Flood Monitoring and Early Warning

Objective

Floods are among the most devastating natural disasters, posing significant threats to human lives, property, and the environment. The objective of establishing a robust flood monitoring and early warning system is to proactively mitigate flood-related risks, safeguard communities, infrastructure, and natural resources, and reduce flood-induced damages while enhancing public safety.

Therefore, the primary objective of flood monitoring and early warning systems is to harness technology and data-driven strategies to predict, detect, and respond to flood events in a manner that minimizes harm and financial losses.

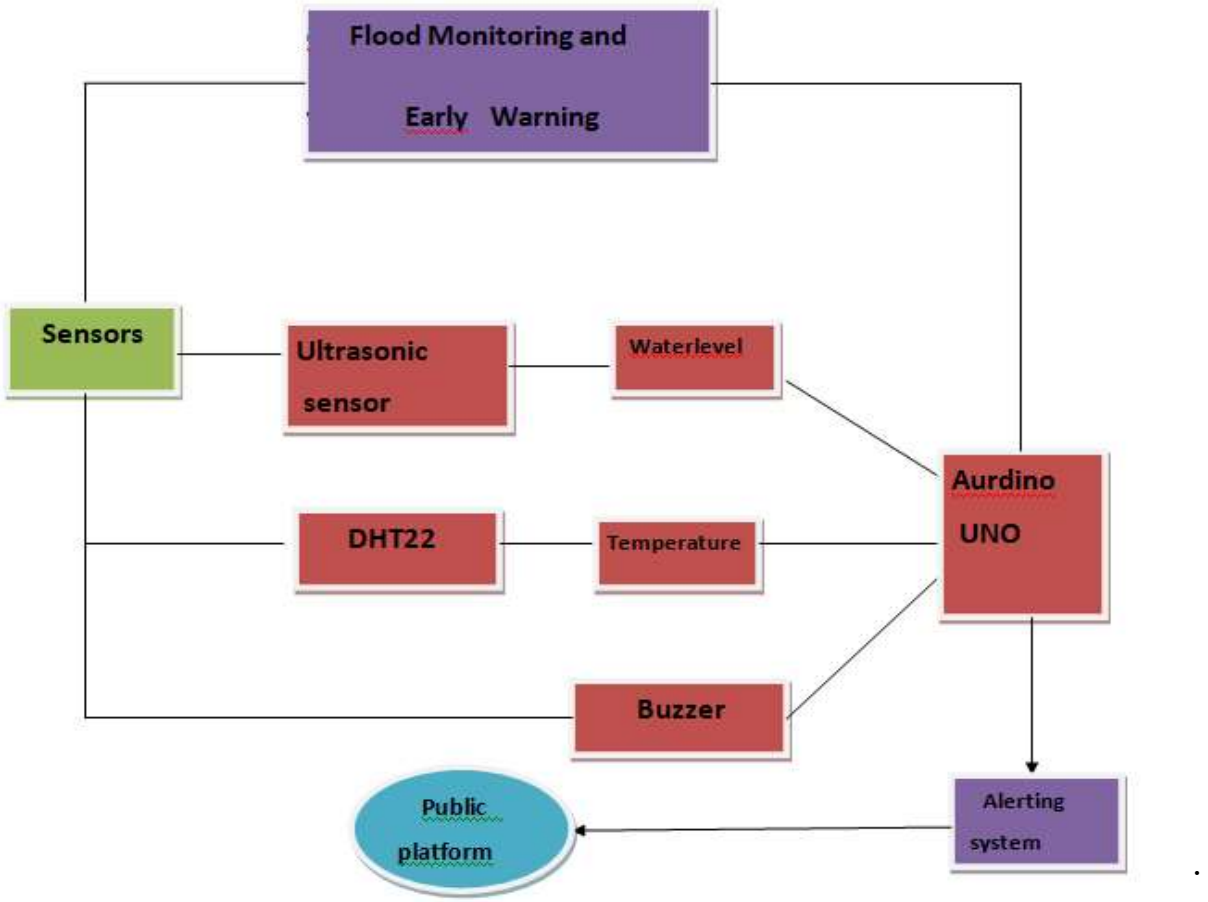
IOT devices setup:

- **Sensor Deployment**

  Water level Sensors like Ultrasonic sensor in Arduino simulator are strategically placed in different water bodies around the city or region under study. These sensors capture water levels and can differentiate between various types of water levels due to raining and cyclones.

