To create a web-based platform for displaying real-time water consumption data, you'll need to create a web page with HTML, style it with CSS, and use JavaScript to fetch and display the data from your IoT sensors. Here's a basic example to get you started:

Create an HTML file (index.html):

```
<!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>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Water Consumption Dashboard</h1>
        <div class="data-display">
            <p id="water-level">Loading...</p>
            <p id="connection-status">Connecting...</p>
            <p id="mode">Mode: Loading...</p>
            <p id="relay-status">Relay: Loading...</p>
            <p id="buzzer-status">Buzzer: Loading...</p>
        </div>
    </div>

    <script src="script.js"></script>
</body>
</html>
```

Create a CSS file (styles.css):

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
}

.container {
    max-width: 600px;
    margin: 0 auto;
    padding: 20px;
```

```
}

.data-display {
  border: 1px solid #ccc;
  padding: 10px;
  margin-top: 20px;
  text-align: left;
}

.data-display p {
  margin: 5px 0;
}
```

Create a JavaScript file (script.js):

```
document.addEventListener('DOMContentLoaded', function() {
  const waterLevelDisplay = document.getElementById('water-level');
  const connectionStatusDisplay = document.getElementById('connection-
status');
  const modeDisplay = document.getElementById('mode');
  const relayStatusDisplay = document.getElementById('relay-status');
  const buzzerStatusDisplay = document.getElementById('buzzer-status');

  function updateDisplay(data) {
    waterLevelDisplay.innerText = `Water Level: ${data.waterLevel}%`;
    connectionStatusDisplay.innerText = data.connectionStatus ? 'Status:
Online' : 'Status: Offline';
    modeDisplay.innerText = `Mode: ${data.mode}`;
    relayStatusDisplay.innerText = `Relay: ${data.relayStatus ? 'ON' : 'OFF'}`;
    buzzerStatusDisplay.innerText = `Buzzer: ${data.buzzerStatus ? 'ON' :
'OFF'}`;
  }

  function fetchData() {
    // Make an API request to your IoT device for the data
    // Replace 'apiEndpoint' with the actual endpoint to retrieve data from your
IoT device
    fetch('apiEndpoint')
      .then(response => response.json())
      .then(data => updateDisplay(data))
      .catch(error => console.error('Error:', error));
```

```
    }

    // Update data every 3 seconds
    setInterval(fetchData, 3000);
});
```

Explanation:

index.html: This is the main HTML file. It includes a basic structure with placeholders for displaying water consumption data. It links to the CSS file for styling and the JavaScript file for functionality.

styles.css: This file contains basic styling for the web page.

script.js: This JavaScript file handles fetching data from your IoT device and updating the display. It uses fetch to make an API request to your IoT device. You'll need to replace 'apiEndpoint' with the actual endpoint URL to retrieve data from your device.

Integrate with Your IoT Device:

You'll need to replace 'apiEndpoint' in script.js with the actual URL or API endpoint where your IoT device is providing data.

Ensure that your IoT device is set up to handle requests and respond with the necessary data in a format that the JavaScript code can process.

Ensure your IoT device is connected to the same network as the device running the web page, or configure it to be accessible over the internet (if needed).

This basic example will periodically fetch data from your IoT device and update the web page with the latest information. You can further enhance this with additional features and interactivity as needed.

Team members:
A.Athira- 962821104020
A.P.Bala Bharathy- 962821104023
K.Faisha Marshooka- 962821104030
G.T.Harini- 962821104033