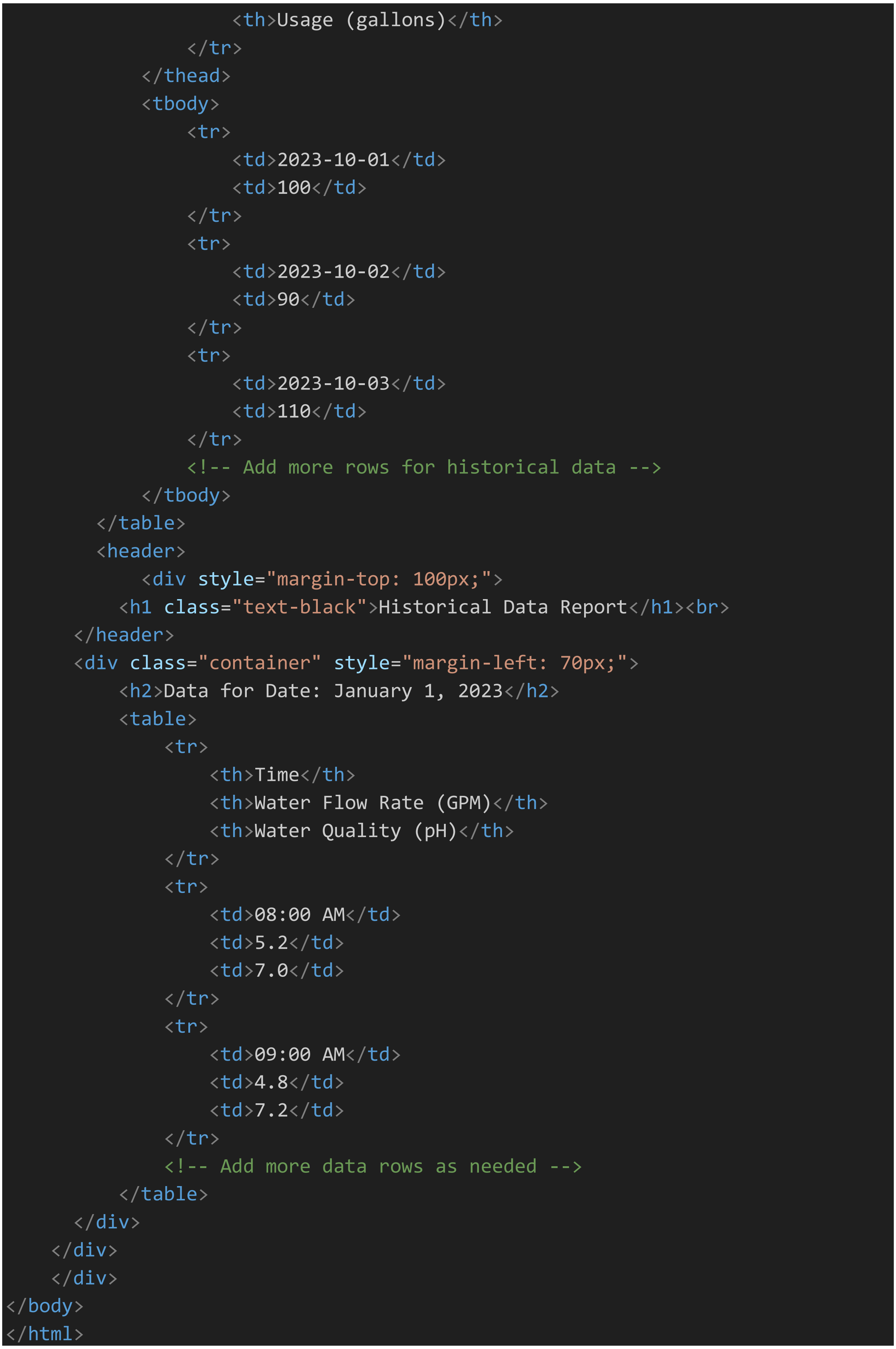
```
            document.getElementById("temperature").textContent =
temperatureValue;
        }

        // Poll for updates every 5 seconds (adjust as needed)
        setInterval(updateData, 5000);
    </script>
     <style>
        table {
            width: 80%;
            border-collapse: collapse;
            margin: 20px auto;
        }
        table, th, td {
            border: 1px solid #ccc;
        }
        th, td {
            padding: 10px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
    <title>Historical Data Report</title>
    <style>
        /* Add your CSS styles here */
        body {
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #ffffff;
            color: #ffffff;
            text-align: center;
        }
        h1 {
            font-size: 24px;
        }
        .container {
            max-width: 800px;
            margin: 20px auto;
            background-color: #fff;
            padding: 20px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
```



```
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        table, th, td {
            border: 1px solid #ccc;
        }
        th, td {
            padding: 8px;
            text-align: left;
        }
        th {
            background-color: #ffffff;
            color: #000000;
        }
    </style>
    </head>
<body>

        <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
            <div class="container-fluid">

              <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#mynavbar">
                <span class="navbar-toggler-icon"></span>
              </button>
              <div class="collapse navbar-collapse" id="mynavbar">
                <ul class="navbar-nav me-auto">
                  <li class="nav-item">
                    <a class="nav-link" href="#home">HOME</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="#notification-
container">NOTIFICATION</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="#ph-level">REAL TIME DATA</a>
                  </li>
                  <li class="nav-item">
                    <a class="nav-link" href="#history">USAGE HISTORY</a>
                  </li>
                </ul>
                <form class="d-flex">
                  <input class="form-control me-2" type="text"
placeholder="Search">
                  <button class="btn btn-primary"
type="button">Search</button>
```