- **Monitoring :**

  If the water level detected by the sensor exceeds a certain threshold, the system will send out a notification. Implement a system of alerts that can inform the appropriate authorities when water levels rise. Explore and put into effect techniques for reducing flood, adaptive traffic management, or changing construction methods. IoT data can be used to assess the effectiveness of these tactic.



Wokwi Simulator:

The virtual sensors to run the application by utilizing the Wokwi simulator. Wokwi supports a wide range of popular microcontrollers, such as Arduino, ESP8266, and Raspberry Pi Pico.

Main.py:

```
import machine
import time
import urequests
import ujson
import network

# Define ultrasonic sensor pins (Trig and Echo pins)
trigger_pin = machine.Pin(8, machine.Pin.OUT)
echo_pin = machine.Pin(9, machine.Pin.IN)
buzzer_pin = machine.Pin(10, machine.Pin.OUT)
led_pin = machine.Pin(11, machine.Pin.OUT)  # Added LED pin for early
warning
```

```
# Define your Wi-Fi credentials
wifi_ssid = 'Wokwi-GUEST'

wifi_password = 'U[8159n2'  # Replace with your actual Wi-Fi password

# Connect to Wi-Fi
wifi = network.WLAN(network.STA_IF)
wifi.active(True)
wifi.connect(wifi_ssid, wifi_password)

# Wait for Wi-Fi connection
while not wifi.isconnected():
    pass

# Initialize LED and buzzer
led_pin.off()
buzzer_pin.off()

# Firebase Realtime Database URL and secret
firebase_url = 'https://flood-monitoring-8b6ff-default-
rtdb.firebaseio.com/'  # Replace with your Firebase URL
firebase_secret = 'AIzaSyC9Se75T8i4tsWsboC4pWELewpXKOITRFY'  # Replace
with your Firebase secret

# Adjust this threshold according to your needs
safe_water_level = 37

def measure_distance():
    # Trigger the ultrasonic sensor
    trigger_pin.on()
    time.sleep_us(10)
    trigger_pin.off()

    # Measure the pulse width of the echo signal
    pulse_time = machine.time_pulse_us(echo_pin, 1, 30000)

    # Calculate distance in centimeters
    speed_of_sound = 34300  # Speed of sound in cm/s
    distance_cm = (pulse_time * speed_of_sound) / 2 # Adjust the
conversion factor
```

