Lab11

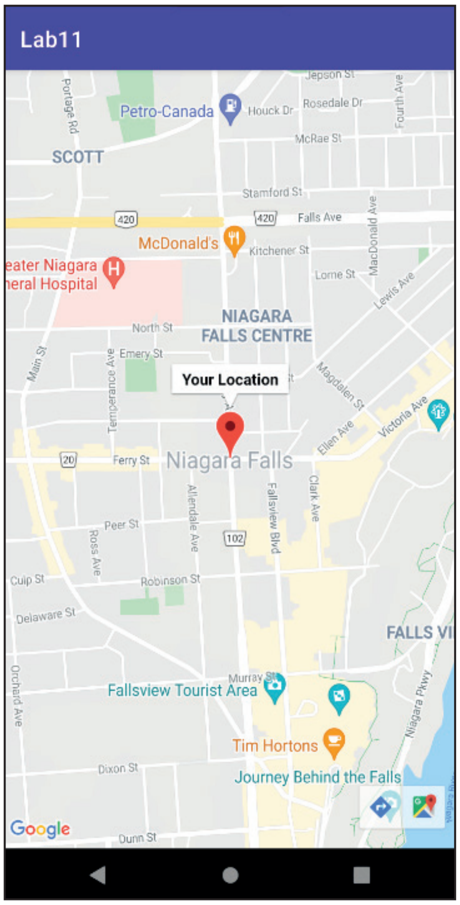
Location-Aware Apps Using the GPS and Google Maps

Objectives:

- **Creating an App Interface and Configuring Google Services Prerequisites.**
- **Creating a Google Map Fragment.**
- **Getting a Google API key.**
- **Configuring Your App to Use Your Google API Key and User App's Permission.**
- **Adding Google Map and Capturing Users' Location.**

Lab Scenario:

In this lab you are going to create an Android application that displays your app user current location on Google map.



Creating an App Interface and Configuring Google Services Prerequisites

1- Open Android Studio, and then click **File → New → New Project.**

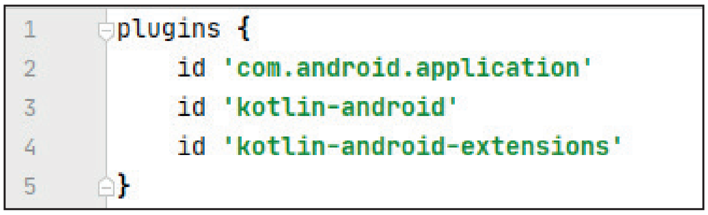
2- Select **Empty Activity**, and click **Next.**

3- Type: **Lab11** for the application name, then click **Finish**.

4- Before you start typing the Kotlin code in your app, check the **build.gardle (Module: Lab11. app)** file and be sure that it has the Kotlin plugin. If not, add the following code:

id 'kotlin-android-extensions'

The configuration should be as follows:



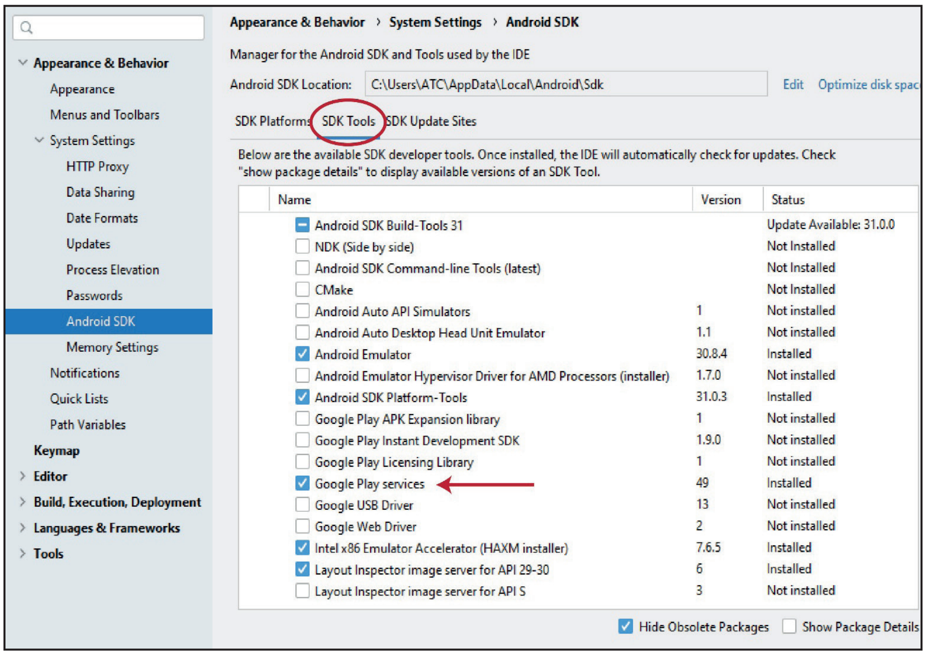
```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
}
```

Add the following Google maps dependencies:

```
implementation 'com.google.android.gms:play-services-location:18.0.0'
implementation 'com.google.android.gms:play-services-maps:17.0.1'
```

Then, click the **Sync Now**.

