Tutorial on Google Map && GPS

EECS397/EECS600, Mobile Computing && Sensor Network, Fall 2015

Advisor: Prof. Ming-Chun Huang

1. Introduction:

In this project, you will learn to build an app to capture photos with existing camera application and track the device's location with location services. The photo and location information are very useful for information retrieval. For example, if we take a photo of buildings, and search the photo and your current location from the internet or a built database, we may obtain some relevant information, such as building's name, suggested parking lot around it, opening hours or short description of this building. Such app will provide us many interesting information and make your life more convenient. This is just an example. You can implement camera and location service to create many fancy functions in your own apps. With this tutorial, you will be familiar with google map API access, camera control, GPS data manipulation and image overlay on map.

Disclaimer: This tutorial is for educational purpose only and it is not a 100% step by step tutorial, which means students still need to finish some parts by themselves. It is intended for students who would like to quickly overview the Google Map API in Android. Without considering security and reliability, according to this tutorial, students can build a fast-working prototype for their class project.

2. Pre-requisite:

1. EECS 132 and EECS 233
2. Android Studio (recommended).

3. Homework Overview:

Apply for the google map API key first (part I), then you can import the codebase using API key. Revise the codebase to build your own app and test your app by taking photos for at least ten different buildings on or around the campus. Overlay the recorded photos on the map. The final result is a map with overlayed images and snippets (see Figure 3). Write a report to describe your work. Please submit your entire Android project and report in a zipped file, and name it as "Lastname1_Lastname2.zip".

Your app should have a similar user interface as Figure 1 and could achieve the following functions:

1. When the user clicks "TAKE A PICTURE", he will enter an interface of camera (see Figure 2). He could take a photo and save it (Part II).

2. When a photo is saved, your app also saves the photo's information, such as time stamp and GPS coordinates (Part III).

3. After saving the information and photos, by clicking cross button, the interface could return to the map and automatically overlay the photos as thumbnails (MARKERS) on the map according to the GPS coordinates (Part IV).

4. Once a thumbnails is clicked, a textbox will show up and the user should be able to change the title and snippet (Part V).

Figure 1

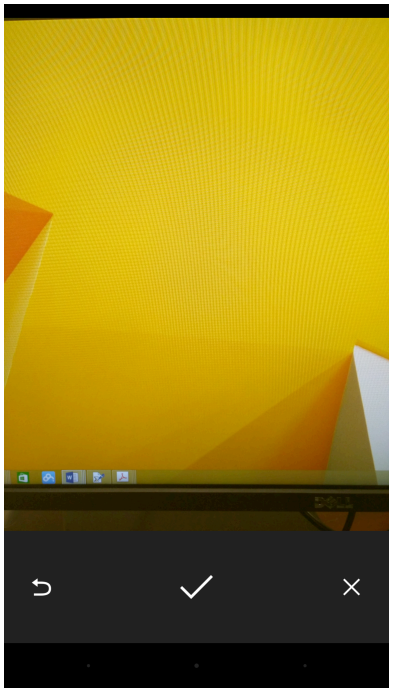


Figure 2

Figure 3

Report: You should write a 2-3 page report to describe your code and work for homework 1-4. There is no standard format of the report, but please include these contents:

1. Code explanation: explain your source code in detail, describe how to use your app and tell us your save path and file name.

2. Experiment result: take at least 10 photos of builds. Please show the screenshot of map with thumbnails and CSV file, and give a brief explanation. In total, you should have at least ten markers on the map and at least ten lines in CSV file. If you do not finish the project, please explain the reason.

