CS407 Fall 2023

Pair Programming: **ALLOWED**
Due: **12:00PM CT on November 6th**

Lab06 APIs and Maps

1 Overview

This lab is designed to walk you through integrating external APIs into your Android application. This knowledge will prove invaluable if you plan to incorporate external services or data into your projects. Specifically, we'll focus on the integration of Google Maps API. This lab teaches you how to generate API keys tailored for Google Maps. These keys are essential for setting up a `SupportMapFragment`, a specialized fragment used for displaying maps on Android, akin to how `AlertDialogFragments` are employed for displaying alerts. Additionally, you'll gain proficiency in retrieving the device's location and plotting a route between your current location and any other point on the map.

- In **Milestone 1**, Setting Up Maps Activity with Marker at Bascom Hall
- In **Milestone 2**, Adding Another Marker at Current Location with a Polyline

Learning Objectives

By completing this lab, a cs407 student will be able to:

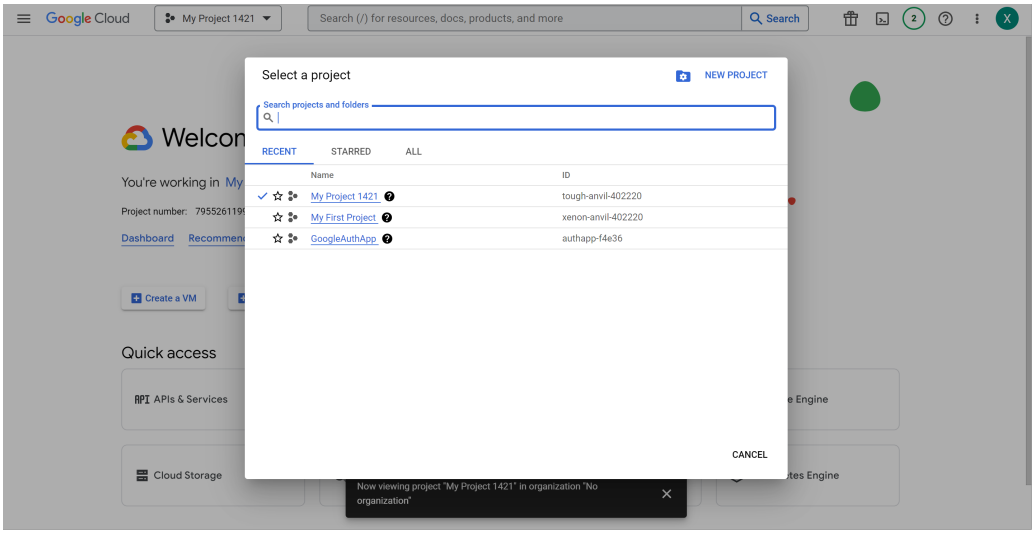
1. **Understand** the process of obtaining a Google Maps API key.
2. **Set up** a new Android project and **configure** it for Google Maps API integration.
3. **Implement** the necessary dependencies and permission in the Android project to support the Google Maps APIs integration.
4. **Display** maps on Android and **customize** its appearance.
5. **Access** and **retrieve** the device's current location using location services.
6. **Plot** a route between the current location of the device and another point on the map.

