Flood Monitoring and Early Warning System

Phase 4 submission

Phase 4: Development Part 2

Topic: Flood Monitoring and Early Warning System

Continue building the project by developing the early warning platform. Use web development technologies (e.g., HTML, CSS, JavaScript) to create a platform that displays real-time water level data and flood warnings. Design the platform to receive and display water level data from IoT sensors and issue flood warnings when necessary.

HTML Code (index.html)

```
<!DOCTYPE html>
<html>
<head>
<title>Real-Time Flood Warning System</title>
<link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
<nav class="nav-2">
<a href="#data-display" class="nav-1">Water Level Data</a> 
<a href="#warning-section"class="nav-1">Flood Warnings</a>
<a href="#settings" class="nav-1">Settings</a>
</nav>
<header>
<h1>Real-Time Water Level Data and Flood Warnings</h1>
</header>
<section id="data-display">
<h2>Water Level Data</h2>
<table>
<thead>
<tr>
<th>Location</th>
<th>Water Level (meters)</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody id="data-table">
<!-- Real-time data will be displayed here -->
```

```
</tbody>
</table>
</section>
<section id="warning-section">
<h2>Flood Warnings</h2>
<ul id="warnings-list">
<!-- Real-time flood warnings will be displayed here -->
</ul>
</section>
<section id="settings">
<h2>System Settings</h2>
<form id="settings-form">
<label for="server-url">Server URL:</label>
<input type="text" id="server-url" name="server-url"
placeholder="Enter the server URL" required> <label for="sensorthreshold">Sensor
Threshold (meters):</label>
<input type="number" id="sensor-threshold" name="sensor-threshold"
step="0.01" placeholder="Enter the sensor threshold" required>
<label for="notification-interval">Notification Interval
(seconds):</label>
<input type="number" id="notification-interval" name="notificationinterval"
placeholder="Enter the notification interval" required>
<button type="submit">Save Settings</button>
</form>
</section>
<div id="water-level-chart">
<!-- JavaScript will populate and render the chart here -->
</div>
<section id="notifications">
<h2>Notifications</h2>
<ul>
<li>System is operational.</li>
<!-- Display real-time flood warnings here -->
```

```
</ul>
</section>
<footer>
<div style="align-content: center;">
<p>&copy; 2023 YourCompany. All rights reserved.</p>
<p><a href="privacy-policy.html">Privacy Policy</a> | <a href="terms-ofservice.html">Terms of Service</a></p></div>
</footer>
<script>
// Simulate fetching and updating real-time data
function fetchData() {
// Simulate data from IoT sensors (you should replace this with real
data)
const locations = ["Location A", "Location B", "Location C"];
const waterLevels = [Math.random() * 10, Math.random() * 10,
Math.random() * 10];
// Get the data table element
const dataTable = document.getElementById("data-table");
// Clear previous data
dataTable.innerHTML = '';
// Loop through the data and populate the table
for (let i = 0; i < locations.length; i++) {
const row = document.createElement("tr");
row.innerHTML = `
<td>${locations[i]}</td>
<td>${waterLevels[i].toFixed(2)}</td>
<td>${new Date().toLocaleString()}</td>
`;
dataTable.appendChild(row);
// Check for flood warnings (you can set your own thresholds)
if (waterLevels[i] > 7.0) {
const warningsList = document.getElementById("warningslist");
const warning = document.createElement("li");
```

```
warning.textContent = `Flood Warning: High water level
detected at ${locations[i]}.`;
warningsList.appendChild(warning);
}
}
}
document.getElementById("settings-form").addEventListener("submit",
function (event) {
event.preventDefault(); // Prevent the default form submission
// Get values from the input fields
const serverUrl = document.getElementById("server-url").value;
const sensorThreshold = parseFloat(document.getElementById("sensorthreshold").value);
const notificationInterval = parseInt(document.getElementById("notificationinterval").value);
// Use the values to update system settings (you can send them to the server
or store them locally)
// For example, display the updated settings in the console
console.log("Server URL:", serverUrl);
console.log("Sensor Threshold:", sensorThreshold);
console.log("Notification Interval:", notificationInterval);
// You can also send these settings to your server for further processing
// Optionally, provide feedback to the user that settings were saved
alert("Settings saved successfully!");
}); // Update data every 5 seconds (adjust the interval as needed)
setInterval(fetchData, 5000);
</script>
</body>
</html>
```

This is an HTML document that represents a web page for a "Real-Time Flood Warning System." Let me explain the structure and functionality of this HTML code:

1. Document Type Declaration (<!DOCTYPE html>):

This declaration specifies that the document is an HTML5 document.

