

CSE 162 - Lab 3

Activity Recognition

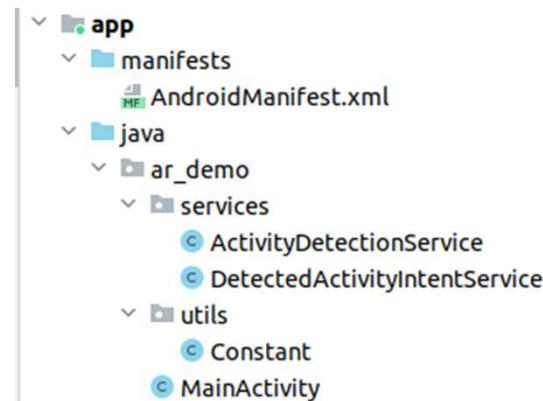
Goal

- Display the current activity of the user
 - Learn to use the Google Activity Recognition API

Overview of tasks

- obtain the permissions
- register for activity updates
 - create pending intents
- deregister for activity updates
- process activity events

Take care of these files



Obtain permissions

AndroidManifest.xml

- AndroidManifest.xml file:
 - Request permission
 - declare activity
 - declare services

The screenshot shows the XML code for the AndroidManifest.xml file. Red arrows point from the list items to specific parts of the code:

- A red arrow points from the "Request permission" item to the `<uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />` line.
- A red arrow points from the "declare activity" item to the `<activity>` section, which includes the `android:name=".MainActivity"`, `android:screenOrientation="fullSensor"`, and `<intent-filter>` blocks.
- A red arrow points from the "declare services" item to the `<service android:name=".services.DetectedActivityIntentService" />` and `<service android:name=".services.ActivityDetectionService" />` lines.

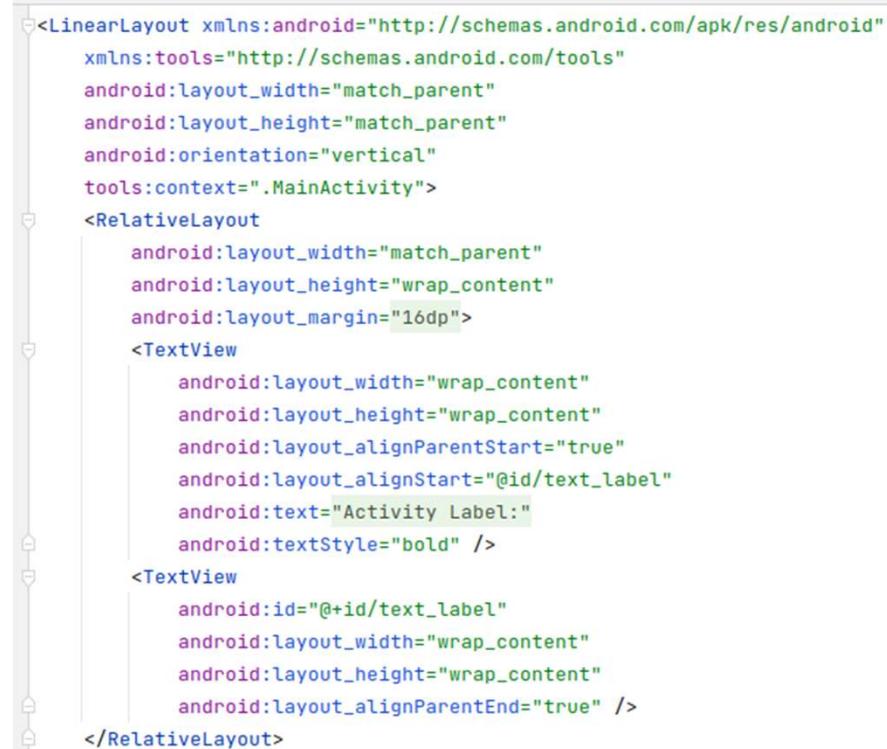
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ar_demo">

