LOCATION and MAP

Note: starting from Android 5.0 (API level 21) or higher, in addition to having the following in AndroidManifest.xml,

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

You will also need to add the following

```
<uses-feature
android:name="android.hardware.location.gps" />
```

There are two ways to get the user's location in Android:

- using Android APIs: android.location.LocationListener
- using Google Play Services API: com.google.android.gms.location.LocationListener
- Better to use Google Play Services API as it provides better accuracy for indoor environment. So we will use only Google Play Services for this set of notes.

Android's API (we won't use this)

uses 3 different providers:

- LocationManager.GPS_PROVIDER

- LocationManager.NETWORK_PROVIDER
  – cell tower and wifi access points

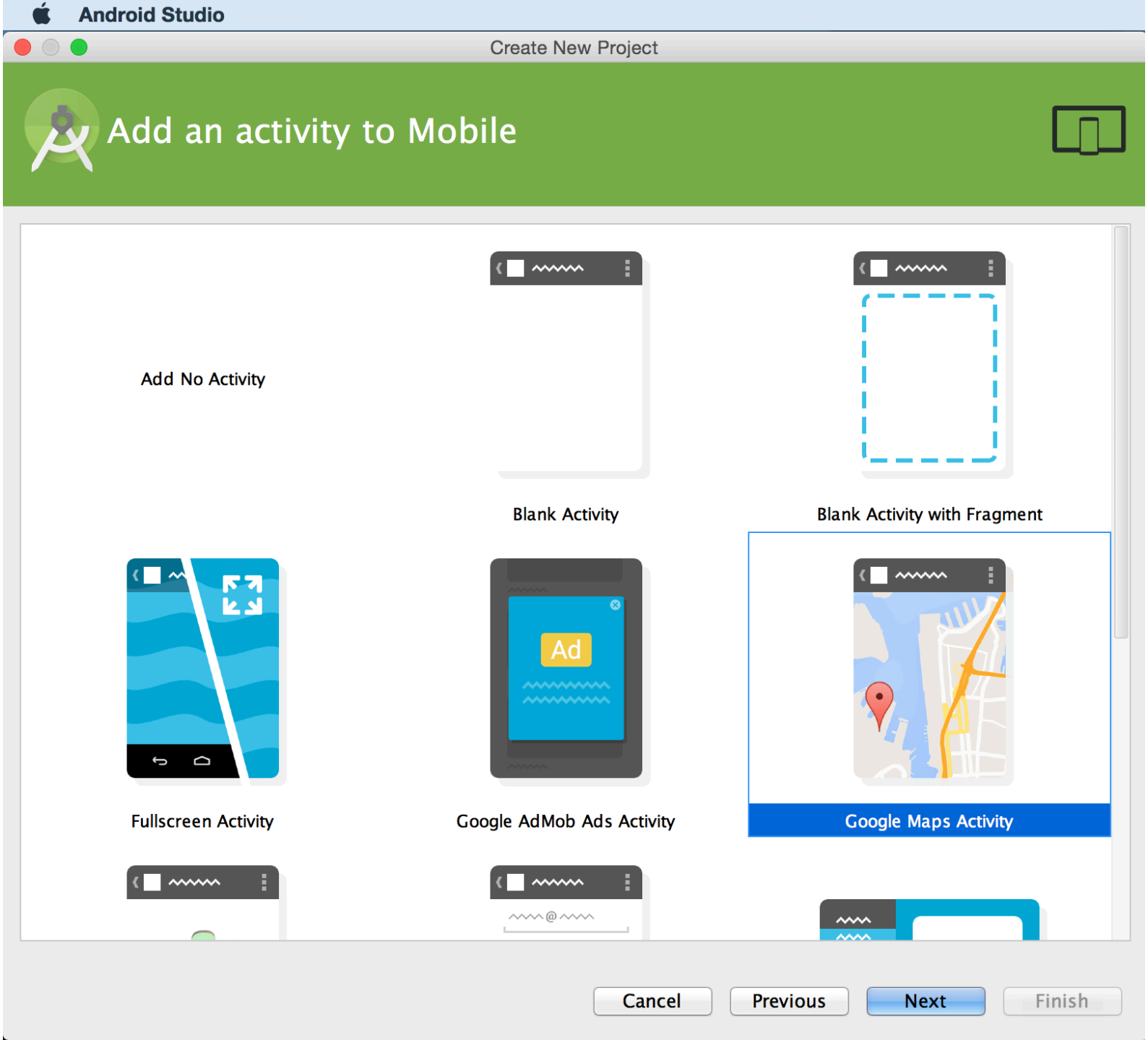
- LocationManager.PASSIVE_PROVIDER
  – passively receive location updates without actually requesting the locations yourself (no constant update, save battery)

Essentially, you need to get an object of LocationManager, implement the LocationListener, and call requestLocationUpdates on LocationManager.

