Lab 6: Working With APIs and Maps

Due Nov 8 by 3:45pm **Points** 10 **Submitting** a text entry box or a file upload
Available Oct 29 at 1:45pm - Nov 30 at 3:45pm about 1 month

This assignment was locked Nov 30 at 3:45pm.

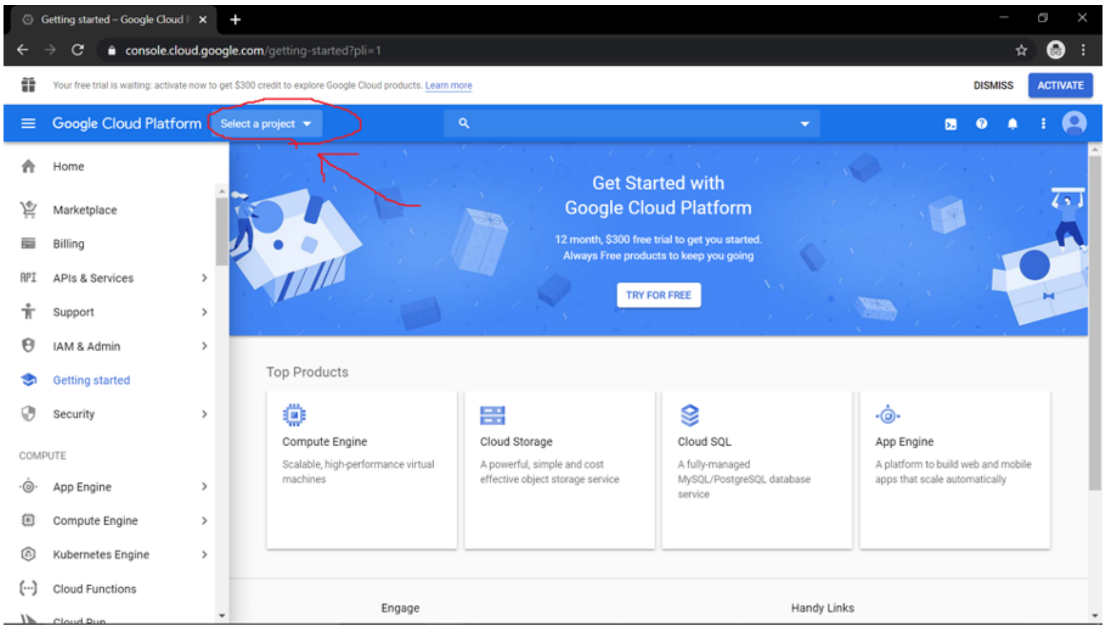
Introduction:

This lab will show you how to integrate external APIs into your Android app. You will find this helpful if you are looking at using external APIs in your projects. We will be using Google Maps as an example. In this lab you will generate API keys to be used with Google Maps to set up a SupportMapFragment, which is the fragment used to display Maps on Android (just like AlertDialogFragments are used to display alerts). You'll learn how to query the device for its location and display a route between your location and any other location on the map.

Prerequisites:

If you have not generated Google Maps API keys before, here are the steps to do the same:

1. Go to the Google Cloud Platform Console at https://console.cloud.google.com/ (https://console.cloud.google.com/)
2. Agree to the terms and services, if a dialog pops up (But feel free to read through it!)
3. Click the 'Select a Project' drop-down as shown in the screenshot below:

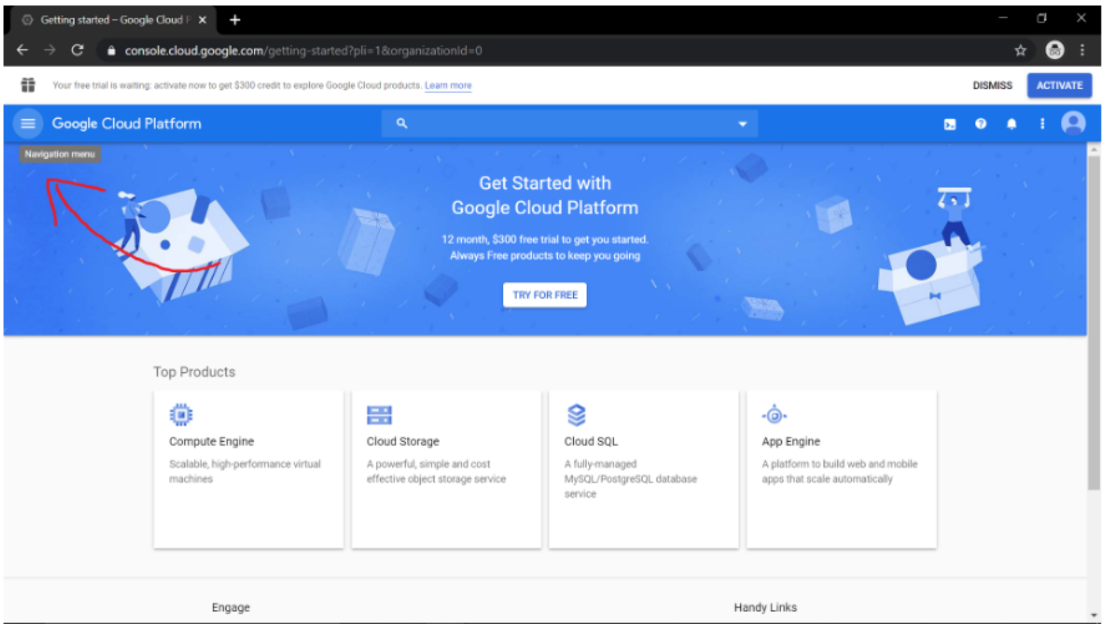

Getting started – Google Cloud
console.cloud.google.com/getting-started?pli=1
Your free trial is waiting: activate now to get $300 credit to explore Google Cloud products. Learn more
DISMISS
ACTIVATE
Google Cloud Platform
Select a project
Home
Marketplace
Billing
APIs & Services
Support
IAM & Admin
Getting started
Security
COMPUTE
App Engine
Compute Engine
Kubernetes Engine
Cloud Functions
Get Started with Google Cloud Platform
12 month, $300 free trial to get you started.
Always Free products to keep you going
TRY FOR FREE
Top Products
Compute Engine
Scalable, high-performance virtual machines
Cloud Storage
A powerful, simple and cost effective object storage service
Cloud SQL
A fully-managed MySQL/PostgreSQL database service
App Engine
A platform to build web and mobile apps that scale automatically
Engage
Handy Links

(**Note**: If you've created a Google Platform project in the past, the drop down may have the name of that project. You don't have to worry about it as we will be creating a new project.)

