**NOISE POLLUTION MONITORING**

**Project objective:**

The IoT-based flood warning and early warning system project aims to collect real-time data on water levels and other relevant parameters in flood-prone areas and use that information for analysis and timely warnings to mitigate potential flood risks. This project typically involves the deployment of sensors, data processing, and reporting mechanisms.

The objective of an IoT (Internet of Things) based flood warning and early warning system is to monitor, analyze, and provide early warnings for potential flooding in a given area using connected sensors and devices. Flooding can pose significant risks to communities and infrastructure, and an IoT-based solution can offer critical information and tools to help prevent and manage flood-related emergencies.

**IOT Device setup:**

* **Deployment of sensors:**

Ultrasonic water level sensors, ESP32 microcontrollers, and warning buzzer devices are strategically placed in flood-prone areas of the city or region under study. The ultrasonic sensors measure water levels, the ESP32 microcontrollers collect and process data on water levels, rainfall, and environmental conditions, and the warning buzzers are used to provide audible alerts, allowing for the early detection and continuous monitoring of potential flooding events.

* **Integrate data:**

The water level and environmental sensors, including ultrasonic water level sensors and ESP32 microcontrollers, collect data continuously and transmit it to a central server or cloud platform using wireless communication protocols like Wi-Fi or cellular connectivity. This approach ensures real-time monitoring capabilities. The collected data, which includes information on water levels, rainfall, and environmental conditions, is processed and analyzed to detect changes and patterns related to potential flooding events, providing early warning and monitoring of flood risks.

* **Monitoring :**

If the water levels detected by the sensors exceed predefined thresholds, the system will initiate a notification process. Implement an alert system to promptly inform the relevant authorities and residents when water levels indicate potential flooding. Explore and implement flood mitigation techniques, such as the activation of flood barriers, traffic rerouting, or adjustments to construction activities. IoT data can be leveraged to assess the effectiveness of these strategies in preventing or mitigating flood events.

**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, ESP32,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 = ''  # 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://iot-ibm-669fd-default-rtdb.firebaseio.com'  # Replace with your Firebase URL

firebase\_secret = 'AIzaSyCLAxyN8n069Civr2LMZmRWxgQPVcEmFjQ'  # 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": "suriya",

  "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" ] ],

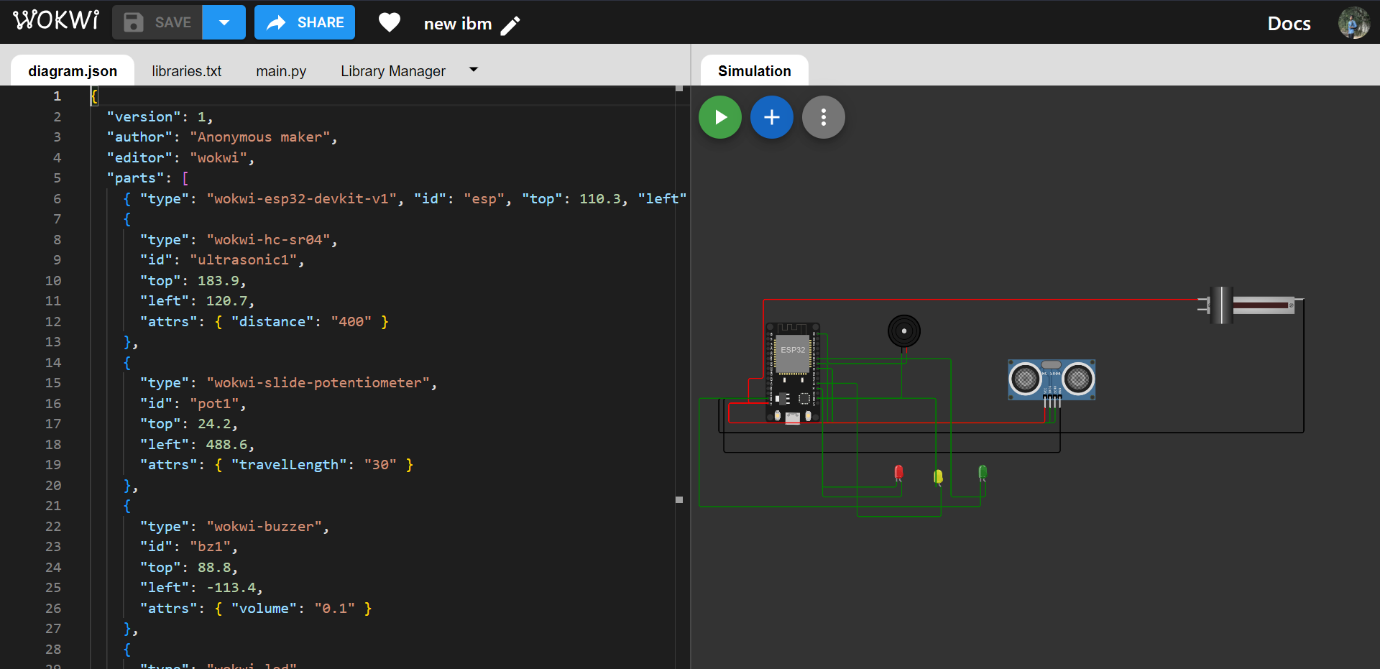
    [ "led3:C", "esp:GND.2", "green", [ "v48", "h-546.8", "v-211.2" ] ],