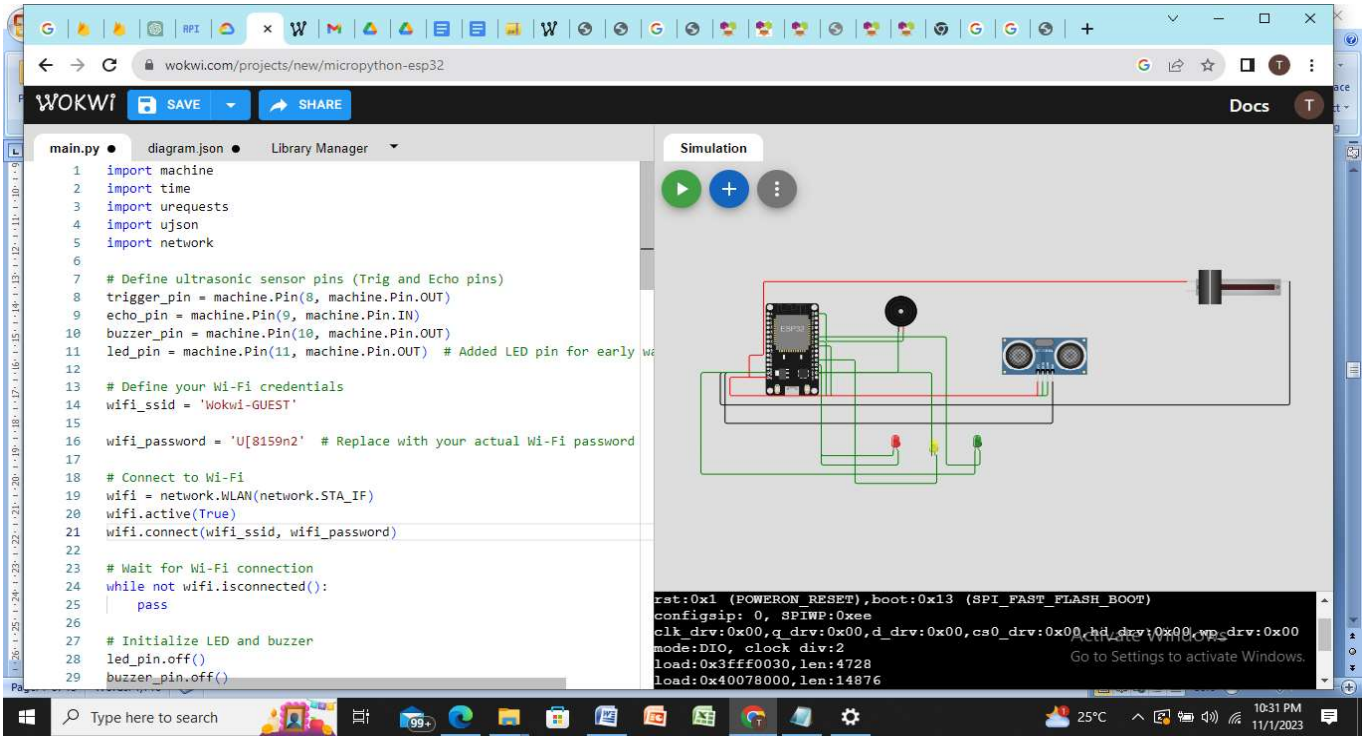
```
        return distance_cm

    def send_data_to_firebase(distance):
        data = {
            "waterLevel": distance
        }
        url = f'{firebase_url}/sensor_data.json?auth={firebase_secret}'

        try:
            response = urequests.patch(url, json=data)
            if response.status_code == 200:
                print("Data sent to Firebase")
            else:
                print(f"Failed to send data to Firebase. Status code:
    {response.status_code}")
        except Exception as e:
            print(f"Error sending data to Firebase: {str(e)}")

    def flood_alert(distance):
        if distance < safe_water_level:
            print("Flood Alert!")
            buzzer_pin.on()
            led_pin.on()


            # Send data to Firebase when a flood alert is triggered
            send_data_to_firebase(distance)
        else:
            print("Safe")
            buzzer_pin.off()
            led_pin.off()

    try:
        while True:
            distance = measure_distance()
            print("Distance: {:.2f} cm".format(distance))
            flood_alert(distance)
            time.sleep(5)  # Delay for 5 seconds before the next reading

    except KeyboardInterrupt:
        print("Monitoring stopped")
```