3. Division of work: what's your contribution and your partner's contribution.

4. Project description

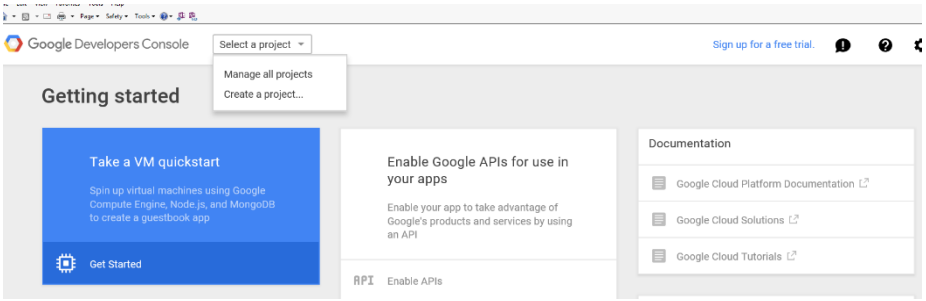
If you import the given project successfully, you will see the google map with a button and a normal marker is displayed on the map too. You need to implement the camera by yourself. In addition, you also need to locate to Case campus and access the GPS data by yourself. In the end, you can do the image overlay (use thumbnail of the photo as marker).

Part I: Accessing the Google Map Android APIs

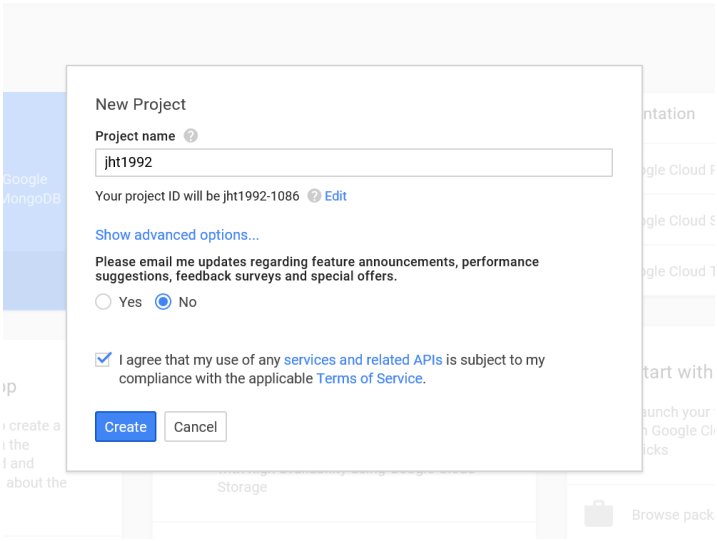
All Google Map Android APIs applications require authentication using an API key. To get you started with this tutorial, we will guide you through https://console.developers.google.com to do a few things first.

1. Create or choose a project.
2. Activate the Google Maps Android API
3. Create appropriate keys

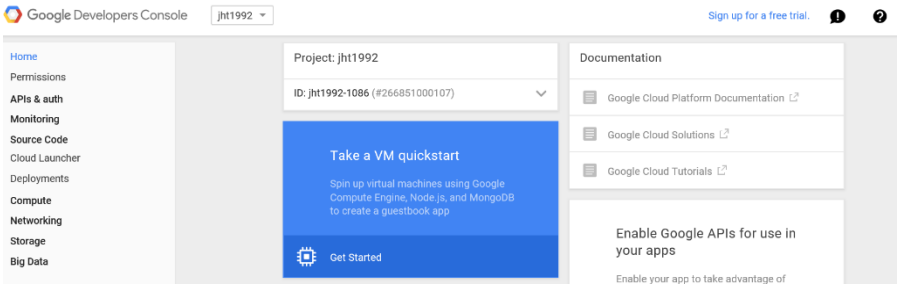
Go to the Google Developers Console and create a project for you.