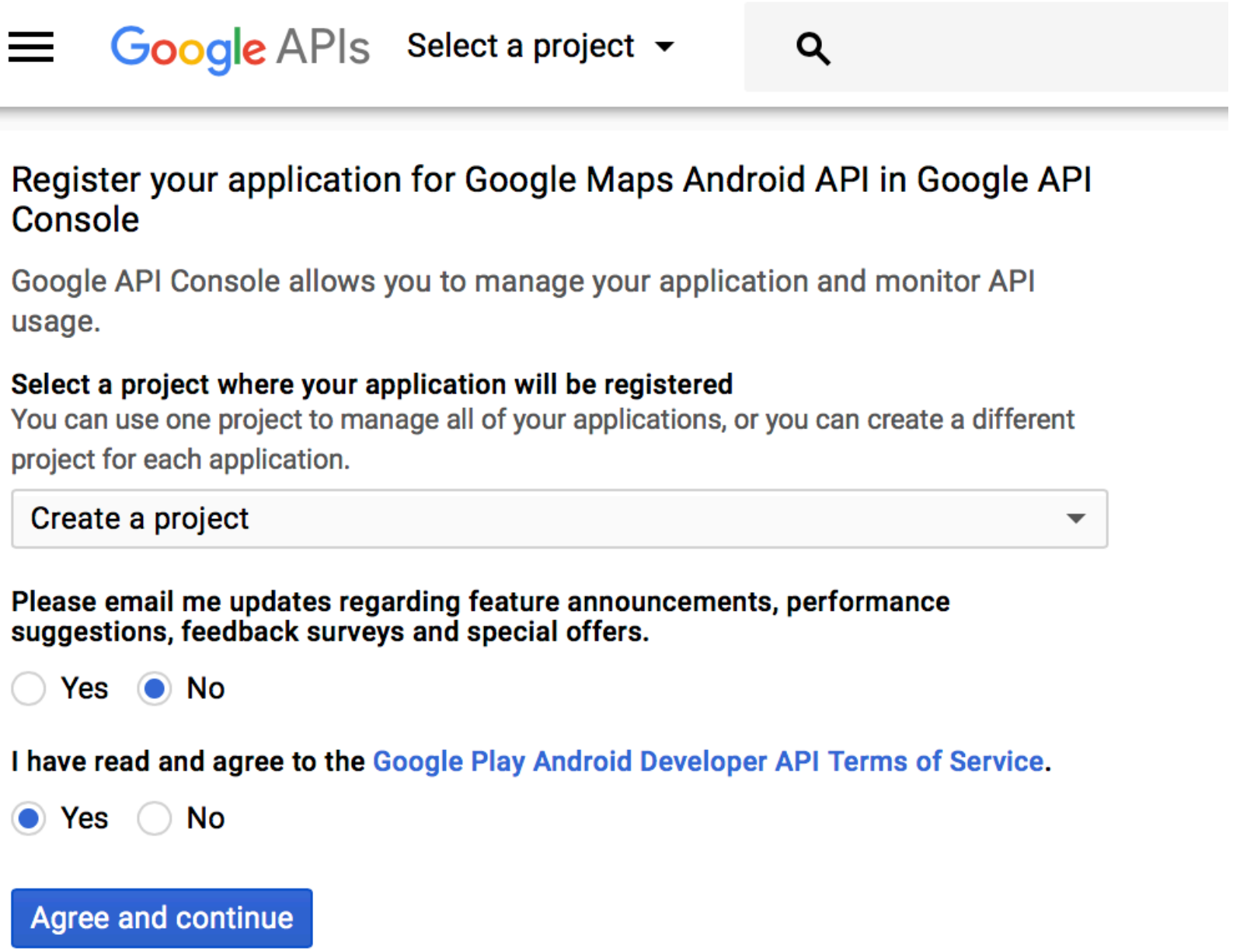
Google's Location Services API

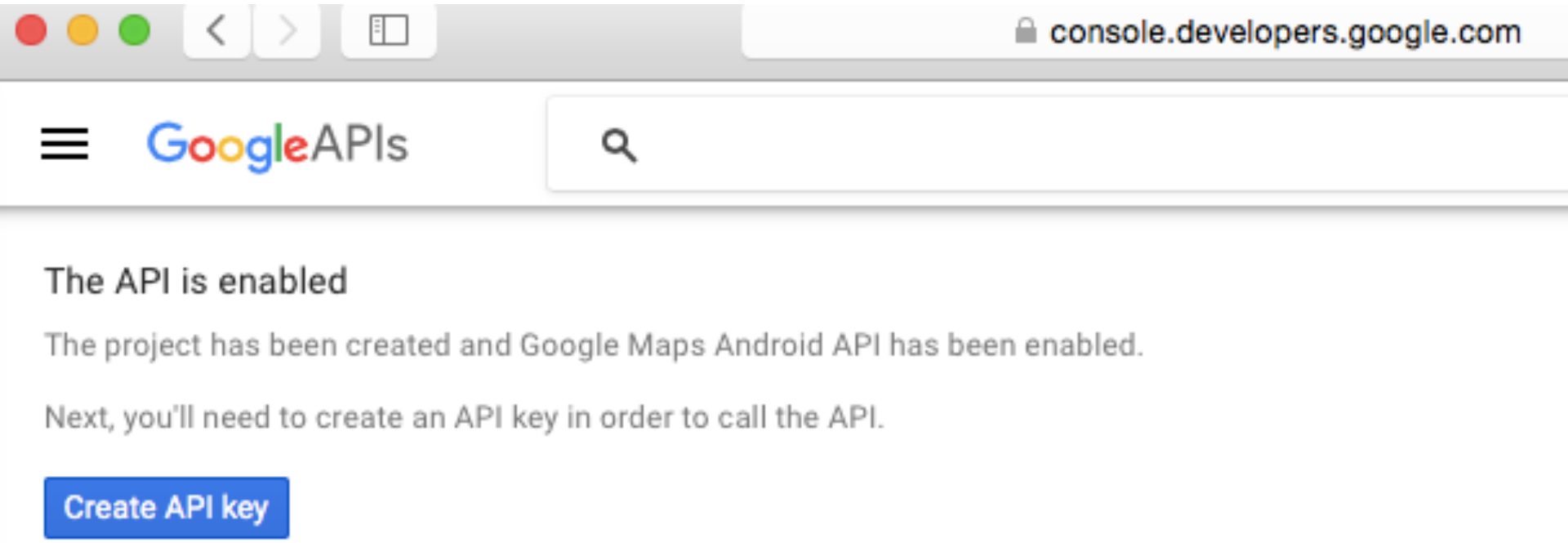
Google's Location Services API is part of Google Play Services. They are built on top of Android's API.

