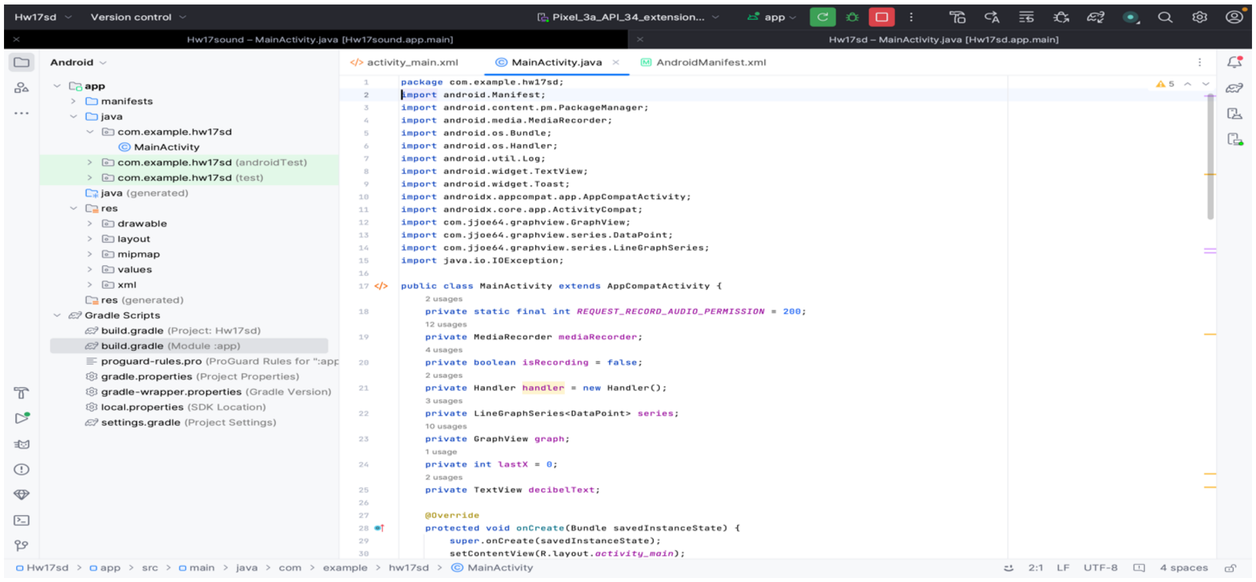
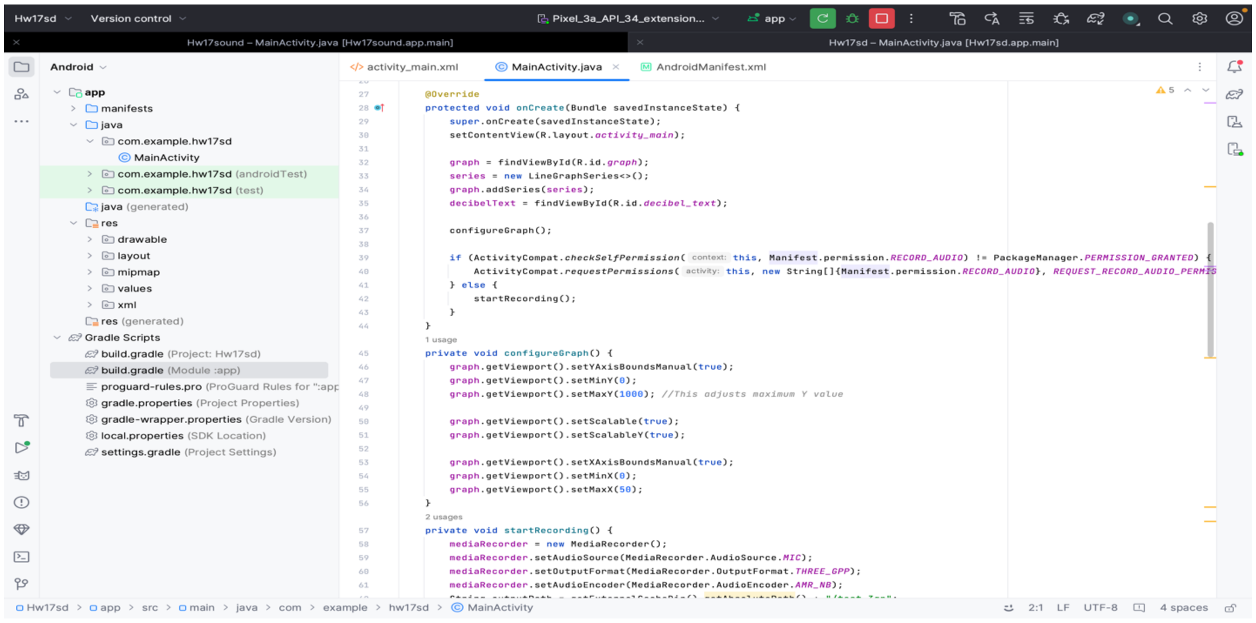
Build a sound level detection app.
* Extension of the audio capture app, do not save audio
* Use maximum amplitude, and display the sound level graphically using line chart or bar chart
*https://developer.xamarin.com/api/property/Android.Media.MediaRecorder.MaxAmplitude/