2. <html> Element: The root element of the HTML document.

3. <head> Section:

• <title> Element: Sets the title of the web page to "Real-Time Flood Warning System."

• <link> Element: Links an external CSS file named "styles.css" for styling the page.

4. <body> Element: The main content of the web page is contained within the <body> element.

5. Navigation Bar (<nav>):

• This section defines a navigation bar with links to different sections of the page.

• The links point to various sections of the page, including "Water Level Data," "Flood Warnings," and "Settings."

6. <header> Element:

• Contains the main heading, which is the title of the web page: "RealTime Water Level Data and Flood Warnings."

7. Sections:

• Three main sections are defined within the <body>:

• "Water Level Data" (<section id="data-display">): This section displays water level data in a table.

• "Flood Warnings" (<section id="warning-section">): This section displays flood warnings as a list.

• "System Settings" (<section id="settings">): This section contains a form for configuring system settings.

8. Data Display:

• The "Water Level Data" section includes a table with columns for "Location," "Water Level (meters)," and "Timestamp."

• Real-time data will be populated in this table via JavaScript.

9. Flood Warnings:

• The "Flood Warnings" section includes an unordered list (<ul>) with the id "warnings-list." Real-time flood warnings will be displayed in this list.

10.System Settings:

• The "System Settings" section includes a form (<form>) with various input fields for setting system parameters, including server URL, sensor threshold, and notification interval.

• A "Save Settings" button allows users to submit their settings. The JavaScript code handles the form submission and provides feedback when settings are saved.

11.Water Level Chart ( ):

This empty div is a placeholder for a chart that will be populated and rendered using JavaScript.

12.Notifications:

• The "Notifications" section is an unordered list that currently contains a single list item indicating that the system is operational. Real-time flood warnings can be added to this list.

13.<footer> Element: The footer of the web page, containing copyright information and links to the Privacy Policy and Terms of Service pages.

14.JavaScript:

• The JavaScript code in the <script> element at the end of the document simulates fetching and updating real-time data. It fetches data from imaginary IoT sensors, populates the data table, and adds flood warnings based on a threshold.

• It also handles the form submission for system settings, logging the values to the console and providing an alert when settings are saved.

• Data is updated every 5 seconds using setInterval.