4. Tap "New Project".

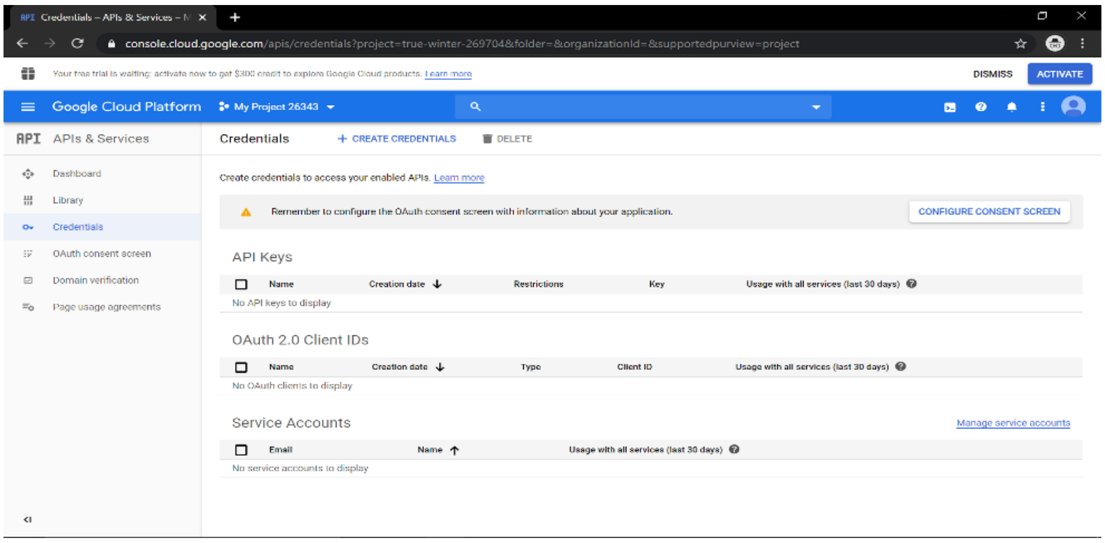
5. You can leave the defaults that appear,make a note of the name of the project that is being created and tap Create.

6. Once the project is created, click the menu button (as shown below):


Getting started – Google Cloud
console.cloud.google.com/getting-started?pli=1&organizationId=0
Your free trial is waiting: activate now to get $300 credit to explore Google Cloud products. Learn more
DISMISS
ACTIVATE
Google Cloud Platform
Navigation menu
Get Started with Google Cloud Platform
12 month, $300 free trial to get you started. Always Free products to keep you going
TRY FOR FREE
Top Products
Compute Engine
Scalable, high-performance virtual machines
Cloud Storage
A powerful, simple and cost effective object storage service
Cloud SQL
A fully-managed MySQL/PostgreSQL database service
App Engine
A platform to build web and mobile apps that scale automatically
Engage
Handy Links

7. Select APIs & Services > Credentials.

8. If your credentials page doesn't look similar to the screenshot shown below, you may have to tap 'Select Project' and select the project you created in Step-5.


Credentials – APIs & Services
console.cloud.google.com/apis/credentials?project=true-winter-269704&folder=&organizationId=&supportedpurview=project
Your free trial is waiting: activate now to get $300 credit to explore Google Cloud products. Learn more
DISMISS
ACTIVATE
Google Cloud Platform
My Project 26343
APIs & Services
Dashboard
Library
Credentials
OAuth consent screen
Domain verification
Page usage agreements
CREATE CREDENTIALS
DELETE
CONFIGURE CONSENT SCREEN
Manage service accounts

Credentials

Create credentials to access your enabled APIs. Learn more

Remember to configure the OAuth consent screen with information about your application.

API Keys

Name	Creation date ↓	Restrictions	Key	Usage with all services (last 30 days)

No API keys to display

OAuth 2.0 Client IDs

Name	Creation date ↓	Type	Client ID	Usage with all services (last 30 days)

No OAuth clients to display

Service Accounts

Email	Name ↑	Usage with all services (last 30 days)

No service accounts to display

9. On the Credentials page, click Create credentials → API key.

10. The dialog that appears upon clicking API key, displays your newly created API key. Make a note of this as you will be using it in your project.

11. Click Close

Milestone 1 - Setup Maps Activity with Marker at Bascom Hall

1. Make sure you have a MainActivity.java and an associated layout XML file.
2. In the layout xml file, delete its contents and add a SupportMapFragment:

```xml
<?xml version="1.0" encoding="utf-8"?>
<fragment
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:id="@+id/fragment_map"
   android:name="com.google.android.gms.maps.SupportMapFragment" />
   <!-- The name attribute represents the fragment is a MapFragment-->
```

