CSE 162 Mobile Computing

Lab 4 Location and Map Programming

Tasks

- Creating a google map

- Using Location services

Maps SDK for Android

- Set up the development environment
- Set up an Android device
- Create a Google Maps project in Android Studio
- Set up in Cloud Console
- Add the API key to your app
- Deploy and run the app

Set up the development environment

- Android Studio is required. If you haven't already done so, download and install it.
- Add the Google Play services SDK to Android Studio. The Maps SDK for Android is distributed as part of the Google Play services SDK, which you can add through the SDK Manager.

Set up an Android device

- To run an app that uses the Maps SDK for Android, you must deploy it to an Android device or Android emulator that is based on Android 4.0 or higher and includes the Google APIs.

Create a Google Maps project in Android Studio

- Open Android Studio, and click Create New Project in the Welcome to Android Studio window.
- In the New Project window, under the Phone and Tablet category, select the Google Maps Activity, and then click Next.
- Complete the Google Maps Activity form:
  - Set Language to Java
  - Set Minimum SDK to an SDK version compatible with your test device. You must select a version greater than the minimum version required by the Maps SDK for Android version 18.0.x, which is currently Android API Level 19 (Android 4.4, KitKat) or higher.

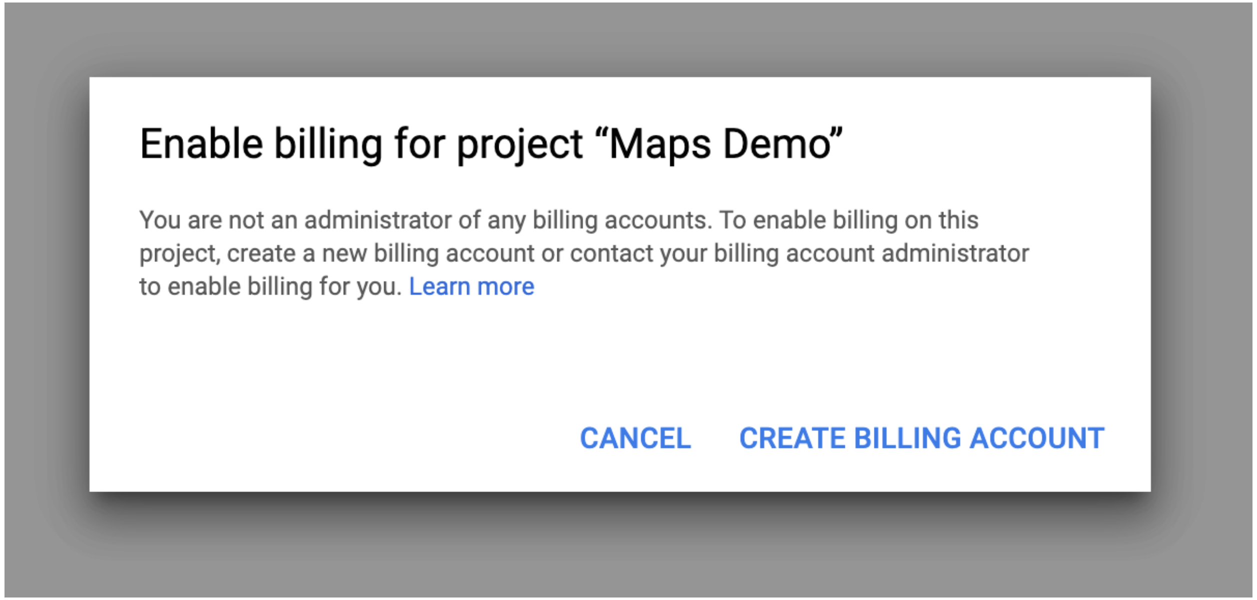
Create a Google Maps project in Android Studio

- When the build is finished, Android Studio opens the google_maps_api.xml and MapsActivity files. Your activity may have a different name, but it will be the one you configured during setup.