5- Be sure the **Google Play service** is installed on your Android Studio. To do this, click the **SDK Manager** (**Tools → SDK Manager**), and as illustrated in the figure below, click **SDK Tools**. If this service is not installed, select **Google Play services**, then click **Apply** to install it.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\ATC\AppData\Local\Android\Sdk   Edit   Optimize disk spac

Below are the available SDK developer tools. Once installed, the IDE will automatically check for updates. Check "show package details" to display available versions of an SDK Tool.

Name	Version	Status
▣ Android SDK Build-Tools 31		Update Available: 31.0.0
☐ NDK (Side by side)		Not Installed
☐ Android SDK Command-line Tools (latest)		Not Installed
☐ CMake		Not Installed
☐ Android Auto API Simulators	1	Not installed
☐ Android Auto Desktop Head Unit Emulator	1.1	Not installed
☑ Android Emulator	30.8.4	Installed
☐ Android Emulator Hypervisor Driver for AMD Processors (installer)	1.7.0	Not installed
☑ Android SDK Platform-Tools	31.0.3	Installed
☐ Google Play APK Expansion library	1	Not installed
☐ Google Play Instant Development SDK	1.9.0	Not installed
☐ Google Play Licensing Library	1	Not installed
☑ Google Play services	49	Installed
☐ Google USB Driver	13	Not installed
☐ Google Web Driver	2	Not installed
☑ Intel x86 Emulator Accelerator (HAXM installer)	7.6.5	Installed
☑ Layout Inspector image server for API 29-30	6	Installed
☐ Layout Inspector image server for API S	3	Not installed

☑ Hide Obsolete Packages ☐ Show Package Details

Creating a Google Map Fragment

6- Open the **activity_main.xml** file in **Design** mode and **delete** the “Hello World!” text.

7- Open the **activity_main.xml** file in the **Code** mode and Replace: **androidx.constraintlayout.widget.ConstraintLayout** in the first line with: **androidx.fragment.app.FragmentContainerView**

8- Add the following attribute to your fragment tag in the **activity_main.xml** file:

```
android:name=”com.google.android.gms.maps.SupportMapFragment”
```

Your **activity_main.xml** file code should be as follows:

```
<?xml version=”1.0” encoding=”utf-8”?>
<androidx.fragment.app.FragmentContainerView
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:name=”com.google.android.gms.maps.SupportMapFragment”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    tools:context=”.MainActivity”/>
```

9- Add a fragment **id** attribute with value: **myMap** to your **activity_main.xml** file. The full code of the **activity_main.xml** file will be as follows:

```
<?xml version=”1.0” encoding=”utf-8”?>
<androidx.fragment.app.FragmentContainerView
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:id=”@+id/myMap”
    android:name=”com.google.android.gms.maps.SupportMapFragment”
    android:layout_width=”match_parent”
    android:layout_height=”match_parent”
    tools:context=”.MainActivity”/>
```

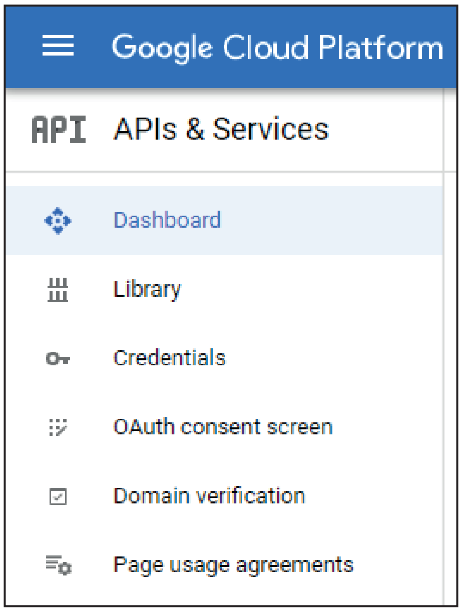
Getting a Google API key

10- Go to : **https://console.developers.google.com**

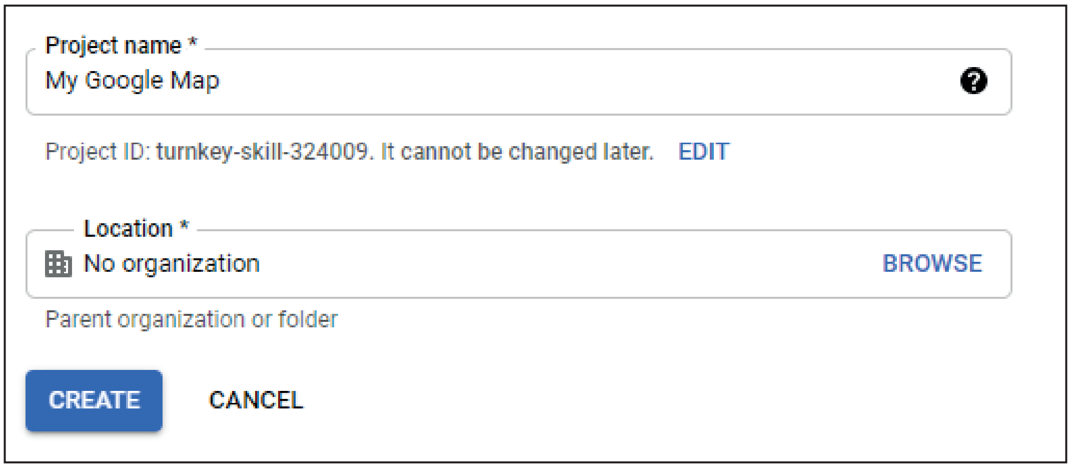
11- Login using your Gmail user name and password. If you don’t have an account, create one.