CSS Code : (style.css)

```
/* CSS styles for your Real-Time Flood Warning System */
/* Reset some default browser styles */
body, h1, h2, ul, li, p {
margin: 0;
padding: 0;
```

```
}
/* Define general styles for your page */
body {
font-family: Arial, sans-serif;
background-color: #f2f2f2;
}
header {
background-color: white;
color: black;
padding: 20px;
text-align: center;
}
h1 {
margin: 0;
}
section {
background-color: #fff;
padding: 20px;
margin: 10px;
}
table {
width: 100%;
border-collapse: collapse;
}
table, th, td {
border: 1px solid #ddd;
}
th, td {
padding: 10px;
text-align: center;
}
ul {
```

```
list-style-type: none;
padding: 0;
}
li {
margin: 5px 0;
}
#data-display {
background-color: #f9f9f9;
}
#warning-section {
background-color: #FF4136;
color: #fff;
}
#data-table th {
background-color: #333;
color: #fff;
}
.nav-2 {
background-color: #716666;
display: flex;
justify-content: left;
padding: 10px;
}
.nav-1 {
text-decoration: none;
color: #fff;
padding: 10px 20px;
transition: background-color 0.3s;
}
/* Change link color on hover */
.nav-1:hover {
background-color: #555; /* Background color on hover */
```

```
}
/* Style the system settings section */
#settings {
background-color: #f7f7f7;
padding: 20px;
margin: 20px;
box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
}
/* Style form elements */
#settings-form {
max-width: 400px;
margin: 0 auto;
}
label {
display: block;
margin-bottom: 10px;
font-weight: bold;
}
input[type="text"],
input[type="number"] {
width: 100%;
padding: 10px;
margin-bottom: 20px;
border: 1px solid #ccc;
border-radius: 4px;
font-size: 16px;
}
input[type="submit"] {
background-color: #0074D9;
color: #fff;
border: none;
padding: 10px 20px;
```

```
cursor: pointer;
}
input[type="submit"]:hover {
background-color: #0056b3;
}
body {font-family: Arial, Helvetica, sans-serif;}
form {border: 3px solid #f1f1f1;}
input[type=text], input[type=password] {
width: 100%;
padding: 12px 20px;
margin: 8px 0;
display: inline-block;
border: 1px solid #ccc;
box-sizing: border-box;
}
button {
background-color: #04AA6D;
color: white;
padding: 14px 20px;
margin: 8px 0;
border: none;
cursor: pointer;
width: 100%;
}
button:hover {
opacity: 0.8;
}
.cancelbtn {
width: auto;
padding: 10px 18px;
background-color: #f44336;
}
```

```
.imgcontainer {
text-align: center;
margin: 24px 0 12px 0;
}
img.avatar {
width: 40%;
border-radius: 50%;
}
.container {
padding: 16px;
}
span.psw {
float: right;
padding-top: 16px;
}
/* Change styles for span and cancel button on extra small screens */
@media screen and (max-width: 300px) {
span.psw {
display: block;
float: none;
}
.cancelbtn {
width: 100%;
}
}
footer {
text-align: center;
padding: 3px;
background-color: DarkSalmon;
color: white;
}
```

The provided CSS code contains styles for various elements and sections of a

"Real-Time Flood Warning System" web page. Here's an explanation of the CSS styles:

1. Reset Styles:

• The initial styles reset default browser margins and paddings for body, h1, h2, ul, li, and p elements to make the layout consistent.

2. General Page Styles:

• body is styled with a background color of light gray (#f2f2f2) and a fallback font of Arial or sans-serif.

• The header section has a white background, black text, padding, and is centered.

• h1 within the header has its margin set to zero.

3. Section Styles:

• Sections (<section>) have a white background, padding, and margin to separate them.

• The table within sections has a 100% width and a collapsed border.

• Table cells (th and td) have padding, a thin border, and text centered.

4. List Styles:

• Unordered lists (ul) have no list-style and no padding.

• List items (li) have margin space on top and bottom.

5. Specific Section Styles:

• The "Water Level Data" section (#data-display) has a light gray background (#f9f9f9).

• The "Flood Warnings" section (#warning-section) has a background color of red (#FF4136) with white text.

6. Navigation Bar Styles:

• The navigation bar (nav-2) has a dark gray background, is displayed as a flex container, and has left-justified content with padding.

• Navigation links (nav-1) have no text decoration, white color, padding, and a background color transition on hover.

7. System Settings Styles:

• The "System Settings" section (#settings) has a light gray background, padding, margin, and a box shadow.

• Form elements are styled, including labels, text input fields, number input fields, and the submit button.

• The submit button has a blue background color that changes on hover.

8. Form Styles:

• Form elements like text and password inputs are styled with width, padding, margin, and border properties.

• The submit button is styled with background color and padding.

