Exercise 7: Map & Location

Due: Apr. 25, 23:59 PDT

Overview

In this exercise, you will build an app (Fig. 1) that allows users to get the current location and switch between different map views. It also demonstrates how to use a floating action button (FAB).

When creating the project, select "Google Map Activity". Name the project as "Exercise7YourName".

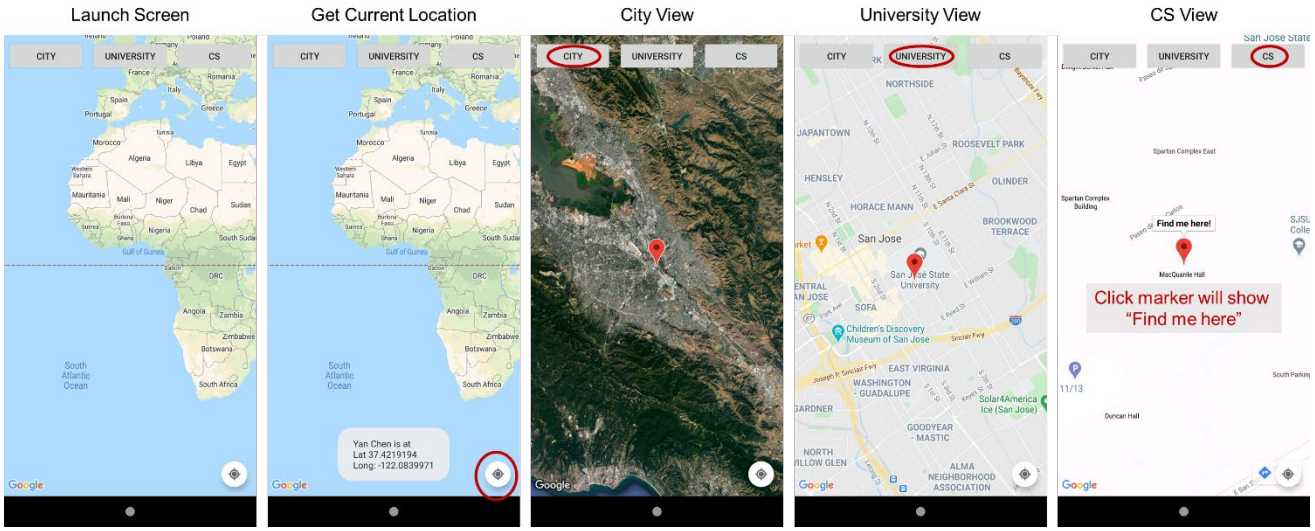

Fig. 1 Sample run of the app

Prerequisites:

- Know the process of submitting your work (exercise 0)
- Be familiar with previous topics related to project structure and UI (exercise 1 - 4, lesson 2 - 4)
- Has set up an emulator (API 29)

Procedure related to the above topics will not be provided in the instruction. Refer to corresponding exercise/lecture notes if needed.

If you set any different view id's, filenames, etc., remember to modify the corresponding part of code.

Step 0. Set up project

0.1 Set Google Map API

After creating the project with a Google map activity, you will be directed to google_maps_api.xml file. Follow the steps in the comment to add a Google API key, or watch this video if needed. Note that the last step in the video seems unnecessary now but no harm to add it.

0.2 Request permissions

Request following permissions in the manifest file.

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

0.3 Add Dependency

We will use Google Location Service to get the current location. So add the following dependency in the module level build.gradle file and click sync. We won't use binding so there's no need to set it.

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

Step 1. Setup UI

In this app, we will have a floating action button (FAB) for displaying the current location. A FAB is a button that appears as a circle with an icon "floating" on the screen. It is still a button, so just set it as you would set a button.

To add a FAB, you can either search for "FAB" in palette of the design mode and drag it onto the screen, or add a "com.google.android.material.floatingactionbutton.FloatingActionButton" element in the code (if you type "<FAB", it should appear in the dropdown menu). When adding a FAB, you need to choose an icon. For this exercise, search for the "location" icon. In the code, it should be android:src="@android:drawable/ic_menu_mylocation". You can also set its background color by android:backgroundTint attribute, set its border by app:borderWidth, or/and set its size (mini or normal) by app:fabSize attribute. Feel free to explore other attributes for a FAB.

Here are all the elements you need to add. As usual, the position does not matter. Note for the fragment, set android:name="com.google.android.gms.maps.SupportMapFragment".

