GOOGLE SOCIAL NETWORK TUTORIAL

- **Vedant Parikh**
  N12058928
  NYU Polytechnic School of Engineering

I have divided the tutorial into 4 Parts.

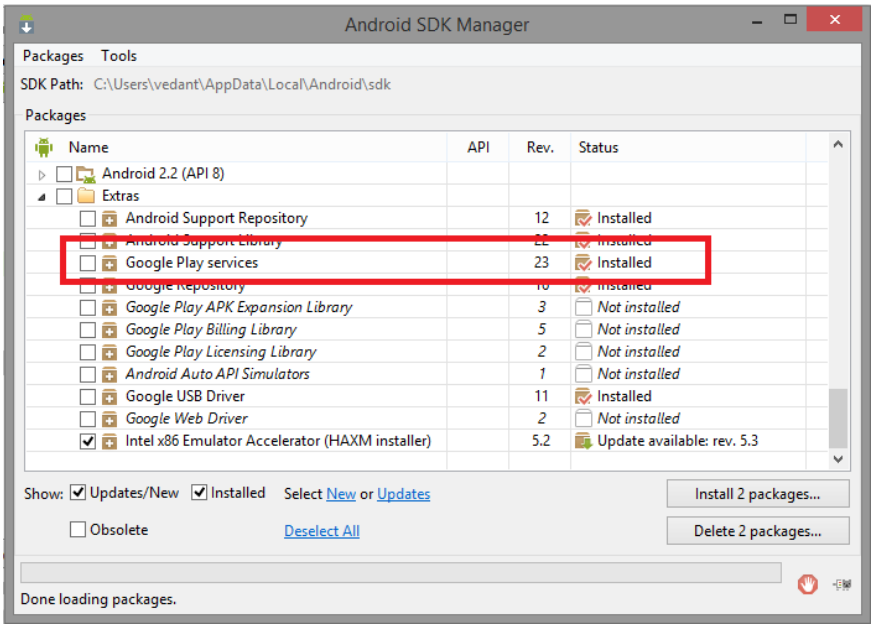
- Part 1: Enable the Google+ API
- Part 2: Setup the Project with minimum requirements along with adding user permissions and dependencies
- Part 3: Create the Sign in button in First Activity
- Part 4: Create share button in Second Activity

PART 1

I am writing the same instruction here as shown in the presentation documentation.

Some of the Requirements before Getting started is as follows:-

- You will need a compatible android device that runs android 2.3 and higher for developing and testing or You need to have an emulator with AVD that runs Google API based on Android 4.2.2 or higher
- Your project should be Compiled against android 2.3 or Higher.
- You need the latest update of Android SDK
- Install Google Play Service SDK. Here is an Image Attached to show you how to install Google Play service SDK.



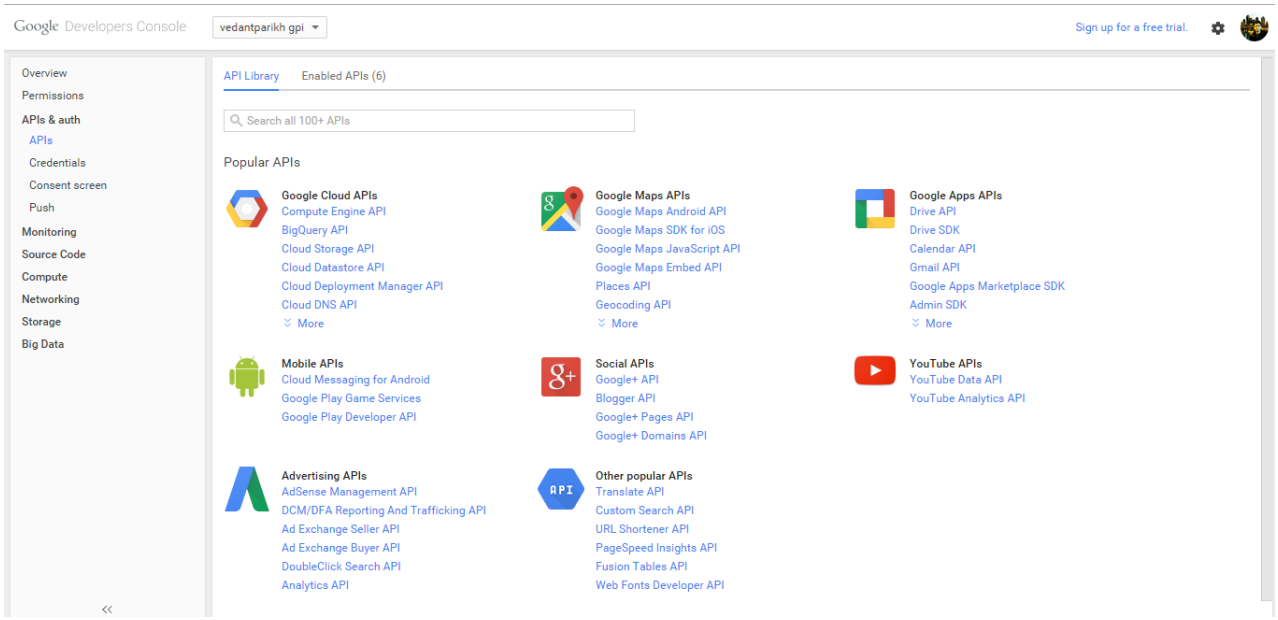
To Begin:

Before you start coding Google+ integration in your Application, You need to enable the Google+ API.

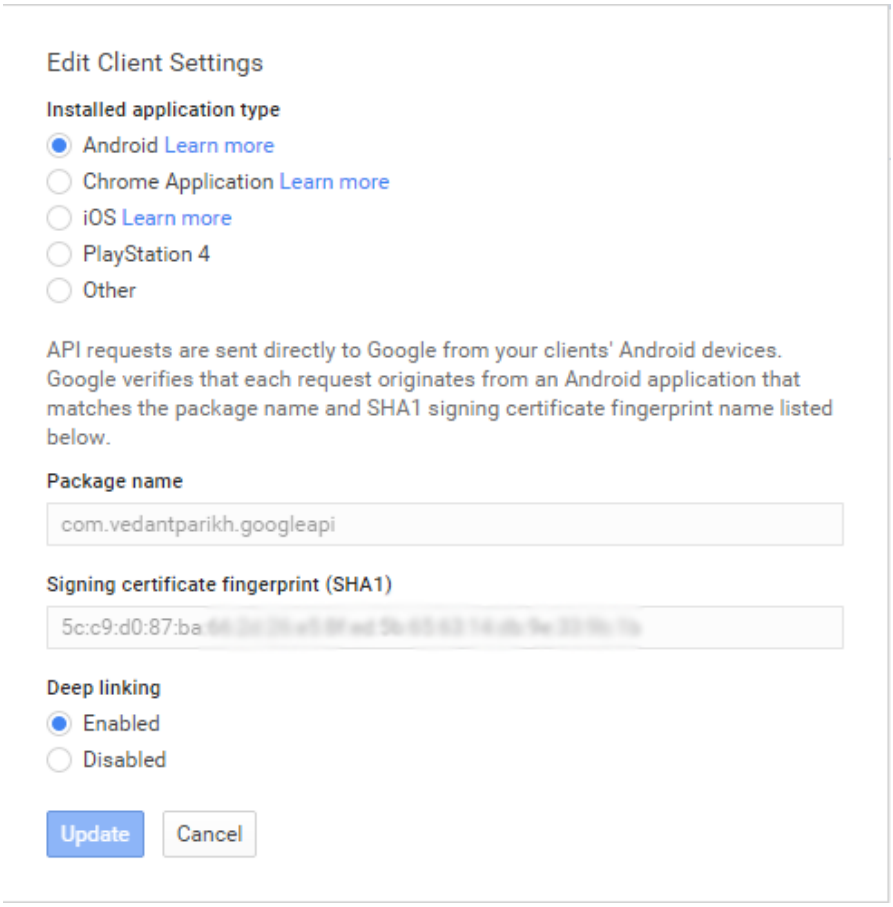
Steps to enable Google+ API is shown below:

- Go to the *Google Developers Console*.
  URL: https://console.developers.google.com/project
- Create a new Project and Give a project ID to the Project. Project ID should be unique worldwide so I would recommend to go with the project ID created by the console.

- After creating the Project, go to the APIs & auth in the left sidebar, Search for the ***Google+ API*** and set its status to ON. As you can see in the below image, you can find API of many different services provided by Google. Just enable the API and you can use it in your application.