9. Extra Small Screens:

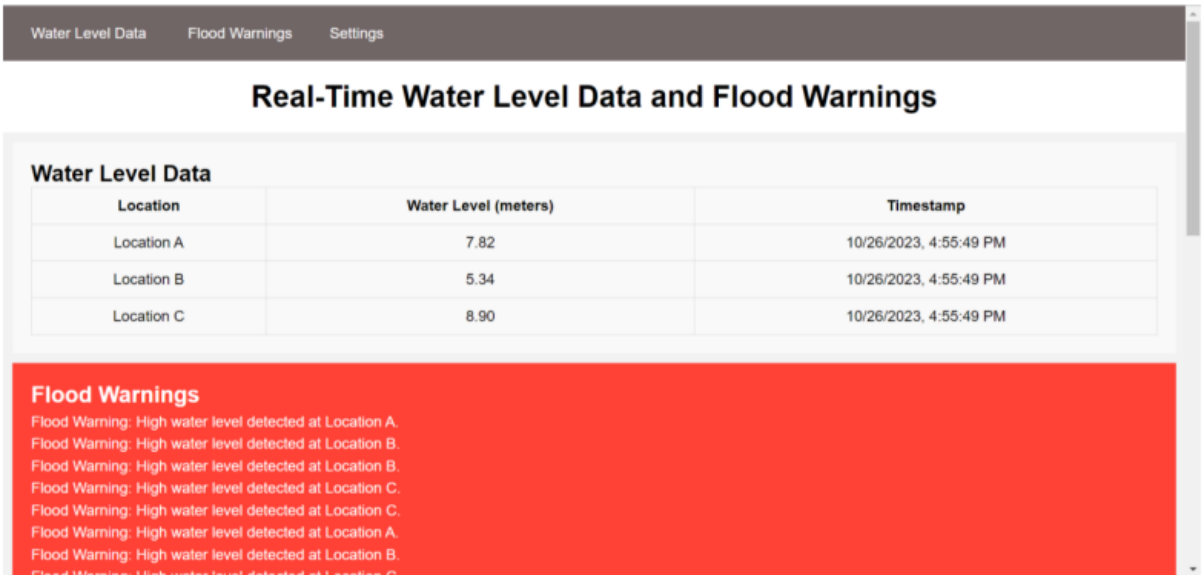
• There is a media query that changes styles for extra small screens (max-width: 300px). It modifies the span.psw and .cancelbtn styles.

10.Footer Styles:

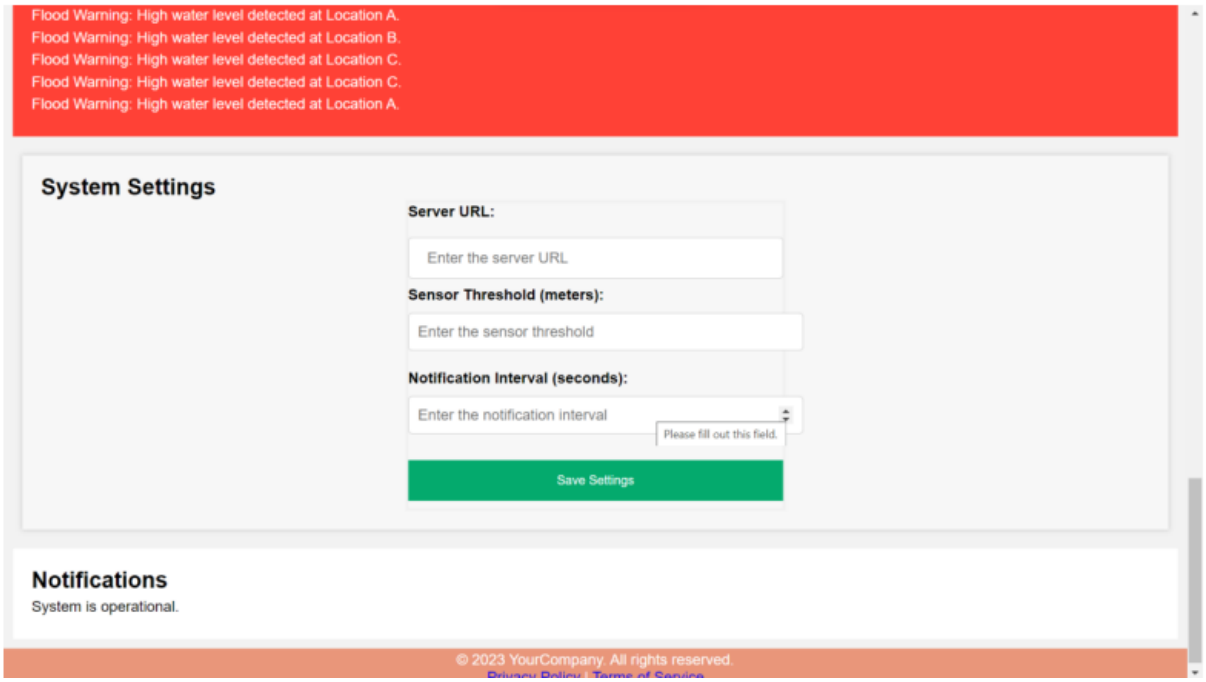
• The footer (footer) is styled with white text on a Dark Salmon background, with centered text and a little padding.

