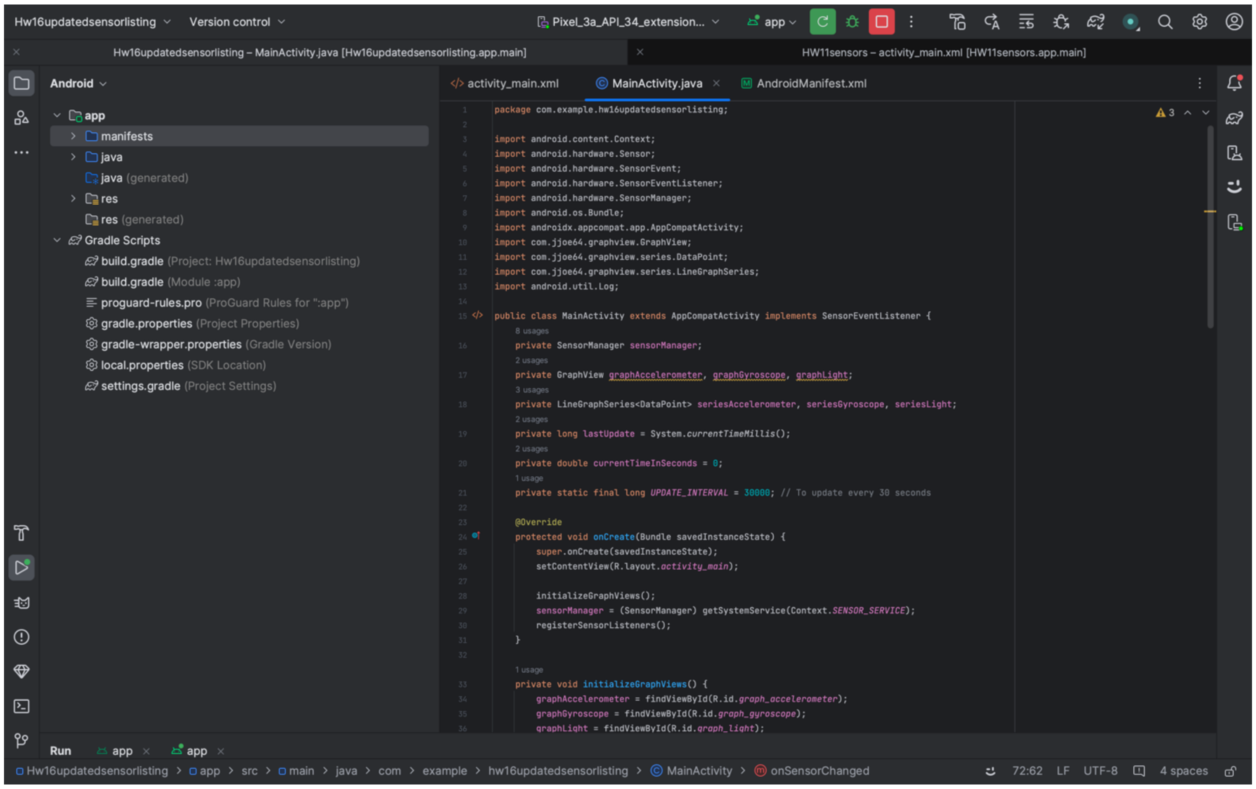
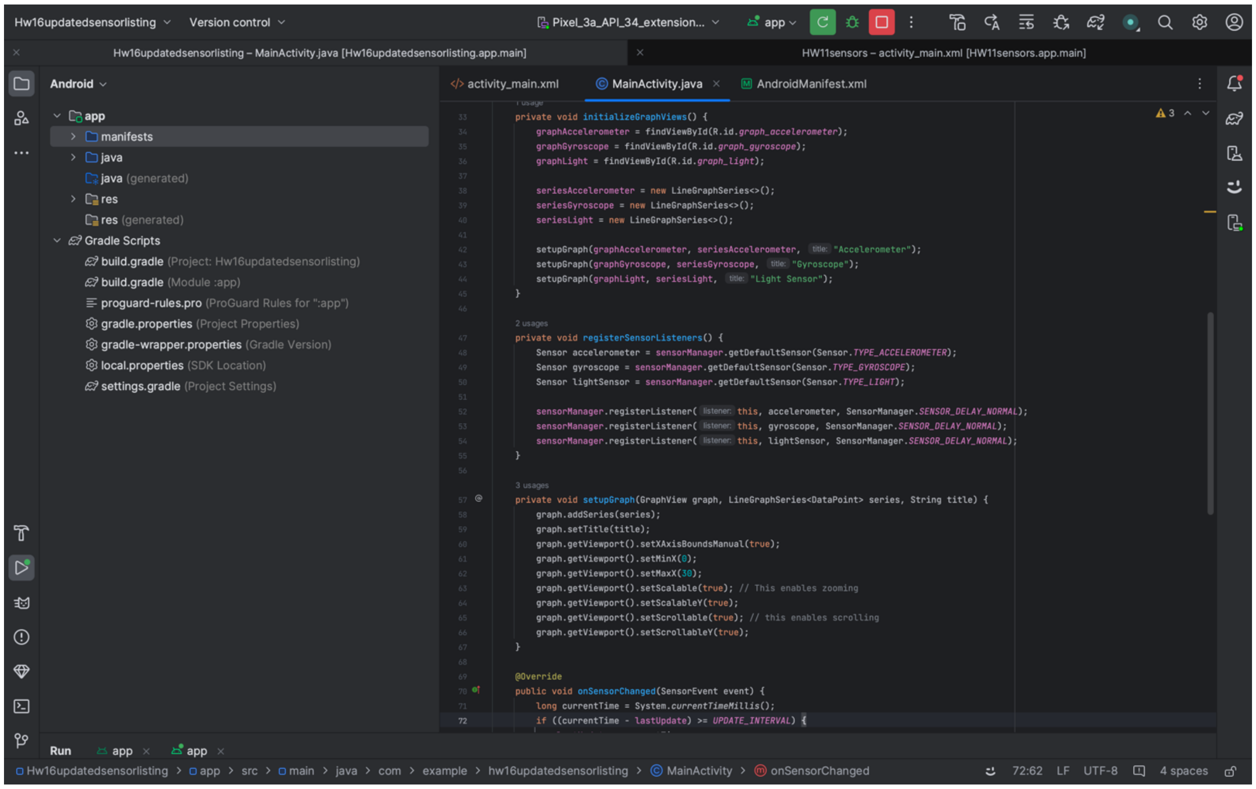
Revise the sensor listing app so that the sensor values are displayed in a chart. You may use any charting library. The chart should display the most recent 30-second of the data collected for each sensor.

