**activity\_main.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*<**LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout\_width="match\_parent"
 android:layout\_height="match\_parent"
 android:orientation="vertical"**>

 <**fragment
 android:id="@+id/mapNearBy"
 android:name="com.google.android.gms.maps.MapFragment"
 android:layout\_width="match\_parent"
 android:layout\_height="0dp"
 android:layout\_below="@+id/rvToolbar"
 android:layout\_weight="1"** />

 <**Button
 android:id="@+id/btnGetDirection"
 android:text="Get Direction"
 android:layout\_width="match\_parent"
 android:layout\_height="wrap\_content"
 android:background="?attr/selectableItemBackground"** />
</**LinearLayout**>

**MainActivity.java**

**package** in.edu.vpt.mapproject;

**import** android.os.Bundle;
**import** android.support.annotation.Nullable;
**import** android.support.v7.app.AppCompatActivity;
**import** android.util.Log;
**import** android.view.View;
**import** android.widget.Button;

**import** com.google.android.gms.maps.CameraUpdateFactory;
**import** com.google.android.gms.maps.GoogleMap;
**import** com.google.android.gms.maps.MapFragment;
**import** com.google.android.gms.maps.OnMapReadyCallback;
**import** com.google.android.gms.maps.model.CameraPosition;
**import** com.google.android.gms.maps.model.LatLng;
**import** com.google.android.gms.maps.model.MarkerOptions;
**import** com.google.android.gms.maps.model.Polyline;
**import** com.google.android.gms.maps.model.PolylineOptions;

**public class** MainActivity **extends** AppCompatActivity **implements** OnMapReadyCallback, TaskLoadedCallback {

 **private** GoogleMap **mMap**;
 **private** MarkerOptions **place1**, **place2**;
 Button **getDirection**;
 **private** Polyline **currentPolyline**;

 @Override
 **protected void** onCreate(@Nullable Bundle savedInstanceState) {
 **super**.onCreate(savedInstanceState);
 setContentView(R.layout.***activity\_main***);

 **getDirection** = findViewById(R.id.***btnGetDirection***);
 **getDirection**.setOnClickListener(**new** View.OnClickListener() {
 @Override
 **public void** onClick(View view) {
 **new** FetchURL(MainActivity.**this**).execute(getUrl(**place1**.getPosition(), **place2**.getPosition(), **"driving"**), **"driving"**);
 }
 });**place1** = **new** MarkerOptions().position(**new** LatLng(19.0216798,72.867759)).title(**"Location 1"**);
 **place2** = **new** MarkerOptions().position(**new** LatLng(19.0186086, 72.8596901)).title(**"Location 2"**);
 MapFragment mapFragment = (MapFragment) getFragmentManager()
 .findFragmentById(R.id.***mapNearBy***);
 mapFragment.getMapAsync(**this**);
 }

 @Override
 **public void** onMapReady(GoogleMap googleMap) {
 **mMap** = googleMap;
 Log.*d*(**"mylog"**, **"Added Markers"**);
 **mMap**.addMarker(**place1**);
 **mMap**.addMarker(**place2**);

 CameraPosition googlePlex = CameraPosition.*builder*()
 .target(**new** LatLng(19.0216798,72.867759))
 .zoom(15)
 .bearing(0)
 .tilt(45)
 .build();

 **mMap**.animateCamera(CameraUpdateFactory.*newCameraPosition*(googlePlex), 5000, **null**);
 }

 **private** String getUrl(LatLng origin, LatLng dest, String directionMode) {String str\_origin = **"origin="** + origin.**latitude** + **","** + origin.**longitude**;String str\_dest = **"destination="** + dest.**latitude** + **","** + dest.**longitude**;String mode = **"mode="** + directionMode;String parameters = str\_origin + **"&"** + str\_dest + **"&"** + mode;String output = **"json"**;String url = **"https://maps.googleapis.com/maps/api/directions/"** + output + **"?"** + parameters + **"&key="** + getString(R.string.***google\_maps\_key***);
 **return** url;
 }

 @Override
 **public void** onTaskDone(Object... values) {
 **if** (**currentPolyline** != **null**)
 **currentPolyline**.remove();
 **currentPolyline** = **mMap**.addPolyline((PolylineOptions) values[0]);
 }
}