- The google_maps_api.xml file contains instructions on getting a Google Maps API key and then adding it to the file. Do not add your API key to the google_maps_api.xml file. Doing so stores your API key less securely. Instead, follow the instructions in the next sections to create a Cloud project and configure an API key.

Set up in Cloud Console

- Navigate to the Google Maps Getting Started guide. You will need to setup a billing account and enable the Google Maps API.

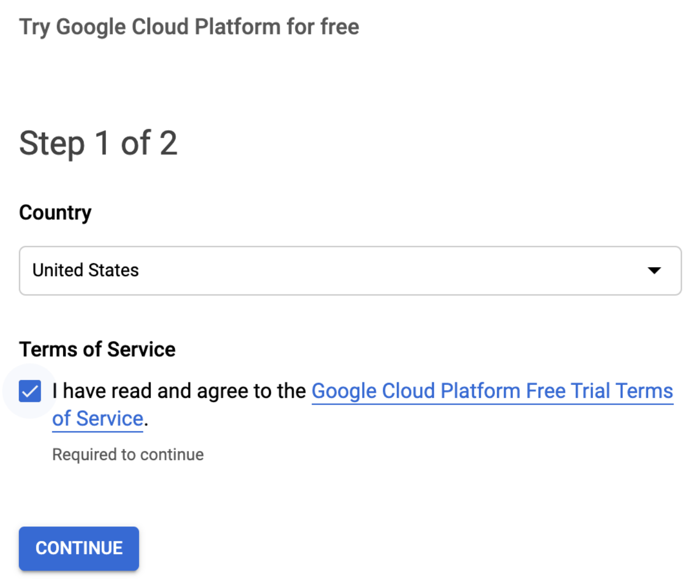


Enable billing for project “Maps Demo”

You are not an administrator of any billing accounts. To enable billing on this project, create a new billing account or contact your billing account administrator to enable billing for you. Learn more

Set up in Cloud Console

- You should then be prompted to accept terms of service:



Try Google Cloud Platform for free

Step 1 of 2

Country

United States

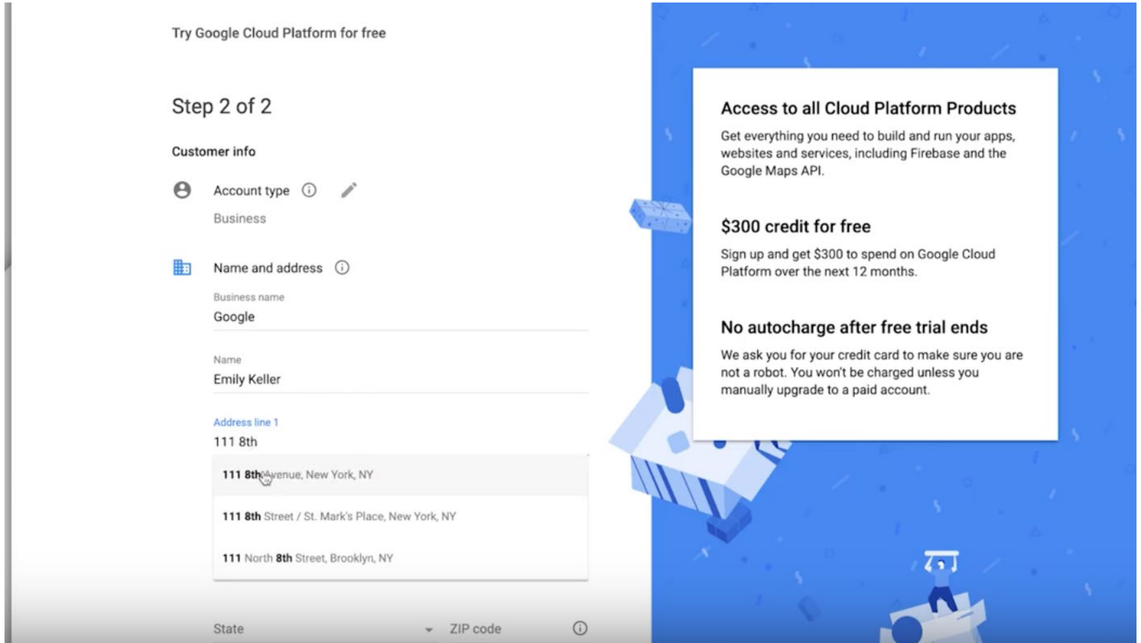
Terms of Service

[✓] I have read and agree to the Google Cloud Platform Free Trial Terms of Service.

