**Smart Water Fountain**

**Building the project by developing the water fountain status platform. Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time water fountain status.**

**Design the platform to receive and display real-time water fountain data, including water flow rate and malfunction alerts.**

* Creating a real-time water fountain status platform using web development technologies involves several components and coding. Here's a simplified outline for each topic:

**Fountain Information Platform (web);**

**1.HTML for Structure:**

* Start with an HTML file to structure your platform. Here's a basic example of the structure:

**Code;**

<!DOCTYPE html>

<html>

<head>

<title>Water Fountain Status</title>

</head>

<body>

<h1>Water Fountain Status</h1>

<div id="status"></div>

</body>

</html>

**2.CSS for Styling:**

* Use CSS to style your platform. This is a minimal example:

**Code;**

body {

font-family: Arial, sans-serif;

}

h1 {

text-align: center;

}

**3.JavaScript for Real-time Data:**

* JavaScript is essential for fetching and displaying real-time data. You can use technologies like WebSockets or AJAX for this. Here's a basic example using JavaScript with AJAX to periodically fetch data:

**Code;**

function fetchWaterFountainStatus() {

// Make an AJAX request to get data from a server (replace with your API endpoint)

// You'd typically use a library like Axios or Fetch API for this.

// For this example, let's assume you have an API that returns JSON data.

fetch('your-api-endpoint-here')

.then(response => response.json())

.then(data => {

// Update the status on the page

document.getElementById('status').textContent = `Flow Rate: ${data.flowRate} gpm, Malfunction: ${data.malfunction}`;

})

.catch(error => {

console.error('Error fetching data: ' + error);

});

}

// Fetch data every 5 seconds (adjust the timing as needed)

setInterval(fetchWaterFountainStatus, 5000);

**4.Server-Side Code:**

* You'll need a server to provide the real-time data. You can use Node.js, Python (Django/Flask), or any server-side technology. Here's a simple Node.js example using Express to create an API endpoint:

**Code;**

const express = require('express');

const app = express();

// Define a sample endpoint to provide fountain data

app.get('/fountain-status', (req, res) => {

const fountainData = {

flowRate: 5.2, // Example flow rate in gallons per minute

malfunction: false // Example malfunction status

};

res.json(fountainData);

});

app.listen(3000, () => {

console.log('Server is running on port 3000');

});

Please note that this is a simplified example. In a real-world scenario, you should consider security, data persistence, and other best practices. Also, you would need to integrate this with sensors or data sources for actual real-time data.