ANDROID REAL-TIME USER LOCATION TRACKING USING GOOGLE MAP API V2

In this tutorial, we are going to learn how to implement Android Real-time user location tracking using Google Map API v2.

Tracking user location can be beneficial when you want to record the distance and route a user has gone through during exercise or other activities. Based on the distance, you can calculate other parameters and present a meaningful and concise result to the user.

Android real-time user location tracking using Google Map API v2 are usually use in sports and social networking apps.

If you are looking for a way to **draw path between two points on Google Map** or **to measures the distance and duration between two points**. I will suggest you first read my posts on these topics

WHAT WE ARE GOING TO ACHIEVE

1. Create and load a Google Map

2. Create a Background Service that will listener to onLocationChange events

3. Send the location data to a Local Broadcast Receiver

4. Update the UI Map View using the overlay Polyline to draw route path.

Before we start it is important that that we understand what we are planning to achieve in this tutorial. I have add some screen-shot from this tutorial

APP SCREEN-SHOT

- Android Retrofit 2 with JSON API Example
- Android Room with LiveData Example Tutorial
- Android UI App for Food Ordering and Delivery from Multiple Restaurants

CREATE NEW ANDROID PROJECT

Windows 10

Android Studio

Sony Xperia ZL

Min SDK 14

Target SDK 23

To create a new android application project, follow the steps as stipulated below.

Go to File menu

Click on New menu

Click on Android Application

Enter Project name: AndroidRealtimeLocationTracking

Package: com.inducesmile.androidrealtimelocationtracking

Select Map Activity

Name your activity: MapsActivity

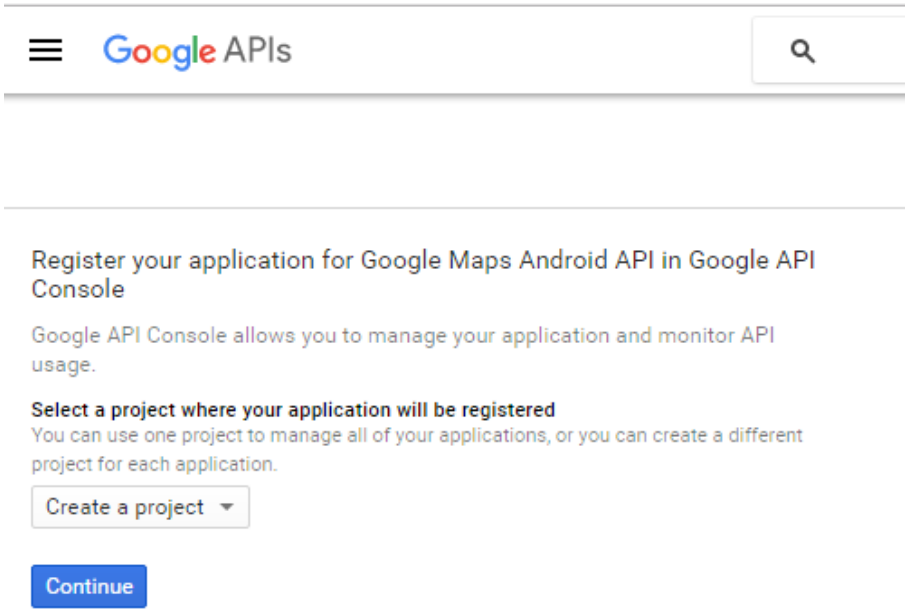
Keep other default selections

Continue to click on next button until Finish button is active, then click on Finish Button.

Since we selected the default android Map Activity template, Android Studio will add an xml file name ***google_maps_api.xml.*** This file is stored in the values folder inside the res folder of your android project.

When you open the file, it contains instruction on how to obtain a Google Map Key. Every request your application send to Google Map Server requires a unique key that will be used to identify your application.