**TaskLoadedCallback.java**

**package** in.edu.vpt.mapproject;

**public interface** TaskLoadedCallback {
 **void** onTaskDone(Object... values);
}

**FetchURL.java**

**package** in.edu.vpt.mapproject;

**import** android.content.Context;
**import** android.os.AsyncTask;
**import** android.util.Log;

**import** java.io.BufferedReader;
**import** java.io.IOException;
**import** java.io.InputStream;
**import** java.io.InputStreamReader;
**import** java.net.HttpURLConnection;
**import** java.net.URL;

**public class** FetchURL **extends** AsyncTask<String, Void, String> {
 Context **mContext**;
 String **directionMode** = **"driving"**;

 **public** FetchURL(Context mContext) {
 **this**.**mContext** = mContext;
 }

 @Override
 **protected** String doInBackground(String... strings) {String data = **""**;
 **directionMode** = strings[1];
 **try** {data = downloadUrl(strings[0]);
 Log.*d*(**"mylog"**, **"Background task data "** + data.toString());
 } **catch** (Exception e) {
 Log.*d*(**"Background Task"**, e.toString());
 }
 **return** data;
 }

 @Override
 **protected void** onPostExecute(String s) {
 **super**.onPostExecute(s);
 PointsParser parserTask = **new** PointsParser(**mContext**, **directionMode**);
 *// Invokes the thread for parsing the JSON data* parserTask.execute(s);
 }

 **private** String downloadUrl(String strUrl) **throws** IOException {
 String data = **""**;
 InputStream iStream = **null**;
 HttpURLConnection urlConnection = **null**;
 **try** {
 URL url = **new** URL(strUrl);urlConnection = (HttpURLConnection) url.openConnection();urlConnection.connect();iStream = urlConnection.getInputStream();
 BufferedReader br = **new** BufferedReader(**new** InputStreamReader(iStream));
 StringBuffer sb = **new** StringBuffer();
 String line = **""**;
 **while** ((line = br.readLine()) != **null**) {
 sb.append(line);
 }
 data = sb.toString();
 Log.*d*(**"mylog"**, **"Downloaded URL: "** + data.toString());
 br.close();
 } **catch** (Exception e) {
 Log.*d*(**"mylog"**, **"Exception downloading URL: "** + e.toString());
 } **finally** {
 iStream.close();
 urlConnection.disconnect();
 }
 **return** data;
 }
}

**PointsParser.java**

**package** in.edu.vpt.mapproject;

**import** android.content.Context;
**import** android.graphics.Color;
**import** android.os.AsyncTask;
**import** android.util.Log;
**import** com.google.android.gms.maps.model.LatLng;
**import** com.google.android.gms.maps.model.PolylineOptions;
**import** org.json.JSONObject;
**import** java.util.ArrayList;
**import** java.util.HashMap;
**import** java.util.List;

**public class** PointsParser **extends** AsyncTask<String, Integer, List<List<HashMap<String, String>>>> {

 TaskLoadedCallback **taskCallback**;
 String **directionMode** = **"driving"**;

 **public** PointsParser(Context mContext, String directionMode) {
 **this**.**taskCallback** = (TaskLoadedCallback) mContext;
 **this**.**directionMode** = directionMode;
 }
@Override
 **protected** List<List<HashMap<String, String>>> doInBackground(String... jsonData) {

 JSONObject jObject;
 List<List<HashMap<String, String>>> routes = **null**;

 **try** {
 jObject = **new** JSONObject(jsonData[0]);
 Log.*d*(**"mylog"**, jsonData[0].toString());
 DataParser parser = **new** DataParser();
 Log.*d*(**"mylog"**, parser.toString());
routes = parser.parse(jObject);
 Log.*d*(**"mylog"**, **"Executing routes"**);
 Log.*d*(**"mylog"**, routes.toString());