Mainactivity.java : To successfully acquire and visualize sensor data, we need to use several critical methods and functionalities. The 'onCreate()' method starts the program by configuring GraphViews and registering sensor listeners. 'initializeGraphViews()' configures the GraphView elements, whereas 'registerSensorListeners()' registers Accelerometer, Gyroscope, and Light Sensor listeners. The GraphView settings are customized using the 'setupGraph()' function. When sensor data changes, the 'onSensorChanged(SensorEvent event)' method logs and updates each sensor type's LineGraphSeries. 'onResume()' re-registers sensor listeners during app resume, and 'onPause()' unregisters them for effective resource management. These components work together to record and visualize accurate real-time sensor data.

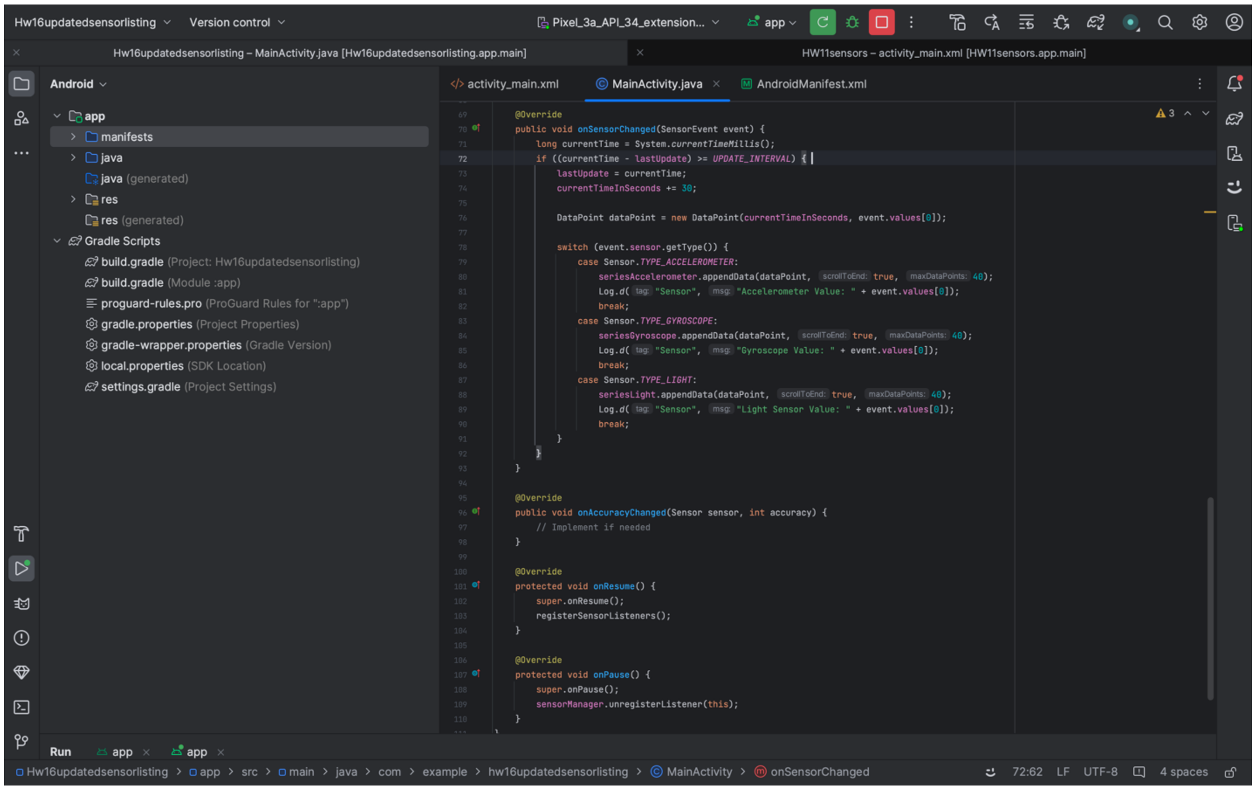
Note : Additionally, I added the zoom in/out functionality with the `setScalable(true)` and `setScalableY(true)` functions and displays sensor names and values on the x-axis by default, contributing to an enhanced user experience and potentially earning extra credits.


Hw16updatedsensorlisting
Version control
Pixel_3a_API_34_extension...
app
Hw16updatedsensorlisting – MainActivity.java [Hw16updatedsensorlisting.app.main]
HW11sensors – activity_main.xml [HW11sensors.app.main]
Android
app
manifests
java
java (generated)
res
res (generated)
Gradle Scripts
build.gradle (Project: Hw16updatedsensorlisting)
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)
activity_main.xml
MainActivity.java
AndroidManifest.xml
Run
app
app
Hw16updatedsensorlisting > app > src > main > java > com > example > hw16updatedsensorlisting > MainActivity > onSensorChanged
72:62 LF UTF-8 4 spaces

```java
package com.example.hw16updatedsensorlisting;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import androidx.appcompat.app.AppCompatActivity;
import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;
import android.util.Log;

public class MainActivity extends AppCompatActivity implements SensorEventListener {
    private SensorManager sensorManager;
    private GraphView graphAccelerometer, graphGyroscope, graphLight;
    private LineGraphSeries<DataPoint> seriesAccelerometer, seriesGyroscope, seriesLight;
    private long lastUpdate = System.currentTimeMillis();
    private double currentTimeInSeconds = 0;
    private static final long UPDATE_INTERVAL = 30000; // To update every 30 seconds

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

        initializeGraphViews();
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        registerSensorListeners();
    }

    private void initializeGraphViews() {
        graphAccelerometer = findViewById(R.id.graph_accelerometer);
        graphGyroscope = findViewById(R.id.graph_gyroscope);
        graphLight = findViewById(R.id.graph_light);
```