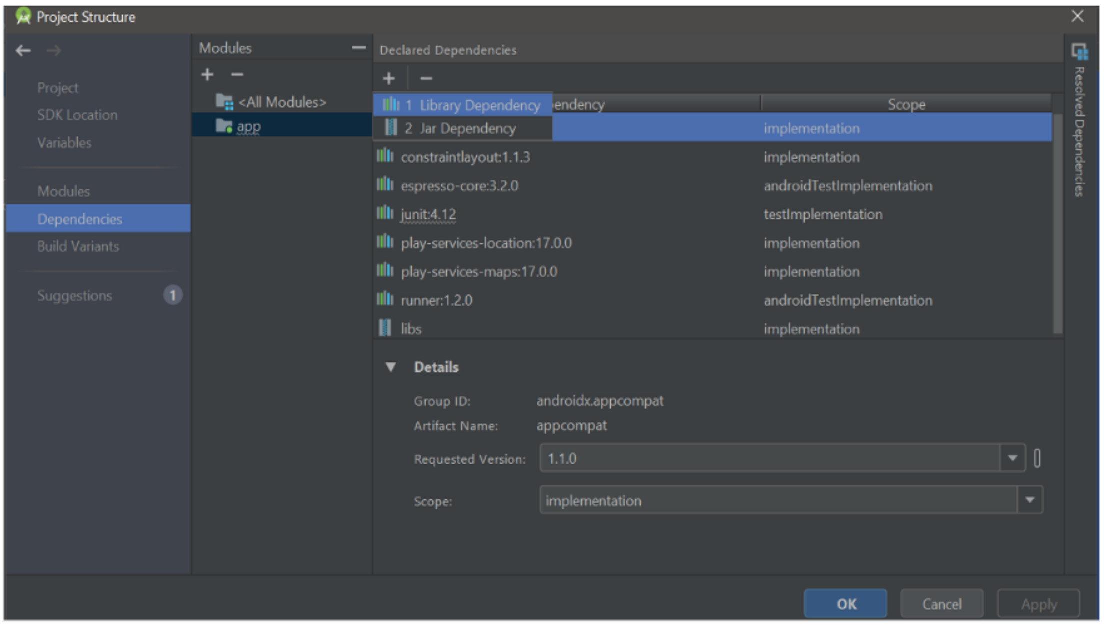
- You'll notice Android Studio having trouble resolving the **android:name attribute** above.

In order to resolve this we need to use certain external libraries.

- The reason why this needs to be done: mapping was introduced to Android a little later after its initial release. They could've bundled the functionality in with the next OS update, but a whole OS update takes a while to be compatible with all devices. Instead, they chose to add the functionality in with the *Play Store* to ensure it'd reach everyone sooner. As you will see in the next set of steps, the dependent library we will be using (or dependency) has 'play-services' in its name.

3. To import the dependency, go to File → Project Structure → Dependencies

and add a new **Library Dependency** in the app module, as shown below:



Dependency	Scope
constraintlayout:1.1.3	implementation
espresso-core:3.2.0	androidTestImplementation
junit:4.12	testImplementation
play-services-location:17.0.0	implementation
play-services-maps:17.0.0	implementation
runner:1.2.0	androidTestImplementation
libs	implementation

Details
Group ID: androidx.appcompat
Artifact Name: appcompat
Requested Version: 1.1.0
Scope: implementation

4. In the resulting dialog, search for:

com.google.android.gms:play-services-maps

Make sure the latest version is selected and press OK. It'll take you back to the screen pictured in Step-3. Just press Ok again.

5. Next, make sure your Main Activity Java file is showing the layout xml and retrieving the SupportMapFragment (the code to be added is highlighted in cyan for your convenience):

```
public class MainActivity extends FragmentActivity {

@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager()
                .findFragmentById(R.id.fragment_map);
    }
}
```

6. Go ahead and try to run your app. If all goes well, it should.... crash. That's because we didn't insert our API key anywhere in the app, and Google requires that we do to display Google Maps in our app (**Here's** (https://stackoverflow.com/questions/481571/why-do-some-api-providers-require-an-api-key) a more detailed discussion on this). To do so, open Manifest.xml and add in the following, replacing YOUR_API_KEY with the key you have:

```
<application
    ...
    <activity android:name=".MainActivity">
        ...
    </activity>
    <meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="YOUR_API_KEY" />
</application>
```

7. Since you're going to use internet in your app (for downloading maps), now's also a good time to declare that permission in your Manifest:

```
<manifest
    ...>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
```

8. Now we're going to go back to our Main Activity java file & work on displaying a marker/pin on the map. To do this we need to ensure that:

- The MapFragment is downloaded and initialized.
- After this, we display a marker.
- Calling **.getMapAsync(..)** on the mapFragment will conveniently notify you when the mapFragment is initialized and is ready, so we can create our markers after the said

notification as shown:

```
public class MainActivity extends FragmentActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
        ...
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.fragment_map);
        mapFragment.getMapAsync(googleMap -> {
                mMap = googleMap;
            // code to display marker
        });
}
```

9. In Step-8, we're saving the GoogleMap object we obtain in another variable (mMap) for later use. Let us define the mMap variable as well as another variable that can represent the latitude & longitude of the marker we want to create.

```
public class MainActivity extends FragmentActivity {

    // Somewhere in Australia
    private final LatLng mDestinationLatLng = new LatLng(-33.8523341, 151.2106085);
    private GoogleMap mMap;

        @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
    }
}
```

10. **Creating the marker:**

To create a fresh new marker and add it to our map, we use the following call:

```
googleMap.addMarker(new MarkerOptions())
```

and we can then just chain commands to customize the marker:

```
mapFragment.getMapAsync(googleMap -> {
            mMap = googleMap;
            // code to display marker
                googleMap.addMarker(new MarkerOptions()
                        .position(mDestinationLatLng)
                        .title("Destination"));
        });
```

- In the above call, ".position" determines where the marker is placed. The LatLng object we created in Step-9 will be used here to set the marker at that location.
- "Destination" is now the text that pops up when you tap on the marker. If nothing shows up on the map, make sure that the "Google Maps Android API v2" is enabled (see trouble shooting for more details).