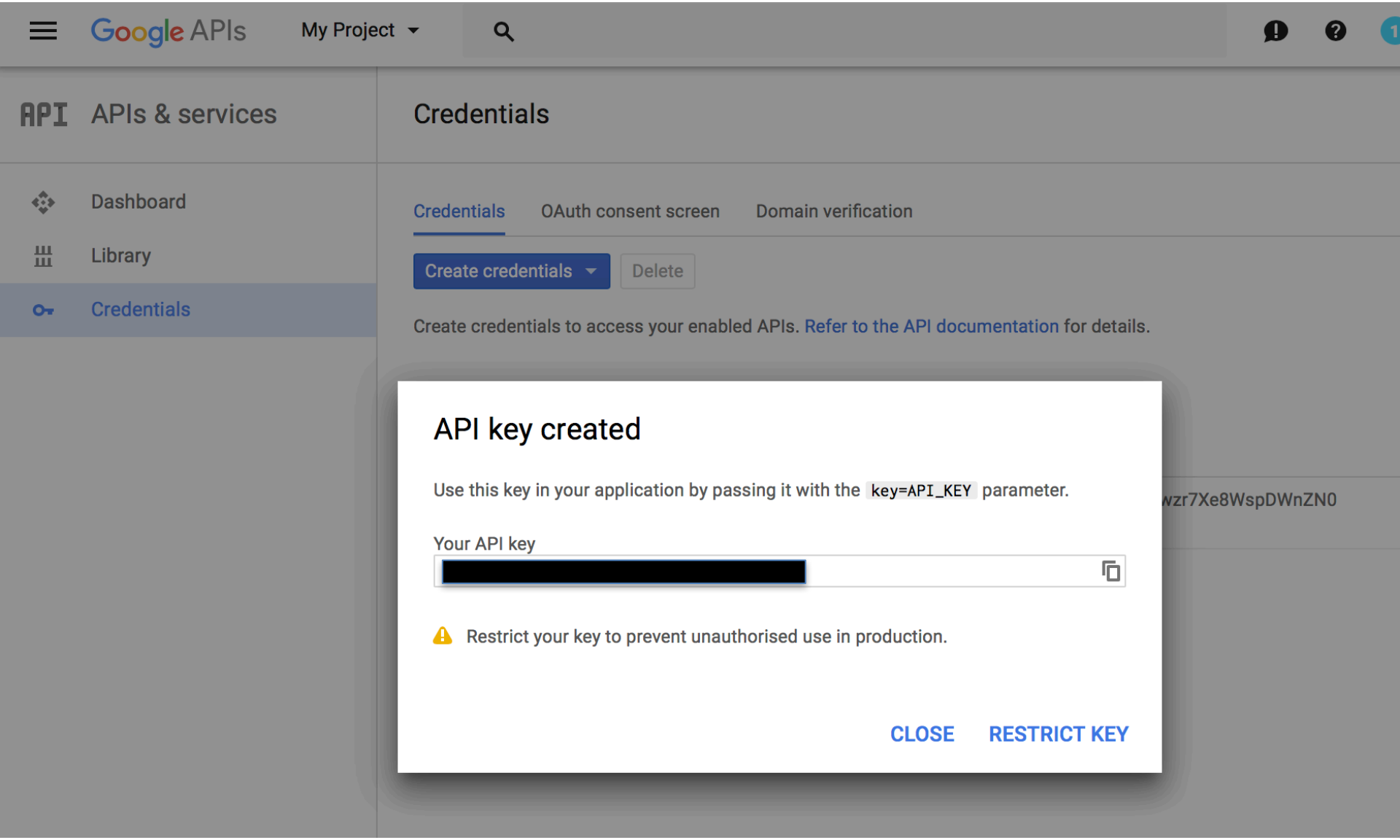
Google's Location Services API provides a "Fused Location Provider" that automatically chooses what provider to use based on accuracy, battery usage, etc. It also provides advanced features such as geofencing.

