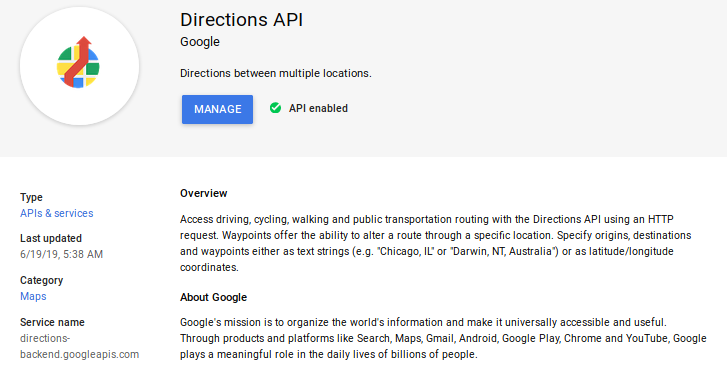
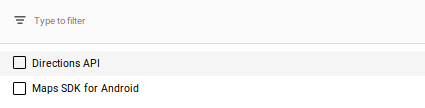
DIRECTION

This tutorial describes the usage of the Google Map API.

**EXAMPLE : Direction Between Two Point In Google Map**

Before starting route between two point, First we need to **enable** the **Directions API**.





The Directions API is a service that calculates directions between locations. You can search for directions for several modes of transportation, including transit, driving, walking, or cycling.

***Sample request and response***

You access the Directions API through an HTTP interface, with requests constructed as a URL string, using text strings or latitude/longitude coordinates to identify the locations, along with your API key.

**For LatLng link**

https://maps.googleapis.com/maps/api/directions/json?origin=26.111150122727047,76.71888649463654&destination=23.089245587708394,78.42668753117323&sensor=false&key=AIzaSyDjLvVE02f5YynKxTt7kTO\_PZvagj1WvsQ

**For Address link**

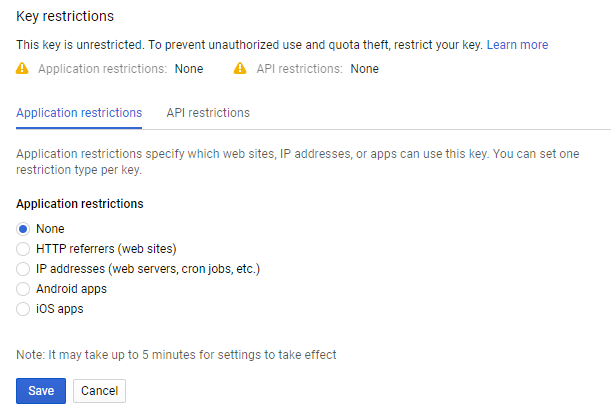
The following example requests the driving directions from Disneyland to Universal Studios Hollywood, in JSON format :

<https://maps.googleapis.com/maps/api/directions/json?origin=Disneyland&destination=Universal+Studios+Hollywood&key=YOUR_API_KEY>

Try it! You can test this request by entering the URL into your web browser (be sure to replace YOUR\_API\_KEY with [your actual API key](https://developers.google.com/maps/documentation/directions/start" \l "get-a-key)). The response returns the driving directions.

But be sure that your key restrictions are well configured. If you have restricted the Key to your app, it may not work because the request made via HTTP are related with your IP address. Meaning that even if you have correctly configured your Android App and manifest, it may result with an ACCESS\_DENIED. This can be verified by changing your restriction.