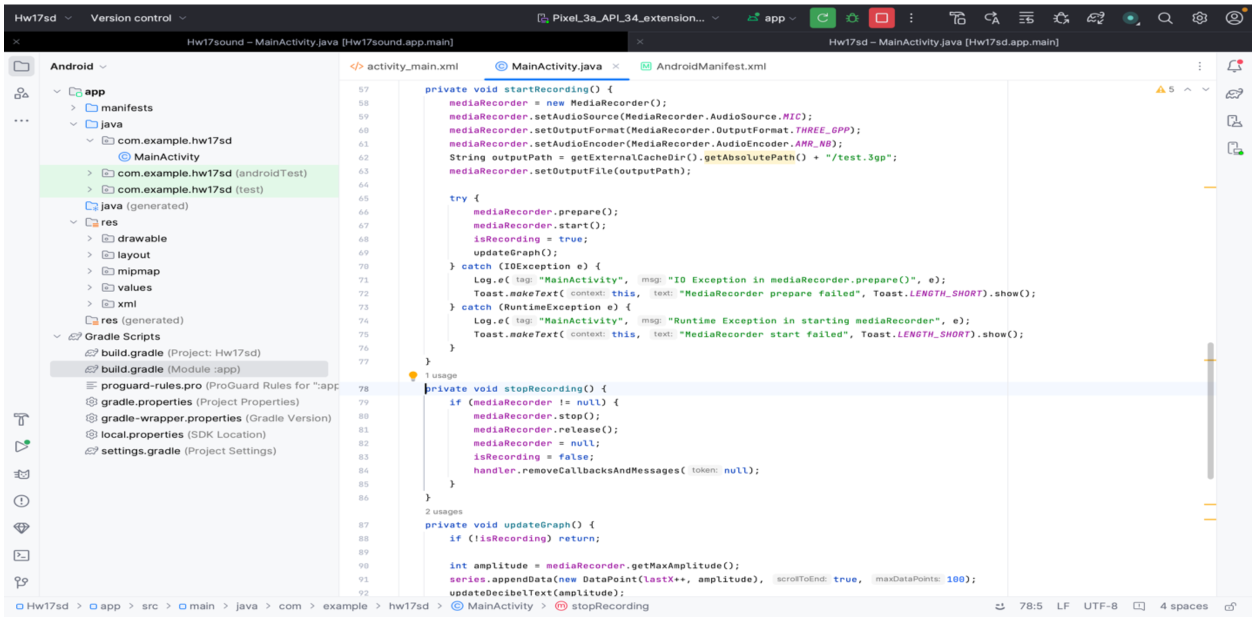
MainActivity.java : This class in this Android app manages audio monitoring and visualization. The essential methods that control audio capture with 'MediaRecorder' are 'startRecording' and 'stopRecording'. The software actively monitors and visualizes audio input, converting sound amplitudes into a real-time graph in a 'GraphView' and calculating matching decibel levels in a 'TextView'. This visualization is refreshed at regular intervals by the 'updateGraph' method, which is invoked at regular intervals. The program also manages audio recording permissions, asking them as needed and appropriately reacting to user approvals or denials, guaranteeing compliance with Android's security rules. The app's fundamental feature is defined by the seamless integration of audio capture, visualization, and permission control, which provides users with an interactive and useful auditory analysis tool.