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

  ],

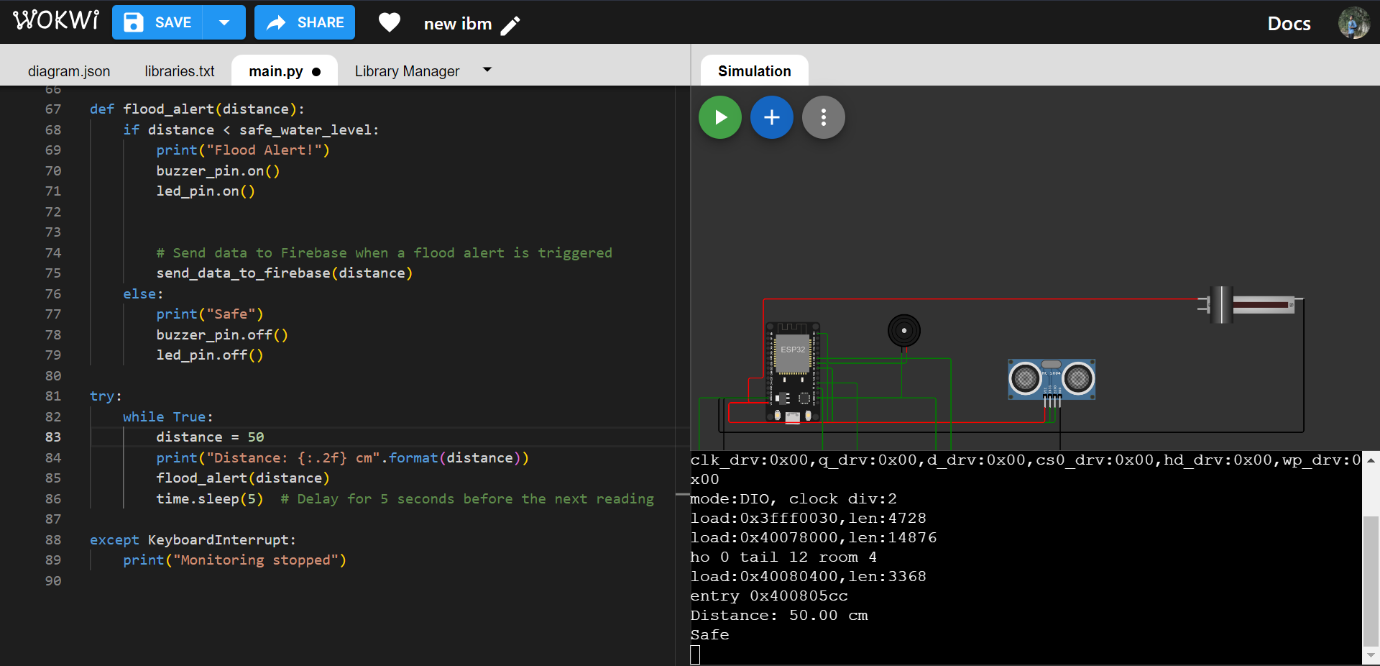
  "dependencies": {}

}



**Wokwi platform address:**

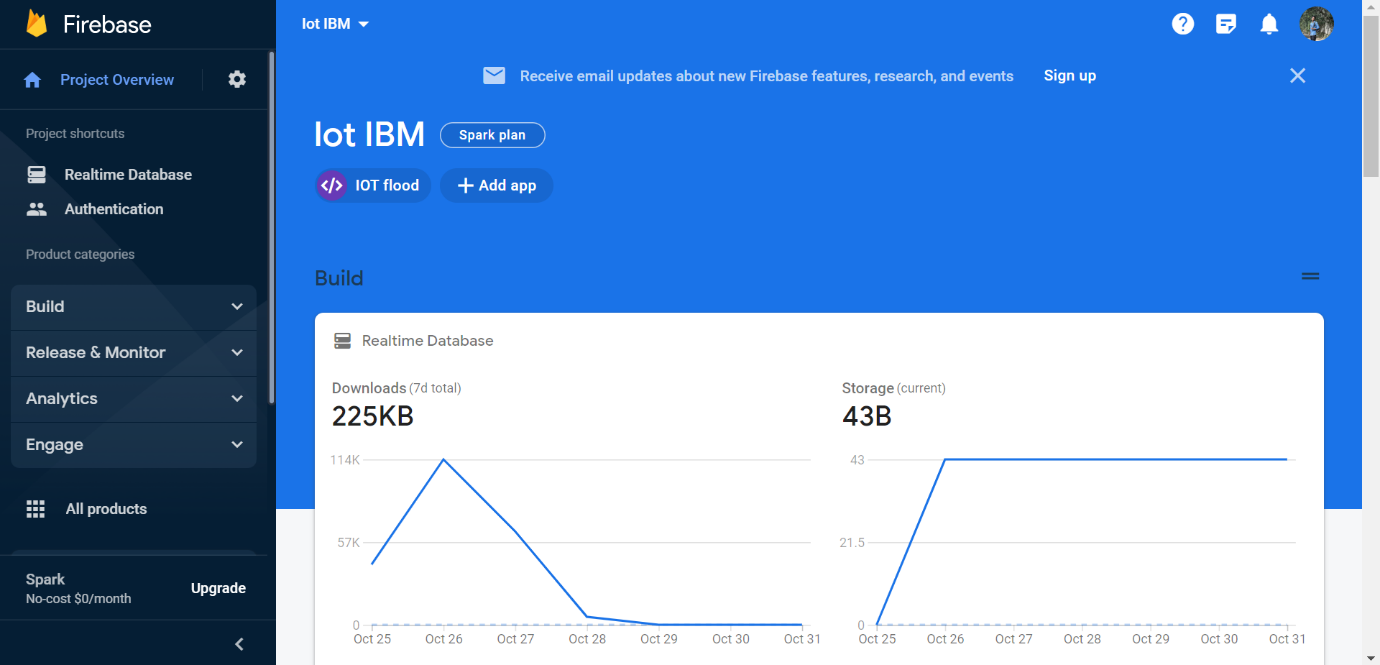
https://wokwi.com/projects/379659587335048193