Example of no restriction for testing purpose :

[](https://i.stack.imgur.com/z1cHh.png)

**STEP 1 :-** build.gradle ( Module: app ) file.

You need to import the Play Services library for maps. In this example, you also need to import the locations Play Services library in order to set an initial position for your map.

apply plugin: 'com.android.application'  
  
android {  
 compileSdkVersion 28  
 defaultConfig {  
 applicationId "map.project.com"  
 minSdkVersion 18  
 targetSdkVersion 28  
 versionCode 1  
 versionName "1.0"  
 testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
 resValue "string", "google\_maps\_key", (project.findProperty("GOOGLE\_MAPS\_API\_KEY") ?: "")  
 }  
 buildTypes {  
 release {  
 minifyEnabled false  
 proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'  
 }  
 }  
}  
  
dependencies {  
 implementation fileTree(dir: 'libs', include: ['\*.jar'])  
 implementation 'com.android.support:appcompat-v7:28.0.0'  
 implementation 'com.android.support.constraint:constraint-layout:1.1.3'  
 testImplementation 'junit:junit:4.12'  
 androidTestImplementation 'com.android.support.test:runner:1.0.2'  
 androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'  
  
 implementation 'com.android.support:design:28.0.0'  
 implementation 'com.google.android.gms:play-services-maps:11.8.0'  
 implementation 'com.google.android.gms:play-services-location:11.8.0'  
}

**STEP 2 :-** AndroidManifest.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="map.project.com">  
 *<!--  
 The ACCESS\_COARSE/FINE\_LOCATION permissions are not required to use  
 Google Maps Android API v2, but you must specify either coarse or fine  
 location permissions for the 'MyLocation' functionality.  
 -->* <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  
 <uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />  
 <uses-permission android:name="android.permission.INTERNET" />  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
 <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
 <uses-permission android:name="com.google.android.providers.gsf.permission.READ\_GSERVICES" />  
  
 <permission  
 android:name="map.project.com.permission.MAPS\_RECEIVE"  
 android:protectionLevel="signature" />  
 <uses-permission android:name="map.project.com.permission.MAPS\_RECEIVE" />  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher\_round"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme">  
  
 <meta-data  
 android:name="com.google.android.gms.version"  
 android:value="@integer/google\_play\_services\_version"/>  
  
 <meta-data  
 android:name="com.google.android.geo.API\_KEY"  
 android:value="@string/google\_maps\_key"/>  
  
 <activity android:name=".MapActivity"/>  
 <activity android:name=".MainActivity">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
   
 </application>  
 *<!--  
 Google Maps Android API requires OpenGL ES version 2 so add following tag to manifest,  
 Required for Google Maps V2  
 -->* <uses-feature  
 android:glEsVersion="0x00020000"  
 android:required="true"/>  
</manifest>

**STEP 3 :-** activity\_main.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <Button  
 android:id="@+id/go\_on\_map\_button"  
 android:layout\_width="200dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="216dp"  
 android:textSize="20sp"  
 android:text="Go On Map" />  
</RelativeLayout>

**STEP 4 :-** activity\_map.xml

*<?*xml version="1.0" encoding="utf-8"*?>*<fragment xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:map="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/map"  
 android:name="com.google.android.gms.maps.SupportMapFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MapActivity"/>

**STEP 5 :-** MainActivity.java

import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
import android.content.Intent;  
import android.view.View;  
import android.widget.Button;  
public class MainActivity extends AppCompatActivity {  
  
 Button GoOnMapButton;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 GoOnMapButton=(Button)findViewById(R.id.*go\_on\_map\_button*);  
  
 GoOnMapButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent i = new Intent(MainActivity.this, MapActivity.class);  
 startActivity(i);  
 }  
 });  
 }  
}

**STEP 6 :-** MapUtils.java

**import** com.google.android.gms.maps.CameraUpdateFactory;  
**import** com.google.android.gms.maps.GoogleMap;  
**import** com.google.android.gms.maps.model.BitmapDescriptorFactory;  
**import** com.google.android.gms.maps.model.CameraPosition;  
**import** com.google.android.gms.maps.model.LatLng;  
**import** com.google.android.gms.maps.model.Marker;  
**import** com.google.android.gms.maps.model.MarkerOptions;  
  