diagram.json

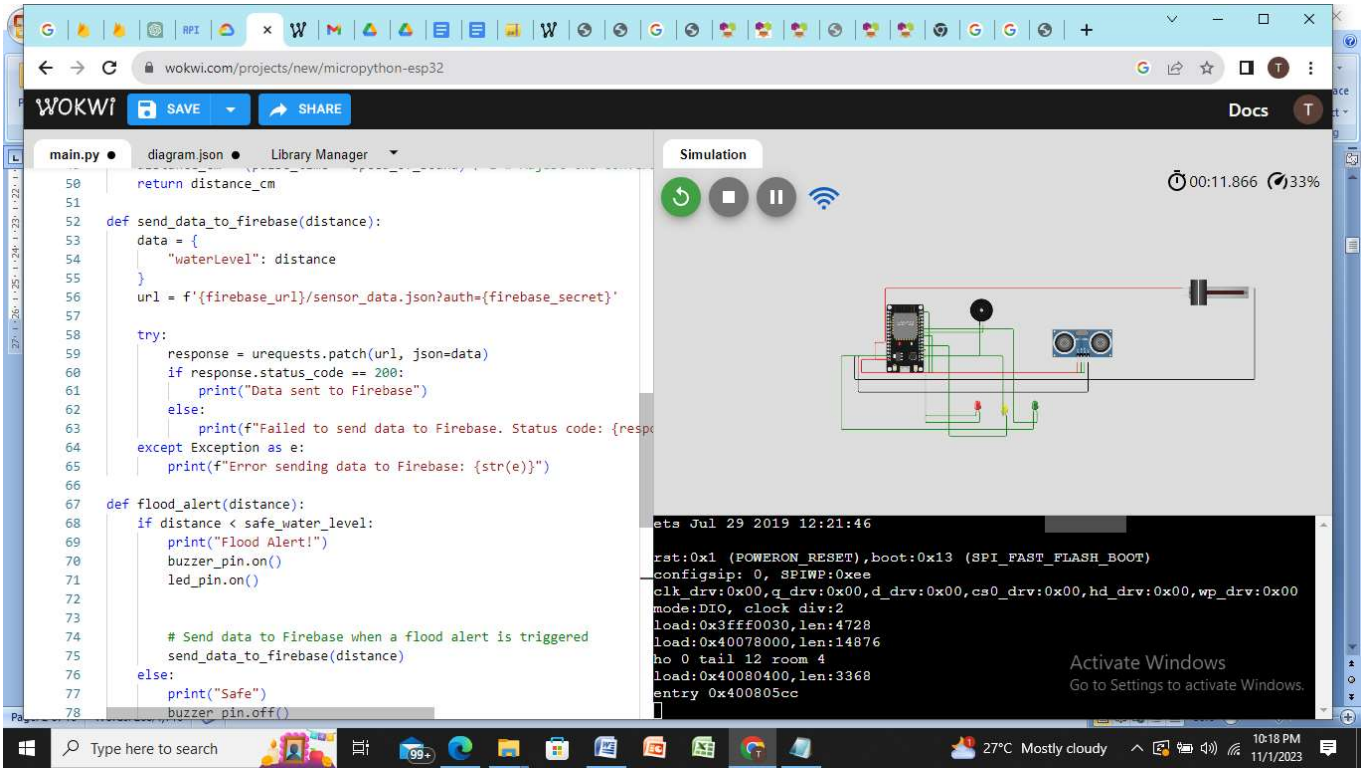
```
{
  "version": 1,
  "author": "Thivya",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": 110.3,
"left": -350.6, "attrs": {} },
    {
      "type": "wokwi-hc-sr04",
      "id": "ultrasonic1",
      "top": 183.9,
      "left": 120.7,
      "attrs": { "distance": "400" }
    },
    {
      "type": "wokwi-slide-potentiometer",
      "id": "pot1",
      "top": 24.2,
      "left": 488.6,
      "attrs": { "travelLength": "30" }
    },
    {
      "type": "wokwi-buzzer",
      "id": "bz1",
      "top": 88.8,
      "left": -113.4,
      "attrs": { "volume": "0.1" }
    },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": 380.4,
      "left": -111.4,
```

```
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-led",
      "id": "led2",
      "top": 390,
      "left": -34.6,
      "attrs": { "color": "yellow" }
    },
    {
      "type": "wokwi-led",
      "id": "led3",
      "top": 380.4,
      "left": 51.8,
      "attrs": { "color": "green" }
    }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "bz1:1", "esp:GND.1", "green", [ "v57.6" ] ],
    [ "bz1:2", "esp:D23", "green", [ "v19.2", "h-154", "v-57.6" ] ],
    [ "ultrasonic1:TRIG", "esp:D5", "green", [ "v28.8", "h-422.8", "v-
105.6" ] ],
    [ "ultrasonic1:ECHO", "esp:D18", "green", [ "v28.8", "h-442.4", "v-
115.2" ] ],
    [ "ultrasonic1:GND", "esp:GND.2", "black", [ "v86.4", "h-654", "v-
105.6" ] ],
    [ "ultrasonic1:VCC", "esp:VIN", "red", [ "v28.8", "h-614.4",
"v28.8" ] ],
    [ "pot1:GND", "esp:GND.2", "black", [ "v259.2", "h-1137.2", "v-
67.2" ] ],
    [ "pot1:VCC", "esp:VIN", "red", [ "h-844.8", "v153.6", "h-28.8",
"v-19.2" ] ],
    [ "led1:C", "esp:GND.1", "green", [ "v9.6", "h-143.6", "v-172.8" ]
],
    [ "led1:A", "esp:D2", "green", [ "v28.8", "h-153.6", "v-211.2" ] ],
    [ "led2:C", "esp:GND.1", "green", [ "v0" ] ],
    [ "led2:A", "esp:D4", "green", [ "v57.6", "h-163.2", "v-259.2" ] ],
    [ "led3:C", "esp:GND.2", "green", [ "v48", "h-546.8", "v-211.2" ]
],
```

```
      [ "led3:A", "esp:D19", "green", [ "v28.8", "h-67.2", "v-163.2" ] ]
    ],
    "dependencies": {}
  }
```