Deliverables to be shown to the instructors for Milestone1:

change the Destination Marker so it's situated at **Bascom Hall** (Hint: Look it up on Google Maps & inspect the URL to get the required location parameters) and show the marker on the AVD/device.

Milestone 2 - Another marker at current location joined via a polyline to existing marker

1. Things start getting interesting here. Android makes it really simple to query a device's location.

- The class we're going to use is **FusedLocationProviderClient**.
- It is a class that gets location (the method we use from this class is called ".getLastLocation()").
- It even tries to balance battery & accuracy by using a combination of GPS & network towers, hence the 'fused' part in the name. To obtain and save an instance of this class, add the following:

```
public class MainActivity extends FragmentActivity {
    private FusedLocationProviderClient mFusedLocationProviderClient; // Save the instance

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...

        // Obtain a FusedLocationProviderClient.
        mFusedLocationProviderClient =
LocationServices.getFusedLocationProviderClient(this);
    }
...
}
```

To add the reference to **LocationServices** above (***another neat class you should** **check out (https://developers.google.com/android/reference/com/google/android/gms/location/LocationServices)** **if you want to do anything location-related in your projects***!), we need to import a dependency.

Go to File → Project Structure → Dependencies. Add the following library in the app module (as we did earlier in Milestone-1):

com.google.android.gms:play-services-location

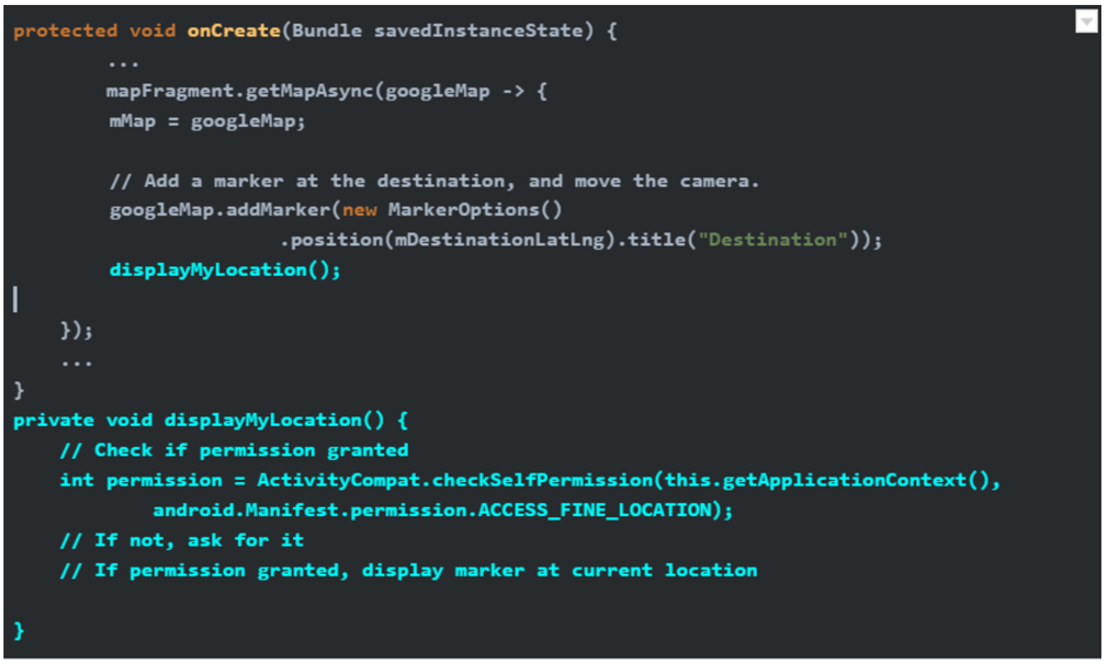
Choose the latest version and press OK. And then OK again.

(**Note:** If you try to run the app at any point after this step and crash, check the troubleshooting section at the end of this lab manual.)

2. Since we're obtaining the user's location, and that is something we need permission for, make sure you include the following permission in your Manifest file:

```
<manifest ...

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

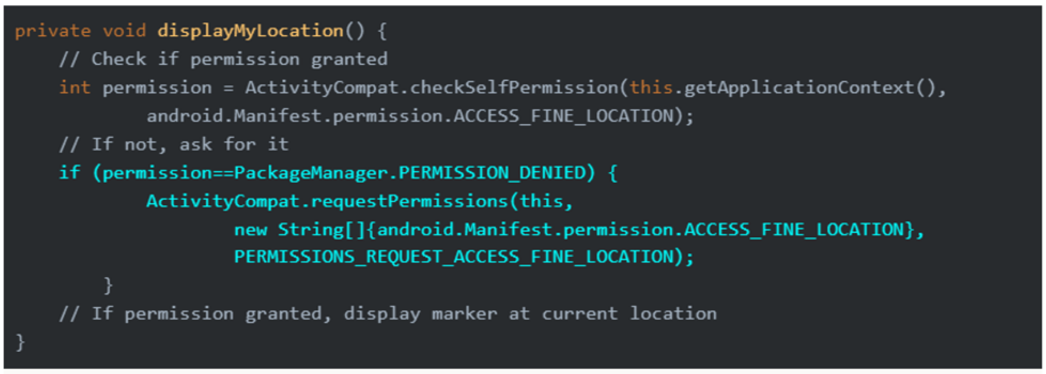
and check whether the permission is granted in your activity as shown:



```
protected void onCreate(Bundle savedInstanceState) {
        ...
        mapFragment.getMapAsync(googleMap -> {
        mMap = googleMap;

        // Add a marker at the destination, and move the camera.
        googleMap.addMarker(new MarkerOptions()
                     .position(mDestinationLatLng).title("Destination"));
        displayMyLocation();

    });
    ...
}
private void displayMyLocation() {
    // Check if permission granted
    int permission = ActivityCompat.checkSelfPermission(this.getApplicationContext(),
            android.Manifest.permission.ACCESS_FINE_LOCATION);
    // If not, ask for it
    // If permission granted, display marker at current location