**public class** MapUtils {  
  
 **public static** Marker setMarker(LatLng point, GoogleMap googleMap, String snippetString, String titleString, Integer resourceId)   
 {  
 *// Creating an instance of MarkerOptions* MarkerOptions markerOptions = **new** MarkerOptions();  
 *// Setting latitude and longitude for the marker* markerOptions.position(point);  
 *// Setting snippet for the InfoWindow* markerOptions.snippet(snippetString);  
 *// Setting title for the InfoWindow* markerOptions.title(titleString);  
 markerOptions.visible(**true**);  
 *// Custom marker icon* markerOptions.icon(BitmapDescriptorFactory.*fromResource*(resourceId));  
 *// Adding marker on the Google Map* Marker marker = googleMap.addMarker(markerOptions);  
 *//Moving Camera to a Location with animation* CameraPosition cameraPosition = **new** CameraPosition.Builder()  
 .target(point) *// Sets the center of the map to location user* .zoom(8) *// Sets the zoom* .bearing(90) *// Sets the orientation of the camera to east* .tilt(40) *// Sets the tilt of the camera to 30 degrees* .build(); *// Creates a CameraPosition from the builder* googleMap.animateCamera(CameraUpdateFactory.*newCameraPosition*(cameraPosition));  
 **return** marker;  
 }  
}

**STEP 7 :-** GoogleMapDirectionApiUtils.java

import android.util.Log;  
import com.google.android.gms.maps.model.LatLng;  
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 GoogleMapDirectionApiUtils {  
  
 public static String getGoogleDirectionsApiUrl(LatLng pickupLocation, LatLng dropLocation, String directionMode)  
 {  
  
 StringBuilder googleDirectionsUrl = new StringBuilder("https://maps.googleapis.com/maps/api/directions/");  
  
 *// Output format* String output = "json";  
 *// Sensor enabled* boolean sensor = false;  
  
 googleDirectionsUrl.append(output);  
 googleDirectionsUrl.append("?");  
 googleDirectionsUrl.append("origin=").append(pickupLocation.latitude).append(",").append(pickupLocation.longitude);  
 googleDirectionsUrl.append("&destination=").append(dropLocation.latitude).append(",").append(dropLocation.longitude);  
 googleDirectionsUrl.append("&sensor=").append(sensor);  
 *// googleDirectionsUrl.append("&units=metric");  
 /\*  
 \* We’ve set the mode=driving in the current application.  
 \* The other modes of transport are:  
 \* driving (default)  
 \* walking  
 \* bicycling  
 \* transit  
 \*/  
 // googleDirectionsUrl.append("&mode=").append(directionMode);* googleDirectionsUrl.append("&key="+"AIzaSyDjLvVE02f5YynKxTt7kTO\_PZvagj1WvsQ");  
 return googleDirectionsUrl.toString();  
 }  
  
 public static String downloadUrl(String responseUrl) throws IOException  
 {  
 String data="";  
 InputStream inputStream = null;  
 HttpURLConnection connection = null;  
 try  
 {  
 URL directionUrl = new URL(responseUrl);  
 connection = (HttpURLConnection) directionUrl.openConnection();  
 connection.connect();  
 inputStream = connection.getInputStream();  
  
 BufferedReader bufferedReader = new BufferedReader(new InputStreamReader(inputStream));  
 StringBuffer stringBuffer = new StringBuffer();  
  
 String line = "";  
 while ((line = bufferedReader.readLine()) != null)  
 {  
 stringBuffer.append(line);  
 }  
  
 data = stringBuffer.toString();  
 bufferedReader.close();  
 }  
 catch (Exception e)  
 {  
 Log.*e*("FETCH : ", "Exception : " + e.toString());  
 return null;  
 }  
 finally  
 {  
 inputStream.close();  
 connection.disconnect();  
 }  
 return data;  
 }  
}