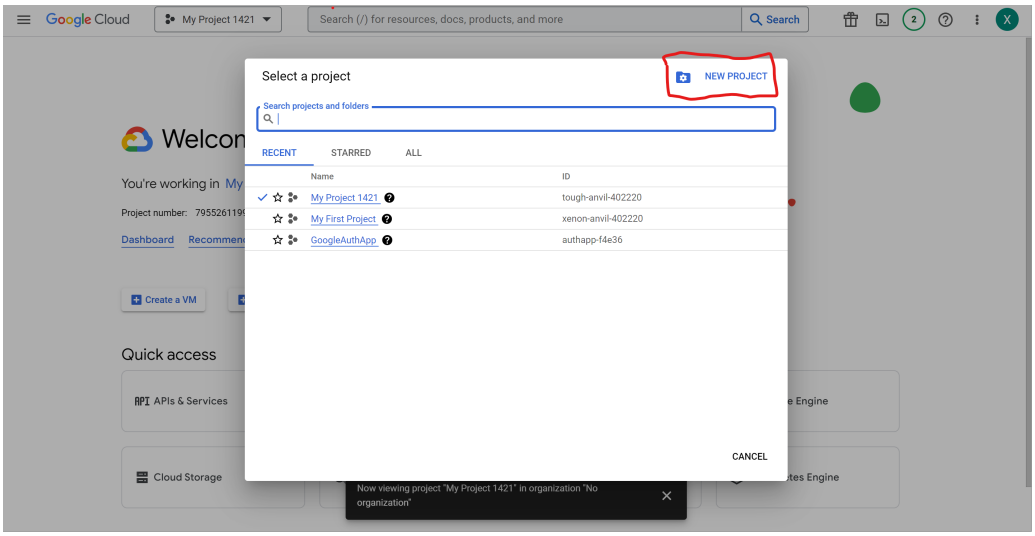
2 Prerequisites

Before we dive in, make sure you have completed the following steps:

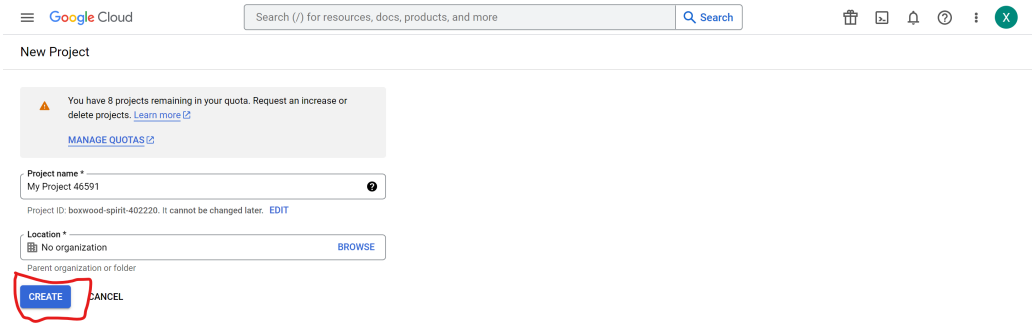
1. **Navigate to Google Cloud Platform Console:** Open a web browser and go to .
2. **Accept Terms and Services:** If prompted, review and accept the terms and services. Don't worry, you can read through them if you'd like!

3. **Select or Create a New Project:** Click the 'Select a Project' drop-down, which you'll find as indicated in the screenshot below.

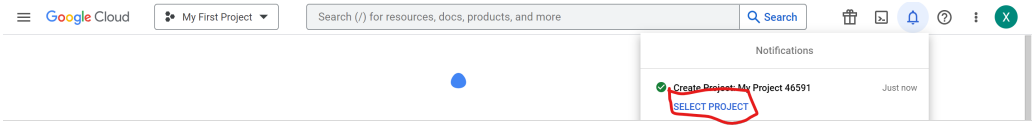




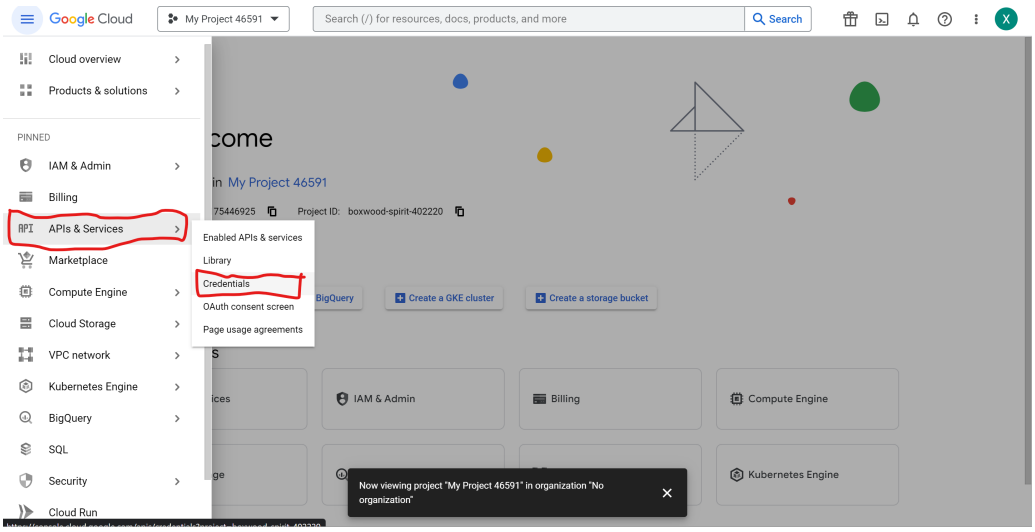
4. **Configure Project Settings:** You can stick with the default settings that appear. Take note of the project name that's being created and then click 'Create'.



