Flood monitoring and early warning system

Phase 4 :

Introduction

In the context of IoT-based environmental monitoring, flood warning systems play a critical role in mitigating potential disasters. Leveraging IoT sensors and web technologies, a flood monitoring system has been designed to monitor water levels and issue warnings. This document showcases the creation of a flood monitoring system through two implementations: a Flutter-based mobile application and a web-based platform using HTML, CSS, and JavaScript.

Flutter Code

lib/main.dart

dart

```
import 'package:flutter/material.dart';
import 'dart:async'; // For simulating sensor data

void main() {
  runApp(FloodMonitoringApp());
}

class FloodMonitoringApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flood Monitoring App',
      home: FloodMonitoringScreen(),
```

```
    );
  }
}

class FloodMonitoringScreen extends StatefulWidget {
  @override
  _FloodMonitoringScreenState createState() => _FloodMonitoringScreenState();
}

class _FloodMonitoringScreenState extends State<FloodMonitoringScreen> {
  double waterLevel = 0.0; // Simulated water level
  late Timer timer;

  @override
  void initState() {
    super.initState();
    // Simulate sensor data (increment water level every 3 seconds)
    timer = Timer.periodic(Duration(seconds: 3), (Timer t) {
      setState(() {
        waterLevel += 0.5;
      });
    });
  }
```

```
@override
void dispose() {
  timer.cancel(); // Cancel the timer to prevent memory leaks
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: Text('Flood Monitoring App'),
    ),
    body: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Text(
            'Water Level: ${waterLevel.toStringAsFixed(1)}',
            style: TextStyle(fontSize: 24),
          ),
          SizedBox(height: 20),
          ElevatedButton(
            onPressed: () {
              if (waterLevel > 5.0) {
```

```
            issueFloodWarning();
          }
        },
        child: Text('Check for Flood'),
      ),
    ],
   ),
  ),
 );
}

void issueFloodWarning() {
 showDialog(
  context: context,
  builder: (BuildContext context) {
   return AlertDialog(
    title: Text('Flood Warning!'),
    content: Text('The water level is critical. Be cautious!'),
    actions: [
     TextButton(
      onPressed: () {
       Navigator.of(context).pop();
      },
      child: Text('Close'),
```

```
          ),
        ],
      );
    },
  );
 }
}
```

Description:

The Flutter code represents a basic mobile application for flood monitoring. It simulates changing water level data every 3 seconds and triggers a flood warning if the water level exceeds a defined threshold (5.0 in this example).

Web-Based Flood Monitoring Platform

index.html

```
html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Flood Monitoring Platform</title>
  <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
    <div class="container">
        <h1>Flood Monitoring Platform</h1>
        <div class="water-level">
            <h2>Water Level:</h2>
            <div id="waterLevel">0.0</div>
        </div>
        <div class="warning">
            <h2>Warning:</h2>
            <div id="warningStatus">No flood</div>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

styles.css

```
css
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    margin: 0;
    padding: 0;
```

```
}

.container {
    width: 80%;
    margin: 0 auto;
    text-align: center;
}

.water-level, .warning {
    margin: 20px 0;
    padding: 20px;
    background-color: #fff;
    border: 1px solid #ccc;
}

.water-level, .warning h2 {
    margin: 0;
}

.water-level #waterLevel, .warning #warningStatus {
    font-size: 24px;
    font-weight: bold;
}
```

script.js

javascript

```
document.addEventListener('DOMContentLoaded', () => {
    let waterLevel = 0.0; // Simulated water level
    const warningThreshold = 5.0; // Define the warning threshold

    setInterval(() => {
        waterLevel += 0.5;
        document.getElementById('waterLevel').innerText = waterLevel.toFixed(1);

        if (waterLevel > warningThreshold) {
            issueFloodWarning();
        }
    }, 3000);

    function issueFloodWarning() {
        document.getElementById('warningStatus').innerText = 'Flood Warning!';
        document.getElementById('warningStatus').style.color = 'red';
    }
});
```

Description

The web-based code simulates a flood monitoring platform. It showcases water level information and triggers a flood warning when the water level surpasses the defined threshold. The JavaScript code updates the water level data every 3 seconds and issues a warning accordingly.

Conclusion

The Flutter-based application and the web-based platform exemplify simulated water level monitoring and flood warnings. While the Flutter code designs a simple mobile application, the web-based platform provides a basic representation of a monitoring system. Real-world implementation would involve actual sensor data integration and enhanced user interfaces for effective flood monitoring.