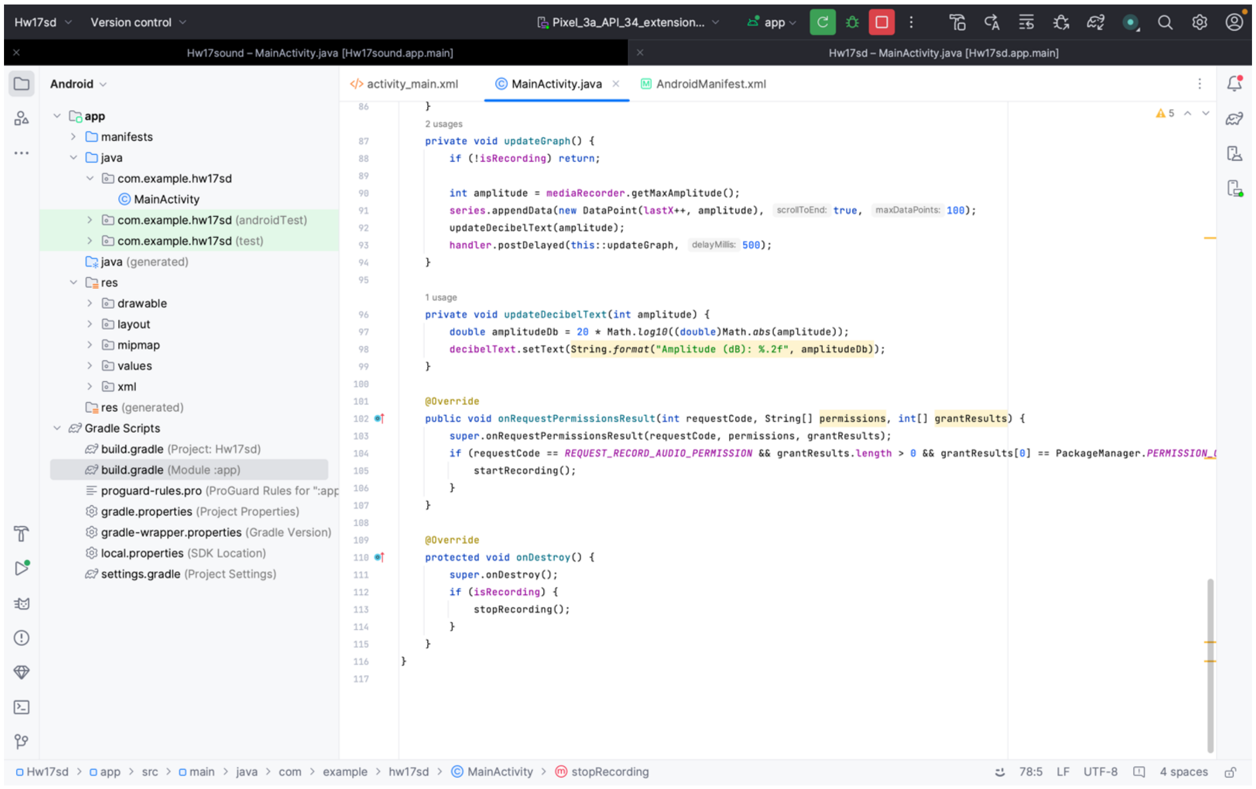
```java
package com.example.hw17sd;
import android.Manifest;
import android.content.pm.PackageManager;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {
    private static final int REQUEST_RECORD_AUDIO_PERMISSION = 200;
    private MediaRecorder mediaRecorder;
    private boolean isRecording = false;
    private Handler handler = new Handler();
    private LineGraphSeries<DataPoint> series;
    private GraphView graph;
    private int lastX = 0;
    private TextView decibelText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
```



```java
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        graph = findViewById(R.id.graph);
        series = new LineGraphSeries<>();
        graph.addSeries(series);
        decibelText = findViewById(R.id.decibel_text);

        configureGraph();

        if (ActivityCompat.checkSelfPermission( this, Manifest.permission.RECORD_AUDIO) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions( this, new String[]{Manifest.permission.RECORD_AUDIO}, REQUEST_RECORD_AUDIO_PERMIS
        } else {
            startRecording();
        }
    }
    private void configureGraph() {
        graph.getViewport().setYAxisBoundsManual(true);
        graph.getViewport().setMinY(0);
        graph.getViewport().setMaxY(1000); //This adjusts maximum Y value

        graph.getViewport().setScalable(true);
        graph.getViewport().setScalableY(true);

        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMinX(0);
        graph.getViewport().setMaxX(50);
    }
    private void startRecording() {
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

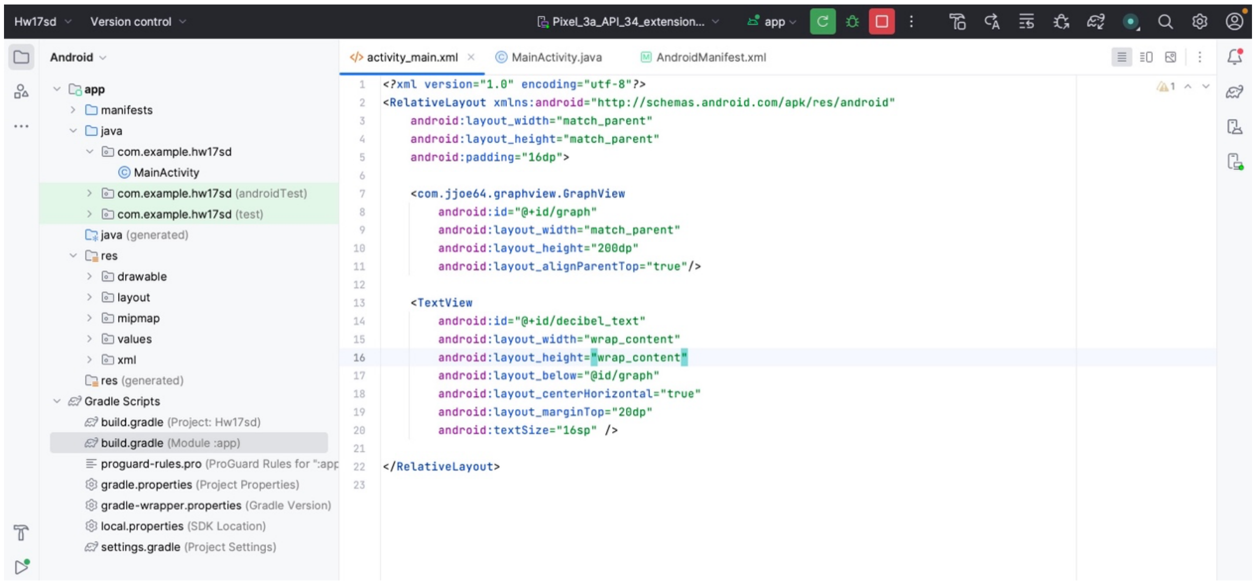