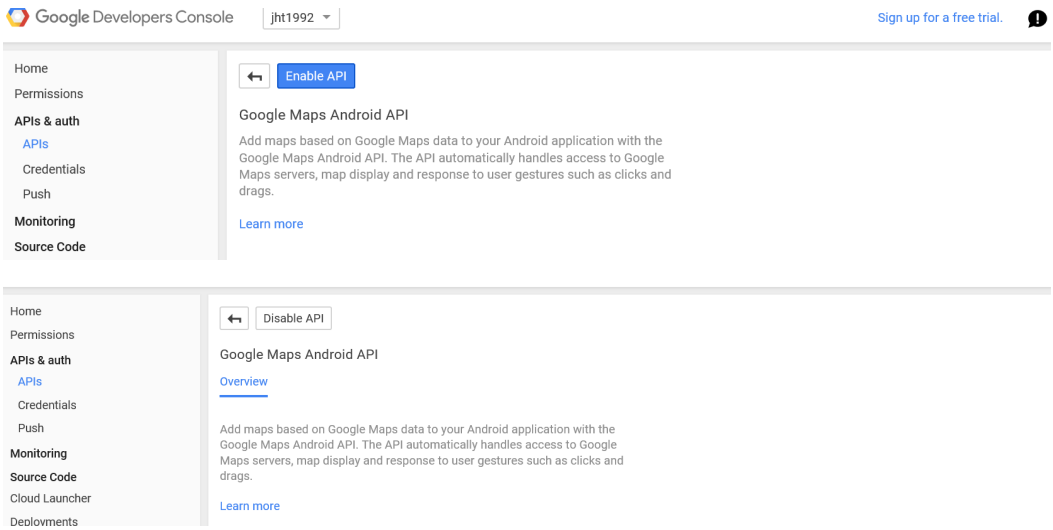
Setting up as the following figure.



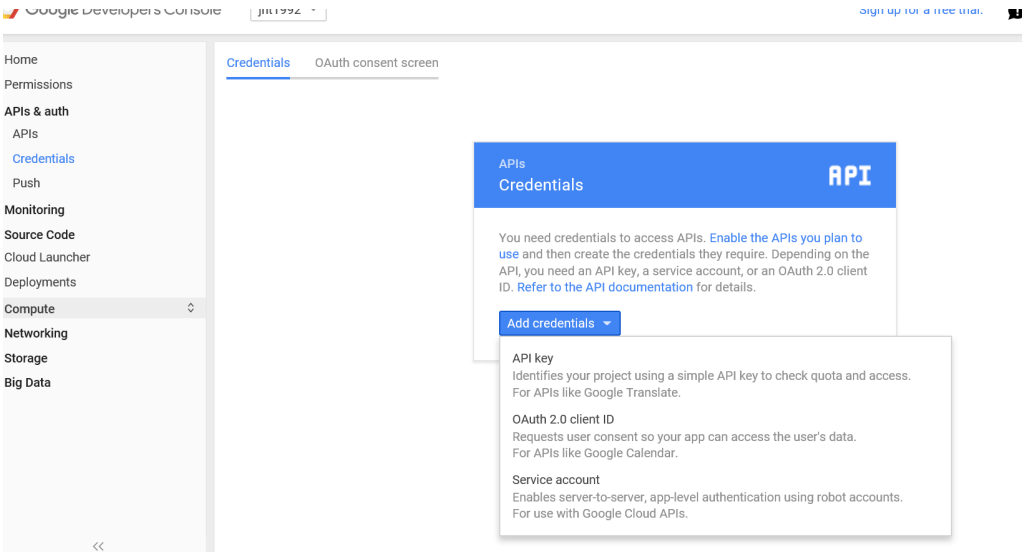
Then, click “Create” and you should see this page.



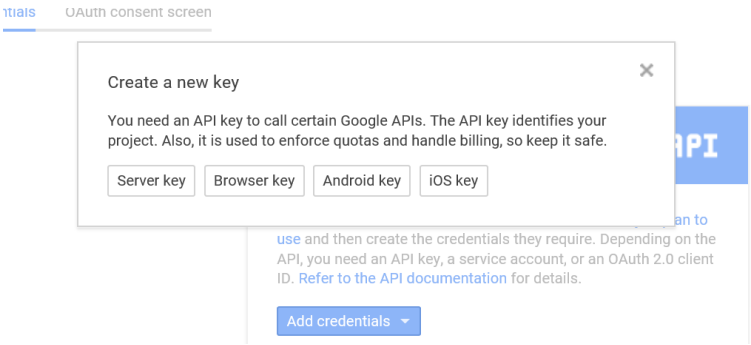
In the left panel, choose APIs&auth/APIs/Google Maps Android API and click “Enable API”.