Required to continue

Set up in Cloud Console

- In the next step, you'll have to go through the steps to enable billing for your project (but you won't get charged for using this API). This requires entering your credit card information:



Try Google Cloud Platform for free

Step 2 of 2

Customer info

Account type
Business

Name and address

Business name
Google

Name
Emily Keller

Address line 1
111 8th

111 8th Avenue, New York, NY
111 8th Street / St. Mark's Place, New York, NY
111 North 8th Street, Brooklyn, NY

State
ZIP code

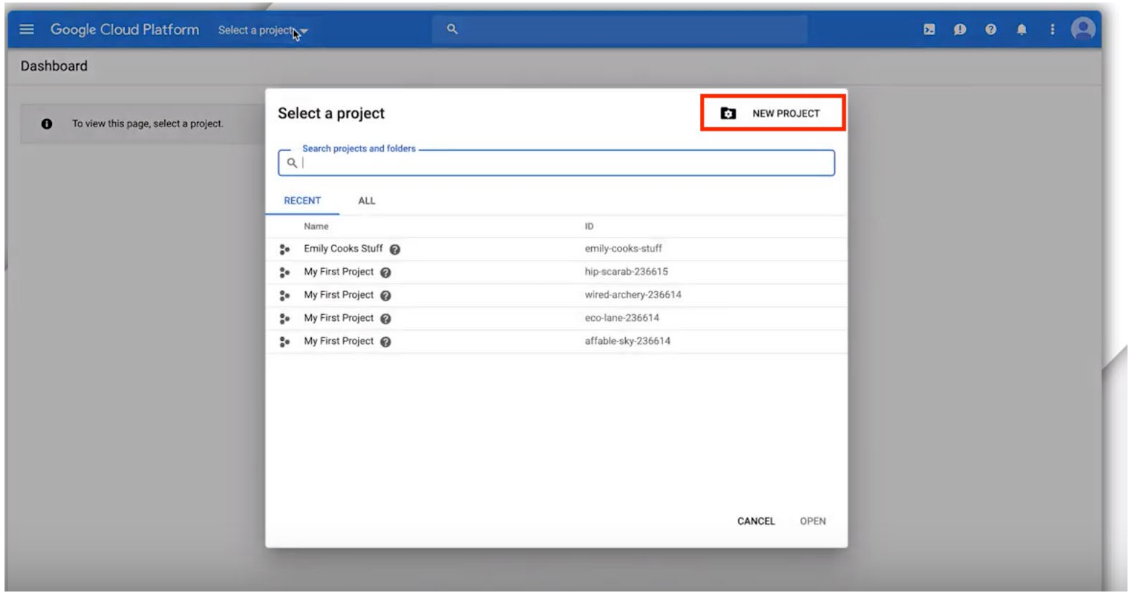
Access to all Cloud Platform Products
Get everything you need to build and run your apps, websites and services, including Firebase and the Google Maps API.

$300 credit for free
Sign up and get $300 to spend on Google Cloud Platform over the next 12 months.

No autocharge after free trial ends
We ask you for your credit card to make sure you are not a robot. You won't be charged unless you manually upgrade to a paid account.