**STEP 8 :-** FetchingDirectionFromGoogleDirectionApi.java

import android.os.AsyncTask;  
import android.util.Log;  
import com.google.android.gms.maps.GoogleMap;  
  
public class FetchingDirectionFromGoogleDirectionApi extends AsyncTask<String, Void, String> {  
  
 String TAG = getClass().getSimpleName();  
 GoogleMap googleMap;  
  
 public FetchingDirectionFromGoogleDirectionApi(GoogleMap googleMap) {  
 this.googleMap = googleMap;  
 }  
  
 @Override  
 protected void onPreExecute() {  
 Log.*d*(TAG + " onPreExecute","On pre Exceute......");  
 super.onPreExecute();  
 }  
  
 @Override  
 protected String doInBackground(String... url) {  
 Log.*d*(TAG + " doInBackground","On doInBackground...");  
  
 */\*  
 \* For storing json response in this variable  
 \*/* String responseFromDirectionApi= "";  
  
 try  
 {  
 responseFromDirectionApi = GoogleMapDirectionApiUtils.*downloadUrl*(url[0]);  
 Log.*d*("Background Task data", responseFromDirectionApi.toString());  
 }  
 catch (Exception e)  
 {  
 Log.*d*("Background Task", e.toString());  
 }  
 return responseFromDirectionApi;  
 }  
  
 @Override  
 protected void onProgressUpdate(Void... values) {  
 Log.*d*(TAG + "onProgressUpdate","On pre Exceute......");  
 super.onProgressUpdate(values);  
 }  
  
 @Override  
 protected void onPostExecute(String result) {  
 Log.*d*(TAG + " onPostExecute", "" + result);  
 ParseDirectionApiJsonResponse parseDirectionApiJsonResponse = new ParseDirectionApiJsonResponse(googleMap);  
 parseDirectionApiJsonResponse.execute(result);  
 super.onPostExecute(result);  
 }  
  
 @Override  
 protected void onCancelled(String s) {  
 super.onCancelled(s);  
 }  
  
 @Override  
 protected void onCancelled() {  
 super.onCancelled();  
 }  
}

**STEP 9 :-** ParseDirectionApiJsonResponse.java

**package** map.project.com;  
**import** android.graphics.Color;  
**import** android.os.AsyncTask;  
**import** android.util.Log;  
**import** com.google.android.gms.maps.GoogleMap;  
**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;  
**import** java.util.Set;  
  
**public class** ParseDirectionApiJsonResponse **extends** AsyncTask<String, Integer, List<List<HashMap<String, String>>>> {  
  
 String **TAG** = getClass().getSimpleName();  
 GoogleMap **googleMap**;  
  
 **public** ParseDirectionApiJsonResponse(GoogleMap googleMap) {  
 **this**.**googleMap** = googleMap;  
 }  
  
 @Override  
 **protected void** onPreExecute() {  
 Log.*d*(**TAG** + **" onPreExecute()"**,**"On pre Exceute......"**);  
 **super**.onPreExecute();  
 }  
  
 *// Parsing the data in non-ui thread* @Override  
 **protected** List<List<HashMap<String, String>>> doInBackground(String... jsonData)  
 {  
 Log.*d*(**TAG** + **" doInBackground"**,**"On doInBackground..."**);  
  
 JSONObject jObject;  
 List<List<HashMap<String, String>>> routes = **null**;  
  
 **try** {  
 jObject = **new** JSONObject(jsonData[0]);  
 Log.*d*(**"ParserTask"**,jsonData[0].toString());  
 DirectionsJSONParser parser = **new** DirectionsJSONParser();  
 Log.*d*(**"ParserTask"**, parser.toString());  
  
 *// Starts parsing data* routes = parser.parse(jObject);  
 Log.*d*(**"ParserTask"**,**"Executing routes"**);  
 Log.*d*(**"ParserTask"**,routes.toString());  
 }  
 **catch** (Exception e)  
 {  
 Log.*d*(**"ParserTask"**,e.toString());  
 e.printStackTrace();  
 }  
 **return** routes;  
 }  
  
