```java
package com.example.rim_test_counter;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.Criteria;
import android.os.Bundle;
import android.util.Log;
import android.util.Pair;
import android.widget.TextView;
import android.widget.Button;
import android.view.View;
import android.content.Intent;
import android.widget.AdapterView;
import android.location.Location;
import android.location.LocationManager;
import android.location.LocationListener;

import android.bluetooth.BluetoothDevice;
import me.aflak.bluetooth.Bluetooth;
import me.aflak.bluetooth.interfaces.BluetoothCallback;
import me.aflak.bluetooth.interfaces.DiscoveryCallback;
import me.aflak.bluetooth.interfaces.DeviceCallback;

import java.util.ArrayList;
import java.util.List;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.google.gson.Gson;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
```

```
import java.util.HashMap;

import org.json.*;
public class MainActivity extends AppCompatActivity {
      // Local Variables
      public TextView counter;
      public TextView streetText;
      public TextView counterStreet2;
      public TextView counterStreet;
      public Button counter_button;
      public Button scan_button;
      public Button reset_button;
      public Button green;
      public Button spring;
      public Button wright;
      public Button gps;
      public TextView coordinates;
      private Bluetooth bluetooth;
      private ArrayList<BluetoothDevice> scannedDevices;
      public boolean scanning = false;
      int tempCount;
      public ListView listView;
      HashMap<String, Pair<Integer,Integer>> streetCounts = new
HashMap<>();

      Location currLoc;
      String currentStreet = "Green St.";
      String msg;
      ArrayAdapter adapter;
      String url = "www.google.com";

      // Instantiate the RequestQueue.
      RequestQueue queue;
      // Request a string response from the provided URL.
      StringRequest stringRequest = new StringRequest(Request.Method.GET,
url,
            new Response.Listener<String>() {
                  @Override
                  public void onResponse(String response) {
                  // Display the first 500 characters of the response
string.
```

```
                streetText.setText("Response is: "+
response.substring(0,500));
                }
            }, new Response.ErrorListener() {
      @Override
      public void onErrorResponse(VolleyError error) {
            streetText.setText("That didn't work!");
      }
      });

      @Override
      protected void onCreate(Bundle savedInstanceState) {
      queue = Volley.newRequestQueue(this);
      streetCounts.put("Green St.", Pair.create(0,0));
      streetCounts.put("Springfield", Pair.create(0,0));
      streetCounts.put("Wright St.", Pair.create(0,0));
      // Ask for permissions
      int MY_PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION = 1;
      ActivityCompat.requestPermissions(this,
                  new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                  MY_PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);

      int MY_PERMISSIONS_REQUEST_ACCESS_INTERNET = 1;
      ActivityCompat.requestPermissions(this,
                  new String[]{Manifest.permission.ACCESS_WIFI_STATE},
                  MY_PERMISSIONS_REQUEST_ACCESS_INTERNET);

      // Setup Bluetooth Library
      setContentView(R.layout.activity_main);
      bluetooth = new Bluetooth(this);
      bluetooth.setBluetoothCallback(bluetoothCallback);
      bluetooth.setDiscoveryCallback(discoveryCallback);
      bluetooth.setDeviceCallback(devicecallBack);

      // Setup list of devices
      listView = findViewById(R.id.device_list);
      adapter = new ArrayAdapter<String>(this,
                  R.layout.activity_listview, new ArrayList<String>());
      if (listView != null) {
            listView.setAdapter(adapter);
            listView.setOnItemClickListener(onScanListItemClick);
      }
```

```
        super.onCreate(savedInstanceState);
        // Initialize counter button and textview
        counter = findViewById(R.id.textView);
        counterStreet2 = findViewById(R.id.counterStreet2);
        counterStreet = findViewById(R.id.counterStreet);
        counter_button = findViewById(R.id.btn_count);
        scan_button = findViewById(R.id.scan_btn);
        counterStreet.setText(String.valueOf(0));
        // Set up counter button
        counter_button.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
                    tempCount = streetCounts.get(currentStreet).first;
                    tempCount += 1;
                    streetCounts.put(currentStreet, Pair.create(tempCount,
streetCounts.get(currentStreet).second));
                    counterStreet.setText(String.valueOf(tempCount));
              }
        });

        // Setup Bluetooth Scan Button
        scan_button.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
                    bluetooth.startScanning();
                    scanning = true;