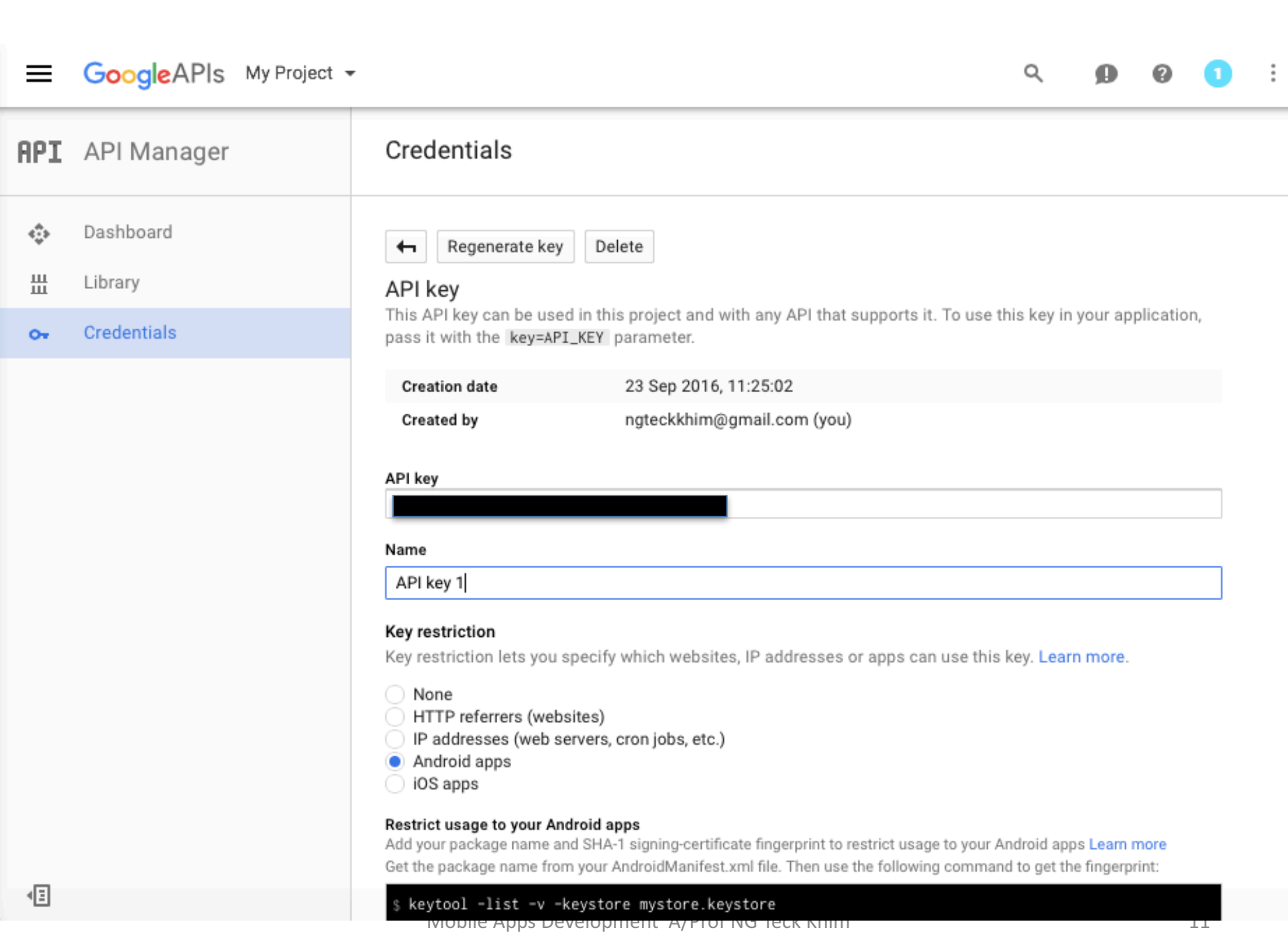

Android Studio
Create New Project
Add an activity to Mobile
Add No Activity
Blank Activity
Blank Activity with Fragment
Fullscreen Activity
Ad
Google AdMob Ads Activity
Google Maps Activity
Cancel
Previous
Next
Finish

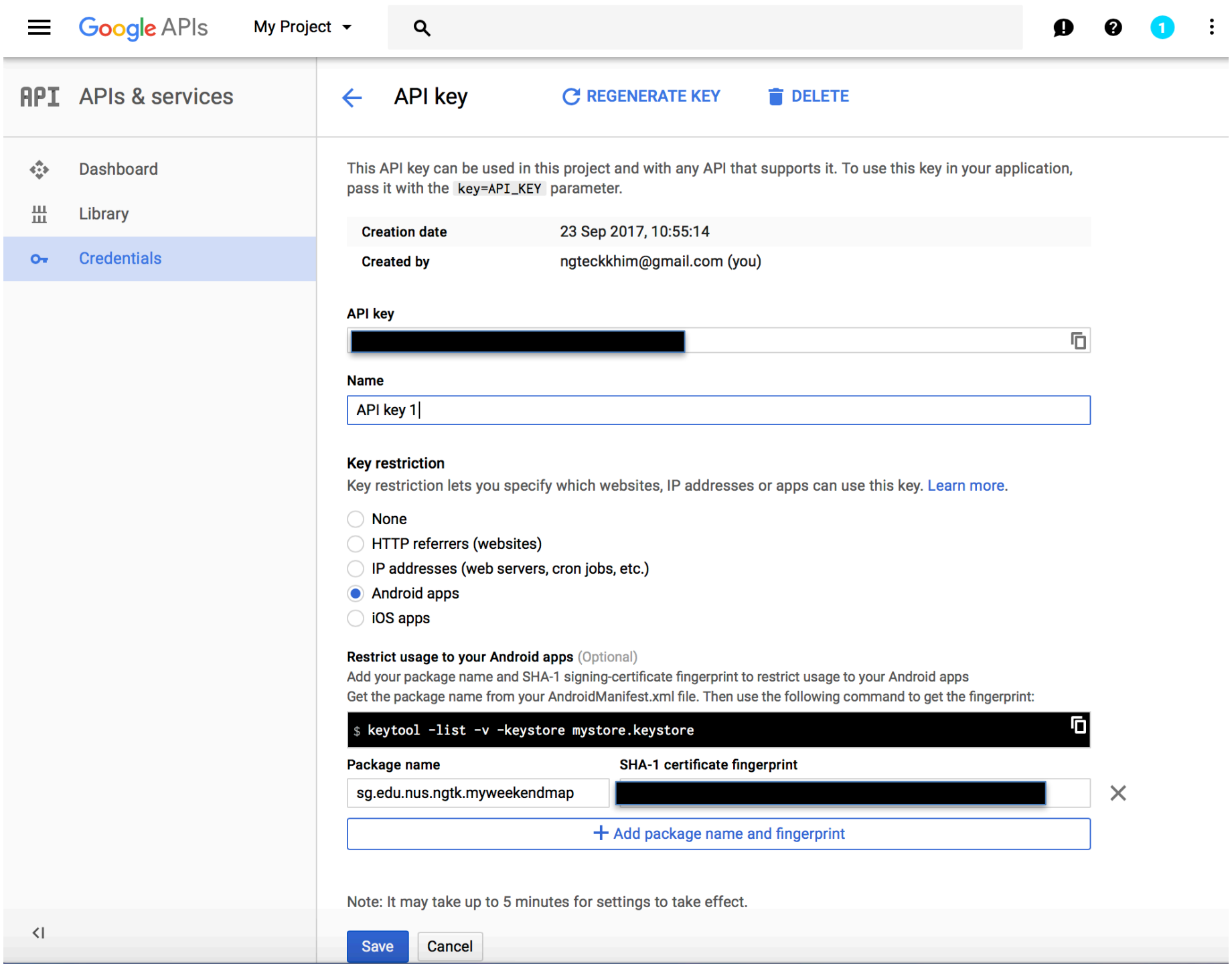
```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=8E:14:BC:04:C4

    You can also add your credentials to an existing key, using this line:
    8E:14:BC:04:C4:A4:30:FD:4D:28:00:FD:2D:A8:C8:66:81:78:57:A4;com.example.ngtk.mylocationmap

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
</resources>
```