    <uses-permission android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="AR-Demo"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:screenOrientation="fullSensor">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".services.DetectedActivityIntentService" />
        <service android:name=".services.ActivityDetectionService" />
    </application>
</manifest>
```

UI design

- activity_main.xml (1)



```
<LinearLayout 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:orientation="vertical"
    tools:context=".MainActivity">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="16dp">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_alignStart="@+id/text_label"
            android:text="Activity Label:"
            android:textStyle="bold" />
        <TextView
            android:id="@+id/text_label"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true" />
    </RelativeLayout>
```

- activity_main.xml (2)

```
<RelativeLayout  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_margin="16dp">  
    <TextView  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentStart="true"  
        android:layout_alignStart="@+id/text_confidence"  
        android:text="Confidence:"  
        android:textStyle="bold" />  
    <TextView  
        android:id="@+id/text_confidence"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:layout_alignParentEnd="true" />  
</RelativeLayout>  
</LinearLayout>
```

register for activity updates

ActivityDetectionService.java

- To start receiving notifications about activity transitions, you must implement the following:
 - An `requestActivityUpdatesHandler` method that specifies what activities and how frequently to request.
 - A `PendingIntent` callback where your app receives notifications. For more information, see [Using a pending intent](#).

Imports for ActivityDetectionService.java

```
1 package ar_demo.services;  
2  
3 import android.app.PendingIntent;  
4 import android.app.Service;  
5 import android.content.Intent;  
6 import android.os.IBinder;  
7 import androidx.annotation.NonNull;  
8 import androidx.annotation.Nullable;  
9 import android.util.Log;  
10  
11 import com.google.android.gms.location.ActivityRecognitionClient;  
12 import com.google.android.gms.tasks.OnFailureListener; ←  
13 import com.google.android.gms.tasks.OnSuccessListener;  
14 import com.google.android.gms.tasks.Task;  
15  
16 import ar_demo.utils.Constant;  
17
```

Google activity
recognition imports

Initialize the update service

- in ar_demo/services/ActivityDetectionService.java, implement:

Create the activity
recognition client

Create the pending intent

Call the handler

```
public int onStartCommand(Intent intent, int flags, int startId) {
    super.onStartCommand(intent, flags, startId);

    Log.d(TAG, msg: "onStartCommand()");
    mActivityRecognitionClient = new ActivityRecognitionClient( context: this);
    Intent mIntentService = new Intent( packageContext: this, DetectedActivityIntentService.class);
    // FLAG_UPDATE_CURRENT indicates that if the described PendingIntent already exists,
    // then keep it but replace its extra data with what is in this new Intent.
    mPendingIntent = PendingIntent.getService( context: this,
                                                requestCode: 1, mIntentService, PendingIntent.FLAG_UPDATE_CURRENT);
    requestActivityUpdatesHandler();

    return START_STICKY;
}
```

- You can register for activity updates by passing your update interval and your PendingIntent object to the requestActivityUpdates() method.
- The requestActivityUpdates() method returns a Task object that you can check for success or failure

Implement requestActivityUpdatesHandler

- in `ar_demo/services/ActivityDetectionService.java`, implement:

activity recognition client

the pending intent

A constant value to
control the frequency of
activity updates. To be
seen soon.

```
60
61     // request updates and set up callbacks for success or failure
62     public void requestActivityUpdatesHandler() {
63         Log.d(TAG, msg: "requestActivityUpdatesHandler()");
64         if(mActivityRecognitionClient != null){
65             Task<Void> task = mActivityRecognitionClient.requestActivityUpdates(
66                 Constant.DETECTION_INTERVAL_IN_MILLISECONDS,
67                 mPendingIntent);
68
69             // Adds a listener that is called if the Task completes successfully.
70             taskaddOnSuccessListener(new OnSuccessListener<Void>() {
71                 @Override
72                 public void onSuccess(Void result) {
73                     Log.d(TAG, msg: "Successfully requested activity updates");
74                 }
75             });
76             // Adds a listener that is called if the Task fails.
77             taskaddOnFailureListener(new OnFailureListener() {
78                 @Override
79                 public void onFailure(@NonNull Exception e) {
80                     Log.e(TAG, msg: "Requesting activity updates failed to start");
81                 }
82             });
83         }
84     }
85 }
```

Deregister activity recognition updates