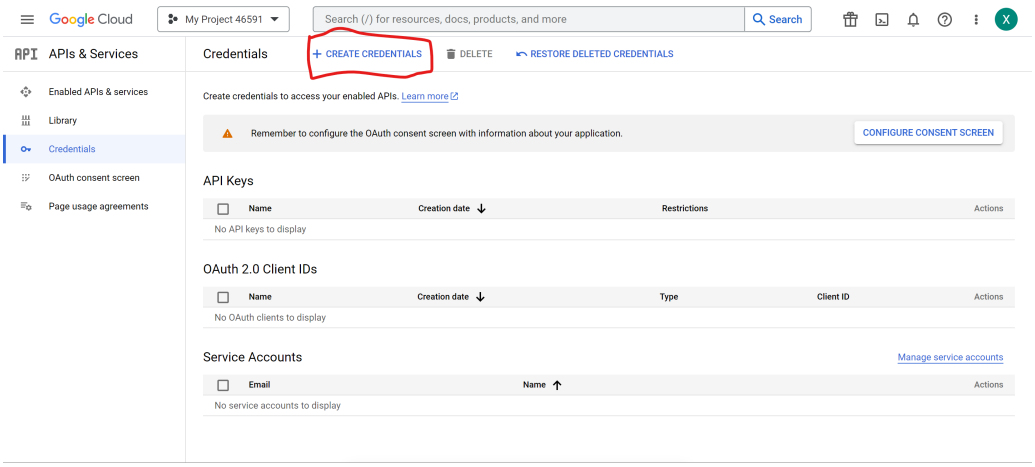
5. **Access Project Menu:** Once the project is successfully created, locate the menu button, typically represented by three dots (as shown below)



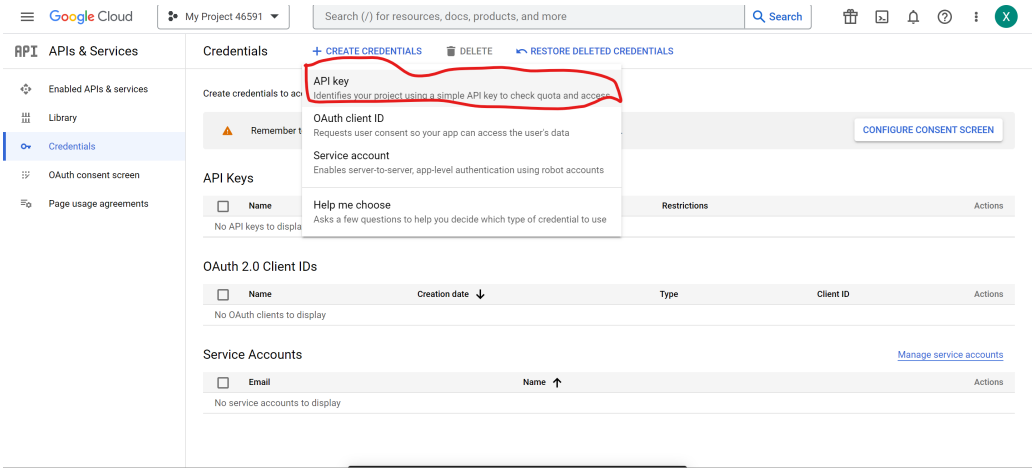
6. **Navigate to APIs & Services Credentials:** Select 'APIs & Services' and then 'Credentials'. If your credentials page doesn't look similar to the screenshot shown below, you may have to tap 'Select Project' and select the project you created in Step 5.



