

Build the pedometer app using the step detector sensor.

Build the sensor listing app using the code provided. In the homework submission please include the Screenshot for all sensors that are supported by your phone. Please show at least three sensors and their values.

MainActivity.java : This Android application's 'MainActivity' class uses the 'SensorManager' to interface with the device's hardware sensors, specifically the accelerometer, gyroscope, and light sensor. Upon launch, the program initializes its user interface and displays information about these sensors, such as their name, kind, vendor, and technical specs, using a 'TextView'. To manage sensor listening properly, lifecycle methods 'onResume()' and 'onPause()' are overridden, registering the app as a listener when in the foreground and unregistering when paused, optimizing for battery economy. While the code offers the architecture for responding to sensor data and accuracy changes via the 'onSensorChanged' and 'onAccuracyChanged' methods, these are not currently implemented.

```

1 package com.example.hw11_2844629;
2
3 import android.content.Context;
4 import android.hardware.Sensor;
5 import android.hardware.SensorEvent;
6 import android.hardware.SensorEventListener;
7 import android.hardware.SensorManager;
8 import android.os.Bundle;
9 import android.widget.TextView;
10 import androidx.appcompat.app.AppCompatActivity;
11 import com.example.hw11_2844629.R;
12 import java.util.List;
13
14 public class MainActivity extends AppCompatActivity implements SensorEventListener {
15     private SensorManager sensorManager;
16     private TextView sensorDataTextView;
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_main);
22
23         sensorDataTextView = findViewById(R.id.sensorData);
24         sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
25
26         int[] sensorTypes = {Sensor.TYPE_ACCELEROMETER, Sensor.TYPE_GYROSCOPE, Sensor.TYPE_LIGHT};
27
28         StringBuilder sensorInfo = new StringBuilder();
29         for (int sensorType : sensorTypes) {
30             Sensor sensor = sensorManager.getDefaultSensor(sensorType);
31             if (sensor != null) {
32                 sensorInfo.append("Sensor Name: ").append(sensor.getName()).append("\n");
33                 sensorInfo.append("Sensor Type: ").append(sensor.getStringType()).append("\n");
34                 sensorInfo.append("Vendor: ").append(sensor.getVendor()).append("\n");
35                 sensorInfo.append("Version: ").append(sensor.getVersion()).append("\n");
36                 sensorInfo.append("Power: ").append(sensor.getPower()).append(" mW\n");
37                 sensorInfo.append("Resolution: ").append(sensor.getResolution()).append("\n");
38                 sensorInfo.append("Maximum Range: ").append(sensor.getMaximumRange()).append("\n\n");
39             }
40         }
41     }
42 }

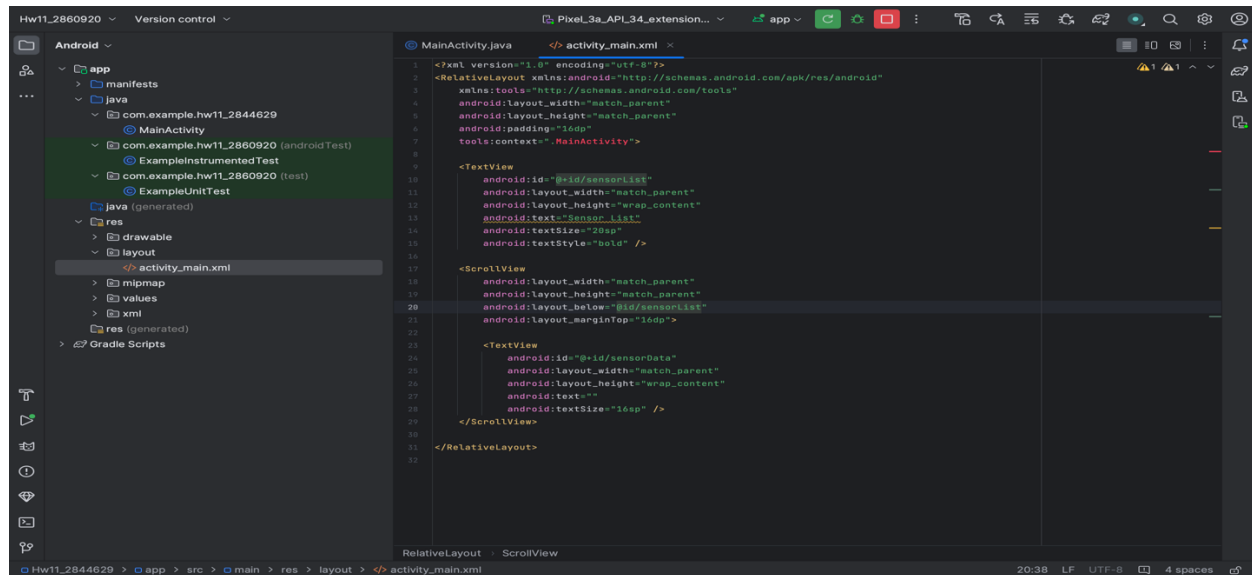
```

```

39             }
40         }
41
42         sensorDataTextView.setText(sensorInfo.toString());
43     }
44
45     @Override
46     protected void onResume() {
47         super.onResume();
48
49         sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER), SensorManager.SENSOR_DELAY_NORMAL);
50         sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE), SensorManager.SENSOR_DELAY_NORMAL);
51         sensorManager.registerListener(this, sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT), SensorManager.SENSOR_DELAY_NORMAL);
52     }
53
54     @Override
55     protected void onPause() {
56         super.onPause();
57         sensorManager.unregisterListener(this);
58     }
59
60     @Override
61     public void onSensorChanged(SensorEvent event) {
62
63     }
64
65     @Override
66     public void onAccuracyChanged(Sensor sensor, int accuracy) {
67
68     }
69 }

```

Activity\_main.xml : This XML layout for an Android app describes a user interface with two primary components within a 'RelativeLayout': a 'TextView' with the ID 'sensorList' at the top, displaying the headline "Sensor List", and a 'ScrollView' holding another 'TextView' with the ID 'sensorData'. If the content in 'sensorData', which is supposed to list sensor details dynamically, exceeds the screen size, the 'ScrollView' assures that it is scrollable. The layout is intended to provide clarity and simplicity of reading sensor data.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:padding="16dp"
7     tools:context=".MainActivity">
8
9     <TextView
10         android:id="@+id/sensorList"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:text="Sensor List"
14         android:textSize="28sp"
15         android:textStyle="bold" />
16
17     <ScrollView
18         android:layout_width="match_parent"
19         android:layout_height="match_parent"
20         android:layout_below="@id/sensorList"
21         android:layout_marginTop="16dp">
22
23         <TextView
24             android:id="@+id/sensorData"
25             android:layout_width="match_parent"
26             android:layout_height="wrap_content"
27             android:text=""
28             android:textSize="16sp" />
29     </ScrollView>
30 </RelativeLayout>
31
32
```

Below are the screenshot of the app :