Widget	id	Text	onClick
Button	city	"City"	switchView
Button	univ	"University"	switchView
Button	cs	"CS"	switchView
FAB	n/a	n/a	getLocation
Fragment	map	n/a	n/a

Step 3. Implement GPSTracker

Create a normal Java class called “GPSTracker” for getting the current location.

3.1 Set a Constructor

We need a Context object for various tasks in this class. Therefore, set a Context object named “context” as a class attribute and initialize it with the argument passed in the constructor. That is all you need for the constructor.

3.2 Check if Location Service is Enabled

User may disable the location service. If that’s the case, ask user to enable it.

First, implement a private helper method to check if location is enabled using LocationManager.

```
private boolean isLocationEnabled() {
    LocationManager manager = (LocationManager)
            context.getSystemService(Context.LOCATION_SERVICE);
    return manager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
            manager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
}
```

Then, implement a private helper method to show settings alert dialog.

```
public void showSettingAlert() {
    AlertDialog.Builder alertDialog = new AlertDialog.Builder(context);
    alertDialog.setMessage("Please enable location service.");
    alertDialog.setPositiveButton("Enable", (dialog, which) -> {
        Intent intent =
                new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
        context.startActivity(intent);
    });
    alertDialog.setNegativeButton("Cancel", (dialog, which) ->
            dialog.cancel());
    alertDialog.show();
}
```

3.3 Check Permission and Request Permission

We also need to request permission for accessing location at runtime, since getting the location is considered as a “dangerous” permission, similar to accessing external storage in exercise 5.

First, implement a private helper method to check if “access location” permissions are given.

```
private boolean checkPermission() {
    int result1 = ActivityCompat.checkSelfPermission
            (context, Manifest.permission.ACCESS_FINE_LOCATION);
    int result2 = ActivityCompat.checkSelfPermission
            (context, Manifest.permission.ACCESS_COARSE_LOCATION);
    return result1 == PackageManager.PERMISSION_GRANTED
            && result2 == PackageManager.PERMISSION_GRANTED;
}
```

Also, implement a private helper method to request the permission, where 100 is the permission code.

```
private void requestPermission() {
    ActivityCompat.requestPermissions((Activity) context,
            new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
                    Manifest.permission.ACCESS_COARSE_LOCATION}, 100);
}
```

3.4 Get Current Location

Devices contain various types of GPS hardware, and activating GPS uses power. Therefore, to get the location information efficiently, we can use FusedLocationProviderClient interface provided by Google Location Service API. It provides a simplified method to get location called “getLastLocation”, which returns the best accuracy available while respecting the location permissions. Note that this method returns a Task that results in a Location object, and it is called after the Task's onSuccess callback method is called to indicate that the Task was successful. So, let’s first implement GPSTracker as a OnSuccessListener<Location> interface and implement the onSuccess call back.

Remember to change the corresponding part of toast message to your name.

```
@Override
public void onSuccess(Location location) {
    if (location != null) {
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        Toast.makeText(context, "Your Name is at \n" +
                        "Lat " + latitude + "\nLong: " + longitude,
                Toast.LENGTH_LONG).show();

    } else
        Toast.makeText(context, "Unable to get location",
                Toast.LENGTH_LONG).show();
}
```

Finally, we can implement the method to get the current location.

```
public void getLocation() {
    FusedLocationProviderClient provider =
            LocationServices.getFusedLocationProviderClient(context);
    if (!isLocationEnabled()) showSettingAlert();
    else if (!checkPermission()) requestPermission();
    else provider.getLastLocation().addOnSuccessListener(this);
}
```

Step 3. Implement MapsActivity

The activity class generated for you is called MapsActivity. Add two class attributes for the location of SJSU and CS department (hardcoded). There is no need to change the onCreate method.

```
private final LatLng LOCATION_UNIV = new LatLng(37.335371, -121.881050);
private final LatLng LOCATION_CS = new LatLng(37.333714, -121.881860);
```

4.1 Implement onMapReady

See the auto-generated comments for what this method does. In this exercise, instead of setting a marker in Sydney, let's set the marker to the CS department.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    mMap = googleMap;
    mMap.addMarker(new MarkerOptions().
            position(LOCATION_CS).
            title("Find me here!"));
}
```

4.2 Implement getLocation

As set in Step 2, the method responds to a click on the FAB is "getLocation", which calls the getLocation method we set in Step 3.4 of a GPSTracker object. Recall that an Activity is a context.