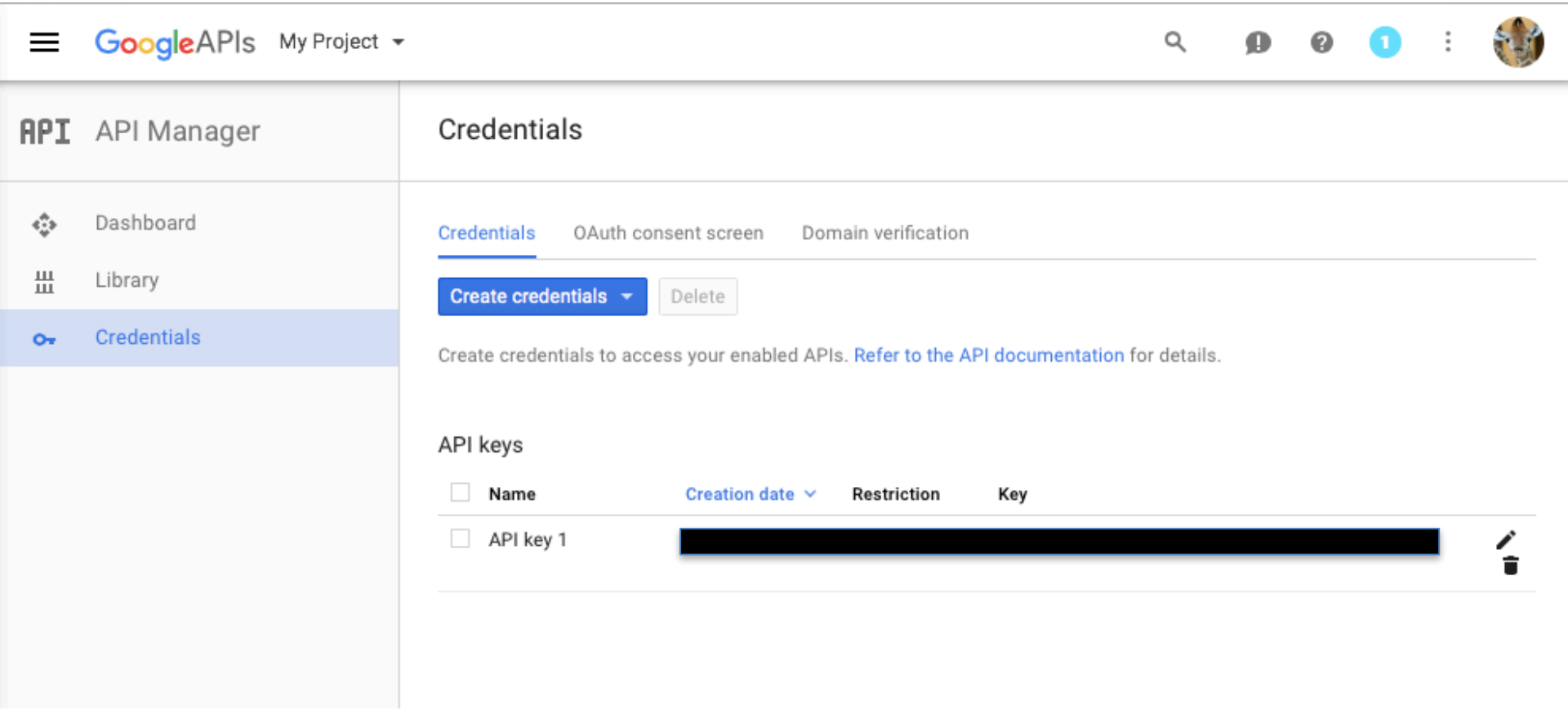

Google APIs
Select a project
Register your application for Google Maps Android API in Google API Console
Google API Console allows you to manage your application and monitor API usage.
Select a project where your application will be registered
You can use one project to manage all of your applications, or you can create a different project for each application.
Create a project
Please email me updates regarding feature announcements, performance suggestions, feedback surveys and special offers.
Yes
No
I have read and agree to the Google Play Android Developer API Terms of Service.
Yes
No
Agree and continue