Set up in Cloud Console

- Once you've created a billing account, you will need to create a new project by clicking the dropdown at the top. Click on the New Project in the corner:

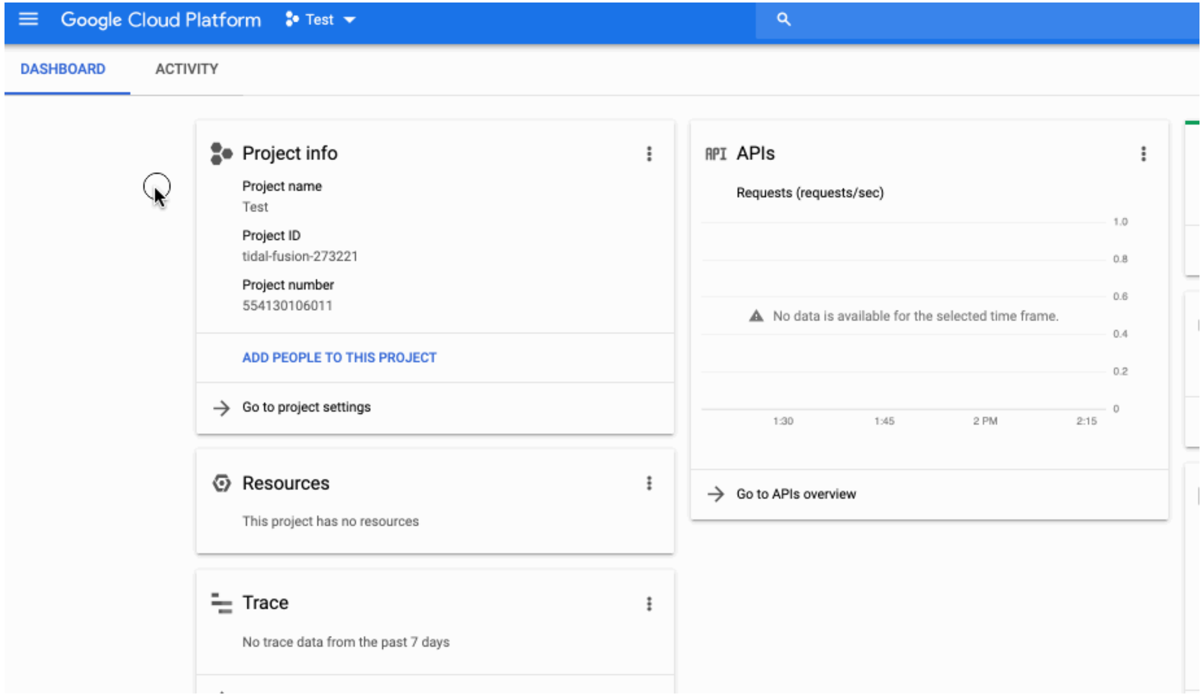


Select a project

Name	ID
Emily Cooks Stuff	emily-cooks-stuff
My First Project	hip-scarab-236615
My First Project	wired-archery-236614
My First Project	eco-lane-236614
My First Project	affable-sky-236614

Set up in Cloud Console

- Once the project is created, navigate to Dashboard, click on Enable APIS and Services, and enable the Maps SDK for Android API:



Project info

Project name
Test

Project ID
tidal-fusion-273221

Project number
554130106011

ADD PEOPLE TO THIS PROJECT

Go to project settings

Requests (requests/sec)

No data is available for the selected time frame.