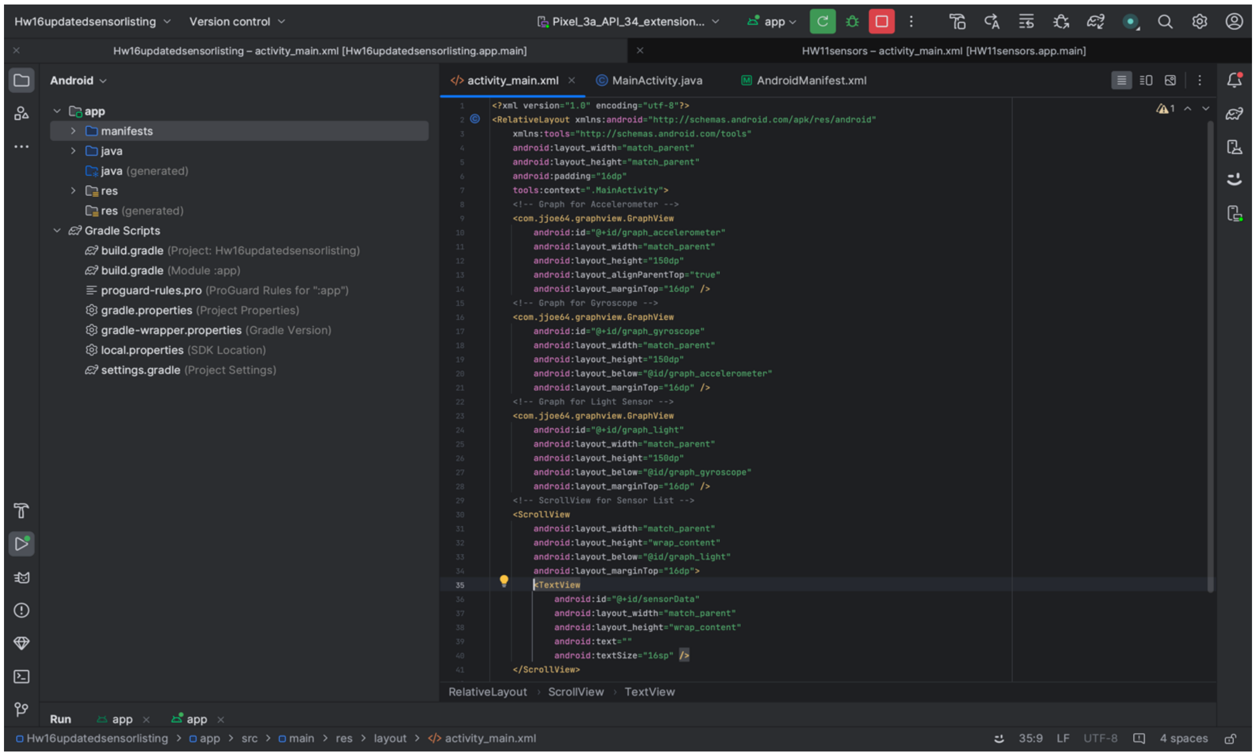


```java
    private void initializeGraphViews() {
        graphAccelerometer = findViewById(R.id.graph_accelerometer);
        graphGyroscope = findViewById(R.id.graph_gyroscope);
        graphLight = findViewById(R.id.graph_light);

        seriesAccelerometer = new LineGraphSeries<>();
        seriesGyroscope = new LineGraphSeries<>();
        seriesLight = new LineGraphSeries<>();

        setupGraph(graphAccelerometer, seriesAccelerometer, "Accelerometer");
        setupGraph(graphGyroscope, seriesGyroscope, "Gyroscope");
        setupGraph(graphLight, seriesLight, "Light Sensor");
    }

    private void registerSensorListeners() {
        Sensor accelerometer = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        Sensor gyroscope = sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
        Sensor lightSensor = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

        sensorManager.registerListener(this, accelerometer, SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, gyroscope, SensorManager.SENSOR_DELAY_NORMAL);
        sensorManager.registerListener(this, lightSensor, SensorManager.SENSOR_DELAY_NORMAL);
    }

    private void setupGraph(GraphView graph, LineGraphSeries<DataPoint> series, String title) {
        graph.addSeries(series);
        graph.setTitle(title);
        graph.getViewport().setXAxisBoundsManual(true);
        graph.getViewport().setMinX(0);
        graph.getViewport().setMaxX(30);
        graph.getViewport().setScalable(true); // This enables zooming
        graph.getViewport().setScalableY(true);
        graph.getViewport().setScrollable(true); // this enables scrolling
        graph.getViewport().setScrollableY(true);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        long currentTime = System.currentTimeMillis();
        if ((currentTime - lastUpdate) >= UPDATE_INTERVAL) {
```