              }
        });
        // Setup Reset Button
        reset_button = findViewById(R.id.resetBTN);
        reset_button.setOnClickListener(new View.OnClickListener() {
              @Override
              public void onClick(View view) {
                    for (String name : streetCounts.keySet())
                    streetCounts.put(name, Pair.create(0,0));

                    counterStreet.setText(String.valueOf(0));
                    counterStreet2.setText(String.valueOf(0));
              }
        });

        // Setup current street text
```

```
streetText = findViewById(R.id.streetText);
green = findViewById(R.id.greenSt);
spring = findViewById(R.id.Springfield);
wright = findViewById(R.id.wright);

green.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
            currentStreet = "Green St.";
            tempCount = streetCounts.get(currentStreet).first;
            counterStreet.setText(String.valueOf(tempCount));
            tempCount = streetCounts.get(currentStreet).second;
            counterStreet2.setText(String.valueOf(tempCount));
            streetText.setText(currentStreet);
      }
});

spring.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
            currentStreet = "Springfield";
            tempCount = streetCounts.get(currentStreet).first;
            counterStreet.setText(String.valueOf(tempCount));
            tempCount = streetCounts.get(currentStreet).second;
            counterStreet2.setText(String.valueOf(tempCount));
            streetText.setText(currentStreet);
      }
});

wright.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View view) {
            currentStreet = "Wright St.";
            tempCount = streetCounts.get(currentStreet).first;
            counterStreet.setText(String.valueOf(tempCount));
            tempCount = streetCounts.get(currentStreet).second;
            counterStreet2.setText(String.valueOf(tempCount));
            streetText.setText(currentStreet);
      }
});
```

```
      gps = findViewById(R.id.gps);
      coordinates = findViewById(R.id.coordinates);

      gps.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                  currLoc =
getLocationWithCheckNetworkAndGPS(view.getContext());
                  String Lat, Long;
                  Lat = String.valueOf(currLoc.getLatitude());
                  Long = String.valueOf(currLoc.getLongitude());
                  coordinates.setText(Lat+","+Long);

                  url =
"https://nominatim.openstreetmap.org/reverse?&format=jsonv2&lat="+Lat+"&lon
="+Long;
                  StringRequest stringRequest = new
StringRequest(Request.Method.GET, url,
                        new Response.Listener<String>() {
                              @Override
                              public void onResponse(String response) {
                              // Display the first 500 characters of the
response string.
                              JsonObject jsonObject = new
JsonParser().parse(response).getAsJsonObject();
                              jsonObject =
jsonObject.get("address").getAsJsonObject();
                              streetText.setText(""+
jsonObject.get("road"));
                              currentStreet =
jsonObject.get("road").getAsString();
                              if
(!streetCounts.containsKey(currentStreet)){
                                    streetCounts.put(currentStreet,
Pair.create(0,0));
                              }
                              tempCount =
streetCounts.get(currentStreet).first;

counterStreet.setText(String.valueOf(tempCount));
                              tempCount =
```

```
streetCounts.get(currentStreet).second;

counterStreet2.setText(String.valueOf(tempCount));
                              }
                        }, new Response.ErrorListener() {
                  @Override
                  public void onErrorResponse(VolleyError error) {

                        streetText.setText(error.getMessage());
                  }
                  });
                  queue.add(stringRequest);
            }
      });


      }

      @Override
      protected void onStart() {
      super.onStart();
      bluetooth.onStart();
      if(bluetooth.isEnabled()){
            String message = "Bluetooth Enabled";
            counter.setText(message);
      } else {
            bluetooth.enable();
      }
      }

      @Override
      protected void onStop() {
      super.onStop();
      bluetooth.onStop();
      }

      @Override
      protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
      super.onActivityResult(requestCode, resultCode, data);
      bluetooth.onActivityResult(requestCode, resultCode);
      }
```

```
	private BluetoothCallback bluetoothCallback = new BluetoothCallback()
{
	@Override public void onBluetoothTurningOn() {}
	@Override public void onBluetoothTurningOff() {}
	@Override public void onBluetoothOff() {}

	@Override
	public void onBluetoothOn() {
		// doStuffWhenBluetoothOn() ...