 @Override  
 **protected void** onProgressUpdate(Integer... values) {  
 Log.*d*(**TAG** + **"onProgressUpdate"**,**"On pre Exceute......"**);  
 **super**.onProgressUpdate(values);  
 }  
  
 *// Executes in UI thread, after the parsing process* @Override  
 **protected void** onPostExecute(List<List<HashMap<String, String>>> lists) {  
 Log.*d*(**TAG** + **" onPostExecute"**, **""** + lists);  
  
 ArrayList<LatLng> points;  
 PolylineOptions lineOptions = **null**;  
  
 *// Traversing through all the routes* **for** (**int** i = 0; i < lists.size(); i++) {  
 points = **new** ArrayList<>();  
 lineOptions = **new** PolylineOptions();  
  
 *// Fetching i-th route* List<HashMap<String, String>> path = lists.get(i);  
  
 *// Fetching all the points in i-th route* **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);  
 }  
  
 *// Adding all the points in the route to LineOptions* lineOptions.addAll(points);  
 lineOptions.width(8);  
 lineOptions.color(Color.*parseColor*(**"#D81B60"**));  
  
 Log.*d*(**"onPostExecute"**,**"onPostExecute lineoptions decoded"**);  
  
 }  
 *// Drawing polyline in the Google Map for the i-th route* **if**(lineOptions != **null**) {  
 **googleMap**.addPolyline(lineOptions);  
  
 HashMap<String, String> durationAndDistance = **new** HashMap<String, String>();  
 *// Starts parsing data* durationAndDistance = DirectionsJSONParser.*durationAndDistance*;  
 Set<String> keys = durationAndDistance.keySet();  
 **for**(String key: keys){  
 System.***out***.println(key);  
 System.***out***.println(durationAndDistance.get(key));  
 }  
 }  
 **else** {  
 Log.*d*(**"onPostExecute"**,**"without Polylines drawn"**);  
 }  
 **super**.onPostExecute(lists);  
 }  
  
 @Override  
 **protected void** onCancelled(List<List<HashMap<String, String>>> lists) {  
 **super**.onCancelled(lists);  
 }  
  
 @Override  
 **protected void** onCancelled() {  
 **super**.onCancelled();  
 }  
}