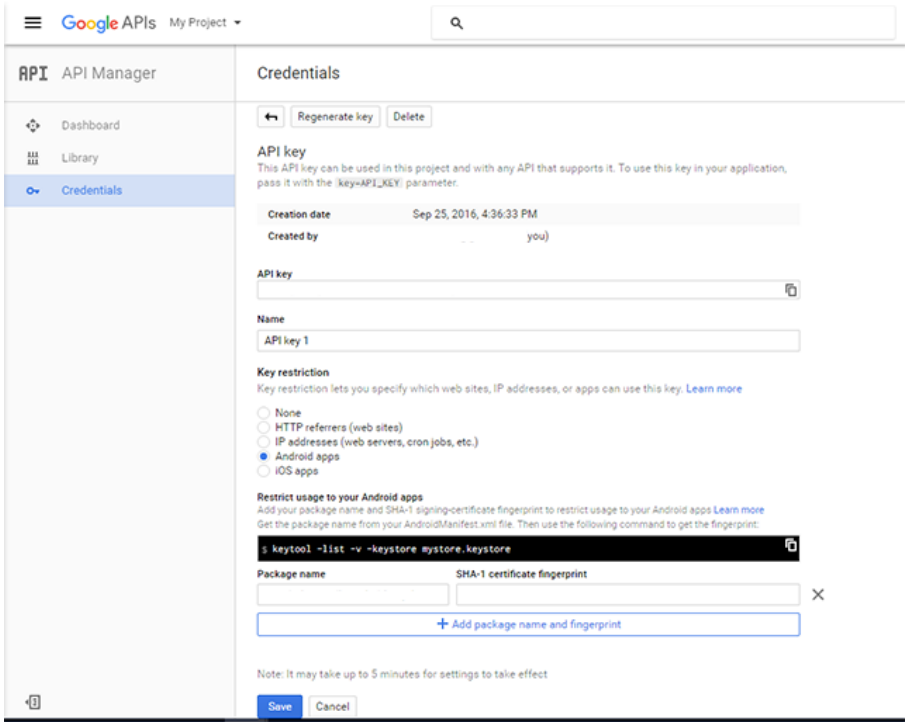
Copy for creating a Google Map Key and paste it in a web browser and hit enter. You will see a page like this.



You can create a new project or use an existing project. Click the continue button to proceed.

Click the Create AOI Key button that will appear in the next page to move over to Google API Manager page.

In the Google API Manager page, click on credentials and the key link to open the page.



Add a name for your key.

Select Android apps to restrict all the request from android apps

Add the your application package name

You can generate a SHA-1 certificate fingerprint. You will find the process on the page.

Click the Save button when you are done.

Finally, copy your application Google map key to the generated ***google_maps_api.xml*** as shown.

google_maps_api.xml </>

```
<resources>
    <!--
    TODO: Before you run your application, you need a Go
    To get one, follow this link, follow the directions
    https://console.developers.google.com/flows/enableap
    You can also add your credentials to an existing key
```

Accept

```
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy
</resources>
```

BUILD.GRADLE

In android, since we are going to make use of user location in drawing path between two points in Google Map API, we are going to use Google Play Services. Android Location Service API is part of Google Play Services.

Since the library is too big and to avoid going beyond 64000 methods which will force use to multiDexEnabled true in the defaultConfig, we will use Location and Map libraries alone.

Inaddition to these libraries, we are going to make use of Volley network library and Gson library.

Open your application build.gradle and add the code below.

```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 24
    buildToolsVersion "24.0.1"
    defaultConfig {
        applicationId "com.inducesmile.androidlocationtr
        minSdkVersion 14
        targetSdkVersion 24
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('progua
        }
```

```
    compile 'com.android.support:appcompat-v7:24.2.1'
    compile 'com.google.android.gms:play-services-maps:9
    compile 'com.google.android.gms:play-services-locati
    compile 'com.readystatesoftware.sqliteasset:sqliteas
    compile 'com.google.code.gson:gson:2.6.1'
    compile 'com.mcxiaoke.volley:library:1.0.19'
}
```

ANDROIDMANIFEST.XML

We are going to update our application androidmanifest.xml. Using Android Location requires that our application must request for user permission before it can access their location. Starting from android 6, location request are run time permission which the user will grant or deny while using the app.