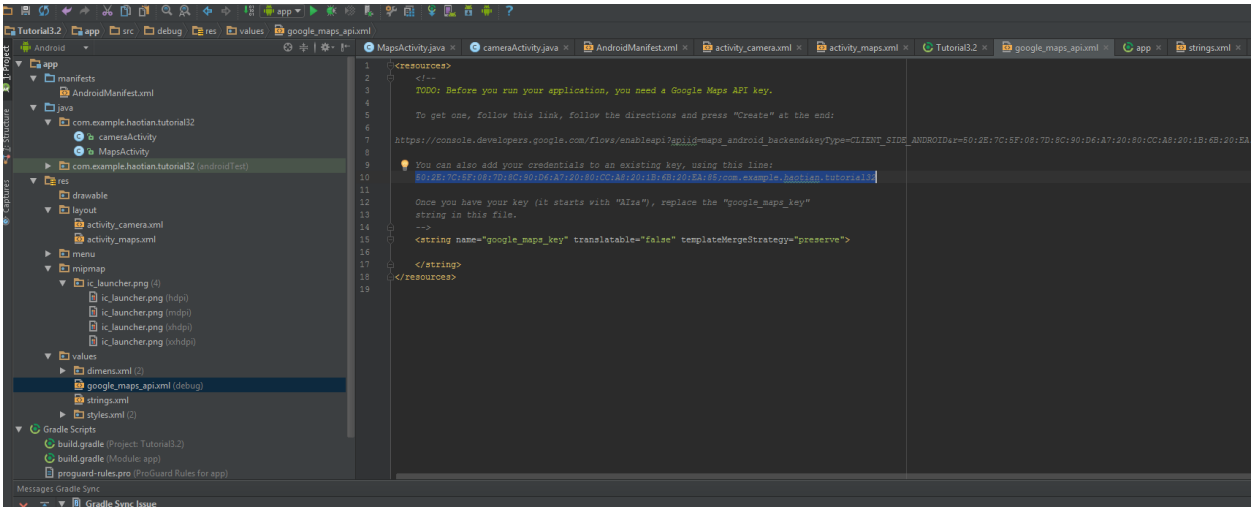
Then, choose Add credentials/API key as following figure.



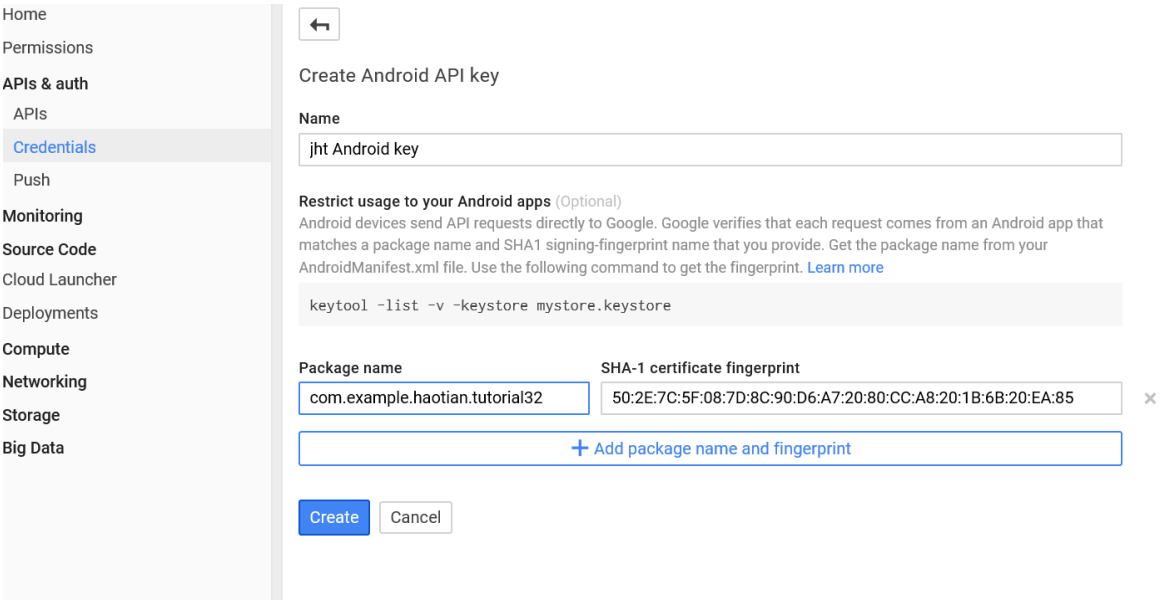
Choose Android key.