- After Enabling the Google+ API, Go to the Credentials in the left Side Bar.
- You will have to create a Client ID. Select *Installed Application* and then select *Android* as the installed application.
- Write down your application's package name into the package name field to enable your package.
- You will also need signing certificate fingerprint i.e. SHA1. You can get this by running keytool utility. Running keytool utility is bit tricky so I will elaborate it in detail further in this chapter.
- Once, the Client ID is created, you can start coding your application by using the registered package name. Below is the image of Creating Client ID tab.


Edit Client Settings
Installed application type
Android Learn more
Chrome Application Learn more
iOS Learn more
PlayStation 4
Other
API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches the package name and SHA1 signing certificate fingerprint name listed below.
Package name
com.vedantparikh.googleapi
Signing certificate fingerprint (SHA1)
5c:c9:d0:87:ba
Deep linking
Enabled
Disabled
Update
Cancel

Keytool Utility:

Using keytool utility is a bit tricky. It has to be run in the terminal of your respective operating system and its format is different for different operating system.

For Mac or Linux, its format is:-

```
$ keytool -exportcert -alias androiddebugkey -keystore ~/.android/debug.keystore -
list -v
```

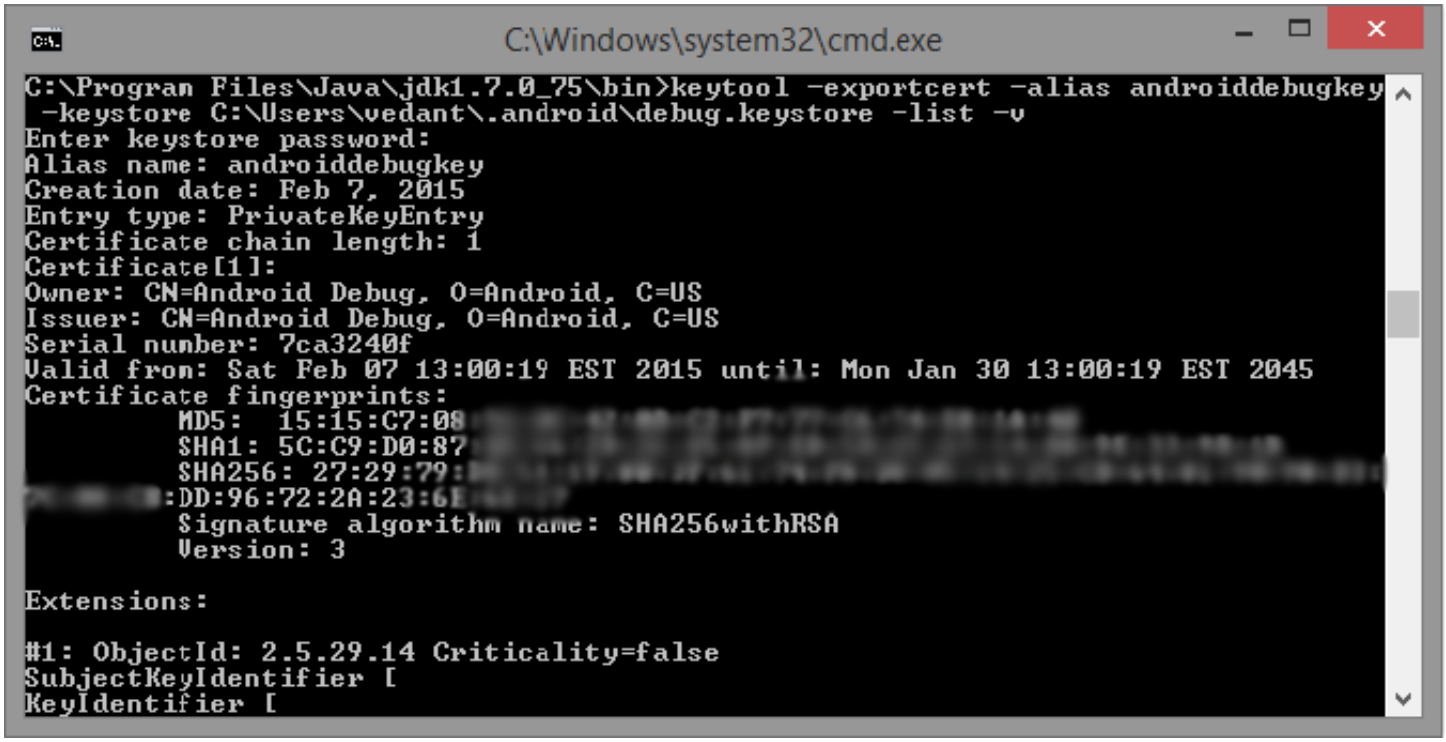
And for Windows, its format is:-

Before you begin, set the path of cmd to your environment variable.

```
keytool -exportcert -alias androiddebugkey -keystore
%USERPROFILE%\.android\debug.keystore -list -v
```

Where %USERPROFILE% = C:\<user name>\.android\debug.keystore. The default password for keytool is “android”

Below is the attached image of keytool utility in windows:



After running the Keytool utility, fetch the SHA1 fingerprint and use it to create the client ID.

PART 2

Once you have completed registering your application package name and enabling the Google+ API, You can follow the below steps for integration:

Create a new project in Android Studio and give it the package name you registered on the Google Developer Console.

- You need to add USER PERMISSIONS and Google Play Services version in your Application Manifest. Add the following lines in AndroidManifest.xml

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


<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
```

- Open your build.gradle and add the following dependencies:

```
compile 'com.google.android.gms:play-services:7.0.0'
compile 'com.google.android.gms:play-services-plus:7.0.0'
```

PART 3

Step 1: Create the first activity. In the xml layout file of your first activity, add the sign in button of Google+.

```
<com.google.android.gms.common.SignInButton
    android:id="@+id/sign_in_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="75dp" />
```