Open your AndroidManifest.xml file and add the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/
    package="com.inducesmile.androidlocationtracking">
    <uses-permission android:name="android.permission.AC
    <uses-permission android:name="android.permission.AC
    <uses-permission android:name="android.permission.IN
    <application
        android:name=".network.CustomApplication"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.act
                <category android:name="android.intent.c
            </intent-filter>
        </activity>
```

```
            android:name=".MapTrackingActivity"
            android:label="@string/title_activity_map_tr
        <activity android:name=".SettingsActivity"></act
        <service
            android:name=".RouteService"
            android:enabled="true"
            android:exported="false"></service>
        <activity android:name=".RecordResultActivity"><
    </application>
</manifest>
```

The meta-data retrieves the Google Map Key that we obtained before.

STRINGS.XML

We are going to update our project strings.xml file located in the values folder inside the res folder. Open the file and add the code below to it.

```
<resources>
    <string name="app_name">Android Real-time Location T
    <string name="title_activity_map_tracking">Map Route
    <string name="track_path_on_map">Walk and route your
    <string name="permission_notice">Please note that yo
    <string name="permission_title">Location Permission
    <string name="cancel_permission_notice">Wow, please
</resources>
```

COLORS.XML

Open the colors.xml file in the same location as the strings.xml file and add the code below to the file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
```

```
    <color name="colorPrimaryText">#212121</color>
    <color name="colorSecondaryText">#757575</color>
    <color name="colorIcons">#FFFFFF</color>
    <color name="colorDivider">#BDBDBD</color>
</resources>
```

ACTIVITY_MAP_TRACKING.XML

The MapTracking class main layout file consist of a Framelayout ViewGroup with a Fragment and Button as children.

Open the layout file and add the code below.

```
<FrameLayout xmlns:android="http://schemas.android.com/a
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:layout_width="match_parent">
    <fragment
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.Suppor
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center"
        tools:context="com.inducesmile.androidlocationtr
    <Button
        android:id="@+id/start_tracking"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="16dp"
        android:background="@color/colorPrimaryDark"
        android:textColor="@color/colorIcons"
        android:text="@string/start_tracking"
        android:layout_gravity="bottom"
        android:layout_marginBottom="40dp"
        android:layout_marginRight="16dp"
        android:layout_marginLeft="16dp"
```

MAPTRACKINGACTIVITY CLASS

The MapTrackingActivity class will implement the GoogleApiClient.ConnectionCallback interface. We will create an instance of the GoogleApiClient which we will use to connect to Google Play Services and interact with the Location Service API.

In the onConnection() callback method, we will create a location request and check if the our device has the right location settings before we can obtain the device current location. The GoogleApiClient object is released in the onStop() callback method.

We have also created an inner RouteBroadCastReceiver class which will receive intent from the background service to update the Map route UI.