If you get any dialog box that is related to the Google terms and services, select your country, click I Agree, then click **AGREE AND CONTINUE**.

You should get the panel below at the left side of Google cloud web site:



12- At the top of this web site, click **CREATE PROJECT**, then you should get the following dialog box. Type: **My Google Map** (or any project name you want) for the project name and click **CREATE**.



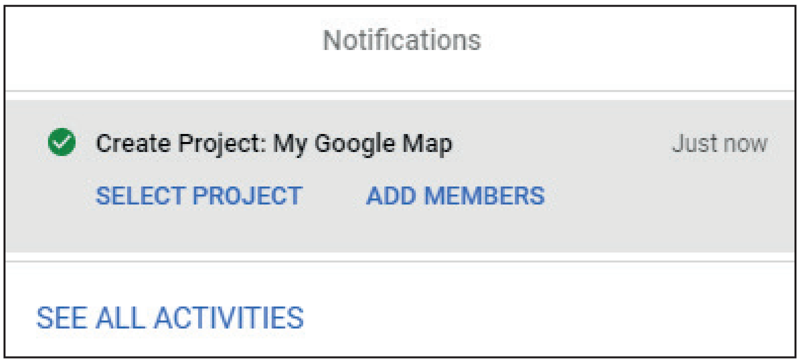
Project name *
My Google Map

Project ID: turnkey-skill-324009. It cannot be changed later. EDIT

Location *
No organization BROWSE

Parent organization or folder

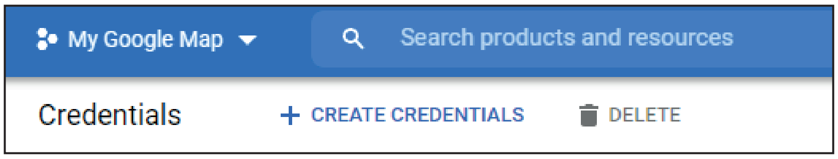
You should get the following notification:



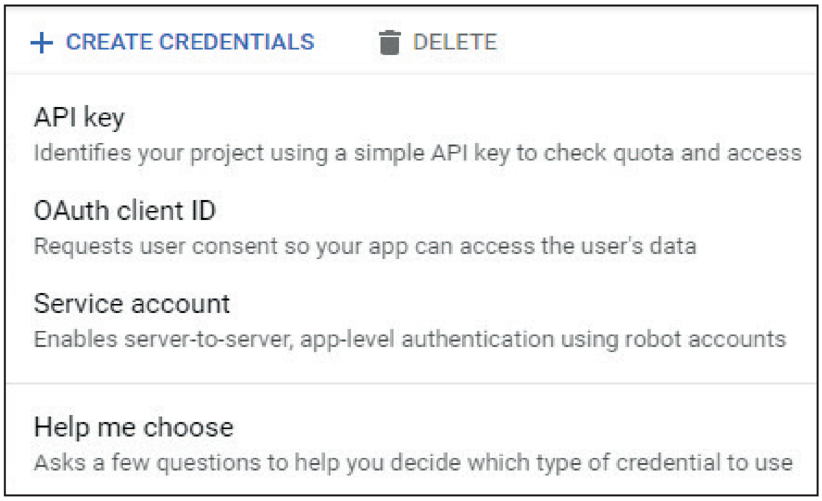
Notifications

Create Project: My Google Map Just now

13- Click **Credentials** from the left panel. Then, as illustrated in the figure below and at the top of your web browser, click **+ CREATE CREDENTIALS.**



You should get a menu as illustrated in the figure below. Select **API Key.**



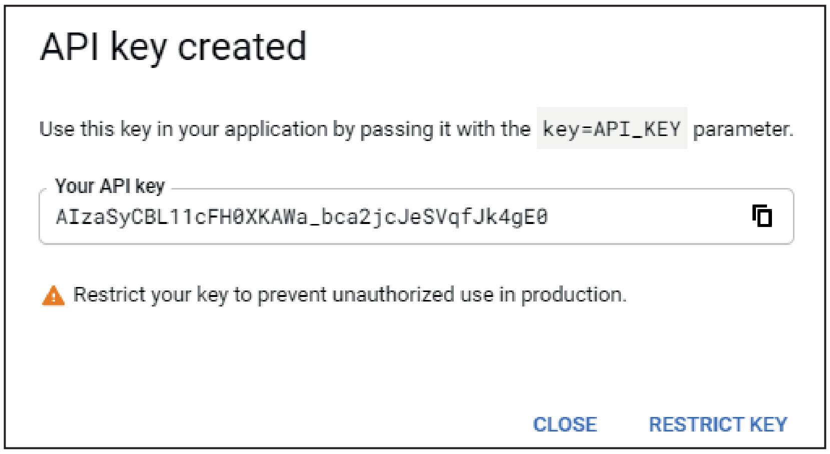
API key
Identifies your project using a simple API key to check quota and access

OAuth client ID
Requests user consent so your app can access the user's data

Service account
Enables server-to-server, app-level authentication using robot accounts

Help me choose
Asks a few questions to help you decide which type of credential to use

Then, you will get your API Key as illustrated in the following figure:



API key created

Use this key in your application by passing it with the `key=API_KEY` parameter.