Resources

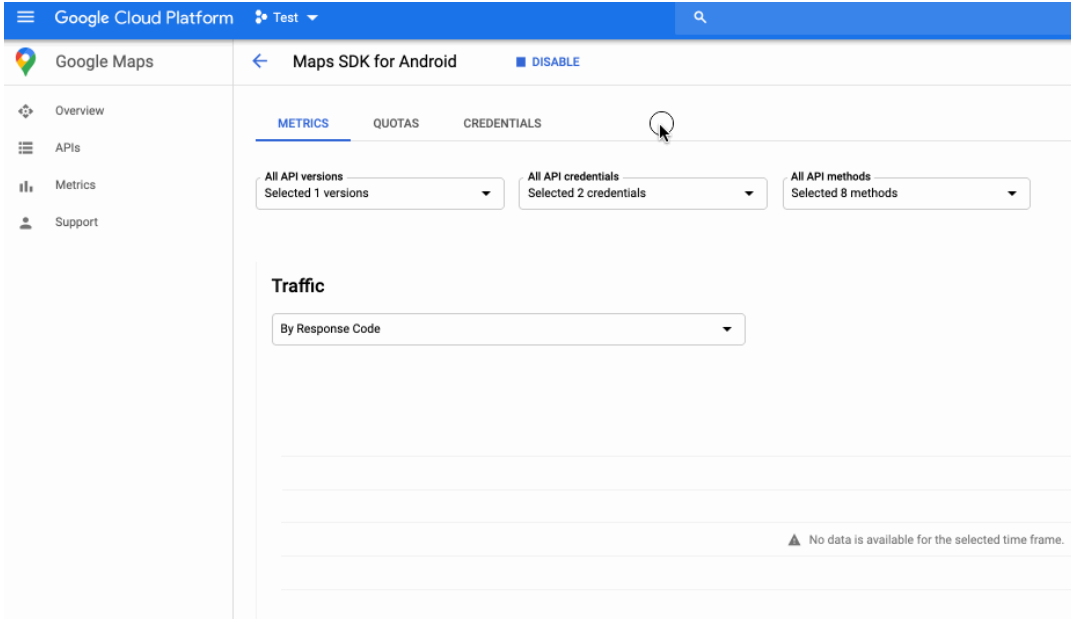
This project has no resources

Trace

No trace data from the past 7 days

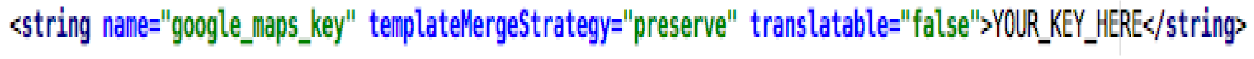
Set up in Cloud Console

- Click the Credentials link in the left sidebar, follow the next link Credentials in API Services, and click on Create Credentials. Select API Key and copy the contents to your clipboard:



Creating a map

Look into **google_maps_api**

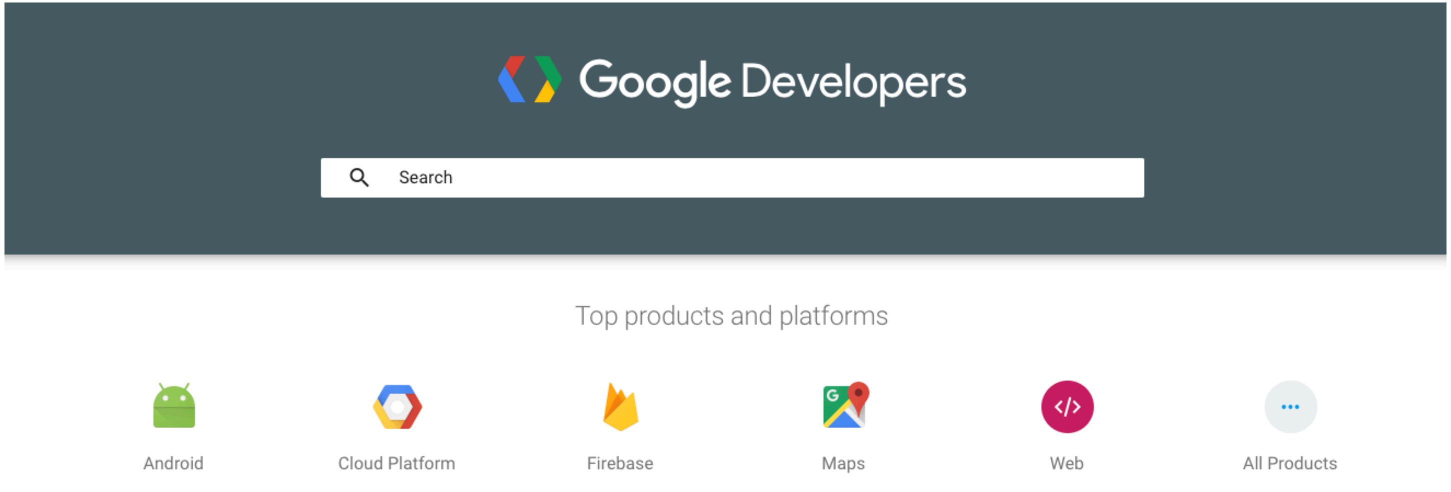


```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">YOUR_KEY_HERE</string>
```