}
```

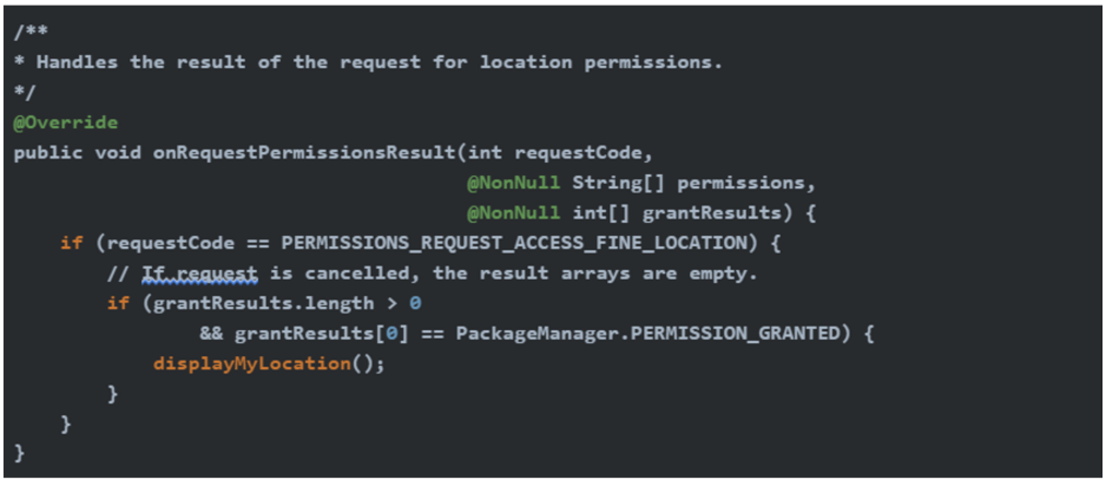
We're calling the new method called **displayMyLocation()** inside .getMapAsync() since the map needs to be ready first to be able to display our marker, and .getMapAsync() is called only after the map is ready.

3. If location permission isn't already granted, we need to ask for it during runtime (since it's a dangerous permission as listed [here (https://developer.android.com/reference/android/Manifest.permission#ACCESS_FINE_LOCATION)](https://developer.android.com/reference/android/Manifest.permission#ACCESS_FINE_LOCATION)) as shown in the code snippet below:



```
private void displayMyLocation() {
    // Check if permission granted
    int permission = ActivityCompat.checkSelfPermission(this.getApplicationContext(),
            android.Manifest.permission.ACCESS_FINE_LOCATION);
    // If not, ask for it
    if (permission==PackageManager.PERMISSION_DENIED) {
            ActivityCompat.requestPermissions(this,
                    new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                    PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
        }
    // If permission granted, display marker at current location
}
```

and call the method again if the permission's granted:

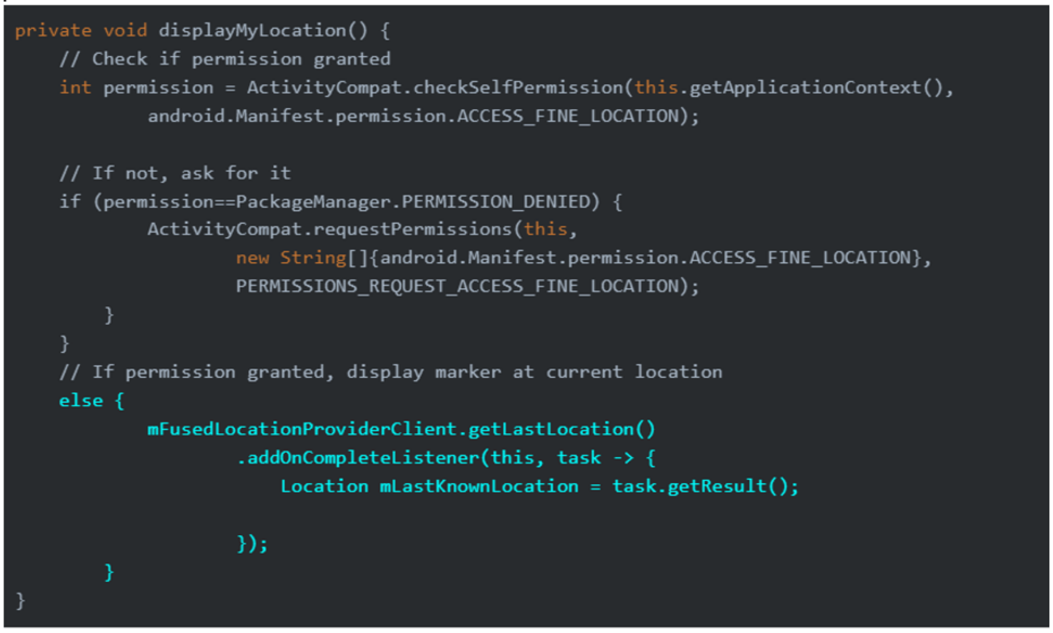


```
/**
* Handles the result of the request for location permissions.
*/
@Override
public void onRequestPermissionsResult(int requestCode,
                                       @NonNull String[] permissions,
                                       @NonNull int[] grantResults) {
    if (requestCode == PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION) {
        // If request is cancelled, the result arrays are empty.
        if (grantResults.length > 0
                && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            displayMyLocation();
        }
    }
}
```

PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION is just a constant defined in the activity to identify a request for this particular permission