```
87         // remove the activity requested updates from Google play.
88
89     @Override
90     public void onDestroy() {
91         super.onDestroy();
92         // need to remove the request to Google play services. Brings down the connection.
93         removeActivityUpdatesHandler();
94     }
95
96     // remove updates and set up callbacks for success or failure
97     public void removeActivityUpdatesHandler() {
98         if(mActivityRecognitionClient != null){
99             Task<Void> task = mActivityRecognitionClient.removeActivityUpdates(
100                 mPendingIntent);
101             // Adds a listener that is called if the Task completes successfully.
102             taskaddOnSuccessListener(new OnSuccessListener<Void>() {
103                 @Override
104                 public void onSuccess(Void result) {
105                     Log.d(TAG, msg: "Removed activity updates successfully!");
106                 }
107             });
108             // Adds a listener that is called if the Task fails.
109             taskaddOnFailureListener(new OnFailureListener() {
110                 @Override
111                 public void onFailure(@NonNull Exception e) {
112                     Log.e(TAG, msg: "Failed to remove activity updates!");
113                 }
114             });
115         }
116     }
```

- Overview

```
19
20  public class ActivityDetectionService extends Service {
21      private static final String TAG = ActivityDetectionService.class.getSimpleName();
22
23      private PendingIntent mPendingIntent;
24      private ActivityRecognitionClient mActivityRecognitionClient;
25
26      public ActivityDetectionService() {
27          //Log.d(TAG, "ActivityDetectionService()");
28      }
29
30
31      @Nullable
32      @Override
33      public IBinder onBind(Intent intent) {
34          Log.d(TAG, msg: "onBind()");
35          return null;
36      }
37
38      @Override
39      public void onCreate() {
40          super.onCreate();
41          Log.d(TAG, msg: "onCreate()");
42      }
43  }
```

- Overview 2

```
44     @SuppressLint("UnspecifiedImmutableFlag")
45     @Override
46     public int onStartCommand(Intent intent, int flags, int startId) {
47         super.onStartCommand(intent, flags, startId);
48
49         Log.d(TAG, msg: "onStartCommand()");
50         mActivityRecognitionClient = new ActivityRecognitionClient(context: this);
51         Intent mIntentService = new Intent(packageContext: this, DetectedActivityIntentService.class);
52         // FLAG_UPDATE_CURRENT indicates that if the described PendingIntent already exists,
53         // then keep it but replace its extra data with what is in this new Intent.
54         mPendingIntent = PendingIntent.getService(context: this,
55             requestCode: 1, mIntentService, PendingIntent.FLAG_UPDATE_CURRENT);
56         requestActivityUpdatesHandler();
57
58         return START_STICKY;
59     }
59
60
61     // request updates and set up callbacks for success or failure
62     public void requestActivityUpdatesHandler() {
63         Log.d(TAG, msg: "requestActivityUpdatesHandler()");
64         if(mActivityRecognitionClient != null){
65             Task<Void> task = mActivityRecognitionClient.requestActivityUpdates(
66                 Constant.DETECTION_INTERVAL_IN_MILLISECONDS,
67                 mPendingIntent);
68
69             // Adds a listener that is called if the Task completes successfully.
70             taskaddOnSuccessListener(new OnSuccessListener<Void>() {
71                 @Override
72                 public void onSuccess(Void result) {
73                     Log.d(TAG, msg: "Successfully requested activity updates");
74                 }
75             });
76             // Adds a listener that is called if the Task fails.
77             taskaddOnFailureListener(new OnFailureListener() {
78                 @Override
79                 public void onFailure(@NonNull Exception e) {
80                     Log.e(TAG, msg: "Requesting activity updates failed to start");
81                 }
82             });
83         }
83
84     }
84 }
```

• Overview 3