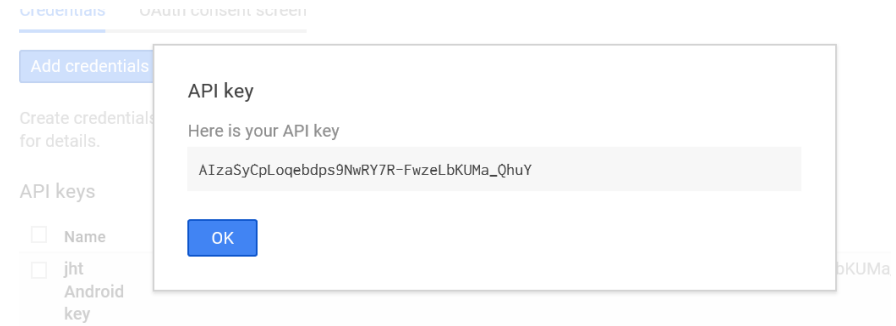
Now, in the zip file, you can find an existing gradle project Tutorial 3. Import it and go to res/values/google_maps_api.xml. In this tutorial, we create the project using API 23. If you cannot import the project successfully, please install the API 23 first in SDK manager. In addition, in the *build.gradle(Module:app)/defaultConfig*, set the *minSdkVersion 15*, *targetSdkVersion 23*.



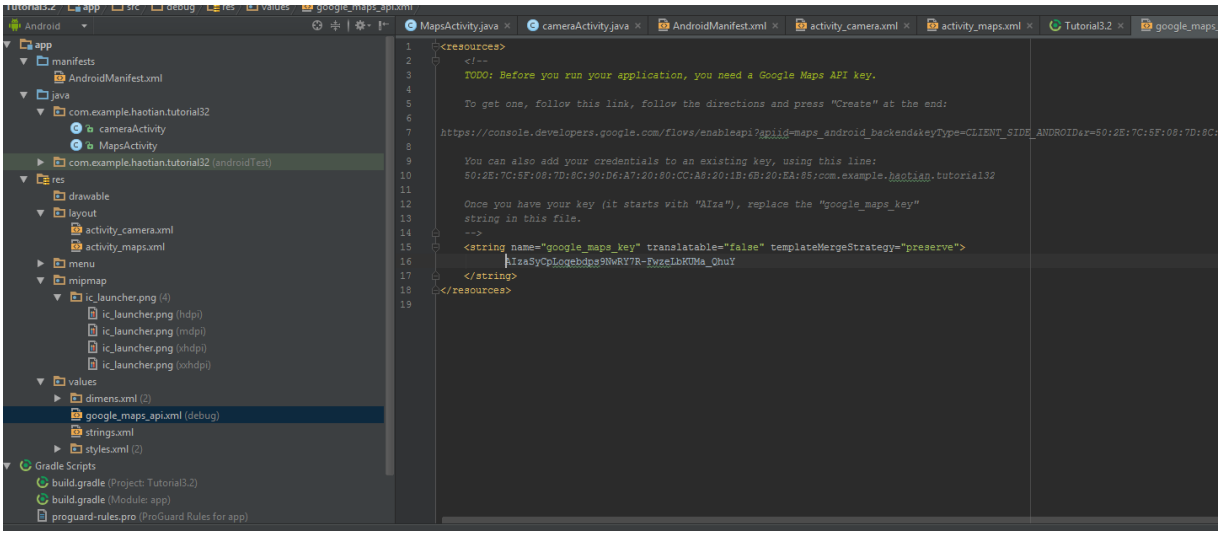
Copy and paste the package name and certificate fingerprint as following, then click “Create”.