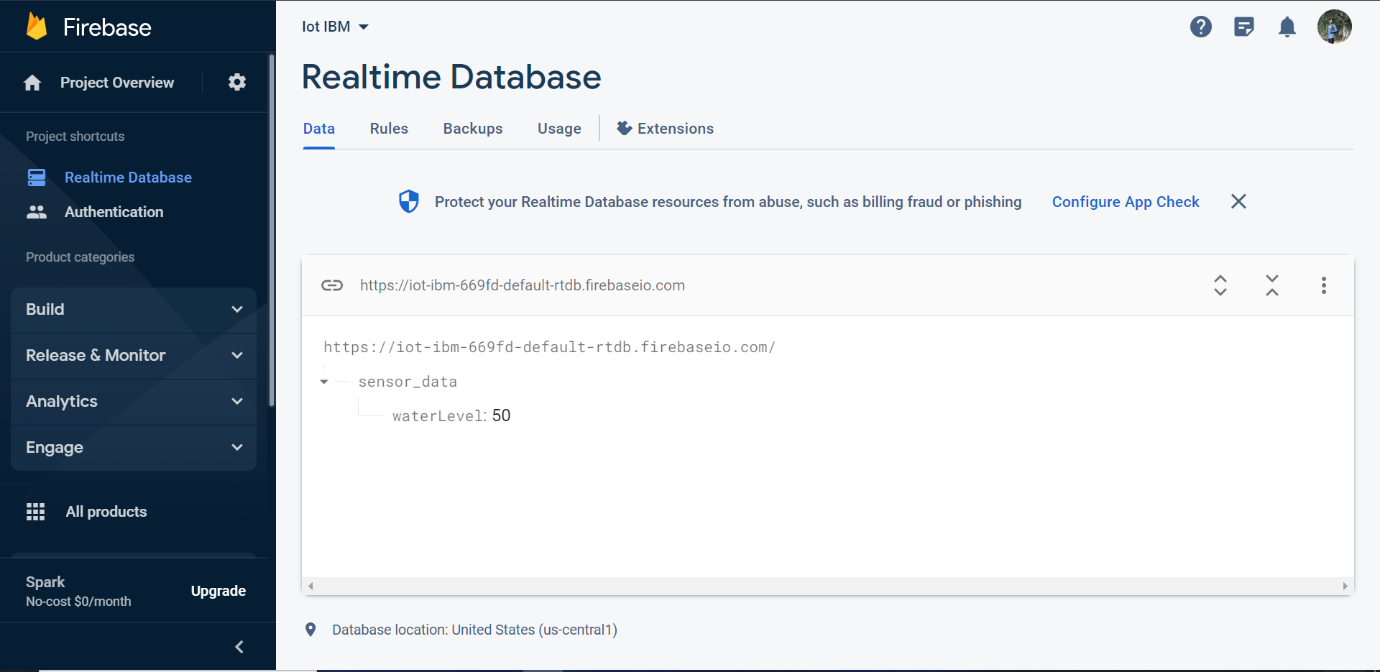
**Firebase platform:**

* 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://iot-ibm-669fd-default-rtdb.firebaseio.com>
* Database\_secret: AIzaSyCLAxyN8n069Civr2LMZmRWxgQPVcEmFjQ

**WEB SITE DEVELOPMENT:**

**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: "AIzaSyCLAxyN8n069Civr2LMZmRWxgQPVcEmFjQ",

authDomain: "iot-ibm-669fd.firebaseapp.com",

databaseURL: "https://iot-ibm-669fd-default-rtdb.firebaseio.com",

projectId: "iot-ibm-669fd",

storageBucket: "iot-ibm-669fd.appspot.com",

messagingSenderId: "980279781293",

appId: "1:980279781293:web:e8a9edd13aa071829a2324",

measurementId: "G-XDQQVJZM3Y"

};

// 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";