```
86      // remove the activity requested updates from Google play.
87
88      @Override
89  ↗    public void onDestroy() {
90          super.onDestroy();
91          // need to remove the request to Google play services. Brings down the connection.
92          removeActivityUpdatesHandler();
93      }
94
95      // remove updates and set up callbacks for success or failure
96      public void removeActivityUpdatesHandler() {
97          if(mActivityRecognitionClient != null){
98              Task<Void> task = mActivityRecognitionClient.removeActivityUpdates(
99                  mPendingIntent);
100             // Adds a listener that is called if the Task completes successfully.
101             task.addOnSuccessListener(new OnSuccessListener<Void>() {
102                 @Override
103  ↗                public void onSuccess(Void result) {
104                     Log.d(TAG, msg: "Removed activity updates successfully!");
105                 }
106             });
107             // Adds a listener that is called if the Task fails.
108             task.addOnFailureListener(new OnFailureListener() {
109                 @Override
110  ↗                public void onFailure(@NonNull Exception e) {
111                     Log.e(TAG, msg: "Failed to remove activity updates!");
112                 }
113             });
114         }
115     }
116 }
117 }
```

Define the update interval

ar_demo/utils/Constant.java

```
public class Constant {

    3 usages
    public static final String BROADCAST_DETECTED_ACTIVITY = "activity_intent";
    // the desired time between activity detections. Larger values will result in fewer activity
    // detections while improving battery life. A value of 0 will result in activity detections
    // at the fastest possible rate.
    1 usage
    public static final long DETECTION_INTERVAL_IN_MILLISECONDS = 1000; // every N seconds
}
```

Process activity events

DetectedActivityIntentService.java

Imports for DetectedActivityIntentService.java

```
1  package ar_demo.services;  
2  
3  import android.app.IntentService;  
4  import android.content.Intent;  
5  import androidx.localbroadcastmanager.content.LocalBroadcastManager;  
6  import android.util.Log;  
7  
8  import com.google.android.gms.location.ActivityRecognitionResult;  
9  import com.google.android.gms.location.DetectedActivity;  
10  
11 import java.util.List;  
12  
13 import ar_demo.utils.Constant;  
14
```

Implement the pending intent

- create the file
ar_demo/services/DetectedActivityIntentService.java
- Create a service:

3 usages

```
public class DetectedActivityIntentService extends IntentService
```

- override onHandleIntent() method

Get the recognition results

```
    @Override
    protected void onHandleIntent(Intent intent) {
        Log.d(TAG, msg: TAG + "onHandleIntent()");
        ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);

        // Get the list of the probable activities associated with the current state of the
        // device. Each activity is associated with a confidence level, which is an int between
        // 0 and 100.

        List<DetectedActivity> detectedActivities = result.getProbableActivities();

        for (DetectedActivity activity : detectedActivities) {
            //Log.d(TAG, "Detected activity: " + activity.getType() + ", " + activity.getConfidence());
            broadcastActivity(activity);
        }
    }
```

send the result out

- create an intent and send out the activity info

```
private void broadcastActivity(DetectedActivity activity) {
    // Log.d(TAG, TAG+ "broadcastActivity()");
    Intent intent = new Intent(Constant.BROADCAST_DETECTED_ACTIVITY);
    intent.putExtra( name: "type", activity.getType());
    intent.putExtra( name: "confidence", activity.getConfidence());
    LocalBroadcastManager.getInstance( context: this).sendBroadcast(intent);
}
```

Overview of DetectedActivityIntentService.java

```
public class DetectedActivityIntentService extends IntentService {
    3 usages
    protected static final String TAG = DetectedActivityIntentService.class.getSimpleName();

    public DetectedActivityIntentService() {
        super(TAG);
        // Log.d(TAG, TAG + "DetectedActivityIntentService()");
    }

    @Override
    public void onCreate() {
        super.onCreate();
        // Log.d(TAG, TAG + "onCreate()");
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        Log.d(TAG, msg: TAG + "onHandleIntent()");
        ActivityRecognitionResult result = ActivityRecognitionResult.extractResult(intent);

        // Get the list of the probable activities associated with the current state of the
        // device. Each activity is associated with a confidence level, which is an int between
        // 0 and 100.