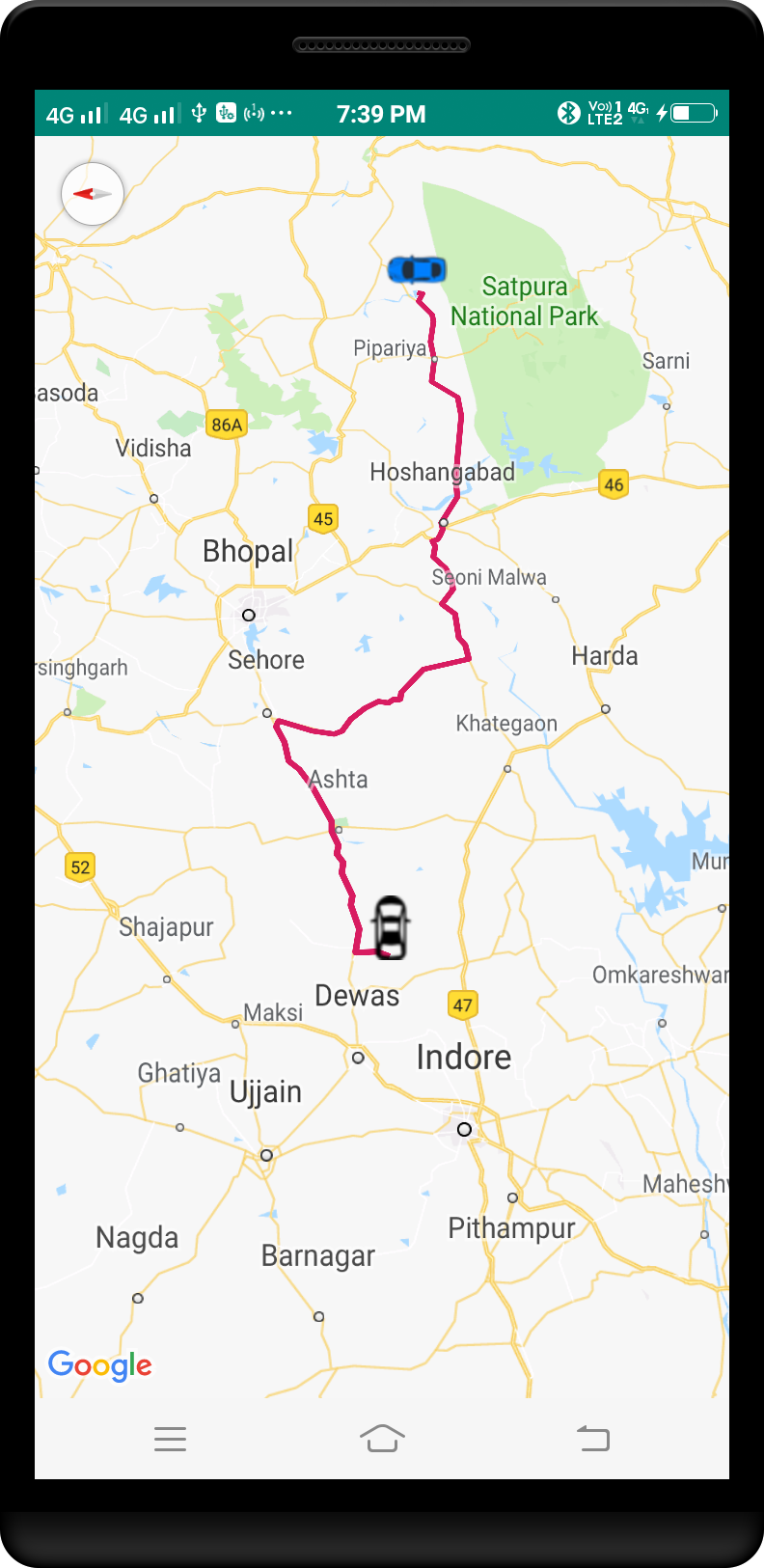
**STEP 10 :-** DirectionsJsonParser.java

package map.project.com;  
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 static HashMap<String, String> *durationAndDistance* = new HashMap<String, String>();  
  
 public HashMap<String,String> getDuration(JSONArray googleDirectionsJson)  
 {  
 HashMap<String,String> googleDirectionsMap = new HashMap<>();  
 String duration = "";  
 String distance ="";  
 String end\_address="";  
 String start\_address="";  
  
 try {  
  
 duration = googleDirectionsJson.getJSONObject(0).getJSONObject("duration").getString("text");  
 distance = googleDirectionsJson.getJSONObject(0).getJSONObject("distance").getString("text");  
 end\_address = googleDirectionsJson.getJSONObject(0).getString("end\_address");  
 start\_address = googleDirectionsJson.getJSONObject(0).getString("start\_address");  
  
 googleDirectionsMap.put("duration" , duration);  
 googleDirectionsMap.put("distance", distance);  
 googleDirectionsMap.put("end\_address", end\_address);  
 googleDirectionsMap.put("start\_address", start\_address);  
  
 } catch (JSONException e) {  
 e.printStackTrace();  
 }  
 return googleDirectionsMap;  
 }  
  
 */\*\*  
 \* Receives a JSONObject and returns a list of lists containing latitude and longitude  
 \*/* 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");  
 *durationAndDistance*=getDuration(jLegs);  
 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<LatLng> 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;  
 }  
  
