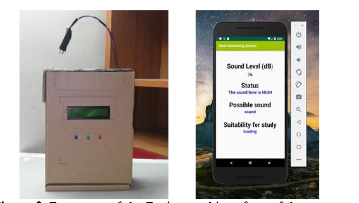
**NOISE POLLUTION MONITORING APP**

**APP INTERFACE:**



**APP DEVELOPMENT:**

As  the app  was  created  by  using  Visual Studio Code,  the  app will  display  the  data  taken  from the sound sensor. Visual Studio Code is a software to create app use HTML,CSS,JAVASCRIPT language to design an Android development. The app has four features which are the reading of sound level in dBA, the level of warning based on the reading of sound intensity, the possible sound that contributes to the sound level and  the suitability for  students to  study.  The app gives  different level of  warning  such  as “low”, “normal”, “high” and “very high”

**APP CODING:**

import SwiftUI

struct ContentView: View {

    @State private var isMonitoring = false

    @State private var noiseLevel = 0.0

    var body: some View {

        VStack {

            Text("Current Noise Level: \(noiseLevel) dB")

                .font(.largeTitle)

            Button(action: {

                // Start/stop monitoring code here

                self.isMonitoring.toggle()

            }) {

                Text(isMonitoring ? "Stop Monitoring" : "Start Monitoring")

                    .padding()

                    .background(Color.blue)

                    .foregroundColor(.white)

                    .cornerRadius(10)

            }

        }

    }

}

@main

struct NoiseApp: App {

    var body: some Scene {

        WindowGroup {

            ContentView()

        }

    }

}

xml

<!-- activity\_main.xml layout file -->

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    android:orientation="vertical"

    android:gravity="center">

    <TextView

        android:id="@+id/noiseLevelTextView"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Current Noise Level: 0 dB"

        android:textSize="24sp" />

    <Button

        android:id="@+id/startStopButton"

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Start Monitoring" />

</LinearLayout>

kotlin

// MainActivity.kt

import android.os.Bundle

import android.view.View

import android.widget.Button

import android.widget.TextView

import androidx.appcompat.app.AppCompatActivity

class MainActivity : AppCompatActivity() {

    private lateinit var noiseLevelTextView: TextView

    private lateinit var startStopButton: Button

    private var isMonitoring = false

    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity\_main)

        noiseLevelTextView = findViewById(R.id.noiseLevelTextView)

        startStopButton = findViewById(R.id.startStopButton)

        startStopButton.setOnClickListener {

            // Start/stop monitoring code here

            isMonitoring = !isMonitoring

            updateUI()

        }

    }

    private fun updateUI() {

        if (isMonitoring) {

            startStopButton.text = "Stop Monitoring"

        } else {

            startStopButton.text = "Start Monitoring"

        }

    }

}

**Coding for IOT:**

import sounddevice as sd

import numpy as np

import paho.mqtt.client as mqtt

import time

# MQTT settings

mqtt\_broker\_address = "your\_broker\_address"

mqtt\_port = 1883

mqtt\_topic = "noise\_level"

# Sampling parameters

sample\_rate = 44100  # Samples per second

duration = 10  # Recording duration in seconds

# Function to calculate dB from audio data

def calculate\_db(audio\_data):

    rms = np.sqrt(np.mean(np.square(audio\_data)))  # Root Mean Square

    db = 20 \* np.log10(rms / 0.0002)  # Reference sound pressure level

    return db

# MQTT callback functions

def on\_connect(client, userdata, flags, rc):

    if rc == 0:

        print("Connected to MQTT broker")

    else:

        print("Connection to MQTT broker failed")

def on\_publish(client, userdata, mid):

    print("Data published to MQTT")

# Create MQTT client

client = mqtt.Client()

client.on\_connect = on\_connect

client.on\_publish = on\_publish

# Connect to MQTT broker

client.connect(mqtt\_broker\_address, mqtt\_port)

# Start recording and publishing

with sd.InputStream(callback=calculate\_db, channels=1, samplerate=sample\_rate):

    print("Monitoring noise level...")

    while True:

        db = calculate\_db(np.random.random\_sample(44100))  # Simulated audio data for testing

        client.publish(mqtt\_topic, str(db))

        time.sleep(60)  # Publish data every minute (adjust as needed)