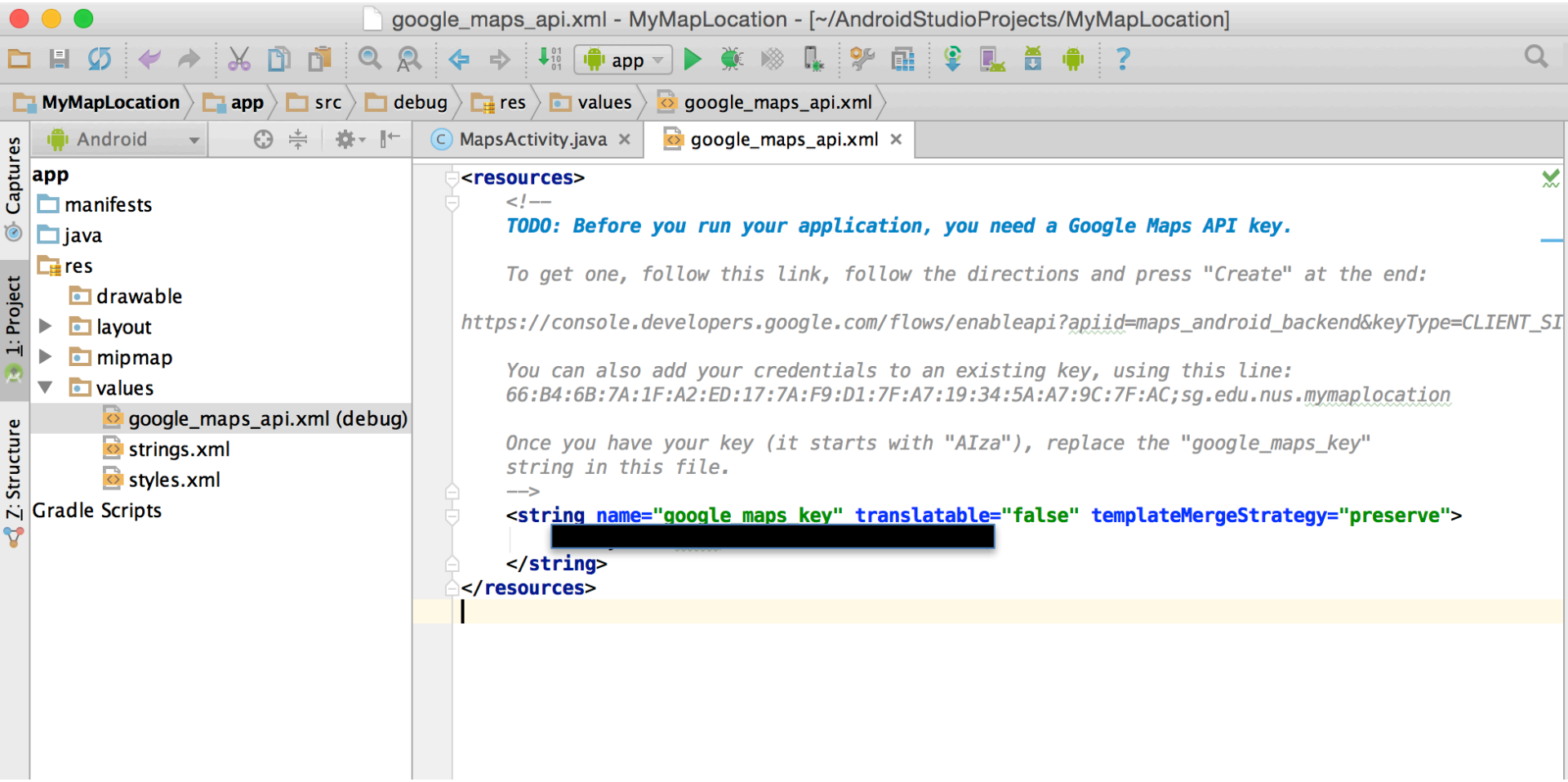

console.developers.google.com
GoogleAPIs
The API is enabled
The project has been created and Google Maps Android API has been enabled.
Next, you'll need to create an API key in order to call the API.
Create API key


Google APIs
My Project
APIs & services
Credentials
Dashboard
Library
Credentials
Credentials
OAuth consent screen
Domain verification
Create credentials
Delete
Create credentials to access your enabled APIs. Refer to the API documentation for details.
API key created
Use this key in your application by passing it with the key=API_KEY parameter.
Your API key
[redacted]
Restrict your key to prevent unauthorised use in production.
CLOSE
RESTRICT KEY


GoogleAPIs My Project
API API Manager
Dashboard
Library
Credentials
Credentials
Regenerate key
Delete
API key
This API key can be used in this project and with any API that supports it. To use this key in your application, pass it with the key=API_KEY parameter.
Creation date 23 Sep 2016, 11:25:02
Created by ngteckkhim@gmail.com (you)
API key
[redacted]
Name
API key 1
Key restriction
Key restriction lets you specify which websites, IP addresses or apps can use this key. Learn more.
None
HTTP referrers (websites)
IP addresses (web servers, cron jobs, etc.)
Android apps
iOS apps
Restrict usage to your Android apps
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps Learn more
Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:
$ keytool -list -v -keystore mystore.keystore

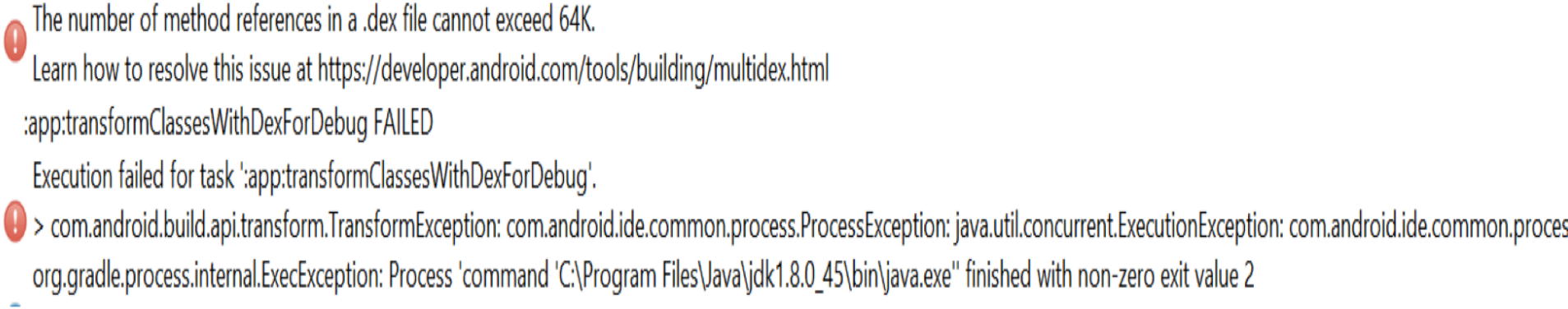

Google APIs
My Project
APIs & services
Dashboard
Library
Credentials
API key
REGENERATE KEY
DELETE
This API key can be used in this project and with any API that supports it. To use this key in your application, pass it with the key=API_KEY parameter.
Creation date
23 Sep 2017, 10:55:14
Created by
ngteckkhim@gmail.com (you)
API key
[redacted]
Name
API key 1
Key restriction
Key restriction lets you specify which websites, IP addresses or apps can use this key. Learn more.
None
HTTP referrers (websites)
IP addresses (web servers, cron jobs, etc.)
Android apps
iOS apps
Restrict usage to your Android apps (Optional)
Add your package name and SHA-1 signing-certificate fingerprint to restrict usage to your Android apps
Get the package name from your AndroidManifest.xml file. Then use the following command to get the fingerprint:
$ keytool -list -v -keystore mystore.keystore
Package name
SHA-1 certificate fingerprint
sg.edu.nus.ngtk.myweekendmap
[redacted]
Add package name and fingerprint
Note: It may take up to 5 minutes for settings to take effect.
Save
Cancel