```java
    private void startRecording() {
        mediaRecorder = new MediaRecorder();
        mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
        mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
        String outputPath = getExternalCacheDir().getAbsolutePath() + "/test.3gp";
        mediaRecorder.setOutputFile(outputPath);

        try {
            mediaRecorder.prepare();
            mediaRecorder.start();
            isRecording = true;
            updateGraph();
        } catch (IOException e) {
            Log.e( tag: "MainActivity",  msg: "IO Exception in mediaRecorder.prepare()", e);
            Toast.makeText( context: this,  text: "MediaRecorder prepare failed", Toast.LENGTH_SHORT).show();
        } catch (RuntimeException e) {
            Log.e( tag: "MainActivity",  msg: "Runtime Exception in starting mediaRecorder", e);
            Toast.makeText( context: this,  text: "MediaRecorder start failed", Toast.LENGTH_SHORT).show();
        }
    }
    1 usage
    private void stopRecording() {
        if (mediaRecorder != null) {
            mediaRecorder.stop();
            mediaRecorder.release();
            mediaRecorder = null;
            isRecording = false;
            handler.removeCallbacksAndMessages( token: null);
        }
    }
    2 usages
    private void updateGraph() {
        if (!isRecording) return;

        int amplitude = mediaRecorder.getMaxAmplitude();
        series.appendData(new DataPoint(lastX++, amplitude),  scrollToEnd: true,  maxDataPoints: 100);
        updateDecibelText(amplitude);
```



```java
    }
    2 usages
    private void updateGraph() {
        if (!isRecording) return;

        int amplitude = mediaRecorder.getMaxAmplitude();
        series.appendData(new DataPoint(lastX++, amplitude),  scrollToEnd: true,  maxDataPoints: 100);
        updateDecibelText(amplitude);
        handler.postDelayed(this::updateGraph,  delayMillis: 500);
    }