7. **Generate API Key:** On the Credentials page, click 'Create credentials' → 'API key'.



A dialog will appear displaying your newly minted API key. This is crucial, so make sure to take note of it as you'll be using it in your project.



8. **Close the Dialog:** After noting down the API key, simply click 'Close'.

Directions

Milestone 1 - Setting Up Maps Activity with Marker at Bascom Hall

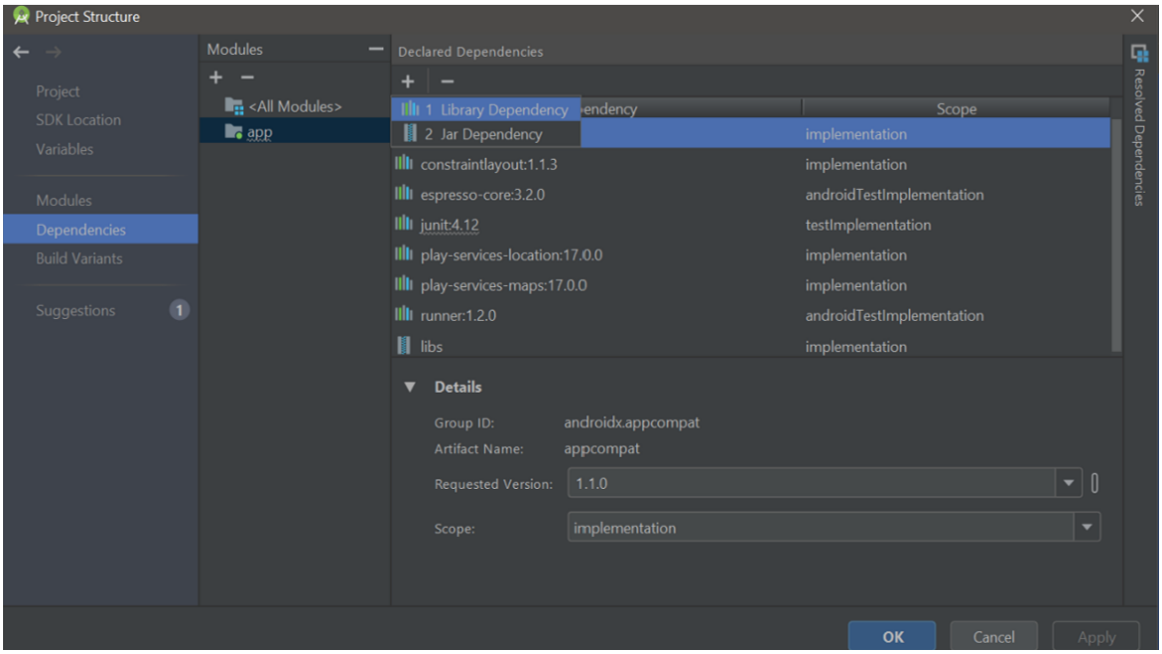
- **Ensure Required Files:** Make sure you have two essential files:
  - MainActivity.java
  - Associated layout XML file (usually named activity main.xml)

- **Add SupportMapFragment:** In the layout XML file, remove its existing content and add a SupportMapFragment.

– 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.

```
<?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-->
```

– 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.



- **Add Dependencies:** To import the dependency, go to File → Project Structure → Dependencies and add a new Library Dependency in the app module, as shown above Ensure the latest version is selected and press OK.

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

- **Update MainActivity:** Make sure your MainActivity.java is referencing the layout XML and retriev- ing the SupportMapFragment. The code to be added is highlighted 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);
    }
}
```

- **Insert API Key:** Open Manifest.xml and add the following, replacing YOUR API KEY with your actual key

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

- **Initialize GoogleMap:** Go back to your MainActivity.java. Ensure that the MapFragment is down- loaded and initialized.

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

- **Display a Marker:** Use .getMapAsync(..) to be notified when the MapFragment is ready, and then create your markers.

- **Save GoogleMap Object:** Save the GoogleMap object obtained in a variable (mMap) for later use.