}

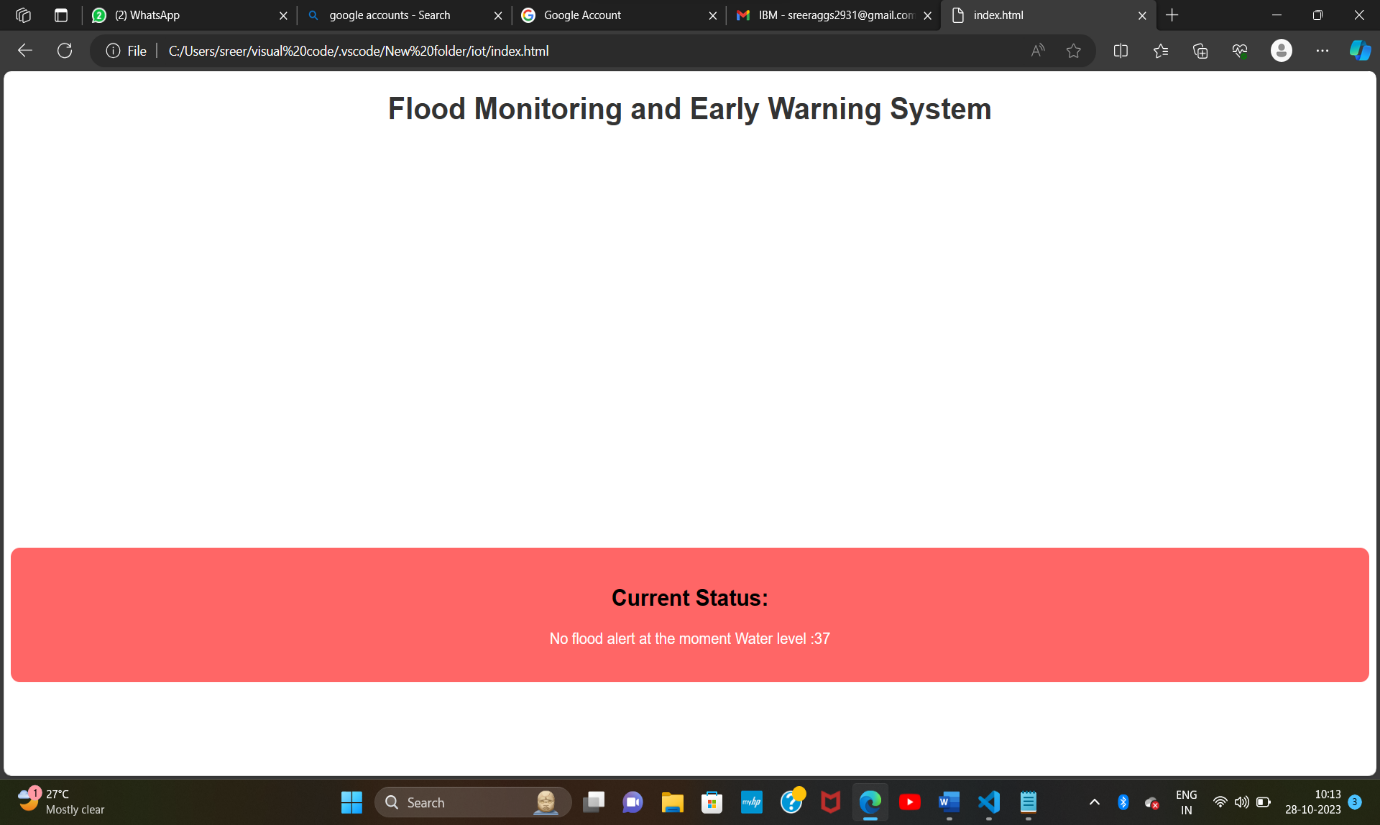
}

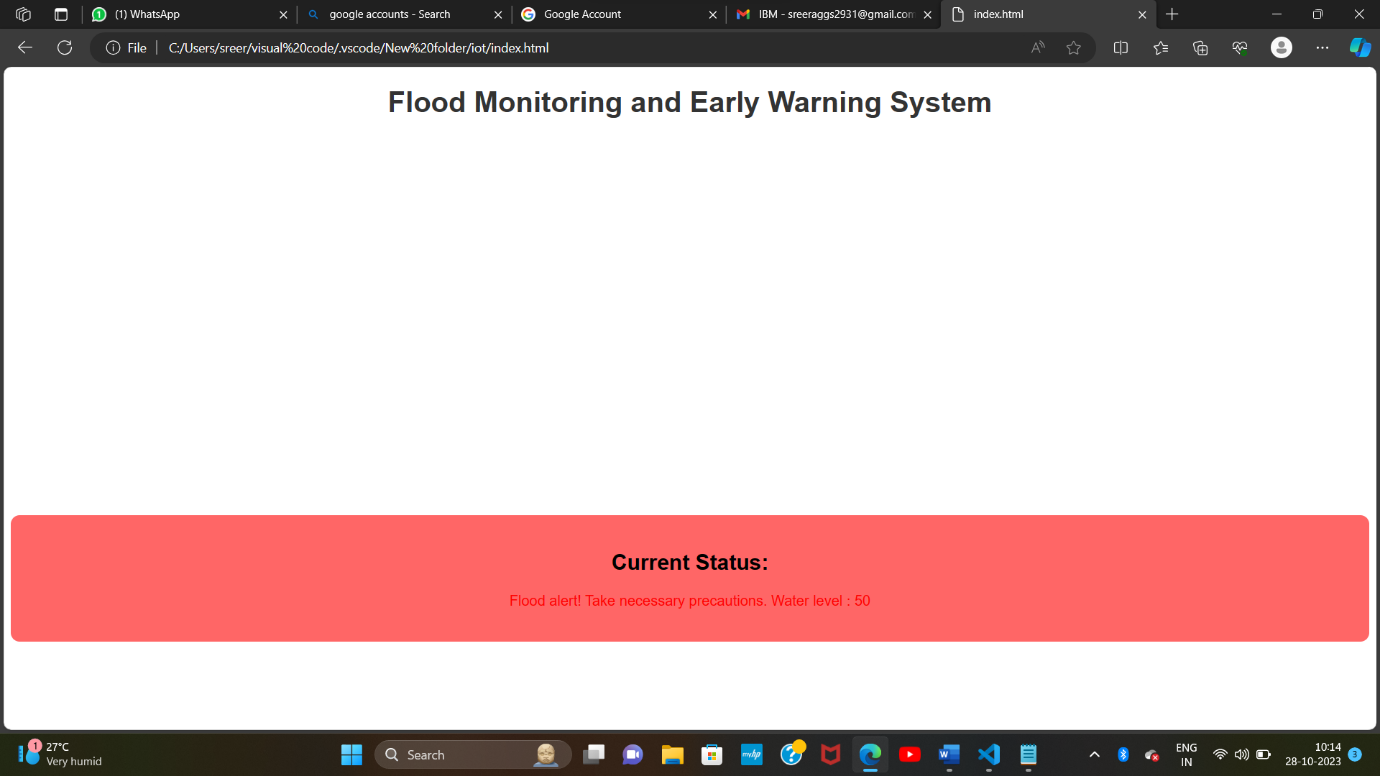
// Simulate data update every 5 seconds

setInterval(updateFloodStatus, 5000);

// Initialize the map (You can use mapping libraries like Leaflet or Google Maps for real maps)

**OUTPUT:**

****

****