**COLLEGE NAME:** JEPPIAAR ENGINEERING COLLEGE

**COLLEGE CODE:** 3108

**DEPARTMENT:** B.E COMPUTER SCIENCE AND ENGINEERING

**SEMESTER:** 05

**COURSE NAME:** Internet of Things

**PROJECT NAME:** FLOOD MONITORING AND EARLY WARMING

**PHASE 4: Development Part 2**

**INTRODUCTION:**

Creating a platform for real-time water level data and flood warnings from IoT sensors is a complex project. The frontend involves HTML, CSS, and JavaScript to display data and alerts. The project includes real-time data retrieval, flood warning logic, user interaction, notifications, error handling, responsive design, testing, and documentation. However, it's important to note that a complete system would also require a backend for data management and user authentication, and security and scalability considerations are vital.

**APPLICATIONS**:

Developing a real-time water level monitoring system is a comprehensive project. Here's an overview of the key components and technologies involved in different forms:

1.**Frontend Web Application:**

- Create a web interface using HTML, CSS, and JavaScript to present water level data and flood warnings.

- Design a user-friendly dashboard with visualisations and maps for data representation.

2. **Backend Server:**

- Set up a backend server using a server-side language like Node.js, Python, or Ruby to manage data processing and communicate with IoT sensors.

3. **IoT Sensor Integration:**

- Integrate IoT sensors capable of measuring water levels and transmitting real-time data to the server.

- Utilise protocols like MQTT, HTTP, or WebSocket for effective sensor data communication.

4. **Data Storage:**

- Implement a database system such as MySQL or MongoDB to store historical sensor data for analysis and reference.

5. **Real-time Data Processing:**

- Develop server-side scripts to process incoming sensor data, perform calculations, and generate water level trends.

6. **Flood Warning Logic:**

- Implement algorithms to detect abnormal water level changes or predefined thresholds that may indicate potential floods.

- Configure the system to trigger flood warnings based on these thresholds.

7. **Notifications:**

- Integrate a notification system that can send alerts through various channels such as email, SMS, or push notifications to notify users and authorities when a flood warning is issued.

8. **Map Integration:**

- Use mapping libraries like Leaflet or Google Maps to display sensor locations and real-time water level data on the frontend for visualization.

9. **User Authentication and Authorization:**

- Implement user authentication and authorization to control access to the platform and ensure data security.

10. **APIs:**

- Create RESTful or GraphQL APIs to facilitate communication between the frontend and backend components, allowing data exchange.

**Hardware and Software Specifications:**

- Programming Languages: Python, HTML5, CSS3, JavaScript

- Hardware Components: Ultrasonic sensor, Buzzer, DHT22 sensor, Raspberry Pi Pico, LED lights, Register (likely for GPIO control)

- These components are used to gather sensor data, alert users, and control hardware devices.

This project combines web development, data processing, IoT integration, and notification systems to provide a comprehensive solution for monitoring water levels and issuing flood warnings in real-time.

**To create a platform that displays real-time water level data and flood warnings.**

**CODE:**

**HTML**

Fmes.html

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Real-Time Water Level Data & Flood Warnings</title>

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

</head>

<body>

<h1>Flood monitoring system model</h1>

<div class="hed">

<div class="header">

<h1>Real-Time Water Level Data & Flood Warnings</h1>

</div>

<div class="content">

<div class="water-level">

<h2>Current Water Level: <span id="waterLevel">loading...</span></h2>

</div>

<div class="flood-warning">

<h2>Flood Warning detection: <span id="floodWarning">loading...</span></h2>

</div>

</div>

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

</div>

</body>

</html>

**CSS:**

Fmes.css

body {

font-family: Arial, sans-serif; background-color: #f0f0f0; margin: 110px;

padding: 10px;

}

.hed{

border-style: solid;

}

.header {

background-color: #d84155; color: white;

text-align: center; padding: 20px;

}

.content { margin: 20px;

text-align: center;

}

.water-level, .flood-warning {

background-color: rgba(18, 179, 207, 0.486); padding: 10px;

margin: 10px;

border: 1px solid #ccc;

}

**JS:**

Fmes.js

const WaterLevelData = () => (Math.random() \* 10).toFixed(2); const FloodWarningData = () => Math.random() > 0.7;

function updateData() {

const waterLevelElement = document.getElementById("waterLevel"); const floodWarningElement = document.getElementById("floodWarning");

const waterLevel = WaterLevelData();

const isFloodWarning = FloodWarningData();

waterLevelElement.textContent = waterLevel + " meters"; floodWarningElement.textContent = isFloodWarning ? "Yes" : "No"; floodWarningElement.style.color = isFloodWarning ? "red" : "green";

}

setInterval(updateData, 5000);

updateData();

**CONCLUSION:**

This project involves creating a real-time water level monitoring system. The frontend uses HTML, CSS, and JavaScript to display data and alerts, while the backend manages data processing and communicates with IoT sensors. The system integrates IoT sensors, stores data in a database, processes real-time data, and issues flood warnings. User notifications, map visualisation, user authentication, and APIs are key components for a comprehensive solution. The code provided represents the frontend, and in practice, a complete system requires a backend and user authentication for data security and functionality.