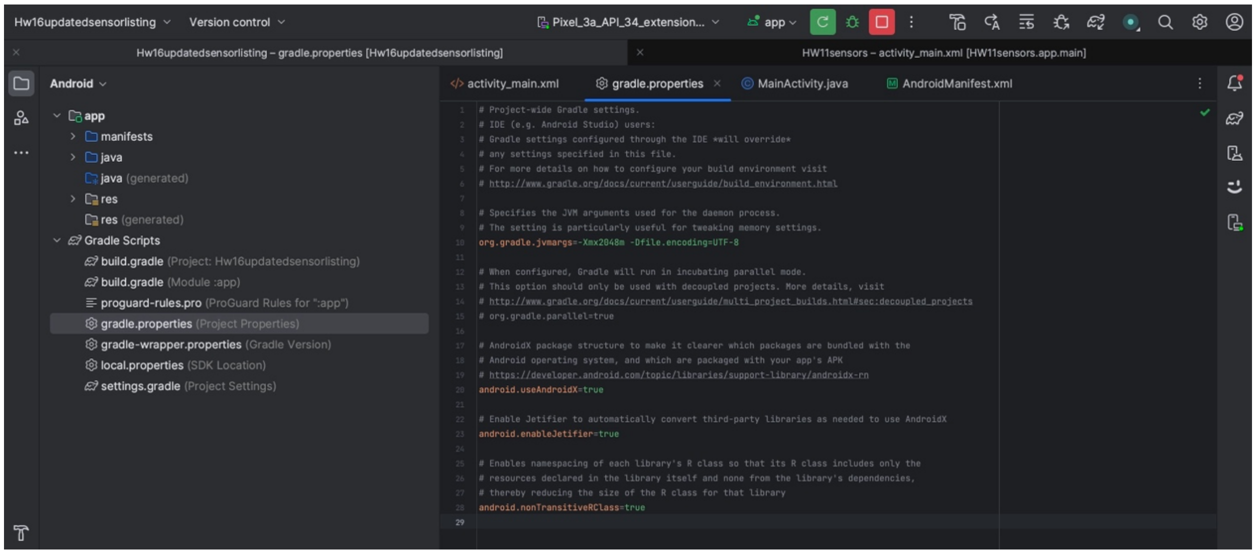
```java
    @Override
    public void onSensorChanged(SensorEvent event) {
        long currentTime = System.currentTimeMillis();
        if ((currentTime - lastUpdate) >= UPDATE_INTERVAL) {
            lastUpdate = currentTime;
            currentTimeInSeconds += 30;

            DataPoint dataPoint = new DataPoint(currentTimeInSeconds, event.values[0]);

            switch (event.sensor.getType()) {
                case Sensor.TYPE_ACCELEROMETER:
                    seriesAccelerometer.appendData(dataPoint, true, 40);
                    Log.d("Sensor", "Accelerometer Value: " + event.values[0]);
                    break;
                case Sensor.TYPE_GYROSCOPE:
                    seriesGyroscope.appendData(dataPoint, true, 40);
                    Log.d("Sensor", "Gyroscope Value: " + event.values[0]);
                    break;
                case Sensor.TYPE_LIGHT:
                    seriesLight.appendData(dataPoint, true, 40);
                    Log.d("Sensor", "Light Sensor Value: " + event.values[0]);
                    break;
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // Implement if needed
    }

    @Override
    protected void onResume() {
        super.onResume();
        registerSensorListeners();
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
```

Activity_main.xml : The XML layout code defines the main user interface for the activity of the Android app. It features three vertically stacked 'GraphView' elements for viewing sensor data (accelerometer, gyroscope, and light sensor). Below them is a 'ScrollView' with a 'TextView' that displays a list of sensor data. The design guarantees that sensor data is visually displayed by graphs and that it may be scrolled through in text format.