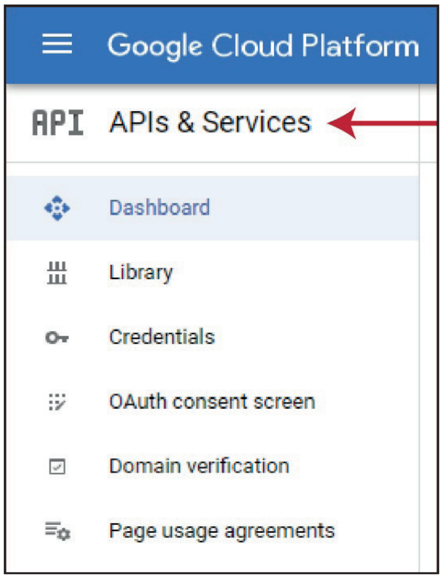
Your API key
AIzaSyCBL11cFH0XKAWa_bca2jcJeSVqfJk4gE0

Restrict your key to prevent unauthorized use in production.

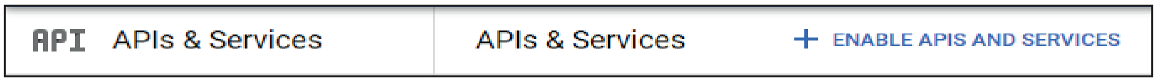
Copy your API key, and paste it in a separate Notepad file to use it later in your Android app configuration. Click **Close**.

Important note: The API key above is provided as an example only. You cannot use exactly the same key. Instead, use the one that you have generated in your Google API console.

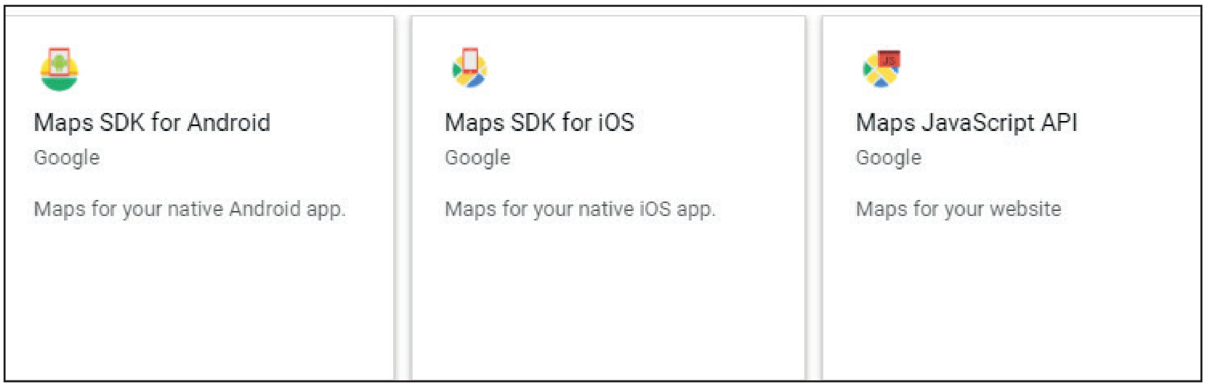
14- Now, to enable **Maps SDK** for your Android app, as illustrated in the figure below, click: **Google Cloud Platform** → **APIs & Services**.



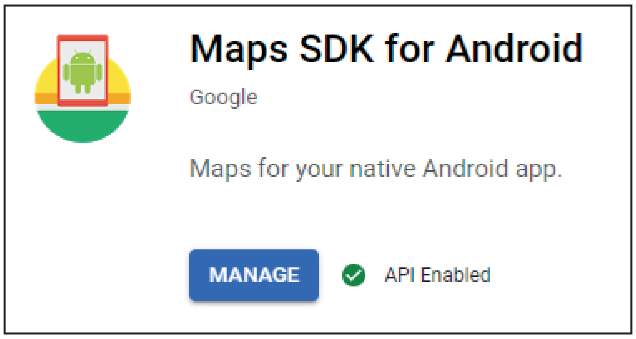
Then, at the top of the web site and as illustrated in the figure below, click **+ ENABLE APIS AND SERVICES.**



Then, you should get the figure below. Click **Maps SDK for Android**.



Then, click **Enable**. You should have the following status of your Maps configuration at Google cloud:



This is all the required configuration to make your Android app and to have an access on Google Maps.

Now, back to your Android Studio to complete the other configuration steps and to display and use Google map on your Android app.

Configuring Your App to Use Your Google API Key and User App's Permission

15- To add your **API key** to your app, go back to Android Studio, open the **AndroidManifest.xml** and add the meta-data tag below before the activity tag:

```
<meta-data android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCBL11cFH0XKAWa_bca2jcJeSVqfJk4gE0"/>
```

Then, add the following permission tags before the **<application>** tag to let your app access the app user's precise location on the Google map:

```
<uses-permission android:name="android.permission.ACCESS_FINE_
LOCATION"/>

<uses-permission android:name="android.permission.ACCESS_COARSE_
LOCATION"/>
```

The full code of the AndroidMainfest.xml file will be as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="com.androidatc.lab11">

    <uses-permission android:name="android.permission.ACCESS_FINE_
LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_
COARSE_LOCATION"/>

    <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/Theme.Lab11">
        <meta-data android:name="com.google.android.geo.API_KEY"
            android:value="AIzaSyCBL11cFH0XKAWa_bca2jcJeSVqfJk4gE0"/>

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.
LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