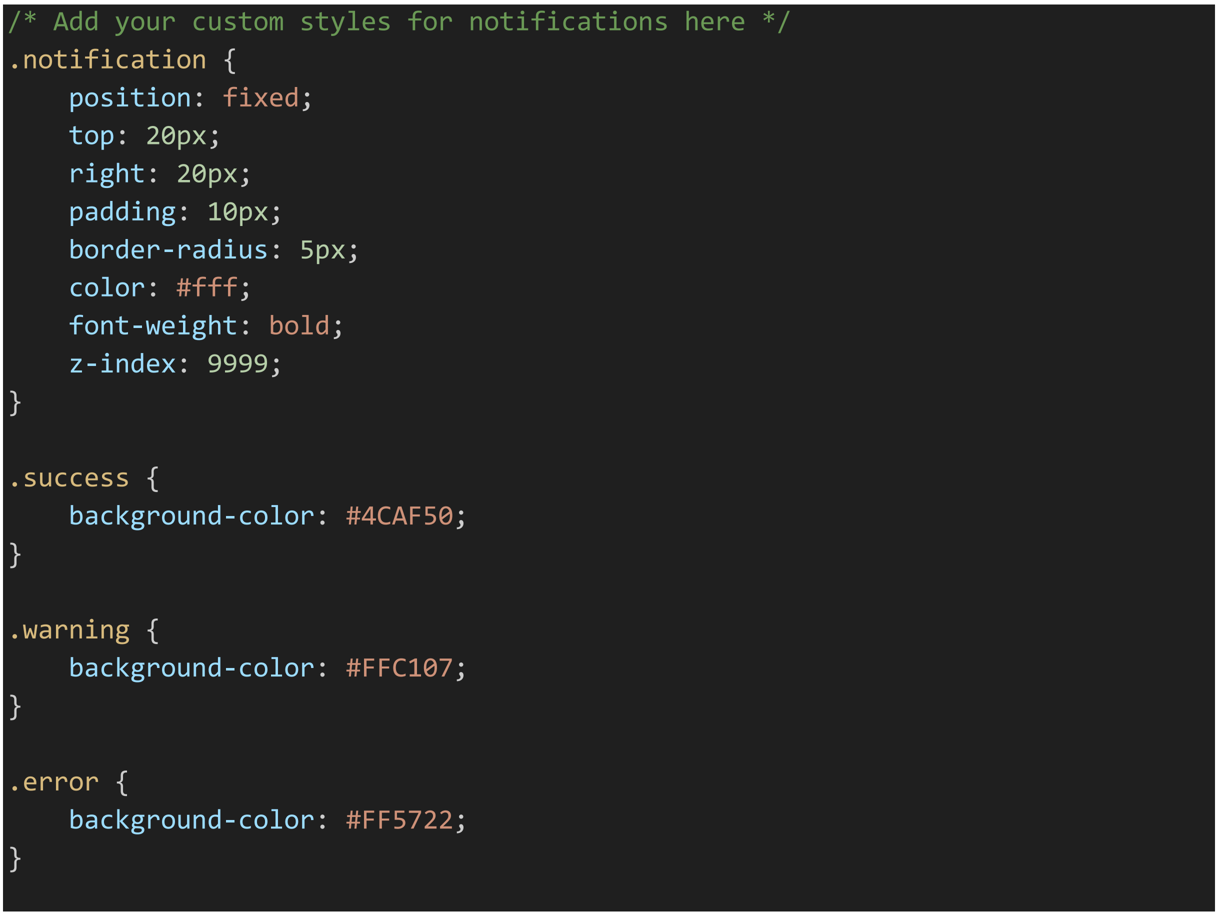


```
                </form>
              </div>
            </div>
          </nav>

          <header>
        <h1 class="text-black">Smart Water System </h1>


    </header>
    <main>
        <section class="sensor-data" id="home">
            <h2>Sensor Data</h2>
            <p>Water Level: <span id="water-level">70%</span></p>
            <p>Temperature: <span id="temperature">22°C</span></p>
            <p>pH Level: <span id="ph-level">7.0</span></p>
        </section>
        <section class="control-panel">
            <h2>Control Panel</h2>
            <button id="start-pump">Start Pump</button>
            <button id="stop-pump">Stop Pump</button>
        </section>
    </main>
    <div style="margin-left:270px ; margin-top: 100px;">
    <h1>Notification and Alert Example</h1>
    <button onclick="showNotification('Success!', 'Your task was completed
successfully.', 'success')">Show Success Notification</button>
    <button onclick="showNotification('Warning!', 'Please be cautious with
this action.', 'warning')">Show Warning Notification</button>
    <button onclick="showNotification('Error!', 'An error occurred. Please try
again.', 'error')">Show Error Notification</button>
    <div id="notification-container"></div>
    </div>
    <script src="script.js"></script>
    <div style="margin-left:270px ; margin-top: 100px;">
    <div>
        <h1>Real-Time Data Updates</h1>
        <p>PH Level: <span id="ph-level">7.0</span></p>
        <p>Conductivity: <span id="conductivity">500 µS/cm</span></p>