        List<DetectedActivity> detectedActivities = result.getProbableActivities();

        for (DetectedActivity activity : detectedActivities) {
            //Log.d(TAG, "Detected activity: " + activity.getType() + ", " + activity.getConfidence());
            broadcastActivity(activity);
        }
    }

    1 usage
    private void broadcastActivity(DetectedActivity activity) {
        // Log.d(TAG, TAG+ "broadcastActivity()");
        Intent intent = new Intent(Constant.BROADCAST_DETECTED_ACTIVITY);
        intent.putExtra( name: "type", activity.getType());
        intent.putExtra( name: "confidence", activity.getConfidence());
        LocalBroadcastManager.getInstance( context: this).sendBroadcast(intent);
    }
}
```

MainActivity.java

Import section for MainActivity.java

```
1  package ar_demo;  
2  
3  import android.content.BroadcastReceiver;  
4  import android.content.Context;  
5  import android.content.Intent;  
6  import android.content.IntentFilter;  
7  import androidx.localbroadcastmanager.content.LocalBroadcastManager;  
8  import androidx.appcompat.app.AppCompatActivity;  
9  import android.os.Bundle;  
10 import android.util.Log;  
11 import android.widget.TextView;  
12  
13 import com.google.android.gms.location.DetectedActivity;  
14  
15 import ar_demo.services.ActivityDetectionService;  
16 import ar_demo.utils.Constant;
```

Handle the activity intent in ar_demo/MainActivity.java

Create intent receiver:

```
57     BroadcastReceiver mActivityBroadcastReceiver = new BroadcastReceiver() {
58
59     @Override
60     public void onReceive(Context context, Intent intent) {
61         // Log.d(TAG, "onReceive()");
62         if (intent.getAction().equals(Constant.BROADCAST_DETECTED_ACTIVITY)) {
63             int type = intent.getIntExtra("type", -1);
64             int confidence = intent.getIntExtra("confidence", 0);
65             handleUserActivity(type, confidence);
66         }
67     }
};
```

- Implement handleUserActivity()

```
private void handleUserActivity(int type, int confidence) {  
    String label = "Unknown";  
    switch (type) {  
        case DetectedActivity.IN_VEHICLE: {  
            label = "In_Vehicle";  
            break;  
        }  
        case DetectedActivity.ON_BICYCLE: {  
            label = "On_Bicycle";  
            break;  
        }  
        case DetectedActivity.ON_FOOT: {  
            label = "On_Foot";  
            break;  
        }  
        case DetectedActivity.RUNNING: {  
            label = "Running";  
            break;  
        }  
        case DetectedActivity.STILL: {  
            label = "Still";  
            break;  
        }  
        case DetectedActivity.TILTING: {  
            label = "Tilting";  
            break;  
        }  
        case DetectedActivity.WALKING: {  
            label = "Walking";  
            break;  
        }  
        case DetectedActivity.UNKNOWN: {  
            break;  
        }  
    }  
    Log.d(TAG, msg: "broadcast:onReceive(): Activity is " + label  
        + " and confidence level is: " + confidence);  
    mTextARLabel.setText(label);  
    mTextConfidence.setText(confidence+"");  
}
```

- in the main activity, call the activity detection service

```
// register the RX and start up the ActivityDetectionService service
@Override
protected void onStart() {
    super.onStart();
    Log.d(TAG, msg: "onStart():start ActivityDetectionService");
    LocalBroadcastManager.getInstance( context: this).registerReceiver(mActivityBroadcastReceiver,
        new IntentFilter(Constant.BROADCAST_DETECTED_ACTIVITY));