Open the MapTrackingActivity class and add the code below to it.

```
import android.Manifest;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.pm.PackageManager;
import android.graphics.Color;
import android.location.Location;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.FragmentActivity;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import com.google.android.gms.common.api.GoogleApiClient
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.LocationRequest;
```

```
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.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
import com.inducesmile.androidlocationtracking.database.
import java.util.ArrayList;
import java.util.List;
public class MapTrackingActivity extends FragmentActivit
    private static final String TAG = MapTrackingActivit
    private GoogleApiClient mGoogleApiClient;
    private Location mLastLocation;
    private LocationRequest mLocationRequest;
    private double latitudeValue = 0.0;
    private double longitudeValue = 0.0;
    private GoogleMap mMap;
    private DatabaseQuery mQuery;
    private RouteBroadCastReceiver routeReceiver;
    private List<LocationObject> startToPresentLocations
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_map_tracking);
        if (mGoogleApiClient == null) {
            mGoogleApiClient = new GoogleApiClient.Build
                    .addConnectionCallbacks(this)
                    .addApi(LocationServices.API)
                    .build();
        }
        mQuery = new DatabaseQuery(this);
        startToPresentLocations = mQuery.getAllLocationO
        mLocationRequest = createLocationRequest();
        routeReceiver = new RouteBroadCastReceiver();
        // Obtain the SupportMapFragment and get notifie
        SupportMapFragment mapFragment = (SupportMapFrag
        mapFragment.getMapAsync(this);
    }
    @Override
    public void onMapReady(GoogleMap googleMap) {
```

```
        mapObject.moveCamera(CameraUpdateFactory.newLatL
    }
    @Override
    public void onConnected(@Nullable Bundle bundle) {
        Log.d(TAG, "Connection method has been called");
        LocationSettingsRequest.Builder builder = new Lo
        PendingResult<LocationSettingsResult> result = L
        result.setResultCallback(new ResultCallback<Loca
            @Override
            public void onResult(@NonNull LocationSettin
                final Status status = result.getStatus()
                switch (status.getStatusCode()) {
                    case LocationSettingsStatusCodes.SUC
                        if (ActivityCompat.checkSelfPerm
                                && ActivityCompat.checkS
                            mLastLocation = LocationServ
                            if (mLastLocation != null) {
                                latitudeValue = mLastLoc
                                longitudeValue = mLastLo
                                Log.d(TAG, "Latitude 4:
                                refreshMap(mMap);
                                markStartingLocationOnMa
                                startPolyline(mMap, new
                            }
                        }
                        break;
                    case LocationSettingsStatusCodes.SET
                        break;
                }
            }
        });
    }
    @Override
    public void onConnectionSuspended(int i) {
    }
    private class RouteBroadCastReceiver extends Broadca
        @Override
        public void onReceive(Context context, Intent in
            String local = intent.getExtras().getString(
            assert local != null;
            if(local.equals("LOCAL")){
                //get all data from database
```

```
                refreshMap(mMap);
                markStartingLocationOnMap(mMap, loca
                drawRouteOnMap(mMap, locationPoints)
            }
        }
    }
}
private List<LatLng> getPoints(List<LocationObject>
    List<LatLng> points = new ArrayList<LatLng>();
    for(LocationObject mLocation : mLocations){
        points.add(new LatLng(mLocation.getLatitude(
    }
    return points;
}
private void startPolyline(GoogleMap map, LatLng loc
    if(map == null){
        Log.d(TAG, "Map object is not null");
        return;
    }
    PolylineOptions options = new PolylineOptions().
    options.add(location);
    Polyline polyline = map.addPolyline(options);
    CameraPosition cameraPosition = new CameraPositi
            .target(location)
            .zoom(16)
            .build();
    map.animateCamera(CameraUpdateFactory.newCameraP
}
private void drawRouteOnMap(GoogleMap map, List<LatL
    PolylineOptions options = new PolylineOptions().
    options.addAll(positions);
    Polyline polyline = map.addPolyline(options);
    CameraPosition cameraPosition = new CameraPositi
            .target(new LatLng(positions.get(0).lati
            .zoom(17)
            .bearing(90)
            .tilt(40)
            .build();
    map.animateCamera(CameraUpdateFactory.newCameraP
}
private void refreshMap(GoogleMap mapInstance){
    mapInstance.clear();
```

```
        mLocationRequest.setFastestInterval(3000);
        mLocationRequest.setPriority(LocationRequest.PRI
        return mLocationRequest;
    }
    @Override
    protected void onResume() {
        super.onResume();
        if(routeReceiver == null){
            routeReceiver = new RouteBroadCastReceiver()
        }
        IntentFilter filter = new IntentFilter(RouteServ
        LocalBroadcastManager.getInstance(this).register
    }
    @Override
    protected void onPause() {
        super.onPause();
        LocalBroadcastManager.getInstance(this).unregist
    }
    @Override
    protected void onStart() {
        mGoogleApiClient.connect();
        super.onStart();
    }
    @Override
    protected void onStop() {
        mGoogleApiClient.disconnect();
        super.onStop();
    }
}
```

ROUTESERVICE CLASS

The RouteService class is an android service class that runs in the background. The service starts when a user toggle the start tracking button and you can as well stop it with the button is the button uses a flag to tracking the state of the service class.

The RouteService class implements the GoogleApiClient

The LocationListener overrides the onLocationChange() method which pass the current location as parameter.

The location is stored in SQLite database and the Service class will notify the Local Broadcast Receiver class of the location update.

