**Smart water management**

**html code;**

**</html>**<!DOCTYPE html>

<html>

<head>

<title>Water Level Monitoring System</title>

<link rel="stylesheet" type="text/css" href="style.css">

</head>

<body>

<div id="water-level">

<h1>Water Level: <span id="level">Loading...</span></h1>

</div>

<div id="flood-warning">

<h2>Flood Warning</h2>

<p id="warning-status">No warnings at this time.</p>

</div>

<script src="script.js"></script>

</body>

</html>

// script.js

const levelElement = document.getElementById("level");

// Simulate receiving real-time data from IoT sensors

function updateWaterLevel(newLevel) {

levelElement.textContent = newLevel + " meters";

}

// Connect to the server for real-time data updates (use appropriate server-side code)

const socket = new WebSocket("ws://your-server-url");

socket.onmessage = (event) => {

const data = JSON.parse(event.data);

updateWaterLevel(data.waterLevel);

};

const warningStatusElement = document.getElementById("warning-status");

// Simulated flood warning logic

function checkFloodWarning(newLevel) {

if (newLevel > 5) {

warningStatusElement.textContent = "Flood warning issued!";

// Trigger notifications or alerts as needed

} else {

warningStatusElement.textContent = "No warnings at this time.";

}

}

// Update water level and check for flood warnings

function updateWaterLevelAndCheckWarning(newLevel) {

updateWaterLevel(newLevel);

checkFloodWarning(newLevel);

}

// Simulate periodic updates from sensors

setInterval(() => {

const randomLevel = Math.random() \* 10; // Simulated water level data

updateWaterLevelAndCheckWarning(randomLevel);

}, 5000); // Update every 5 seconds

**Back-end (Server-Side):**

For the back-end of your system, you will need a server that collects data from IoT sensors and broadcasts it to connected clients through WebSockets. You can use technologies like Node.js with libraries such as Express and Socket.IO to create the server-side logic.

1. Set up a Node.js server with Express to handle HTTP requests.
2. Implement WebSocket functionality (using Socket.IO or a similar library) to send real-time data to clients.
3. Connect to IoT sensors and receive data.
4. Implement logic for flood warnings and send notifications or alerts when necessary.

This is a simplified example, and in a real-world scenario, you would need to consider security, data persistence, and scalability. Additionally, integrating with a database to store historical data and more advanced flood warning algorithms would be crucial for an operational water management system.