    startService(new Intent( packageContext: this, ActivityDetectionService.class));
}
```

- Don't forget common tasks, including unregistering listeners, initiate UI, etc

Overview of MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    4 usages
    public static final String TAG = MainActivity.class.getSimpleName();

    2 usages
    private TextView mTextARLabel;
    2 usages
    private TextView mTextConfidence;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        Log.d(TAG, msg: "onCreate()");
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mTextARLabel = findViewById(R.id.text_label);
        mTextConfidence = findViewById(R.id.text_confidence);
    }

    // register the RX and start up the ActivityDetectionService service
    @Override
    protected void onStart() {
        super.onStart();
        Log.d(TAG, msg: "onStart():start ActivityDetectionService");
        LocalBroadcastManager.getInstance( context: this).registerReceiver(mActivityBroadcastReceiver,
            new IntentFilter(Constant.BROADCAST_DETECTED_ACTIVITY));

        startService(new Intent( packageContext: this, ActivityDetectionService.class));
    }
}
```

Overview of MainActivity.java (2)

```
46
47     // unregister the RX and stop up the ActivityDetectionService service
48
49     @Override
50     protected void onPause() {
51         super.onPause();
52         Log.d(TAG, msg: "onPause():stop ActivityDetectionService");
53         if(mActivityBroadcastReceiver != null){
54             stopService(new Intent(packageContext: this, ActivityDetectionService.class));
55             LocalBroadcastManager.getInstance(this).unregisterReceiver(mActivityBroadcastReceiver);
56         }
57     }
58     BroadcastReceiver mActivityBroadcastReceiver = new BroadcastReceiver() {
59         @Override
60         public void onReceive(Context context, Intent intent) {
61             // Log.d(TAG, "onReceive()");
62             if (intent.getAction().equals(Constant.BROADCAST_DETECTED_ACTIVITY)) {
63                 int type = intent.getIntExtra(name: "type", defaultValue: -1);
64                 int confidence = intent.getIntExtra(name: "confidence", defaultValue: 0);
65                 handleUserActivity(type, confidence);
66             }
67         }
68     };
69 }
```

Overview of MainActivity.java (3)

```
68     private void handleUserActivity(int type, int confidence) {  
69         String label = "Unknown";  
70         switch (type) {  
71             case DetectedActivity.IN_VEHICLE: {  
72                 label = "In_Vehicle";  
73                 break;  
74             }  
75             case DetectedActivity.ON_BICYCLE: {  
76                 label = "On_Bicycle";  
77                 break;  
78             }  
79             case DetectedActivity.ON_FOOT: {  
80                 label = "On_Foot";  
81                 break;  
82             }  
83             case DetectedActivity.RUNNING: {  
84                 label = "Running";  
85                 break;  
86             }  
87             case DetectedActivity.STILL: {  
88                 label = "Still";  
89                 break;  
90             }  
91             case DetectedActivity.TILTING: {  
92                 label = "Tilting";  
93                 break;  
94             }  
95             case DetectedActivity.WALKING: {  
96                 label = "Walking";  
97                 break;  
98             }  
99             case DetectedActivity.UNKNOWN: {  
100                 break;  
101             }  
102         }  
103         Log.d(TAG, msg: "broadcast:onReceive(): Activity is " + label  
104             + " and confidence level is: " + confidence);  
105         mTextARLabel.setText(label);  
106         mTextConfidence.setText(confidence);  
107     }  
108 }  
109 }
```

Build gradle file

```
You can use the Project Structure dialog to view and edit your project configuration
1 apply plugin: 'com.android.application'
2
3 android {
4     compileSdkVersion 29
5     defaultConfig {
6         applicationId "edu.dartmouth.cs.dartnets.ar_demo"
7         minSdkVersion 21
8         targetSdkVersion 27
9         versionCode 1
10        versionName "1.0"
11        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
12    }
13    buildTypes {
14        release {
15            minifyEnabled false
16            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
17        }
18    }
19 }
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'androidx.appcompat:appcompat:1.1.0'
24     implementation ('com.google.android.gms:play-services-location:15.0.0'){
25         exclude group: "com.android.support"
26     }
27     testImplementation 'junit:junit:4.12'
28     androidTestImplementation 'androidx.test.ext:junit:1.1.1'
29     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
30 }
31
```

Final App