Open the RouteService.java file and add the code below.

```
import android.Manifest;
import android.app.Service;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.location.Location;
import android.os.Bundle;
import android.os.IBinder;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.LocalBroadcastManager;
import android.util.Log;
import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.api.GoogleApiClient
import com.google.android.gms.common.api.PendingResult;
import com.google.android.gms.common.api.ResultCallback;
import com.google.android.gms.common.api.Status;
import com.google.android.gms.location.LocationListener;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.location.LocationServices;
import com.google.android.gms.location.LocationSettingsR
import com.google.android.gms.location.LocationSettingsR
import com.google.android.gms.location.LocationSettingsS
import com.inducesmile.androidlocationtracking.database.
import com.inducesmile.androidlocationtracking.helpers.C
public class RouteService extends Service implements Goo
    private static final String TAG = RouteService.class
    public static final String ACTION = "com.inducesmile
    private GoogleApiClient mGoogleApiClient;
    private Location mLastLocation;
    private LocationRequest mLocationRequest;
    private double latitudeValue = 0.0;
    private double longitudeValue = 0.0;
```

Accept

```
@Override
public void onCreate() {
    super.onCreate();
    customSharedPreference = new CustomSharedPrefere
    if(isRouteTrackingOn()){
        startTimeInMilliSeconds = System.currentTime
        Log.d(TAG, "Current time " + startTimeInMill
        Log.d(TAG, "Service is running");
    }
    query = new DatabaseQuery(getApplicationContext(
    mLocationRequest = createLocationRequest();
    if (mGoogleApiClient == null) {
        mGoogleApiClient = new GoogleApiClient.Build
                .addConnectionCallbacks(this)
                .addOnConnectionFailedListener(this)
                .addApi(LocationServices.API)
                .build();
        mGoogleApiClient.connect();
    }
}
@Override
public int onStartCommand(Intent intent, int flags,
    isServiceRunning = true;
    return Service.START_STICKY;
}
@Nullable
@Override
public IBinder onBind(Intent intent) {
    return null;
}
@Override
public void onConnected(@Nullable Bundle bundle) {
    Log.d(TAG, "Connection method has been called");
    LocationSettingsRequest.Builder builder = new Lo
    PendingResult<LocationSettingsResult> result = L
    result.setResultCallback(new ResultCallback<Loca
        @Override
        public void onResult(@NonNull LocationSettin
            final Status status = result.getStatus()
            switch (status.getStatusCode()) {
                case LocationSettingsStatusCodes.SUC
                    if (ActivityCompat.checkSelfPerm
```

Accept

```
                            longitudeValue = mLastLo
                            Log.d(TAG, "Latitude 1:
                            LocationServices.FusedLo
                        }
                    }
                    break;
                case LocationSettingsStatusCodes.SET
                    break;
            }
        }
    });
}
@Override
public void onConnectionSuspended(int i) {
}
@Override
public void onConnectionFailed(@NonNull ConnectionRe
}
protected LocationRequest createLocationRequest() {
    LocationRequest mLocationRequest = new LocationR
    mLocationRequest.setInterval(5000);
    mLocationRequest.setFastestInterval(3000);
    mLocationRequest.setPriority(LocationRequest.PRI
    return mLocationRequest;
}
@Override
public void onLocationChanged(Location location) {
    Log.d(TAG, "Latitude " + location.getLatitude()
    Log.d(TAG, "SERVICE RUNNING " + isServiceRunning
    if(isRouteTrackingOn() && startTimeInMilliSecond
        startTimeInMilliSeconds = System.currentTime
    }
    if(isRouteTrackingOn() && startTimeInMilliSecond
        latitudeValue = location.getLatitude();
        longitudeValue = location.getLongitude();
        Log.d(TAG, "Latitude " + latitudeValue + " L
        // insert values to local sqlite database
        query.addNewLocationObject(System.currentTim
        // send local broadcast receiver to applicat
        Intent localBroadcastIntent = new Intent(ACT
        localBroadcastIntent.putExtra("RESULT_CODE",
        LocalBroadcastManager.getInstance(getApplica
```