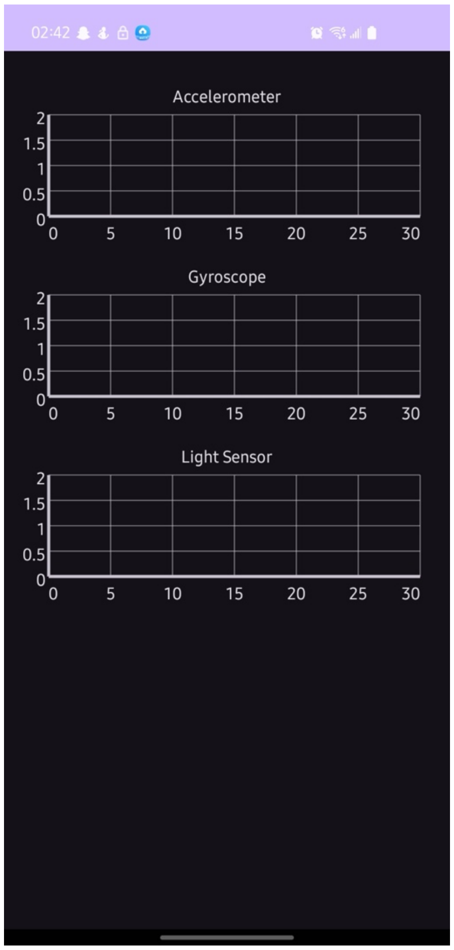
```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    tools:context=".MainActivity">
    <!-- Graph for Accelerometer -->
    <com.jjoe64.graphview.GraphView
        android:id="@+id/graph_accelerometer"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_alignParentTop="true"
        android:layout_marginTop="16dp" />
    <!-- Graph for Gyroscope -->
    <com.jjoe64.graphview.GraphView
        android:id="@+id/graph_gyroscope"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_below="@id/graph_accelerometer"
        android:layout_marginTop="16dp" />
    <!-- Graph for Light Sensor -->
    <com.jjoe64.graphview.GraphView
        android:id="@+id/graph_light"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_below="@id/graph_gyroscope"
        android:layout_marginTop="16dp" />
    <!-- ScrollView for Sensor List -->
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/graph_light"
        android:layout_marginTop="16dp">
        <TextView
            android:id="@+id/sensorData"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text=""
            android:textSize="16sp" />
    </ScrollView>
```

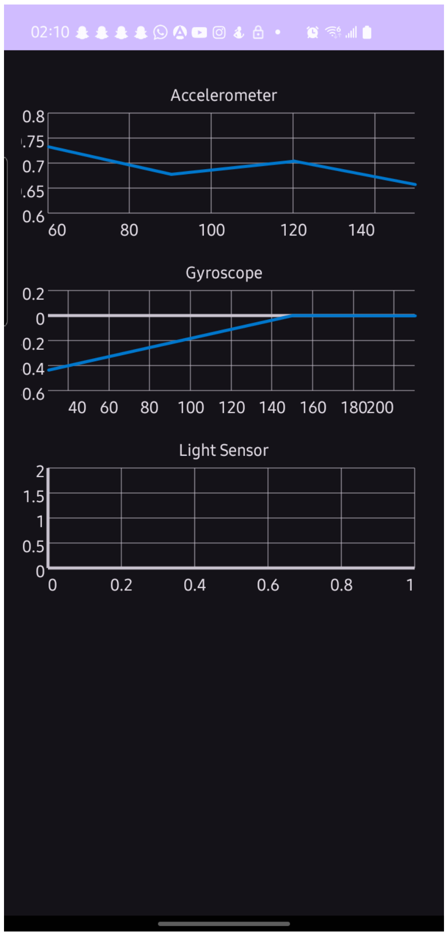
Gradle properties : In the `gradle.properties` file, we need to use the line "android.useAndroidX=true" to enable AndroidX libraries, and "android.enableJetifier=true" is included to enable automatic migration of third-party libraries for AndroidX compatibility.



```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html

# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8

# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
# http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
# org.gradle.parallel=true

# AndroidX package structure to make it clearer which packages are bundled with the
# Android operating system, and which are packaged with your app's APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true

# Enable Jetifier to automatically convert third-party libraries as needed to use AndroidX
android.enableJetifier=true

# Enables namespacing of each library's R class so that its R class includes only the
# resources declared in the library itself and none from the library's dependencies,
# thereby reducing the size of the R class for that library
android.nonTransitiveRClass=true

```

I ran the app on my device for testing as the android studio emulator is not reliable for testing sensors below are the screen shots of the app the app





SS1 before testing. SS2 after testing