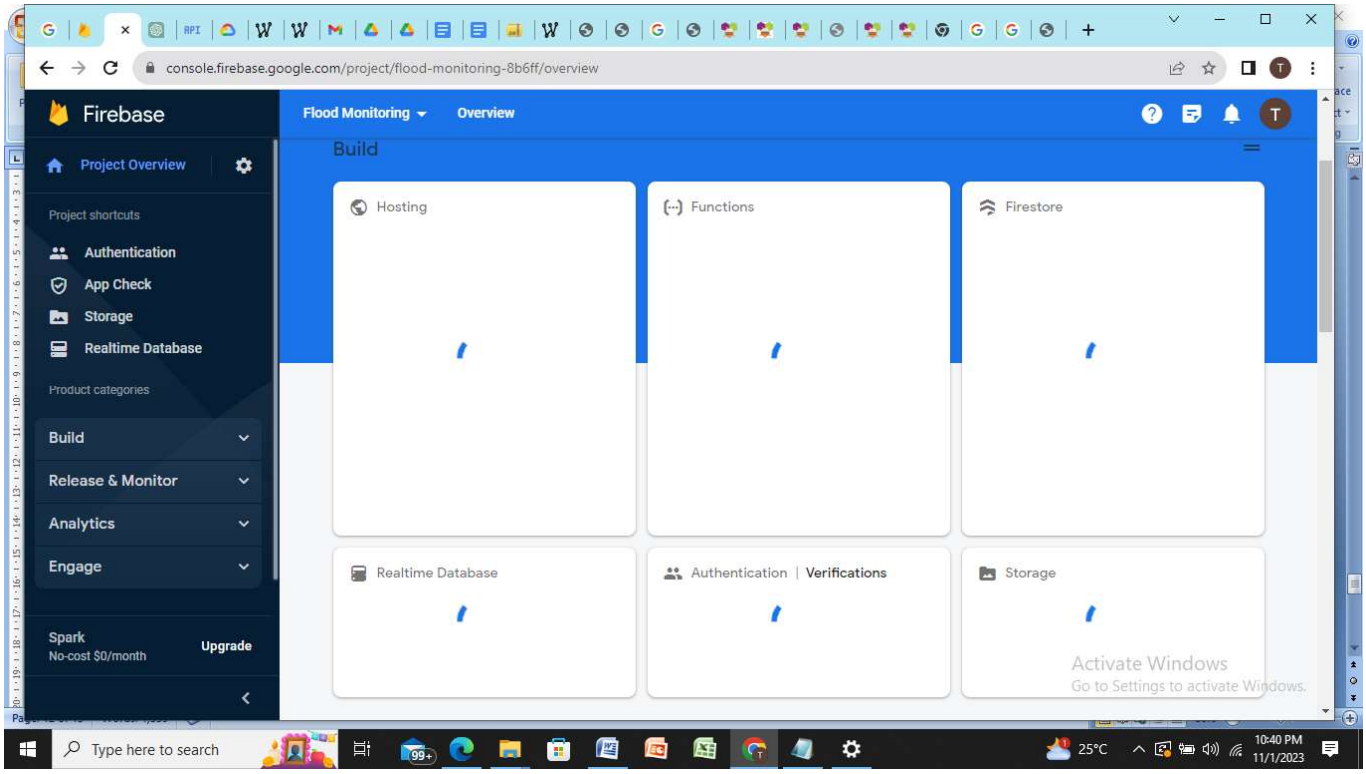


Wokwi platform Address :

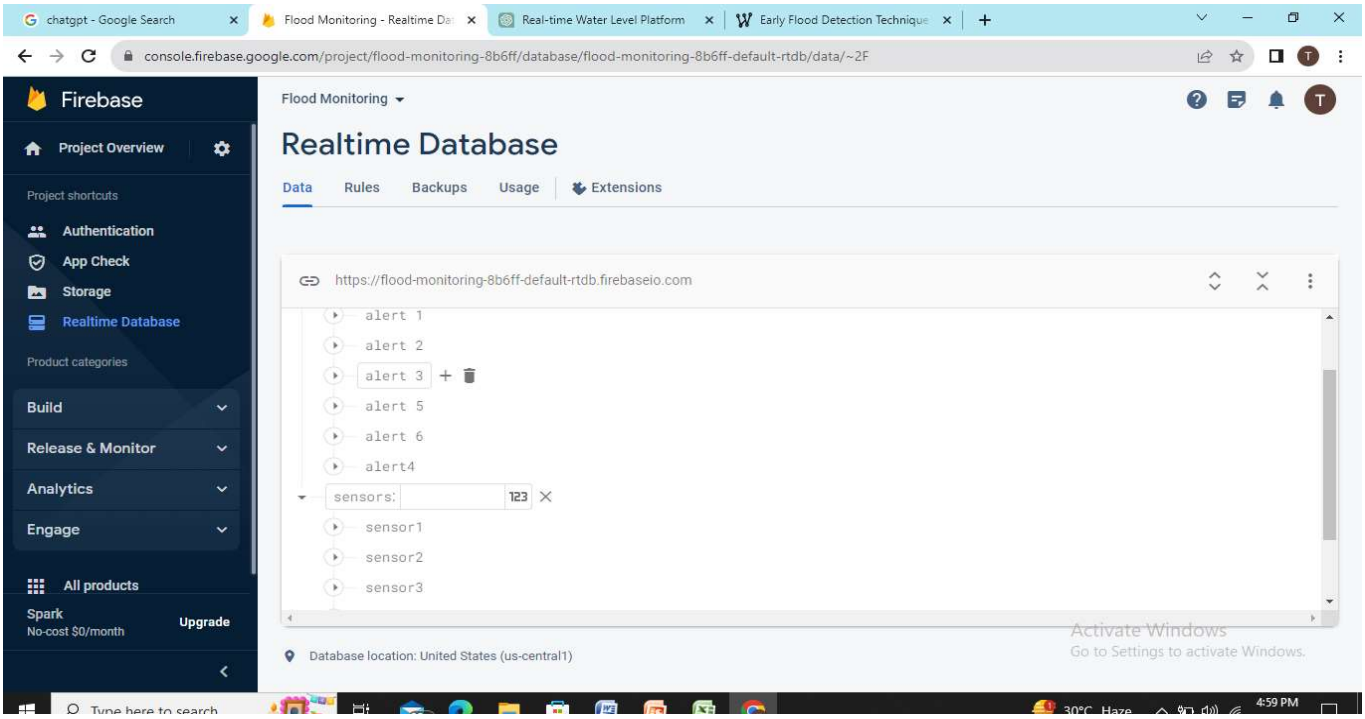


Firebase platform:

Firebase is a mobile and web application development platform that offers various tools and services like Real-time Data Sync to maintain a real-time connection between the server and clients. This means that any changes made to the database are instantly reflected in all connected clients without the need for manual refreshes.