 } **catch** (Exception e) {
 Log.*d*(**"mylog"**, e.toString());
 e.printStackTrace();
 }
 **return** routes;
 }
@Override
 **protected void** onPostExecute(List<List<HashMap<String, String>>> result) {
 ArrayList<LatLng> points;
 PolylineOptions lineOptions = **null**;**for** (**int** i = 0; i < result.size(); i++) {
 points = **new** ArrayList<>();
 lineOptions = **new** PolylineOptions();List<HashMap<String, String>> path = result.get(i);**for** (**int** j = 0; j < path.size(); j++) {
 HashMap<String, String> point = path.get(j);
 **double** lat = Double.*parseDouble*(point.get(**"lat"**));
 **double** lng = Double.*parseDouble*(point.get(**"lng"**));
 LatLng position = **new** LatLng(lat, lng);
 points.add(position);
 }
 lineOptions.addAll(points);
 **if** (**directionMode**.equalsIgnoreCase(**"walking"**)) {
 lineOptions.width(10);
 lineOptions.color(Color.***MAGENTA***);
 } **else** {
 lineOptions.width(20);
 lineOptions.color(Color.***RED***);
 }
 Log.*d*(**"mylog"**, **"onPostExecute lineoptions decoded"**);
 }
**if** (lineOptions != **null**) {**taskCallback**.onTaskDone(lineOptions);

 } **else** {
 Log.*d*(**"mylog"**, **"without Polylines drawn"**);
 }
 }
}

**DirectionsJSONParser.java**

**package** in.edu.vpt.mapproject;

**import** com.google.android.gms.maps.model.LatLng;

**import** org.json.JSONArray;
**import** org.json.JSONException;
**import** org.json.JSONObject;

**import** java.util.ArrayList;
**import** java.util.HashMap;
**import** java.util.List;

**public class** DirectionsJSONParser {

**public** List<List<HashMap<String,String>>> parse(JSONObject jObject){

 List<List<HashMap<String, String>>> routes = **new** ArrayList<List<HashMap<String,String>>>() ;
 JSONArray jRoutes = **null**;
 JSONArray jLegs = **null**;
 JSONArray jSteps = **null**;

 **try** {

 jRoutes = jObject.getJSONArray(**"routes"**);

 */\*\* Traversing all routes \*/* **for**(**int** i=0;i<jRoutes.length();i++){
 jLegs = ( (JSONObject)jRoutes.get(i)).getJSONArray(**"legs"**);
 List path = **new** ArrayList<HashMap<String, String>>();

 */\*\* Traversing all legs \*/* **for**(**int** j=0;j<jLegs.length();j++){
 jSteps = ( (JSONObject)jLegs.get(j)).getJSONArray(**"steps"**);

 */\*\* Traversing all steps \*/* **for**(**int** k=0;k<jSteps.length();k++){
 String polyline = **""**;
 polyline = (String)((JSONObject)((JSONObject)jSteps.get(k)).get(**"polyline"**)).get(**"points"**);
 List list = decodePoly(polyline);

 */\*\* Traversing all points \*/* **for**(**int** l=0;l <list.size();l++){
 HashMap<String, String> hm = **new** HashMap<String, String>();
 hm.put(**"lat"**, Double.*toString*(((LatLng)list.get(l)).**latitude**) );
 hm.put(**"lng"**, Double.*toString*(((LatLng)list.get(l)).**longitude**) );
 path.add(hm);
 }
 }
 routes.add(path);
 }
 }

 } **catch** (JSONException e) {
 e.printStackTrace();
 }**catch** (Exception e){
 }

 **return** routes;
 }

 **private** List decodePoly(String encoded) {

 List poly = **new** ArrayList();
 **int** index = 0, len = encoded.length();
 **int** lat = 0, lng = 0;

 **while** (index < len) {
 **int** b, shift = 0, result = 0;
 **do** {
 b = encoded.charAt(index++) - 63;
 result |= (b & 0x1f) << shift;
 shift += 5;
 } **while** (b >= 0x20);
 **int** dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
 lat += dlat;

 shift = 0;
 result = 0;
 **do** {
 b = encoded.charAt(index++) - 63;
 result |= (b & 0x1f) << shift;
 shift += 5;
 } **while** (b >= 0x20);
 **int** dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
 lng += dlng;

 LatLng p = **new** LatLng((((**double**) lat / 1E5)),
 (((**double**) lng / 1E5)));
 poly.add(p);
 }

 **return** poly;
 }
}

**Output**

****