```
public class MainActivity extends FragmentActivity {

    private static final int PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION = 12; // could've been
any number!
   ...
```

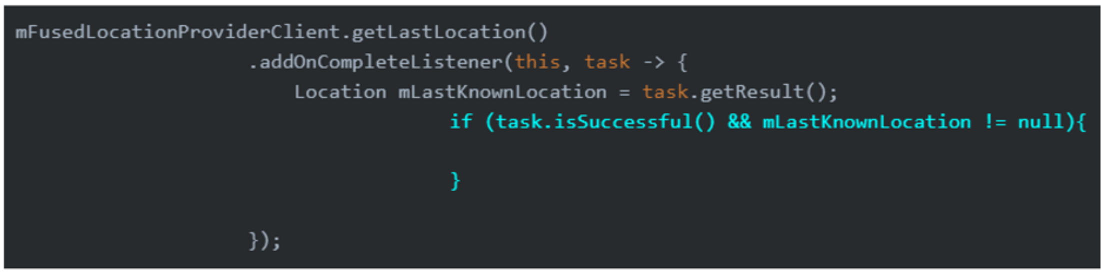
4. We're finally all set to query and display our location on the map:



```
private void displayMyLocation() {
    // Check if permission granted
    int permission = ActivityCompat.checkSelfPermission(this.getApplicationContext(),
            android.Manifest.permission.ACCESS_FINE_LOCATION);

    // If not, ask for it
    if (permission==PackageManager.PERMISSION_DENIED) {
            ActivityCompat.requestPermissions(this,
                    new String[]{android.Manifest.permission.ACCESS_FINE_LOCATION},
                    PERMISSIONS_REQUEST_ACCESS_FINE_LOCATION);
        }
    }
    // If permission granted, display marker at current location
    else {
            mFusedLocationProviderClient.getLastLocation()
                    .addOnCompleteListener(this, task -> {
                        Location mLastKnownLocation = task.getResult();

                    });
        }
}
```

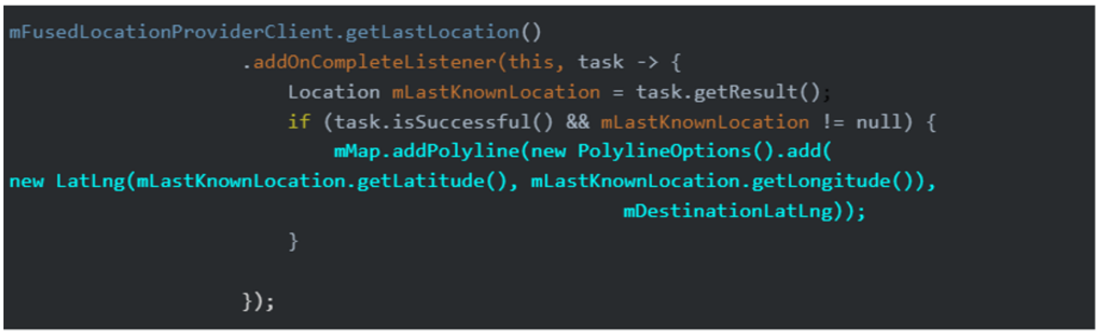
If that seems like a handful, that's because, well, it is. If not, feel free to do a victory lap around the room. In any case, remember, you're just setting up another notification on your location query. And so,**mLastKnownLocation** now stores the result of that query.

5. Before proceeding, let's just do a quick check to make sure the computation was successful:



```
mFusedLocationProviderClient.getLastLocation()
                    .addOnCompleteListener(this, task -> {
                        Location mLastKnownLocation = task.getResult();
                                    if (task.isSuccessful() && mLastKnownLocation != null){

                                    }

                    });
```

And then add a (poly)line between our obtained location and the marker we made in the last milestone



```
mFusedLocationProviderClient.getLastLocation()
                    .addOnCompleteListener(this, task -> {
                        Location mLastKnownLocation = task.getResult();
                        if (task.isSuccessful() && mLastKnownLocation != null) {
                            mMap.addPolyline(new PolylineOptions().add(
new LatLng(mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude()),
                                    mDestinationLatLng));
                        }