        <p>Temperature: <span id="temperature">25°C</span></p>
    </div>
    <div id="history">
        <br><br><br>
        <h1>Water Usage History</h1>
        <table>
            <thead>
                <tr>
                    <th>Date</th>
```



```
                    <th>Usage (gallons)</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>2023-10-01</td>
                    <td>100</td>
                </tr>
                <tr>
                    <td>2023-10-02</td>
                    <td>90</td>
                </tr>
                <tr>
                    <td>2023-10-03</td>
                    <td>110</td>
                </tr>
                <!-- Add more rows for historical data -->
            </tbody>
        </table>
        <header>
            <div style="margin-top: 100px;">
          <h1 class="text-black">Historical Data Report</h1><br>
      </header>
      <div class="container" style="margin-left: 70px;">
          <h2>Data for Date: January 1, 2023</h2>
          <table>
              <tr>
                  <th>Time</th>
                  <th>Water Flow Rate (GPM)</th>
                  <th>Water Quality (pH)</th>
              </tr>
              <tr>
                  <td>08:00 AM</td>
                  <td>5.2</td>
                  <td>7.0</td>
              </tr>
              <tr>
                  <td>09:00 AM</td>
                  <td>4.8</td>
                  <td>7.2</td>
              </tr>
              <!-- Add more data rows as needed -->
          </table>
      </div>
    </div>
    </div>
</body>
</html>
```