You need to create a new key and paste it over the “YOUR_KEY_HERE” placeholder

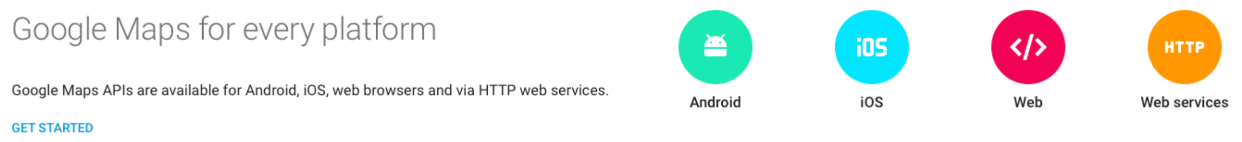
Enabling Google Map API

Head over to https://developers.google.com



Select **Maps** among the options

Enabling Google Map API

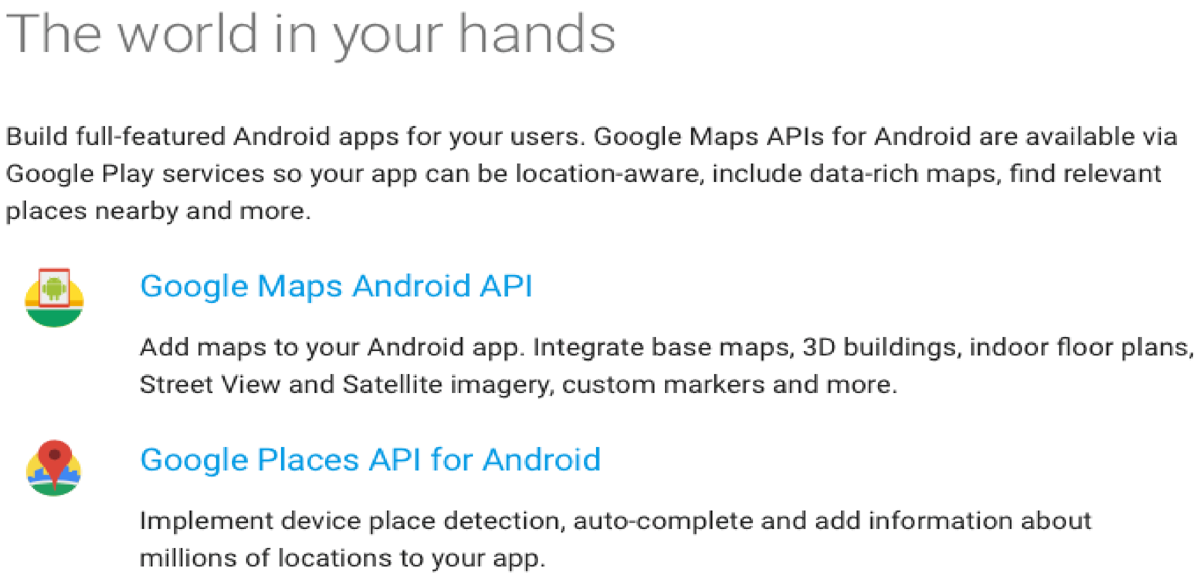


Google Maps for every platform

Google Maps APIs are available for Android, iOS, web browsers and via HTTP web services.

Select Android as the platform for your Map

Enabling Google Map API



The world in your hands

Build full-featured Android apps for your users. Google Maps APIs for Android are available via Google Play services so your app can be location-aware, include data-rich maps, find relevant places nearby and more.

Google Maps Android API

Add maps to your Android app. Integrate base maps, 3D buildings, indoor floor plans, Street View and Satellite imagery, custom markers and more.