```
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
        });
}
```

- **Define Variables:** Define the mMap variable along with another variable representing the latitude and longitude of the marker.

```
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) {
        ...
    }
}
```

- **Create a Marker:** To create a new marker, use:

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

In the above call, .position determines where the marker is placed. The Lat Lng object created earlier will be used to set the marker's location. Destination is the text displayed when you tap the marker.

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

DELIVERABLES:

- Change the destination marker to be situated at Bascom Hall (lookup on Google Maps for the required location parameters).
- Show the marker on the AVD/device.

Milestone 2: Adding Another Marker at Current Location with a Polyline

- **Obtain User's Location:** To obtain the user's location, we'll use the FusedLocationProviderClient. This class provides location data using a combination of GPS and network towers for accuracy and battery efficiency. Add the following code to your MainActivity.java

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

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

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

- **Add Dependency for Location Services:** Navigate to File → Project Structure → Dependencies. Add the following library for the app module: Choose the latest version and press OK.

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

- **Request Location Permission:** Since we're accessing the user's location, we need to ask for permission at run- time. 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>
```

Check whether the permission is granted in your activity

```
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

}
```

- **Display User's Location:** Display the user's location by querying the FusedLocationProviderClient:

```
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
}
```

```
/**
* 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!
   ...
```

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,my Last Known Location now stores the result of that query.

- **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:

- Show the polyline between the obtained location and the marker created in the last milestone.
- Add a marker situated at the current location using the code from the first milestone along with mLastKnownLocation.getLatitude() and mLastKnownLocation.getLongitud If you encounter any issues, refer to the Troubleshooting section provided.

Conclusion

Congratulations on completing this lab! You've covered a basic introduction to Google Maps API, but keep in mind that it can do much more. You can use it for tasks like providing directions, rendering cities in 3D, and even caching maps for offline use. If you're eager to learn more, check out the full API documentation here.

Troubleshooting

- **Conflicting Dependency Versions:** If your app crashes after completing Step 1 of Milestone 2, it might be due to conflicting dependency versions. To resolve this, navigate to **File** > **Project Structure** > **Dependencies** > **App**. Make sure that both play-services-something libraries have the same version. If they don't, assign both of them the higher version.
- **Lambda Expression Error:** If Android Studio throws an error about Lambda expressions ('lambda expressions are not supported in -source 1.7...'), hover your mouse over the problematic code and press Enter + Alt (or Enter + Option on Macs). This will provide an option to change your language level to 8. Choose it, and this should resolve the error.
- **Error:** Ensure "Google Maps Android API v2" is Enabled: If you en- counter this error, it means you may need to enable the Maps Android API manually. Follow these steps:
  - Log in to the Google Developer Console at https://console.cloud.google.com/.
  - Make sure the correct project is selected.
  - Click the menu button.
  - Navigate to APIs and Services Library.
  - Click 'Show All' next to the Maps APIs.
  - Select Maps SDK for Android and click Enable.

Remember, troubleshooting is a normal part of development. Don't hesitate to refer to documentation or seek help from the community if you encounter any challenges. Happy coding!

Gradescope Submission

Congratulations on finishing this CS407 Lab! The only ONE file that you must submit to Gradescope is:

- `Milestone 1's activity_main.xml`.

Your score for this assignment will be based on your **"active"** submission made prior to the deadline of Due: **12:00PM CT on November 6**th.

For group submissions to gradescope, be SURE to follow the instructions given in the **2. Second Step: Add the other group members to that submission** section provided on the How To Make a Group Submission page on canvas.

The second portion of your grade for this lab assignment will be manually graded by our TAs/graders (Demo sessions for **both milestones**). You MUST submit a grading request first! Penalties to your lab's grade are applied if your lab is not manually graded before **5:00PM CT on November 6**th.