```
public void getLocation(View view){
    GPSTracker tracker = new GPSTracker(this);
    tracker.getLocation();
}
```

4.3 Implement switchView

This is the method to respond clicks on other buttons to switch between different views. Check here for the available types of map if interested. The CameraUpdateFactory.newLatLngZoom method takes two parameters, the first one is the location focused on, and the second one is the zoom level.

```
public void switchView(View view) {
    CameraUpdate update = null;
    if (view.getId() == R.id.city) {
        mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
        update = CameraUpdateFactory.newLatLngZoom(LOCATION_UNIV, 10f);
    } else if (view.getId() == R.id.univ) {
        mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        update = CameraUpdateFactory.newLatLngZoom(LOCATION_UNIV, 14f);
    } else if (view.getId() == R.id.cs) {
        mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
        update = CameraUpdateFactory.newLatLngZoom(LOCATION_CS, 18f);
    }
    mMap.animateCamera(update);
}
```

Step 5. Test your app

Here are some potential errors/problems you may encounter and how you may solve them.

Map Not Showing

If the map is blank, it's possible that the Google API key is not set correctly. Open the "run" tab check for the message showed in Fig. 2. Go to the green boxed link, switch to "Credentials" tab and edit the existing key. The information in the two red boxes should match. Otherwise, click to edit the API restrict usage for the key so both credential and package information match your app.