GoogleAPIs
My Project
API Manager
Dashboard
Library
Credentials
Credentials
Credentials
OAuth consent screen
Domain verification
Create credentials
Delete
Create credentials to access your enabled APIs. Refer to the API documentation for details.
API keys
Name
Creation date
Restriction
Key
API key 1


google_maps_api.xml - MyMapLocation - [~/AndroidStudioProjects/MyMapLocation]
app
MyMapLocation
app
src
debug
res
values
google_maps_api.xml
Android
MapsActivity.java
google_maps_api.xml
app
manifests
java
res
drawable
layout
mipmap
values
google_maps_api.xml (debug)
strings.xml
styles.xml
Gradle Scripts
<resources>
<!--
TODO: Before you run your application, you need a Google Maps API key.
To get one, follow this link, follow the directions and press "Create" at the end:
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SI
You can also add your credentials to an existing key, using this line:
66:B4:6B:7A:1F:A2:ED:17:7A:F9:D1:7F:A7:19:34:5A:A7:9C:7F:AC;sg.edu.nus.mymaplocation
Once you have your key (it starts with "AIza"), replace the "google_maps_key"
string in this file.
-->
<string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
</string>
</resources>
1: Project
Captures
Z: Structure

Note:

When creating the map project, if you encounter a problem about exceeding number of method references in a .dex file (error message as follows), then follow the instructions in the next slide.



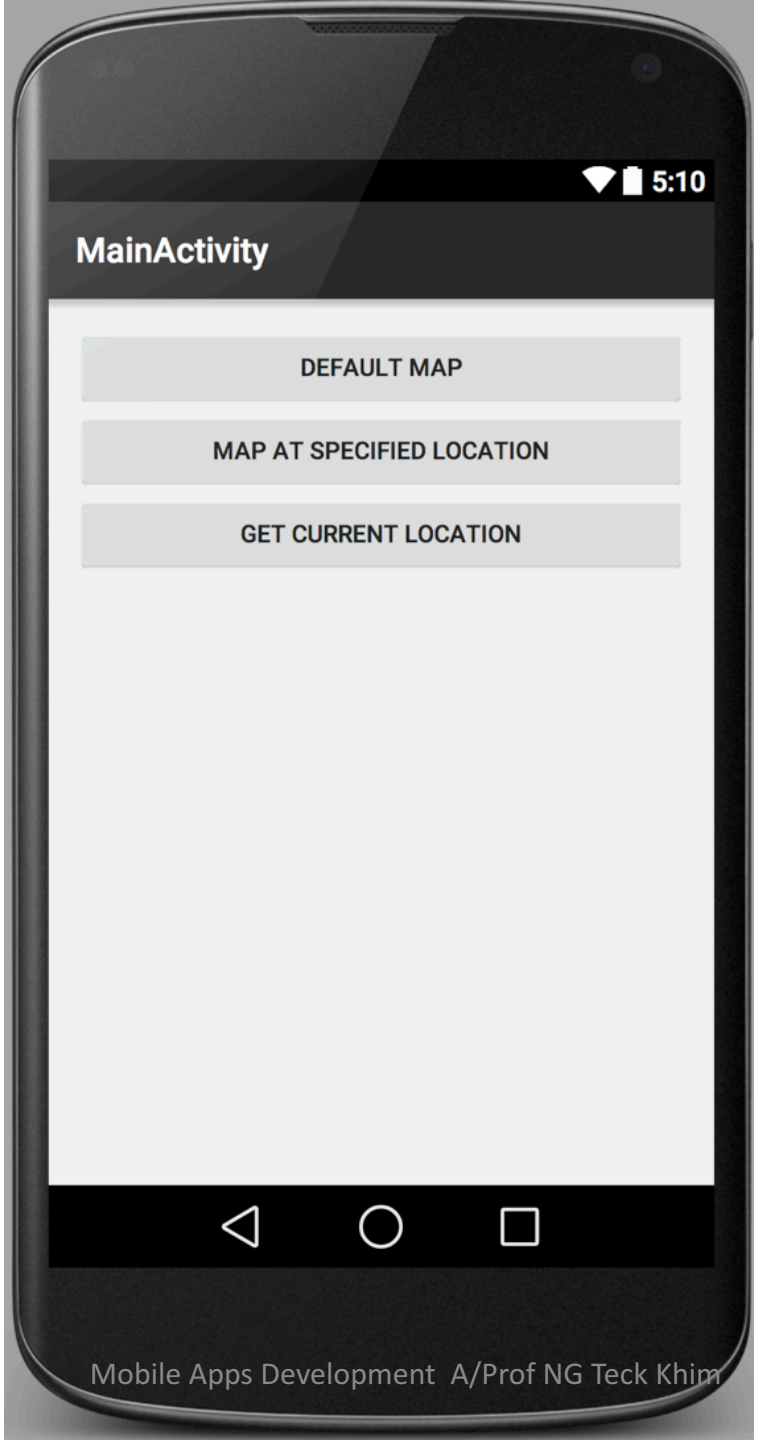
2 solutions:

1) War Tank Solution (enables multiDex in the app. Note that launch time is affected)

```
defaultConfig {

    multiDexEnabled true
```

1) Light solution (compile only maps and location, instead of compiling all google play services). Just replace the default

compile 'com.google.android.gms:play-services:8.4.0'
with
compile 'com.google.android.gms:play-services-maps:8.4.0'
compile 'com.google.android.gms:play-services-location:8.4.0'


5:10
MainActivity
DEFAULT MAP
MAP AT SPECIFIED LOCATION
GET CURRENT LOCATION


11:02 AM
Mongolia
China
South Korea
India
Thailand
Indonesia
Papua New Guinea
Indian Ocean
Australia
New Zealand
Google

Add the following in AndroidManifest.xml :

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

Within <application />, add the following:

```
<meta-data
        android:name=”com.google.android.geo.API_KEY"
        android:value="@string/google_maps_key" />
```

Add the following in build.gradle dependencies:

```
compile 'com.google.android.gms:play-services-maps:11.0.4'
compile 'com.google.android.gms:play-services-location:11.0.4'
```

Note:

- gms == Google Mobile Services.
- com.google.android.gms comes from Google Play Services SDK, which we can attach to our application project as an Android library project.