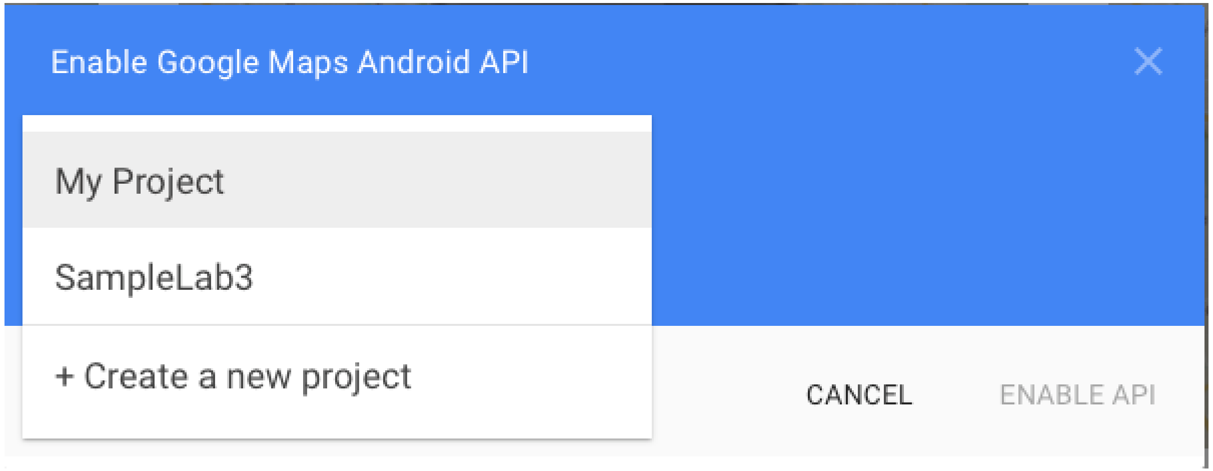
Google Places API for Android

Implement device place detection, auto-complete and add information about millions of locations to your app.

Select Maps for Android Api

Enabling Google Map API

- Now you have to get a key



Enable Google Maps Android API

My Project

SampleLab3

+ Create a new project

Create a new project and enable the API.

Your Android App will now communicate with this API with the key that you get.

Add the API key to your app

- In Android Studio, open your project-level build.gradle file and add the following code to the dependencies element under buildscript.

```
buildscript {
    dependencies {
        // ...
        classpath "com.google.android.libraries.mapsplatform.secrets-gradle-
plugin:secrets-gradle-plugin:2.0.0"
    }
}
```

- Next, open your module-level build.gradle file and add the following code to the plugins element.

```
id 'com.google.android.libraries.mapsplatform.secrets-grad
le-plugin'
```

- Save the file and sync your project with Gradle.

- Open the local.properties in your project level directory, and then add the following code. Replace YOUR_API_KEY with your API key.

```
MAPS_API_KEY=YOUR_API_KEY
```

- In your AndroidManifest.xml file, go to com.google.android.geo.API_KEY and update the android:value attribute as follows:

```
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="${MAPS_API_KEY}" />
```

Deploy and run the app

Location Acquisition

Registering for Location Updates

- The LocationManager handles registrations for GPS and network location updates
- In order for an object to receive updates from GPS, it must implement the LocationListener interface
- Once the LocationManager is obtained, an object registers for updates by calling requestLocationUpdates (there are multiple versions you can use)
- The arguments passed into the requestLocationUpdates method determine the granularity of location changes that will generate an event
  - send updates that are at least X meters apart
  - send updates at least this far apart in time
  - send updates that have this minimum accuracy

The LocationListener Interface

```
public class MyActivity … implements LocationListener{
  …
            // Called when your GPS location changes
            @Override
            public void onLocationChanged(Location location) {
            }

            // Called when a provider gets turned off by the user in the settings
            @Override
            public void onProviderDisabled(String provider) {

            }

            // Called when a provider is turned on by the user in the settings
            @Override
            public void onProviderEnabled(String provider) {

            }

            // Signals a state change in the GPS (e.g. you head through a tunnel and
            // it loses its fix on your position)
            @Override
            public void onStatusChanged(String provider, int status, Bundle extras) {


            }
}
```

location updates

```
public class MapsActivity extends FragmentActivity implements
OnMapReadyCallback,LocationListener {

    private GoogleMap mMap;
    private LocationManager locationManager;
    private double longitude=151;
    private double latitude=-34;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
        locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
        if(ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)== PackageManager.PERMISSION_GRANTED)
            locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 10,
Criteria.ACCURACY_COARSE, this);
    }
```