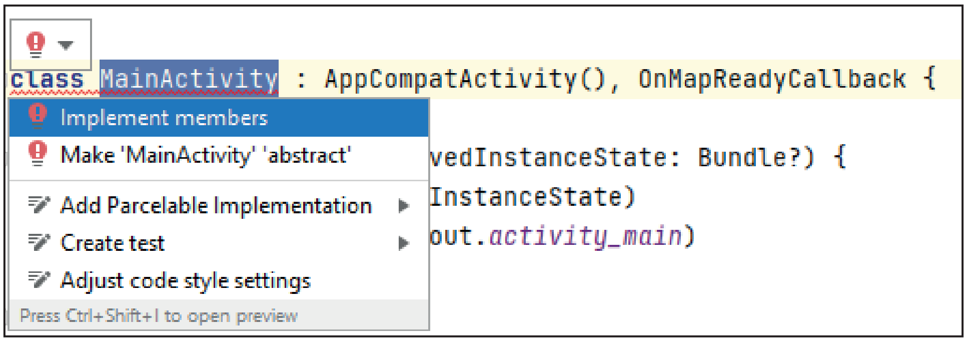
Adding Google Map and Capturing Users' Location

16- Now, you are going to write the code which will show the Google map inside your fragment (**myMap**). Open the **MainActivity** file, then add the following code inside `onCreate()` method:

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

}
```

Double click the `OnMapReadyCallback`, click the red pop-up lamp, and select **Import.**

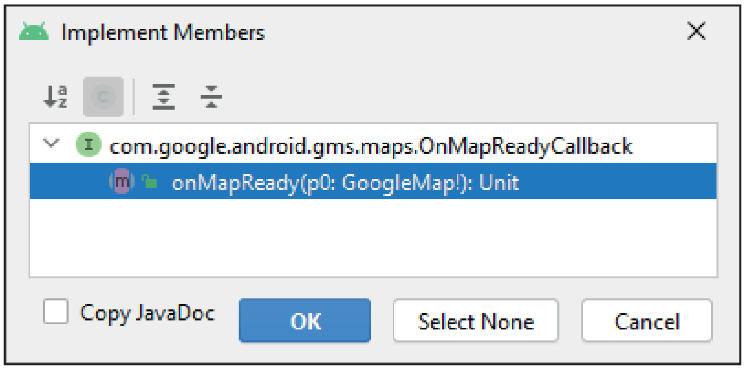
17- Double click **`MainActivity`**, click the red pop-up lamp, and as illustrated in the figure below, select: **Implement members.**



```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {
vedInstanceState: Bundle?) {
InstanceState)
out.activity_main)
```

Implement members
Make 'MainActivity' 'abstract'
Add Parcelable Implementation
Create test
Adjust code style settings
Press Ctrl+Shift+I to open preview

Then as illustrated in the figure below, select **onMapReady**, then click **OK**.



Implement Members
com.google.android.gms.maps.OnMapReadyCallback
onMapReady(p0: GoogleMap!): Unit
Copy JavaDoc
OK
Select None
Cancel

18- Declare the variables: `userLocation`, `fusedLocationClient`, and `REQUEST_CODE` as follows:

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {

 var currentLocation: Location? = null
 var fusedLocationProviderClient: FusedLocationProviderClient? =
null
 private val REQUEST_CODE = 101


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }

    override fun onMapReady(p0: GoogleMap) {
        TODO("Not yet implemented")
    }
}
```

Double click `Location`, click the red pop-up lamp, and select **Import**.

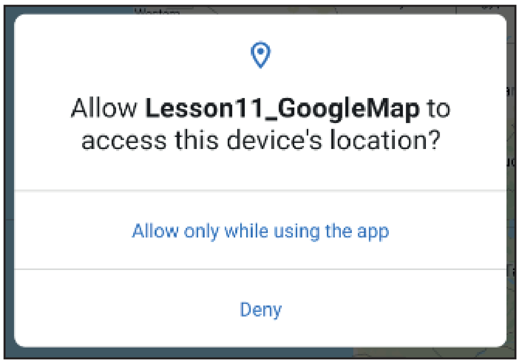
Double click `FusedLocationProviderClient`, click the red pop-up lamp, and select **Import.**

19- Configure your **onCreate** class with `LocationServices` class and `getFusedLocationProviderClient` method as follows:

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {

 var currentLocation: Location? = null
 var fusedLocationProviderClient: FusedLocationProviderClient? =
null
 private val REQUEST_CODE = 101

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

 // Initialize fused location
        fusedLocationProviderClient = LocationServices.
getFusedLocationProviderClient(this)