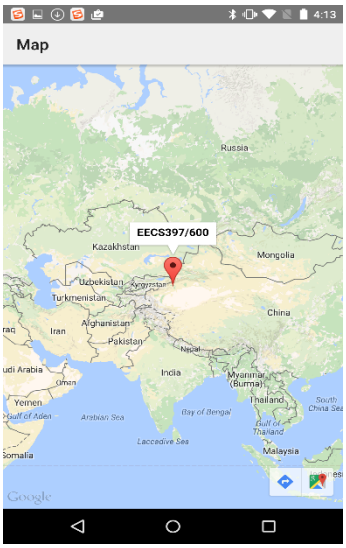
API key appears!



Then, copy the API key to the correct location as following figure.



Here we go! Enjoy!



Part II: Taking Photos

The Android way of delegating actions to other applications is to invoke an Intent that describes what you want done. This process involves three pieces: The Intent itself, a call to start the external Activity, and some code to handle the image data when focus returns to your activity. Here's a function that invokes an intent to capture a photo.

```
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

Homework 1: In the main interface of your app, there should be a “TAKE A PICTURE” button to go to the camera and take photos. After taking a photo, you should have an interface like Figure 2. There are three buttons. If the user is satisfied with the photo, then click tick button to save it into a custom directory under general external storage, otherwise, click return button to discard it and take a new photo. When the user clicks cross button, the app will return to the interface of map and finish taking photo.

Part III: Receiving Location Updates

We highly recommend you to learn this part by yourself! We do the exactly the same things with the official tutorial.

https://developer.android.com/intl/ru/training/location/receive-location-updates.html

There total 5 steps.

1. Connect to Location Services
2. Set up a Location Request
3. Request Location Updates
4. Define the Location Update Callback
5. Stop Location Updates

When you are done, you can get the location update periodically. You can save the last available coordinate data to the CSV file when you take a photo. You can display the latitude/longtitude to the screen dynamically to check the correctness.

Homework 2: In this project, you should take at least ten photos of buildings on or around campus. For each saved photo, the app should also save its information (TimeStamp and GPS coordinates) into CSV file in DCIM folder. Don not save the information of the discarded photo. The format of the CSV file is as follow:

TimeStamp	Latitude	Longitude

Please provide the savepath and name of CSV file in the report.

Part IV: Getting the Thumbnail

Get the Thumbnail: If the simple feat of taking a photo is not the culmination of your app's ambition, then you probably want to get the image back from the camera application and do something with it.

The Android Camera application encodes the photo in the return Intent delivered to *onActivityResult()* as a small Bitmap in the extras, under the key "data". The following code retrieves this image and displays it in an ImageView.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

Homework 3: After taking photos and saving image, when you return to the interface of map, all the photos as thumbnails should overlay on the maps as marker. Do not take two photos too closely, otherwise, these two photos will overlap on the map.

Part V: Changing title and snippet

When a photo is clicked, a textbox should pop up, then the user could add some description in it.

Homework 4: We do not provide any code of adding title and snippet to each photos. This part should be done by yourself. Take a screenshot of your result like Figure 3.