If you have not set any key, redo step 0.1.

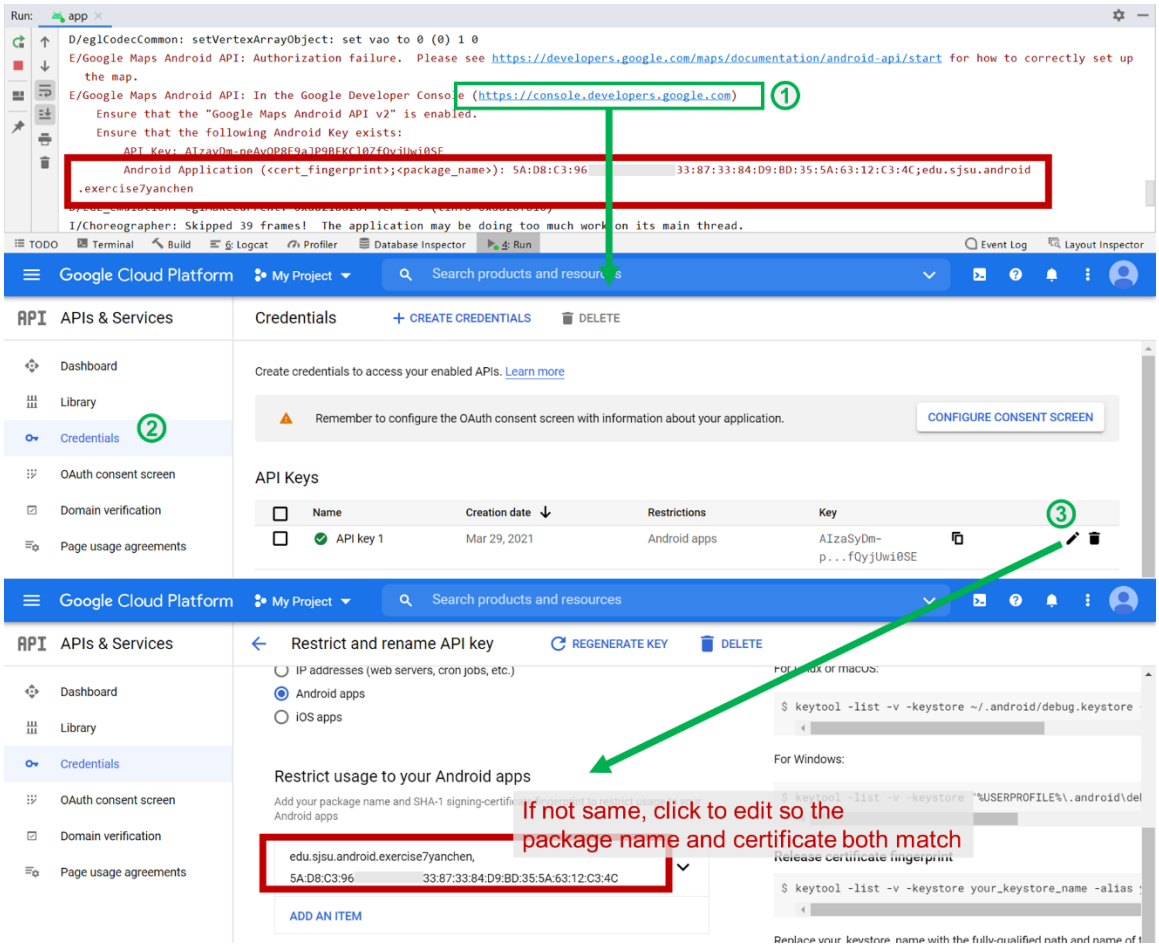

Fig. 2 Check and set information if not match

Unable to Get Location

If after enabling location and giving the permission, it still showed "Unable to get location", try to set the location in the emulator (Fig. 3), or/and turn off the location and turn it back on (Fig. 4). Or change to another device that enables "Google APIs" or "Google Play".

Note that depends on your emulator/Android studio version, setting a location maybe different, but you should be able to either input or select a location. Also, you may need to find out how to go to the settings on your emulator (based on the brand/model of the device you use).

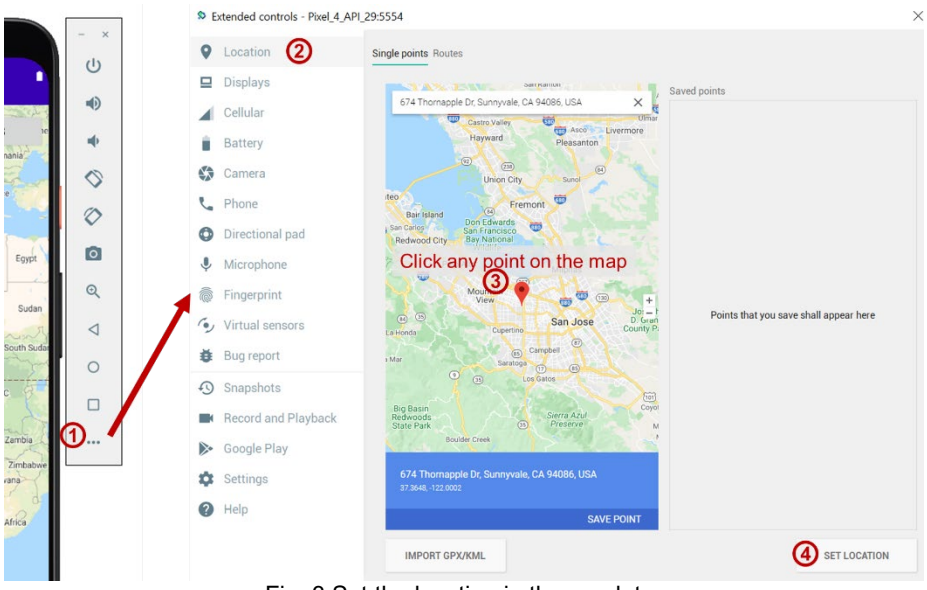

Fig. 3 Set the location in the emulator

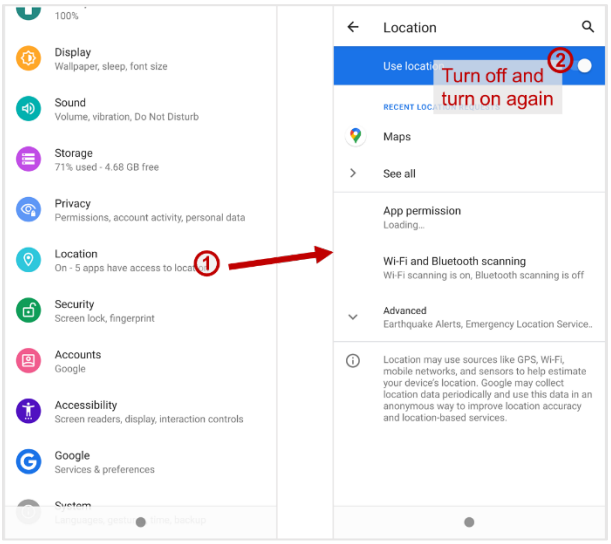

Fig. 4 Turn off and turn on location again

Submission

- Push your project to a Bitbucket repository (name it “exercise7”) by the due date.
- Invite and share your Bitbucket repository the grader (edmond.lin@sjsu.edu) and the instructor (yan.chen01@sjsu.edu).
- Submit repository links, etc. by answering all the questions in the “Exercise 7 - Map & Location” quiz on Canvas.
- Only your last submission before deadline will be graded based on the following criteria:
  2 pts if meets all requirements;
  1 pt if app failed/missing any requirement.