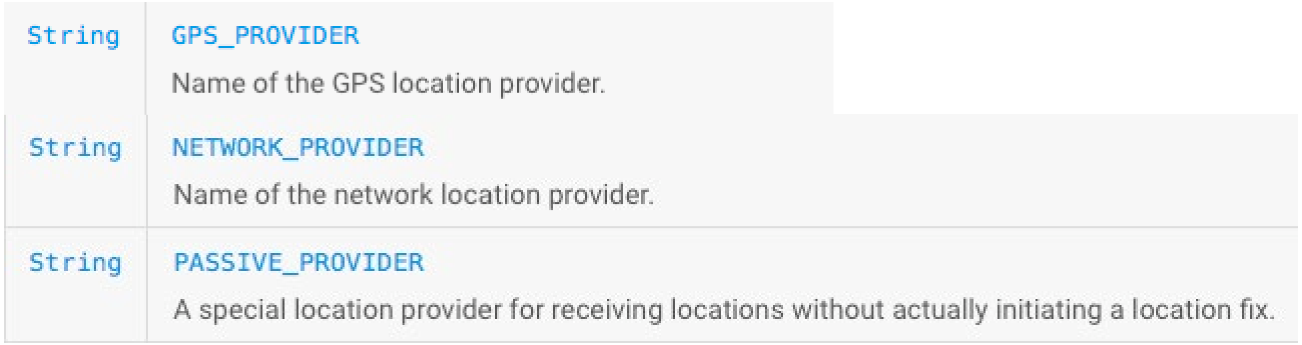
Getting Location Info

```
@Override
public void onLocationChanged(Location location) {
     longitude = location.getLongitude();
     latitude = location.getLatitude();
    AlertDialog alertDialog = new AlertDialog.Builder(MapsActivity.this).create();
    alertDialog.setTitle("Location Detected");
    alertDialog.setMessage(String.valueOf(latitude)+","+String.valueOf(longitude));
    alertDialog.setButton(AlertDialog.BUTTON_NEUTRAL, "OK",
            new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            });
    alertDialog.show();
    LatLng loc = new LatLng(latitude, longitude);
    mMap.addMarker(new MarkerOptions().position(loc).title("Marker in at your location"));
    mMap.moveCamera(CameraUpdateFactory.newLatLng(loc));
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(loc, 12.0f));

}
```

Location Providers

- The phone’s location can be determined from multiple providers
  - GPS
  - Network
- GPS location updates consume significantly more power than network location updates but are more accurate
  - GPS: 25 seconds * 140mA = 1mAh
  - Network: 2 seconds * 180mA = 0.1mAh
- The provider argument determines which method will be used to get a location for you
- You can also register for the PASSIVE_PROVIDER which only updates you if another app is actively using GPS / Network location



String	GPS_PROVIDER Name of the GPS location provider.
String	NETWORK_PROVIDER Name of the network location provider.
String	PASSIVE_PROVIDER A special location provider for receiving locations without actually initiating a location fix.

Being a Good Citizen…

- It is very important that you unregister your App when you no longer need updates
- For example, you should always unregister your listener when your Activity is paused!
- If you unregister when you pause, you must also reregister when you resume
  - This is true for all sensors!

```
protected void onPause() {
   super.onPause();
   try{
      locationManager.removeUpdates(this);}
   catch(SecurityException e){ Log.e("Err","No Location update permission remover");}
}
protected void onResume() {
   super.onResume();
   if(ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION)== PackageManager.PERMISSION_GRANTED)
      locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
10,Criteria.ACCURACY_COARSE, this);
}
```

Assignments

- Display the map
- Set the map view based on the GPS tracking results