CSS Code : (style2.css)



```css
/* Add your custom styles for notifications here */
.notification {
    position: fixed;
    top: 20px;
    right: 20px;
    padding: 10px;
    border-radius: 5px;
    color: #fff;
    font-weight: bold;
    z-index: 9999;
}

.success {
    background-color: #4CAF50;
}

.warning {
    background-color: #FFC107;
}

.error {
    background-color: #FF5722;
}
```

The provided CSS code contains custom styles for notifications, which are typically displayed to the user as feedback, alerts, or messages. Here's an explanation of the CSS rules and their purpose:

1. **.notification:**

   - This is a class selector that applies styles to elements with the class "notification." In the HTML code, you would apply this class to elements that represent notifications.
   - **position: fixed;**: The notification is fixed in place on the screen, which means it stays in the same position even when the user scrolls the page.
   - **top: 20px; right: 20px;**: These properties position the notification 20 pixels from the top and 20 pixels from the right edge of the viewport, creating a top-right corner placement.

    - **padding: 10px;**: Adds some padding around the notification content, making it visually appealing.
    - **border-radius: 5px;**: Rounds the corners of the notification with a radius of 5 pixels, giving it a slightly rounded appearance.
    - **color: #fff;**: Sets the text color to white, making it stand out against the background.
    - **font-weight: bold;**: Makes the text inside the notification bold, emphasizing it.
    - **z-index: 9999;**: Sets a high z-index value, ensuring the notification appears on top of other elements on the page.

2. **.success, .warning, .error:**
    - These are class selectors for different types of notifications: success, warning, and error.
    - They define background colors for each type of notification.
        - **.success** has a green background.
        - **.warning** has a yellow background.
        - **.error** has a red background.
    - The background color helps convey the severity or type of the notification.

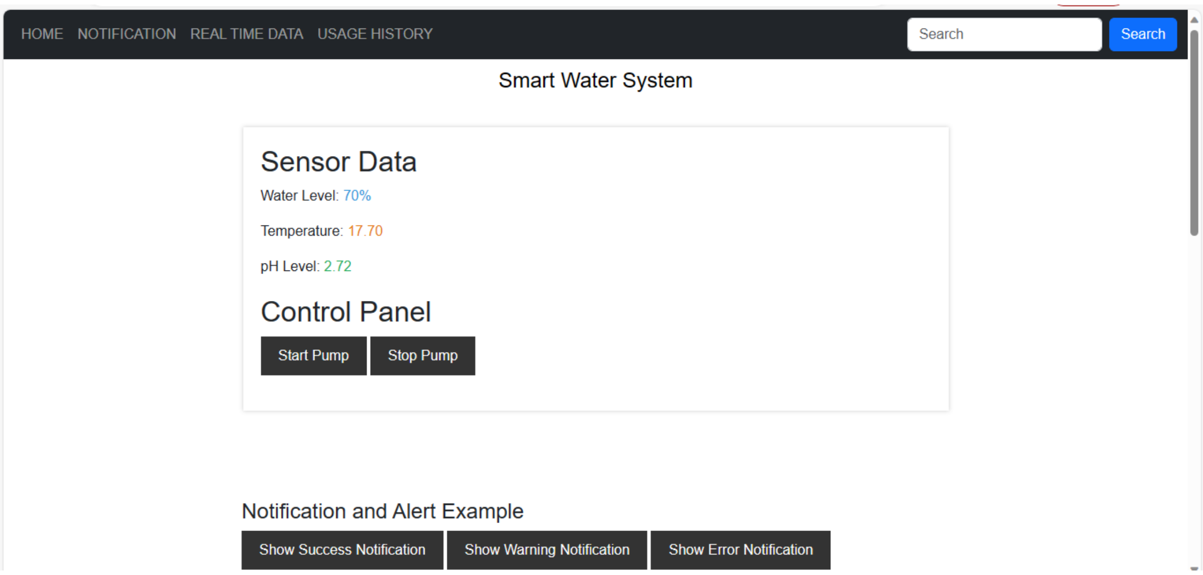
When you want to display a notification in your HTML, you would typically create an element (e.g., a **<div>**) and give it the "notification" class. Additionally, you would add one of the classes (.success, .warning, .error) to indicate the type of notification. For example:

htmlCopy code

```
<div class="notification success"> Your task was completed successfully. </div>
```

This would create a success notification with a green background that appears fixed to the top-right corner of the screen with the specified styles from the CSS. You can customize the styles further or use JavaScript to control the appearance and behavior of notifications.