    }

    override fun onMapReady(p0: GoogleMap) {
        TODO("Not yet implemented")
    }
}
```

Double click **`LocationServices`**, click the red pop-up lamp, and select **Import**.

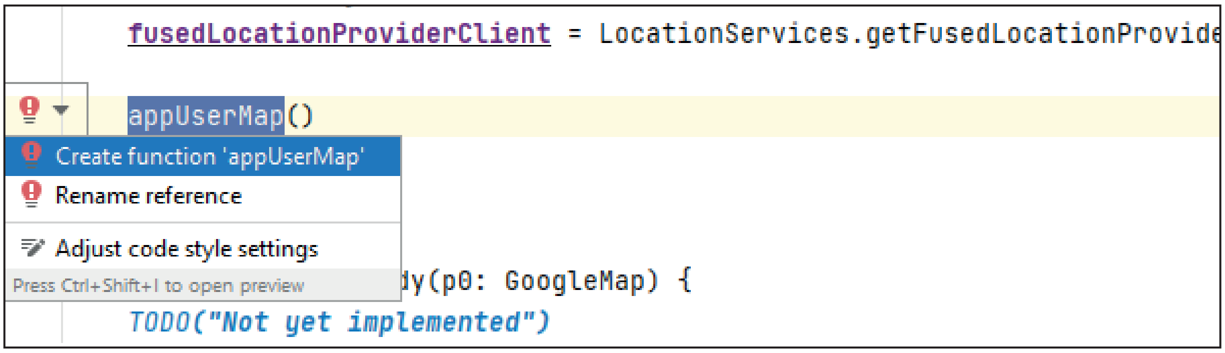
20- You should ask the app user to accept or allow the Android app to display his/her location on the Google map. This is what is called location access permission. In this step, you will add a function called **`appUserMap`** (or any method name) to ask the app user a permission to display his/her coordinates on Google map. This function or method will appear a dialog box similar to the figure below. In case the app user selects the allow option, the app user location should be displayed on the Google map.


Allow **Lesson11_GoogleMap** to access this device's location?

This function should be added as illustrated in the gray highlighted part of the following code:

```
class MainActivity : AppCompatActivity(), OnMapReadyCallback {

 var currentLocation: Location? = null
 var fusedLocationProviderClient: FusedLocationProviderClient? =
null
 private val REQUEST_CODE = 101


    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        // Initialize fused location
        fusedLocationProviderClient = LocationServices.
getFusedLocationProviderClient(this)

        appUserMap()

    }

    override fun onMapReady(p0: GoogleMap) {
        TODO("Not yet implemented")
    }
}
```

21- As illustrated in the figure below, double click the function **appUserMap()**, click the red pop-up lamp, and select: **Create function 'appUserMap'**



```
    fusedLocationProviderClient = LocationServices.getFusedLocationProvide

    appUserMap()
                                 dy(p0: GoogleMap) {
    TODO("Not yet implemented")
```

Create function 'appUserMap'
Rename reference
Adjust code style settings
Press Ctrl+Shift+I to open preview

The following function should be added to your code:

```
private fun appUserMap() {
    TODO("Not yet implemented")
}
```

22- Delete the TODO part of the **appUserMap()** function, and add to this function the following check permission code:

```
private fun appUserMap() {

    if (ActivityCompat.checkSelfPermission(this, Manifest.
permission.ACCESS_FINE_LOCATION)
        != PackageManager.PERMISSION_GRANTED && ActivityCompat.
checkSelfPermission(
            this,
            Manifest.permission.ACCESS_COARSE_LOCATION
        )
        != PackageManager.PERMISSION_GRANTED
    ) {
        ActivityCompat.requestPermissions(
            this,
            arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
            REQUEST_CODE
        )
        return
    }
    // initialize task location
    val task = fusedLocationProviderClient!!.lastLocation
    task.addOnSuccessListener { location ->
        if (location != null) {
            currentLocation = location
            val supportMapFragment =
                (supportFragmentManager.findFragmentById(R.
id.myMap) as SupportMapFragment?)
            supportMapFragment!!.getMapAsync(this)
        }
    }

}
```

Double click **ActivityCompat**, click the red pop-up lamp, and select **Import**.

Double click **Manifest**, click the red pop-up lamp, and select **Import**.

Double click **PackageManager**, click the red pop-up lamp, and select **Import**.

Double click **SupportMapFragment**, click the red pop-up lamp, and select **Import**.

23- Now in the **onMapReady** function, delete the *TODO*(**"Not yet implemented"**, and add the following configuration for this function and for the `onRequestPermissionsResult` function:

```
override fun onMapReady(mMap: GoogleMap) {
    val myUserLocation = LatLng(currentLocation!!.latitude,
currentLocation!!.longitude)

    val markerOptions = MarkerOptions().position(myUserLocation).
title("Your Location")

//zoom map
mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(myUserLocation,
15f))

// add the maker on the map
    mMap.addMarker(markerOptions)
}

override fun onRequestPermissionsResult(
    requestCode: Int,
    permissions: Array<out String>,
    grantResults: IntArray) {
    super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
    when (requestCode) {
        REQUEST_CODE -> {
            if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                appUserMap()
            }
        }
    }