    1 usage
    private void updateDecibelText(int amplitude) {
        double amplitudeDb = 20 * Math.log10((double)Math.abs(amplitude));
        decibelText.setText(String.format("Amplitude (dB): %.2f", amplitudeDb));
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == REQUEST_RECORD_AUDIO_PERMISSION && grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_
            startRecording();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        if (isRecording) {
            stopRecording();
        }
    }
}
```

Activity_main.xml : This XML layout has a 'RelativeLayout' as its base and contains two key UI elements. The 'GraphView' component, which occupies a fixed height at the top, is intended for real-time graphical representation of audio data. A 'TextView' is positioned below the graph to display textual information such as decibel levels. It is centrally aligned below the graph and dynamically updates in sync with the audio data being processed. This layout mixes graphical and textual data presentations in a seamless and user-friendly manner.


Hw17sd
Version control
Pixel_3a_API_34_extension...
app
Android
activity_main.xml
MainActivity.java
AndroidManifest.xml
app
manifests
java
com.example.hw17sd
MainActivity
com.example.hw17sd (androidTest)
com.example.hw17sd (test)
java (generated)
res
drawable
layout
mipmap
values
xml
res (generated)
Gradle Scripts
build.gradle (Project: Hw17sd)
build.gradle (Module :app)
proguard-rules.pro (ProGuard Rules for ":app
gradle.properties (Project Properties)
gradle-wrapper.properties (Gradle Version)
local.properties (SDK Location)
settings.gradle (Project Settings)

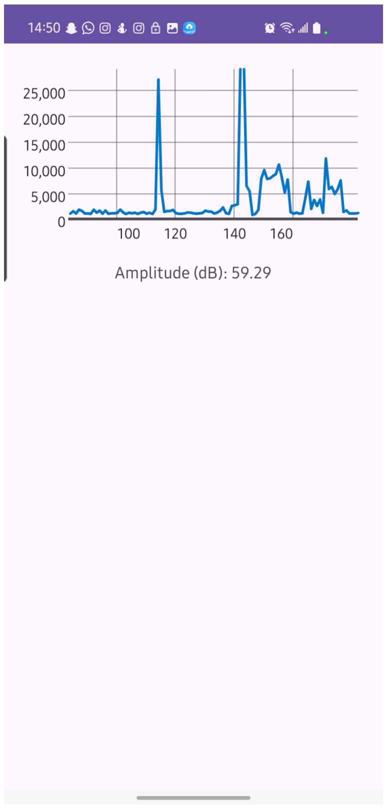
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">

    <com.jjoe64.graphview.GraphView
        android:id="@+id/graph"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:layout_alignParentTop="true"/>

    <TextView
        android:id="@+id/decibel_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/graph"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="20dp"
        android:textSize="16sp" />

</RelativeLayout>
```

Android_manifest.xml : The lines in the Android manifest file 'uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>' and 'uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>' are permissions requests for accessing the device's external storage. These permissions are required for the app to work, which includes reading and writing data to the device's external storage, such as saving audio recordings or accessing files for processing.

Build.gradle : The line 'implementation 'com.jjoe64:graphview:4.2.2'' is included in the app's Build.gradle file to integrate the GraphView library, which provides the functionality to display real-time graphs in the app. 'implementation 'androidx.core:core:1.12.0'' is also used to include the AndroidX Core library, which provides crucial functionalities for current Android development, such as compatibility and utility APIs. These dependencies are essential for improving the app's graphical data display and assuring compatibility with the latest Android features and standards.

Gradle.properties : The line 'android.enableJetifier=true' is included in the gradle.properties file to enable Jetifier. Jetifier automatically changes existing third-party libraries to be compatible with AndroidX. This ensures that libraries that rely on the older Android Support Library are effortlessly updated to work with the modern AndroidX architecture, preserving app compatibility and functionality.

Additionally, I added a TextView which shows the sound in decibels so just to know that the mic is working below is the screen shot of the app and I teste it on my device as the android studio emulator is not reliable for testing hardware.i even added the zoom in and zoom out feature so it will be easy to to see the values as the graph amplitude is varying with the change in sound.



Amplitude (dB): 59.29