                    });
```

Deliverables to be shown to the instructors for Milestone 2:

1. make a marker situated at our current location. You should be able to do this with code you've already seen today. Having trouble? Here's a hint: Use the code written for the first milestone as well as mLastKnownLocation.getLatitude(), mLastKnownLocation.getLongitude(). Note that a Troubleshooting section has also been added at the bottom for nefarious errors.
2. show the polyline between our device location and the marker we made in the last milestone.
3. update our device location and show the change of the marker and the polyline.

Conclusion:

Nice job getting through this lab! The Google Maps API can do a lot more than we explored in this lab. You can use it to give directions, show cities in 3D, and cache maps offline. If you're interested in learning more, you can see the full API here: **Get Started | Maps SDK for Android (https://developers.google.com/maps/documentation/android-sdk/start)** (there's a tutorial there for automating most of the maps setup as well)

Video Option for Lab Checkoff:

In addition to the in-person (during class) and virtual (TA office hours) check-off, we will be adding a video option for lab checkoff. The video option is for you to do a screen recording of the workings of your app with your voice explaining what you are doing. These videos will need to be submitted via Canvas.

Here is the video instruction for lab 6:

Note:

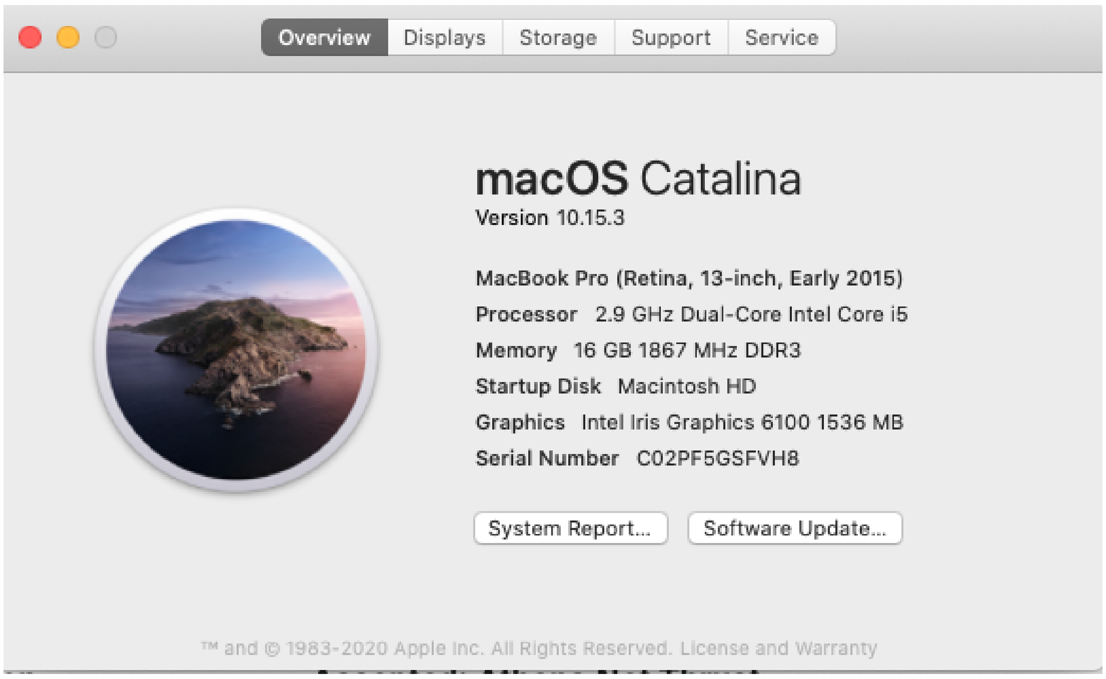
1. Please do the screen recording as a single video and definitely with voice over. It should not be multiple videos that are edited and stitched together.

2. We recommend using Zoom's screen recording feature to capture the video, but you can do so by other means as well.

3. Please include a link your Github repos in the provided text box

4. Device identification:

For mac, go to the Apple icon in the top left, and click "About this Mac".

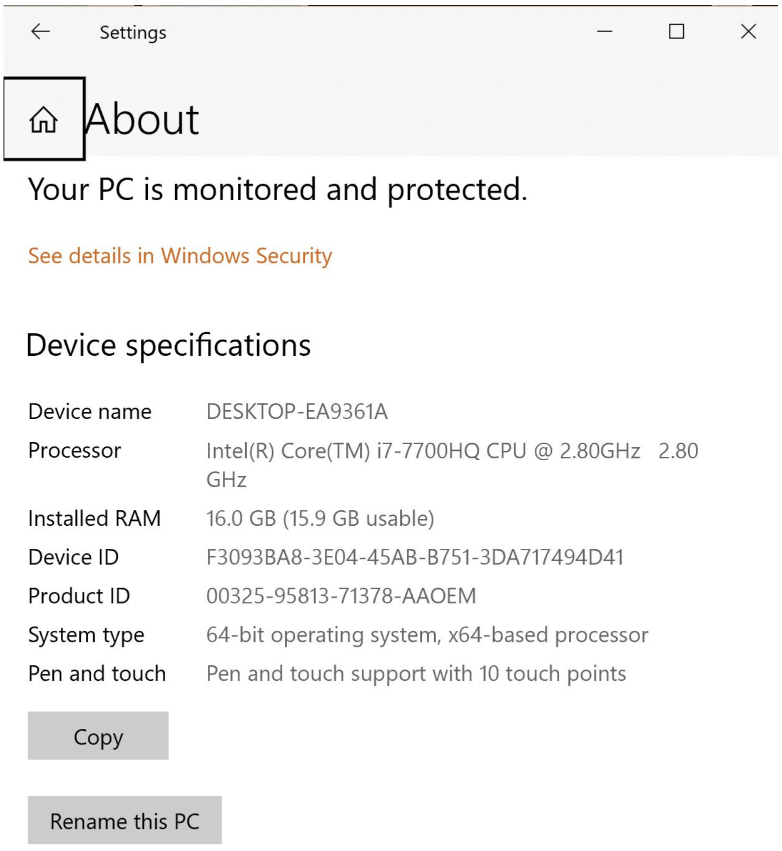


macOS Catalina
Version 10.15.3

MacBook Pro (Retina, 13-inch, Early 2015)
Processor 2.9 GHz Dual-Core Intel Core i5
Memory 16 GB 1867 MHz DDR3
Startup Disk Macintosh HD
Graphics Intel Iris Graphics 6100 1536 MB
Serial Number C02PF5GSFVH8

™ and © 1983-2020 Apple Inc. All Rights Reserved. License and Warranty

For windows, simply search 'About your PC' in the bottom left search bar



Your PC is monitored and protected.

See details in Windows Security

Device specifications

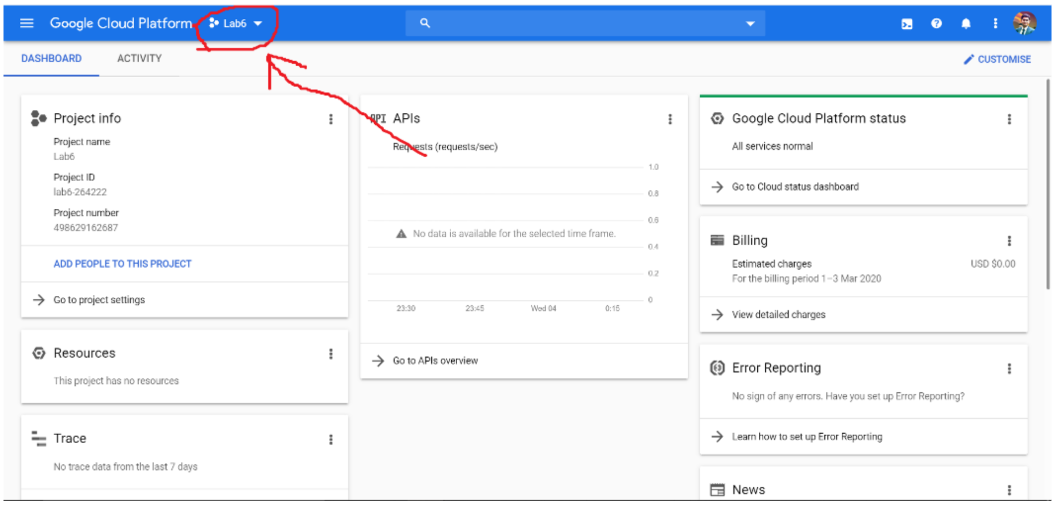
Device name	DESKTOP-EA9361A
Processor	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.80 GHz
Installed RAM	16.0 GB (15.9 GB usable)
Device ID	F3093BA8-3E04-45AB-B751-3DA717494D41
Product ID	00325-95813-71378-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	Pen and touch support with 10 touch points

Troubleshooting:

- If your app crashes when you run it and you're past Step 1 of Milestone 2, conflicting dependency versions might be the issue. Go to File > Project Structure > Dependencies > App and make sure the 2 play-services-something libraries have the same version by selecting them. If they don't, assign both of them the higher version of the two.
- If Android Studio issues an error on using Lambdas (‘lambda expressions are not supported in -source 1.7 (use -source 8 or higher to enable lambda expressions)’), hover your mouse over the erratic code and press Enter + alt (or Enter + the Option key on Macs). That should give you an option to change your language level to 8. Choose it, and that should fix the error
- Error: Ensure that the “Google Maps Android API v2” is enabled
  - It seems you may have to go enable Maps Android API manually:

1. Login to google developer console at **https://console.cloud.google.com/ (https://console.cloud.google.com/)**

2. Ensure that the project you created before is selected:



Project info
Project name
Lab6
Project ID
lab6-264222
Project number
498629162687
ADD PEOPLE TO THIS PROJECT
Go to project settings

Requests (requests/sec)
No data is available for the selected time frame.
Go to APIs overview

All services normal
Go to Cloud status dashboard

Estimated charges USD $0.00
For the billing period 1 – 3 Mar 2020
View detailed charges

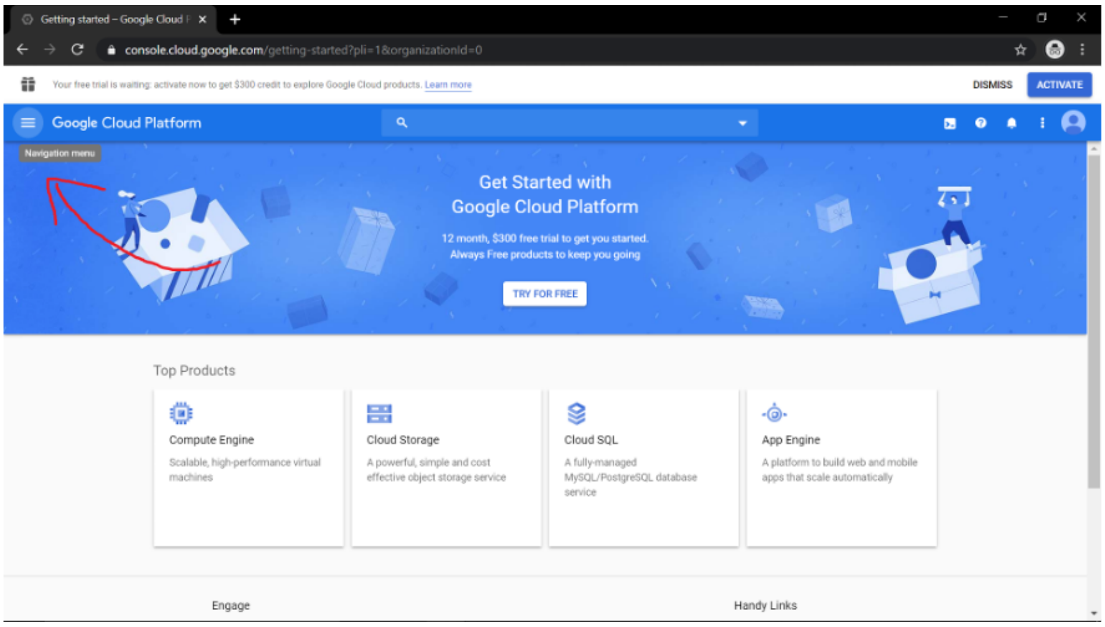
No sign of any errors. Have you set up Error Reporting?
Learn how to set up Error Reporting

This project has no resources

No trace data from the last 7 days

(Note that your project name could be different)

3.Click the menu button:


Getting started – Google Cloud P
console.cloud.google.com/getting-started?pli=1&organizationId=0
DISMISS
ACTIVATE
Google Cloud Platform
Navigation menu
TRY FOR FREE
Top Products
Engage
Handy Links

Your free trial is waiting: activate now to get $300 credit to explore Google Cloud products. Learn more

Get Started with Google Cloud Platform
12 month, $300 free trial to get you started.
Always Free products to keep you going

Compute Engine
Scalable, high-performance virtual machines

Cloud Storage
A powerful, simple and cost effective object storage service

Cloud SQL
A fully-managed MySQL/PostgreSQL database service

App Engine
A platform to build web and mobile apps that scale automatically

4. Tap APIs & Services -> Library

5. Click ‘Show All’ next to the Maps APIs

6. Click Maps SDK for Android -> Enable

References:

- Google's sample app that shows how to obtain device's location
- https://www.bignerdranch.com/blog/embedding-custom-views-with-mapview-v2/ (https://www.bignerdranch.com/blog/embedding-custom-views-with-mapview-v2/)

lab 6 rubric

Criteria	Ratings		Pts
Milestone 1	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
Milestone 2	**5 pts** **Full Marks**	**0 pts** **No Marks**	5 pts
			Total Points: 10