MAP

- To specify latitude-longitude of locations:

```
LatLng NUS         = new LatLng(1.2956, 103.776);
LatLng OrchardRd = new LatLng(1.3051, 103.831);
```

In onMapReady(), add the following:

```
LatLng NUS          = new LatLng(1.2956, 103.776);
LatLng OrchardRd  = new LatLng(1.3051, 103.831);

Marker markerNUS = mMap.addMarker(new
MarkerOptions().position(NUS).title("Marker in NUS").snippet("This is NUS"));

Marker markerOrchard = mMap.addMarker(new
MarkerOptions().position(OrchardRd).title("Marker in Orchard
Road").snippet("This is Orchard
Road").icon(BitmapDescriptorFactory.fromResource(R.drawable.location_pin)));

// move camera to NUS with a zoom of 20

mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(NUS, 20));

// zoom in, animating the camera

mMap.animateCamera(CameraUpdateFactory.zoomTo(10), 2000, null);
```

Location
- Dealing with Dangerous Permissions

Add the following in AndroidManifest.xml:

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

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

Add the following in build.gradle:

```
compile 'com.google.android.gms:play-services-maps:11.0.4'
compile 'com.google.android.gms:play-services-location:11.0.4'
```

Specify Dangerous Permissions in Activity

(note: we won't use camera here. It is included here just for illustrating how to deal with multiple permissions)

```
// for permissions
private static final int REQUEST_ACCESS_FINE_LOCATION_PERMISSION = 200;
private static final int REQUEST_TO_USE_CAMERA_PERMISSION = 300;

private boolean permissionToAccessFineLocationAccepted = false;
private boolean permissionToUseCamera = false;


private String[] permissions =
      {Manifest.permission.ACCESS_FINE_LOCATION,
       Manifest.permission.CAMERA};
```

In onCreate(), add the following:

```
// request permission from user to access fine location
if (ContextCompat.checkSelfPermission(this,
      Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
   ActivityCompat.requestPermissions(this,
         permissions,
         REQUEST_ACCESS_FINE_LOCATION_PERMISSION);
}

if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
      == PackageManager.PERMISSION_DENIED) {
   ActivityCompat.requestPermissions(this, permissions,
REQUEST_ACCESS_FINE_LOCATION_PERMISSION);
}
```

In onCreate(), add the following:

```
// request permission from user to access Camera
if (ContextCompat.checkSelfPermission(this,
      Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
   ActivityCompat.requestPermissions(this,
         permissions,
         REQUEST_TO_USE_CAMERA_PERMISSION);
}

if (ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA)
      == PackageManager.PERMISSION_DENIED) {
   ActivityCompat.requestPermissions(this, permissions,
REQUEST_TO_USE_CAMERA_PERMISSION);
}
```

Add the following in the Activity:

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
   super.onRequestPermissionsResult(requestCode, permissions, grantResults);

   switch (requestCode) {
      case REQUEST_ACCESS_FINE_LOCATION_PERMISSION:
         if (grantResults.length>0) {
            permissionToAccessFineLocationAccepted = grantResults[0] ==
                  PackageManager.PERMISSION_GRANTED;
         }
         break;

      case REQUEST_TO_USE_CAMERA_PERMISSION:
         if (grantResults.length>1) {
            permissionToUseCamera = grantResults[1] ==
PackageManager.PERMISSION_GRANTED;
         }
         break;
   }
```

```
    if ((!permissionToAccessFineLocationAccepted) ||
            (!permissionToUseCamera))
        finish();

}
```

Now we are ready to get location information

Get Current Location

- Specify that the Activity will implement the following:
  com.google.android.gms.location.LocationListener,
  GoogleApiClient.ConnectionCallbacks,
  GoogleApiClient.OnConnectionFailedListener

- Define the following variables

  LocationRequest **mLocationRequest**;

  GoogleApiClient **mGoogleApiClient**;

In onCreate()

mLocationRequest = new LocationRequest();

mLocationRequest.setInterval(***10000***);

mLocationRequest.setFastestInterval(***5000***);

mLocationRequest.setPriority(LocationRequest.***PRIORITY_HIGH_ACCURACY***);

In onCreate()

```
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addApi(LocationServices.API)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .build();
```

In onStart(), add the following after super.onStart()

mGoogleApiClient.connect();

In onStop(), add the following before super.onStop()

mGoogleApiClient.disconnect();

In onPause(), add the following before super.onPause()

LocationServices.***FusedLocationApi***.removeLocationUpdates (**mGoogleApiClient**, **this**);

In onResume(), add the following after super.onResume();

```
if (mGoogleApiClient.isConnected()) {
    if (ContextCompat.checkSelfPermission(this,
            android.Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {

        PendingResult<Status> pendingResult =
            LocationServices.FusedLocationApi.requestLocationUpdates(
                                mGoogleApiClient, mLocationRequest, this);
}
```

Note: for <Status>, there is ambiguity in autoimport. The correct one is

import com.google.android.gms.common.api.Status;

In onConnected(@Nullable Bundle bundle), add

```
if (ContextCompat.checkSelfPermission(this,
        android.Manifest.permission.ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED) {

       PendingResult<Status> pendingResult =

LocationServices.FusedLocationApi.requestLocationUpdates(
                    mGoogleApiClient, mLocationRequest, this);
}
```

```
@Override
public void onConnected(Bundle bundle) {
```

```
Location mCurrentLocation;

@Override
public void onConnectionSuspended(int i) {
}

@Override
public void onConnectionFailed(@NonNull ConnectionResult
connectionResult) {
    Log.d(TAG, "Connection failed: " + connectionResult.toString());
}
```

```
@Override
public void onLocationChanged(Location location) {

    mCurrentLocation =  location;

    if (mCurrentLocation != null) {

      String lat = String.valueOf(mCurrentLocation.getLatitude());
      String lng = String.valueOf(mCurrentLocation.getLongitude());

      // get the location with accuracy and also provider, using
      // mCurrentLocation.getAccuracy()
      // mCurrentLocation.getProvider());
   }
```