Result :



Smart Water System

Sensor Data

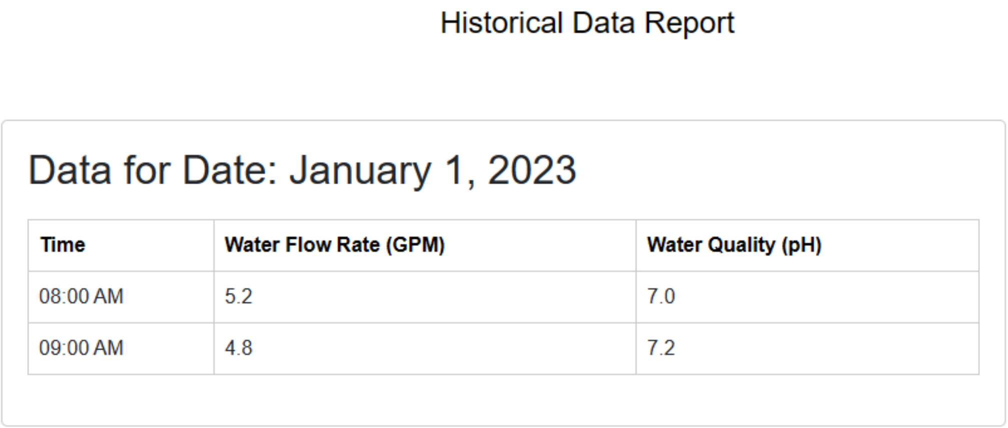
Water Level: 70%

Temperature: 17.70

pH Level: 2.72

Control Panel

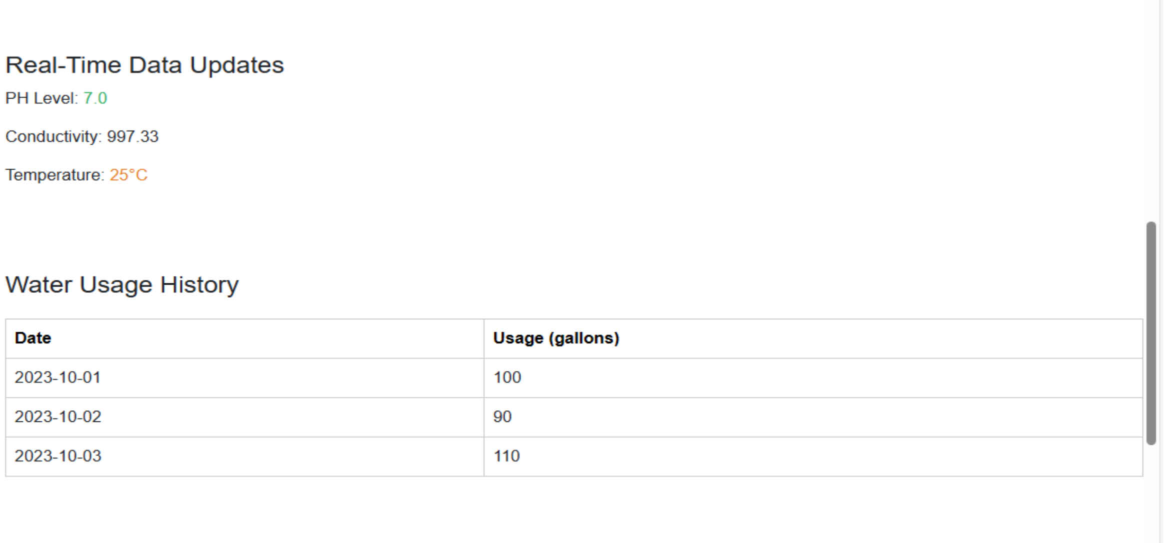
Notification and Alert Example



Historical Data Report

Data for Date: January 1, 2023

Time	Water Flow Rate (GPM)	Water Quality (pH)
08:00 AM	5.2	7.0
09:00 AM	4.8	7.2



Real-Time Data Updates

PH Level: 7.0

Conductivity: 997.33

Temperature: 25°C

Water Usage History

Date	Usage (gallons)
2023-10-01	100
2023-10-02	90
2023-10-03	110

Conclusion:

The Phase 1 of the "Smart Water System" project, which focuses on project definition and design thinking, lays a solid foundation for the project's successful implementation. By defining clear objectives centered around real-time water consumption monitoring, public awareness, water conservation, and sustainable resource management, the project aligns with its mission to promote responsible water use.

The IoT sensor design considerations ensure that the project will accurately capture water consumption data from strategically deployed sensors. The real-time transit information platform, in the form of a user-friendly mobile app, promises to engage users and facilitate data access, further enhancing public awareness.

Lastly, the integration approach ensures efficient data collection, processing, and sharing, utilizing IoT technology and Python. As the project progresses, it will continue to address key factors like data privacy, security, scalability, and sustainability to achieve its water conservation goals effectively. Phase 1 sets the stage for a promising and impactful Smart Water System project.