	}

	@Override
	public void onUserDeniedActivation() {
		// handle activation denial...
	}
	};

	private DiscoveryCallback discoveryCallback = new DiscoveryCallback()
{
	public void onDiscoveryStarted() {
		scannedDevices = new ArrayList<>();
	}
	@Override public void onDiscoveryFinished() {}
	@Override public void onDeviceFound(BluetoothDevice device) {
		String message = "Device Found";
		counter.setText(message);
		scannedDevices.add(device);
		adapter.add(device.getAddress()+" : "+device.getName());
	}
	@Override public void onDevicePaired(BluetoothDevice device) {}
	@Override public void onDeviceUnpaired(BluetoothDevice device) {}
	@Override public void onError(int errorCode) {

	}
	};

	private DeviceCallback devicecallBack = new DeviceCallback() {
	@Override public void onDeviceConnected(BluetoothDevice device) {
		String message = "Device Connected";
	}
```

```
      @Override public void onDeviceDisconnected(BluetoothDevice device,
String message) {}
      @Override public void onMessage(byte[] message) {
            msg = new String(message);
            Log.d("myTag", msg);
            if (msg.equals("<POT>")){
                  tempCount = streetCounts.get(currentStreet).first;
                  tempCount += 1;
                  streetCounts.put(currentStreet, Pair.create(tempCount,
streetCounts.get(currentStreet).second));
            }
            else if (msg.equals("<DEB>")){
                  tempCount = streetCounts.get(currentStreet).second;
                  tempCount += 1;
                  streetCounts.put(currentStreet,
Pair.create(streetCounts.get(currentStreet).first, tempCount));
            }
            runOnUiThread(new Runnable() {

                  @Override
                  public void run() {
                  setStatus(msg);

counterStreet.setText(String.valueOf(streetCounts.get(currentStreet).first)
);

counterStreet2.setText(String.valueOf(streetCounts.get(currentStreet).secon
d));
                  }
            });
      }
      @Override public void onError(int errorCode) {}
      @Override public void onConnectError(BluetoothDevice device, String
message) {}
      };

      private void setStatus(String message){
      counter.setText(message);
      }

      private AdapterView.OnItemClickListener onScanListItemClick = new
AdapterView.OnItemClickListener() {
```

```
      @Override
      public void onItemClick(AdapterView<?> adapterView, View view, int i,
long l) {
            if (scanning) {
                  bluetooth.stopScanning();
            }
            // Pair
            bluetooth.pair(scannedDevices.get(i));
            String message = "Device Paired";
            setStatus(message);
            bluetooth.connectToDevice(scannedDevices.get(i));
            message = "Device Connected";
            setStatus(message);
            listView.setVisibility(View.GONE);
      }
      };


      // Location Get
      public static Location getLocationWithCheckNetworkAndGPS(Context
mContext) {
      /*
      LocationManager lm = (LocationManager)
                  mContext.getSystemService(Context.LOCATION_SERVICE);
      assert lm != null;
      boolean isGpsEnabled =
lm.isProviderEnabled(LocationManager.GPS_PROVIDER);
      boolean isNetworkLocationEnabled =
lm.isProviderEnabled(LocationManager.NETWORK_PROVIDER);

      Location networkLocation = null, gpsLocation = null, finalLoc = null;
      if (isGpsEnabled)
            if (ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

                  return null;
            }gpsLocation =
lm.getLastKnownLocation(LocationManager.GPS_PROVIDER);
```

```
      if (isNetworkLocationEnabled)
            networkLocation =
lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);

      if (gpsLocation != null && networkLocation != null) {

            //smaller the number more accurate result will
            if (gpsLocation.getAccuracy() > networkLocation.getAccuracy())
                  finalLoc = networkLocation;
            else
                  finalLoc = gpsLocation;

      } else {

            if (gpsLocation != null) {
                  finalLoc = gpsLocation;
            } else if (networkLocation != null) {
                  finalLoc = networkLocation;
            }
      }
      */
      LocationManager lm =
(LocationManager)mContext.getSystemService(Context.LOCATION_SERVICE);
      assert lm != null;
      if (true)
            if (ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(mContext,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {

                  return null;
            }
      Location loc =
lm.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
      return loc;
      }
};
```