Accept

```
                Log.d(TAG, "SERVICE HAS BEEN STOPPED");
                this.stopSelf();
            }
        }
        if(!isRouteTrackingOn()){
            Log.d(TAG, "SERVICE HAS BEEN STOPPED 1");
            isServiceRunning = false;
            Log.d(TAG, "SERVICE STOPPED " + isServiceRun
            Intent dialogIntent = new Intent(this, Recor
            dialogIntent.addFlags(Intent.FLAG_ACTIVITY_N
            this.startActivity(dialogIntent);
            this.stopSelf();
        }
    }
    private boolean isRouteTrackingOn(){
        Log.d(TAG, "SERVICE STATE " + customSharedPrefer
        return customSharedPreference.getServiceState();
    }
    @Override
    public void onDestroy() {
        mGoogleApiClient.disconnect();
        super.onDestroy();
    }
}
```

This brings us to the end of this tutorial. I hope that you have learn something. Run your app and take a work around you will see how your location is being drawn on the map.

You can download the code for this tutorial below. If you are having hard time downloading the tutorial, kindly contact me.

Remember to subscribe with your email address to be among the first to receive my new android blog post once it is published.

OTHER INTERESTING POSTS:

Android Studio – Add .Jar File as Android Library

Download Remote Audio File in Android Application

Classic Chemistry Android App – One week after release

How to integrate Android Admob Interstitials Recommendation (App load and exit)

Android Animation Example Tutorial

Android Retrofit 2 with JSON API Example

Tags: Android Real-time User Location Tracking Using Google Map API v2, location tracking in android, location tracking with Google Map

ABOUT THE AUTHOR

Inducesmile

I learn and write about Android, iOS, Javascript, Php, Node.js,

25 COMMENTS

roger | October 27, 2016 Log in to Reply

Hi Henry,

This is very useful tutorial. I just wonder if you can also post class source code of LocationObject as well as the database source code so that to make it as a complete example that I can learn and try.

Thanks a lot
Ling

Henry Author | October 27, 2016
Log in to Reply

Thanks, I will do that when time permits me. Kindly keep checking back

amad arshad | April 10, 2017
Log in to Reply

Hi, Do you have any further update on this request?

jagriti | May 5, 2017
Log in to Reply

hii henry plz provide the source code of database and location object class

ratnesh_k | July 21, 2017

Accept

Can u share the source code?

Henry Author | July 28, 2017 Log in to Reply

I will add the source code soon. Kindly check back. Thanks

Aurelius | August 14, 2017

Hi Henry,
The tutorial is awesome but the download link is missing or something and I cant find it. Could you please add it or the database source code and the LocationObject class or something?

Henry Author | August 20, 2017

Sorry about that. When I have time I will upload the source code. Thanks

anurag31 | August 12, 2017 Log in to Reply

hi, please provide the provide the code, thanx

Accept

how would i save the data travelled in sqlite and use it when i click button reroute ???

Henry Author | October 27, 2016

Log in to Reply

I will soon provide more information with regards to saving real-time location data in SQLite database.

roger | October 29, 2016

Log in to Reply

Thanks a lot! Look forward to see the SQLite database for location data

roger | October 29, 2016 Log in to Reply

By the way , what does RecordResultActivity.class do in this app?

Is it a new activity class that record the data? How is it different fromMapTrackingActivity?

amad arshad | April 10, 2017 Log in to Reply

Hi Henry, Can not see any code to download in your post. Could you please provide download link for your code?

Accept

islahudin | April 18, 2017 Log in to Reply

where the download link?

venu | June 16, 2017 Log in to Reply

where can i get download link.

Poongodi | June 19, 2017 Log in to Reply

Can u share the source code?

ASWATHI | July 12, 2017 Log in to Reply

wat shud i perform on button click???..plz share the download link…

Ahdia | July 15, 2017 Log in to Reply

Hi Henry,
This is very useful tutorial. Can you share the source code ?

Vinit Thada | October 6, 2017 Log in to Reply

Hi Henry,
Please provide whole source code.

Accept

specifically the shared preferences.

monstor | April 23, 2018 Log in to Reply

source code please

Rahul | May 22, 2018 Log in to Reply

Give me download link of this project

Ramesh A | July 13, 2018 Log in to Reply

Hi Henry,
Please provide whole source code.

ADD A COMMENT

You must be logged in to post a comment.