}
```

Double click **`LatLng`**, click the red pop-up lamp, and select **Import**.

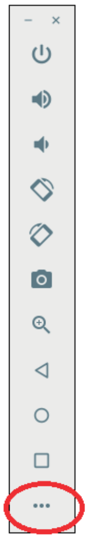
Double click **`MarkerOptions`**, click the red pop-up lamp, and select **Import**.

Double click **`CameraUpdateFactory`**, click the red pop-up lamp, and select **Import**.

24- The full code of your **MainActivity** file will be as follows:

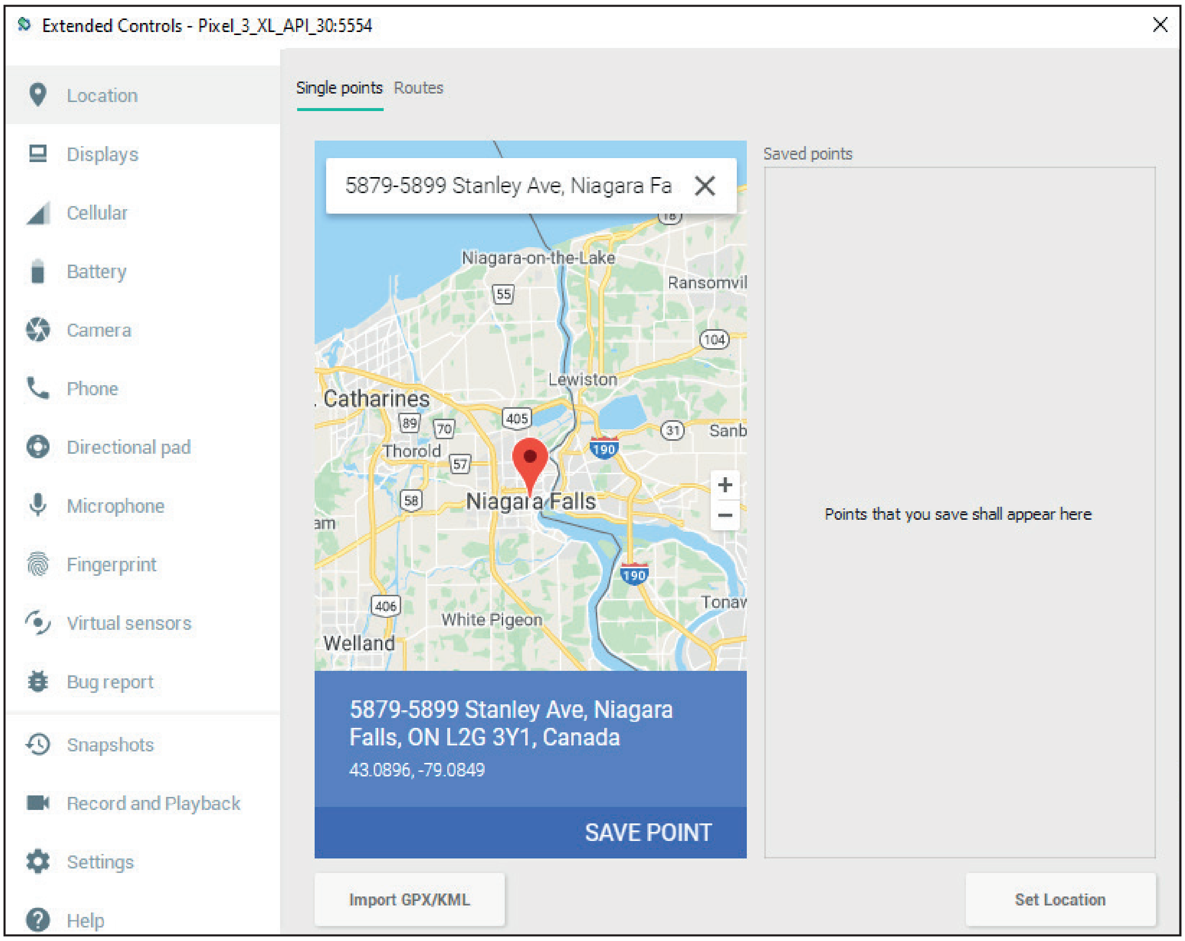
```
package com.androidatc.lab11

import android.Manifest
import android.content.pm.PackageManager
import android.location.Location
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.core.app.ActivityCompat
import com.google.android.gms.location.FusedLocationProviderClient
import com.google.android.gms.location.LocationServices
import com.google.android.gms.maps.CameraUpdateFactory
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.LatLng
import com.google.android.gms.maps.model.MarkerOptions

class MainActivity : AppCompatActivity(), OnMapReadyCallback {

  var currentLocation: Location? = null
  var fusedLocationProviderClient: FusedLocationProviderClient? =
null
    private val REQUEST_CODE = 101

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

 // Initialize fused location
        fusedLocationProviderClient = LocationServices.
getFusedLocationProviderClient(this)

        appUserMap()

    }

    private fun appUserMap() {
        if (ActivityCompat.checkSelfPermission(this, Manifest.
permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.
checkSelfPermission(
                this,Manifest.permission.ACCESS_COARSE_LOCATION)
            != PackageManager.PERMISSION_GRANTED
        ) {
            ActivityCompat.requestPermissions(
                this,
                arrayOf(Manifest.permission.ACCESS_FINE_LOCATION),
                REQUEST_CODE)
            return
        }

// initialize task location
        val task = fusedLocationProviderClient!!.lastLocation
        task.addOnSuccessListener { location ->
            if (location != null) {
                currentLocation = location
                val supportMapFragment =
                    (supportFragmentManager.findFragmentById(R.
id.myMap) as SupportMapFragment?)
                supportMapFragment!!.getMapAsync(this)
            }
        }

    }

    override fun onMapReady(mMap: GoogleMap) {

        val myUserLocation = LatLng(currentLocation!!.latitude,
currentLocation!!.longitude)

        val markerOptions = MarkerOptions().
position(myUserLocation).title("Your Location")

//zoom map
        mMap.animateCamera(CameraUpdateFactory.
newLatLngZoom(myUserLocation, 15f))

// add the maker on the map
        mMap.addMarker(markerOptions)
    }

 override fun onRequestPermissionsResult(
     requestCode: Int,
     permissions: Array<out String>,
     grantResults: IntArray) {
     super.onRequestPermissionsResult(requestCode, permissions,
grantResults)
     when (requestCode) {
         REQUEST_CODE -> {
             if (grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    appUserMap()
                }
            }
        }