 */\*\*  
 \* Method to decode polyline points  
 \* Courtesy : http://jeffreysambells.com/2010/05/27/decoding-polylines-from-google-maps-direction-api-with-java  
 \*/* private List<LatLng> decodePoly(String encoded) {  
  
 List<LatLng> poly = new ArrayList<LatLng>();  
 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;  
 }  
}

**STEP 11 :-** MapActivity.java

import android.support.v4.app.FragmentActivity;  
import android.os.Bundle;  
import com.google.android.gms.maps.GoogleMap;  
import com.google.android.gms.maps.OnMapReadyCallback;  
import com.google.android.gms.maps.SupportMapFragment;  
import com.google.android.gms.maps.CameraUpdateFactory;  
import com.google.android.gms.maps.model.CameraPosition;  
import com.google.android.gms.maps.model.LatLng;  
import android.util.Log;  
import com.google.android.gms.maps.model.Marker;  
import com.google.android.gms.maps.model.MarkerOptions;  
import java.util.ArrayList;  
  
public class MapActivity extends FragmentActivity implements OnMapReadyCallback {  
  
 private GoogleMap googleMap;  
 private SupportMapFragment mapFragment;  
  
 */\*\*  
 \* Two LatLng and Two marker icon  
 \*/* private Integer[] markerIcons = {R.drawable.*car*,R.drawable.*car\_1*};  
 ArrayList<LatLng> MarkerPoints;  
  
 */\*\*  
 \* Direction Mode  
 \*/* public final static String *MODE\_DRIVING* = "driving";  
 public final static String *MODE\_WALKING* = "walking";  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_map*);  
  
 mapFragment = (SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.*map*);  
 assert mapFragment != null;  
 mapFragment.getMapAsync(this);  
  
 */\*  
 \* Initializing Two LatLng ArrayList  
 \*/* MarkerPoints = new ArrayList<>();  
 }  
  
 @Override  
 public void onMapReady(GoogleMap map) {  
 googleMap = map;  
  
 LatLng india = new LatLng(20.5937, 78.9629);  
 googleMap.addMarker(new MarkerOptions().position(india).title("INDIA"));  
 CameraPosition cameraPosition = new CameraPosition.Builder().target(india).zoom(4).build();  
 googleMap.animateCamera(CameraUpdateFactory.*newCameraPosition*(cameraPosition));  
  
 googleMap.setOnMapClickListener(new GoogleMap.OnMapClickListener() {  
  
 @Override  
 public void onMapClick(LatLng point) {  
  
 */\*  
 \* Already two locations  
 \* Store only two LatLng in ArrayList if more  
 \* then two then clear map and again add two marker  
 \*/* if (MarkerPoints.size() > 1)  
 {  
 MarkerPoints.clear();  
 googleMap.clear();  
 }  
  
 */\*  
 \* Adding new LatLng to the ArrayList  
 \*/* MarkerPoints.add(point);  
  
 */\*\*  
 \* For the start location, the black car and  
 \* for the end location, the blue car.  
 \*/* if (MarkerPoints.size() == 1) {  
 Marker startMarker = MapUtils.*setMarker*(point,googleMap,"Start Location","Black Car",markerIcons[0]);  
  
 } else if (MarkerPoints.size() == 2) {  
 Marker endMarker = MapUtils.*setMarker*(point,googleMap,"End Location","Blue Car",markerIcons[1]);  
 }  
  
 */\*  
 \* Checks, whether start and end locations are captured  
 \*/* if (MarkerPoints.size() >= 2)  
 {  
 LatLng pickupLocation = MarkerPoints.get(0);  
 LatLng dropLocation = MarkerPoints.get(1);  
  
 */\*  
 \* Getting URL to the Google Directions API  
 \*/* String googleDirectionsApiUrl = GoogleMapDirectionApiUtils.*getGoogleDirectionsApiUrl*(pickupLocation, dropLocation,*MODE\_DRIVING*);  
 Log.*d*("DIRECTION API URL : ", googleDirectionsApiUrl);  
  