Implement a real-time database and connect to our wokwi simulator to collect data.



The realtime database of firebase that linked to wokwi simulator by

Database_url: https://flood-monitoring-8b6ff-default-rtdb.firebaseio.com/

Platform development

Creating a platform to convey updated flood alerts using HTML, CSS, and JavaScript is an excellent approach for providing timely information to the public and authorities.Design a clean and user-friendly web interface using HTML and CSS to ensure easy navigation and readability.Use responsive design principles to make the platform accessible on various devices, including desktops, tablets, and smartphones. Develop a section on the platform's homepage to display flood alerts.Display alerts prominently using color coding and clear text to indicate the severity of the alert. Design the platform to handle a growing number of users and data as the service gains popularity.

INDEX.html:

```
<!DOCTYPE html>
        <html>
        <head>
```

```
    <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
    <h1>Flood Monitoring and Early Warning System</h1>
    <div id="map"></div>
    <div id="warning">
        <h2>Current Status:</h2>
        <p id="status">No flood alert at the moment</p>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

STYLE.css:

```
body {
    font-family: Arial, sans-serif;
    text-align: center;
    }
h1 {
    color: #333;
}
#map {
    width: 80%;
    height: 400px;
    margin: 0 auto;
}
#warning {
    background-color: #ff6666;
    padding: 20px;
    border-radius: 10px;
    margin-top: 20px;
    }
#status {
    color: #fff;
}
```

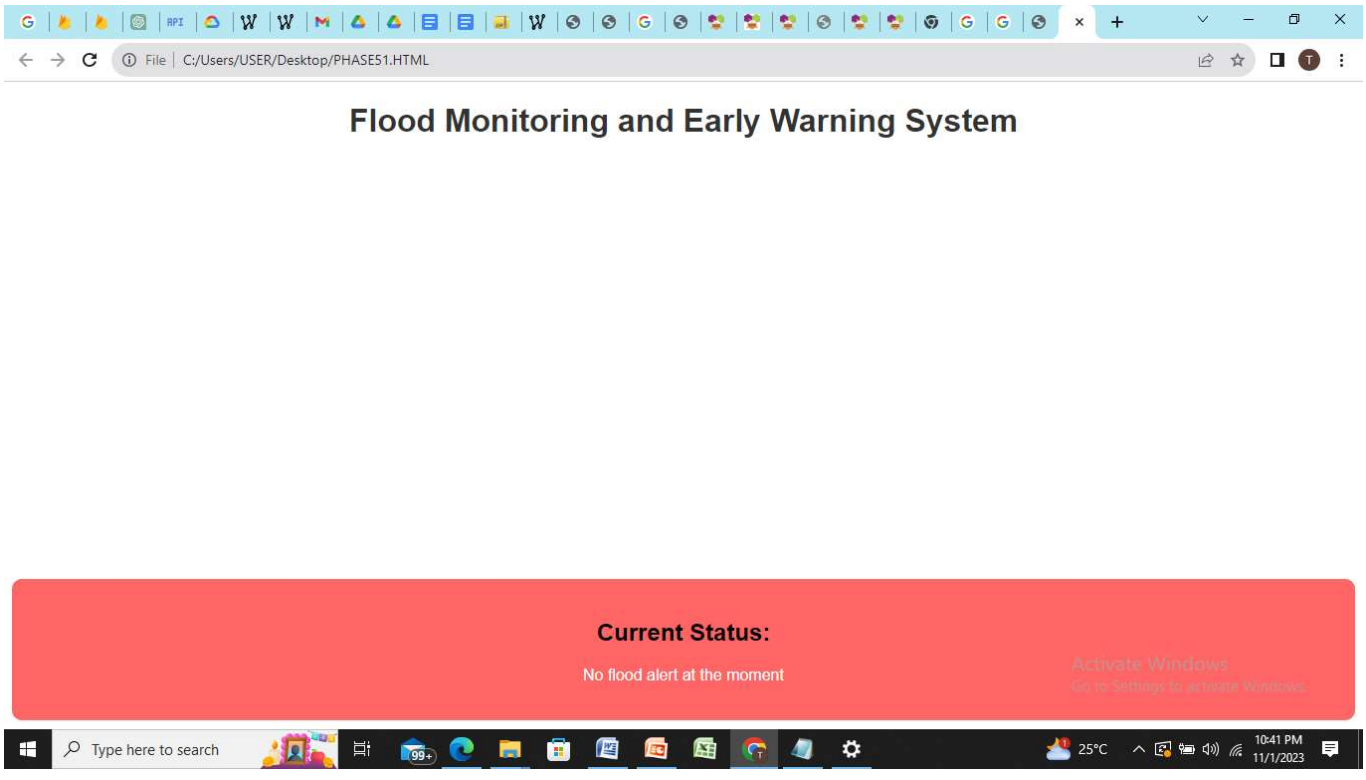
Script.js:

```
import { initializeApp } from "firebase/app";

import { getAnalytics } from "firebase/analytics";

let currentFloodStatus = 0;

var firebaseConfig = {

    apiKey: "AIzaSyC9Se75T8i4tsWsboC4pWELewpXKOITRFY",

    authDomain: "flood-monitoring-8b6ff.firebaseapp.com”,
```

```
    databaseURL: "https://flood-monitoring-8b6ff-default-rtdb.firebaseio.com/",
    projectId: "flood-monitoring-8b6ff",
    storageBucket: "flood-monitoring-8b6ff.appspot.com",
    messagingSenderId: "710537660792",
    appId: "1:710537660792:web:2880fd5c5e26b8a500608d",
    measurementId: "G-FQ96GJTWXY"
};
// Initialize Firebase
firebase.initializeApp(firebaseConfig);
// Reference to your Firebase Realtime Database
// Function to retrieve and display data
var infoGet = firebase.database().ref('sensor_data');
infoGet.on("value",function(snapshot){
    console.get(snapshot.val());
});
updateFloodStatus(infoGet);
// Function to update the flood status and display warnings
function updateFloodStatus(data) {
    currentFloodStatus = data;// Simulated data


    const statusElement = document.getElementById("status");
    if (currentFloodStatus === 1) {
```

```
        statusElement.textContent = "Flood alert! Take necessary precautions.\n Water level : 50";
        statusElement.style.color = "#ff0000";
    } else {
        statusElement.textContent = "No flood alert at the moment \n Water level :37";
        statusElement.style.color = "#fff";
    }
}// Simulate data update every 5 seconds
setInterval(updateFloodStatus, 5000);
```

Output



Public Safety and Awareness

- A real-time flood monitoring and early warning system plays a critical role in enhancing public safety and emergency response coordination by providing timely and accurate information to both the public and relevant authorities.
- Here's how such a system can have a positive impact. Real-time monitoring systems use various sensors and data sources (e.g., rainfall gauges, river level sensors, weather forecasts) to detect changes in water levels, weather conditions, and other relevant parameters.
- When unusual conditions indicative of a flood risk are detected, the system can trigger automated alerts to relevant authorities, emergency services, and the public through various communication channels, including mobile apps, SMS, emails, and sirens.
- Early warning systems provide the public with accurate and timely information about flood risks and developments, allowing individuals and communities to take preventive measures.
- People can receive alerts on their mobile devices, allowing them to prepare for evacuation, move valuable assets to safer locations, and stock up on essential supplies.