    }

}
```

25- Before running your app, remember that you are working with a virtual device (emulator); therefore, configure your location (latitude and longitude) manually by clicking the **more** button of your virtual device tools bar. (See illustration below.)



As illustrated in the figure below, type: **Niagara Falls, Canada**, then click **SAVE POINT.** Click **OK.,** click **Set Location**, then close this dialog box.

Note: You may configure the location for your emulator using your address or any address you want to test your app configuration. However, if you test your app on your physical connected Android phone, it is enough to enable the Location setting in your phone, then the app will determine exactly your location on the Google map.

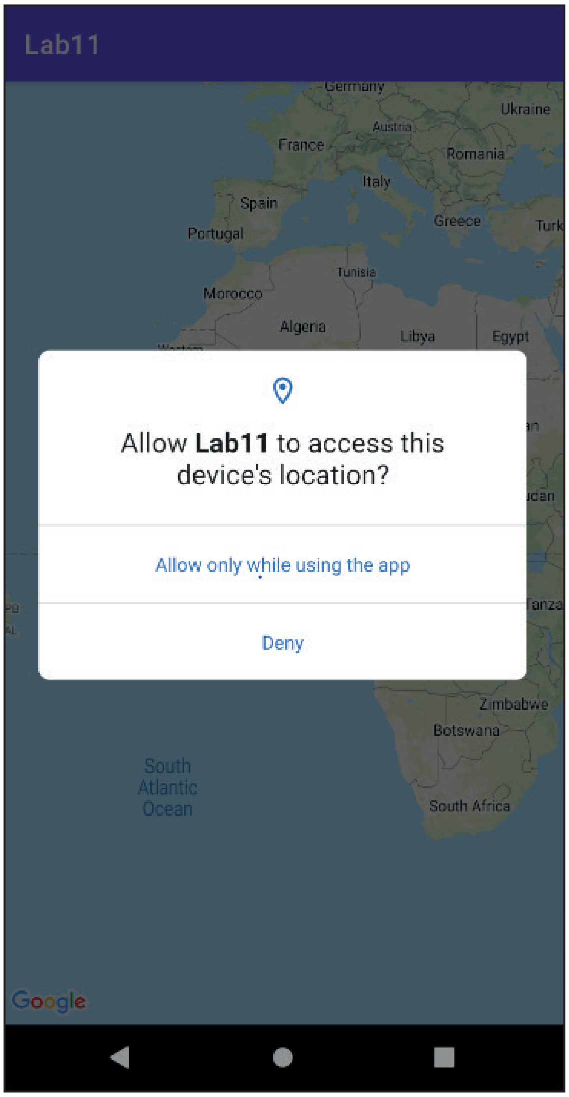

Extended Controls - Pixel_3_XL_API_30:5554

5879-5899 Stanley Ave, Niagara Fa

5879-5899 Stanley Ave, Niagara Falls, ON L2G 3Y1, Canada
43.0896, -79.0849

Points that you save shall appear here

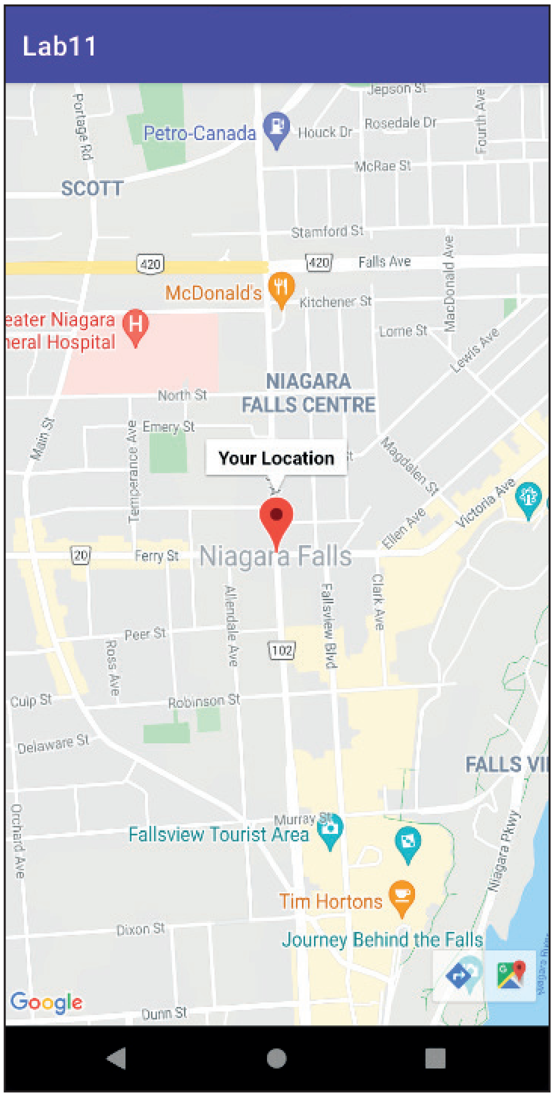
41- **Run** your app. Tap **Allow** to get the next figures which display the user's current location. Click the map maker to get the location title.


Lab11

Allow **Lab11** to access this device's location?

Allow only while using the app

Deny


Lab11

This is the end of lesson 11. Thank you.