 */\*  
 \* Start downloading json data from Google Directions API  
 \*/* FetchingDirectionFromGoogleDirectionApi FetchUrl = new FetchingDirectionFromGoogleDirectionApi(googleMap);  
 FetchUrl.execute(googleDirectionsApiUrl);  
  
 *//move map camera* googleMap.moveCamera(CameraUpdateFactory.*newLatLng*(pickupLocation));  
 googleMap.animateCamera(CameraUpdateFactory.*zoomTo*(11));  
 }  
 }  
 });  
 }  
}

**STEP 12 :-** Run Application



**EXAMPLE : Calculate Distance between ride starts to end Using Distance Matrix Api**

***MY STORY WHEN WORKING WITH GOOGLE DISTANCE MATRIX API***

***https://stackoverflow.com/questions/30161395/im-trying-to-search-nearby-places-such-as-banks-restaurants-atms-inside-the-d***

A few months ago when I’m working on my freelancing projecthttp://wptrafficanalyzer.in/blog/android-geocoding-showing-user-input-location-on-google-map-android-api-v2/. This project is like to build an application like Careem or Uber. You see an application like Careem or Uber required a lot of working with Google Maps, Places and Distance Matrix API and many others. I’ve tackled a lot of difficulties when working on this application but there’s one problem where I stuck for days.

**PROBLEM**

The Uber and Careem have their driver app at which passenger request for the ride. After accepting the passenger request driver needs to reach passenger location and then the driver starts the ride. When the ride starts the driver application start tracking the distance from origin to the passenger destination. At first, I think that’s easy store the location when a driver starts the ride and execute the distance matrix request with the destination location. That’s it!

**DISTANCE TRACKING PROBLEM**

One day, when I’m testing the app I start the ride from my house go to the market came back then complete the ride. But when I see the total traveling distance is just two meters I shocked. I said, to myself, the market is two kilometers away from my home and that’s the one-way distance. So the total distance needs to be 4 kilometers. After an hour of thinking I came up with the problem that my start ride co-ordinates and end ride co-ordinates are the same.

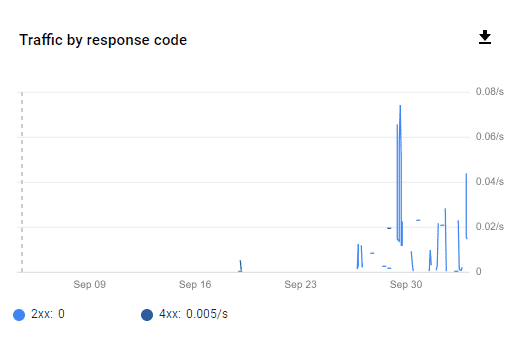
**DISTANCE TRACKING SOLUTION**

Another solution which I come up with after searching on **[StackOverflow](https://stackoverflow.com/)** and talking with other developers on **[Reddit](https://www.reddit.com/)** is to execute the Distance Matrix API request whenever the driver current location changed. This approach is better than the previous one and most of all calculate the correct tracking distance. I need to do two extra things with this approach.

1. I need to request the Google Distance Matrix API every time when the driver current location changed.
2. Need to store the result of Distance Matrix API, store the destination location and again when the current location change executes the request with the previously stored location and so on…

**DISTANCE MATRIX BILLING PROBLEM**

Everything seems to be working fine unless I see the graph of Distance Matrix API graph in [google console](https://console.cloud.google.com/). After every three seconds, one request is executed for distance calculation.



**DISTANCE MATRIX BILLING SOLUTION**

Another day at work and need to find out another solution for distance tracking. At the end of the day I finally able to find a solution which generates less billing and better in performance from the previous one. Now instead of executing distance request whenever the driver current location changed, I started a background service and also increase the interval for Distance Matrix network request. The new interval is ten seconds, so it means the API billing is three times less.

<https://codinginfinite.com/android-location-distance-tracking-app>