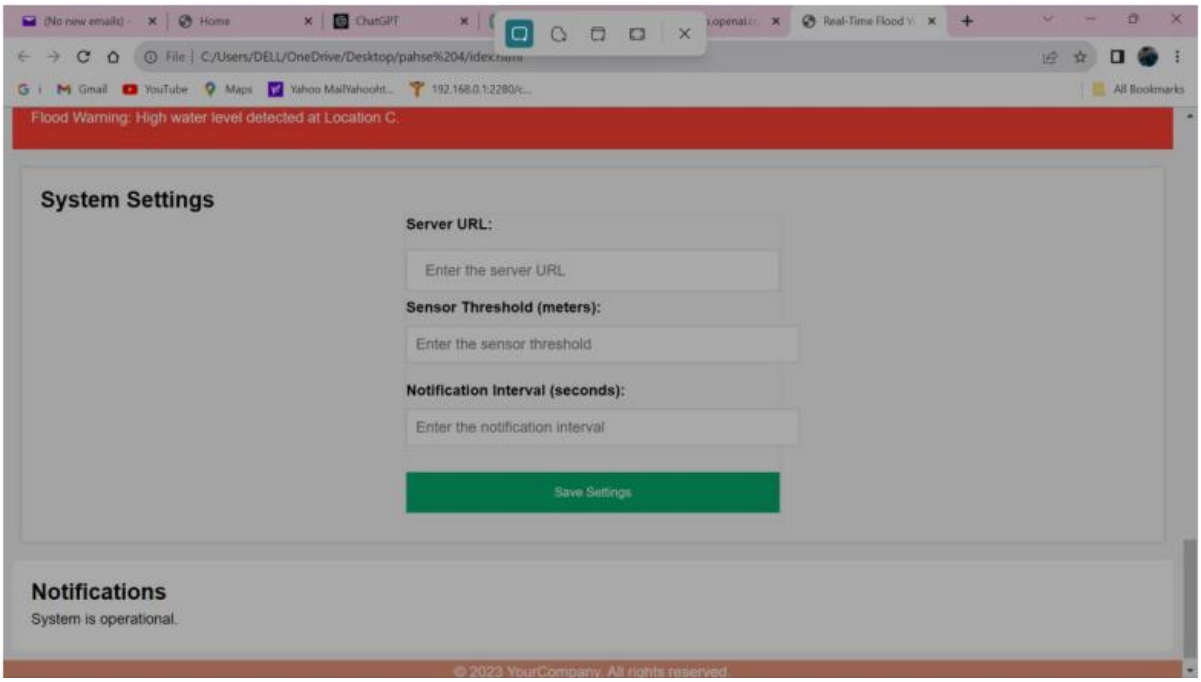
These CSS styles provide a clear and consistent design for the web page, making it user-friendly and visually appealing. Additionally, it offers responsiveness for extra small screens. However, please note that the styles related to the form elements and buttons seem to be repeated in the provided code; you can choose which styles to keep based on your design preferences.

Result




Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.


Flood Warning: High water level detected at Location A.
Flood Warning: High water level detected at Location B.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location C.
Flood Warning: High water level detected at Location A.
System Settings
Server URL:
Enter the server URL
Sensor Threshold (meters):
Enter the sensor threshold
Notification Interval (seconds):
Enter the notification interval
Please fill out this field.
Save Settings
Notifications
System is operational.
© 2023 YourCompany. All rights reserved.
Privacy Policy | Terms of Service


Flood Warning: High water level detected at Location C.
System Settings
Server URL:
Enter the server URL
Sensor Threshold (meters):
Enter the sensor threshold
Notification Interval (seconds):
Enter the notification interval
Save Settings
Notifications
System is operational.
© 2023 YourCompany. All rights reserved.