Step 2: You need to implement three interfaces in your activity to make it listen the established connection or failed connection.

```
public class MainActivity extends Activity implements
        ConnectionCallbacks, OnConnectionFailedListener, View.OnClickListener {
```

Step 3: In your first Activity, initialize the GoogleApiClient object on the onCreate handler of your activity. There are 2 types of initial scope of your application. If you want to access only basic information, set the scope to PLUS_PROFILE else if you want to look user's identity, ability to write application activities and its social graph, set its scope to PLUS_LOGIN.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    GoogleApi = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(Plus.API)
            .addScope(Plus.SCOPE_PLUS_LOGIN)
            .build();
    findViewById(R.id.sign_in_button).setOnClickListener(this);
}
```

Step 4: Connect the GoogleApiClient object on the onStart handler of the activity.

```
protected void onStart() {
    super.onStart();
    GoogleApi.connect();
}
```

Step 5: Initiate the onConnectionFailed method, Whenever GoogleApiClient object is unable to connect, onConnectionFailed method notifies the implementation.

```
@Override
public void onConnectionFailed(ConnectionResult result) {
    if (!Intent_Progress) {
        if (SignIn_Click && result.hasResolution()) {
            try {
                result.startResolutionForResult(this, RC_SIGN_IN);
                Intent_Progress = true;
            } catch (IntentSender.SendIntentException e) {
                Intent_Progress = false;
                GoogleApi.connect();
            }
        }
    }
}
```

Step 6: Create an onClick method which will check the id of sign in button and whether GoogleApiClient is connected or not. If the Id of the button matches and object is not connected, it will run the connect method to connect the object.

Step 7: Create an onConnected method which will toast a sentence when GoogleApiClient is connected. Make an intent to start a new activity in this method.

```
public void onClick(View view) {
    if (view.getId() == R.id.sign_in_button && !GoogleApi.isConnecting()) {
        SignIn_Click = true;
        GoogleApi.connect();
    }
}

@Override
public void onConnected(Bundle connectionHint) {
    SignIn_Click = false;
    Toast.makeText(this, "connected", Toast.LENGTH_LONG).show();
    startActivity(new Intent(this,ShareActivity.class));
}
```

Step 8: Create an onActivityResult() method to get the result of connection failure.

```
protected void onActivityResult(int requestCode, int responseCode, Intent intent) {
    if (requestCode == RC_SIGN_IN) {
        if (responseCode != RESULT_OK) {
            SignIn_Click = false;
        }
        Intent_Progress = false;
        if (!GoogleApi.isConnected()) {
            GoogleApi.reconnect();
        }
    }
}
```

PART 4

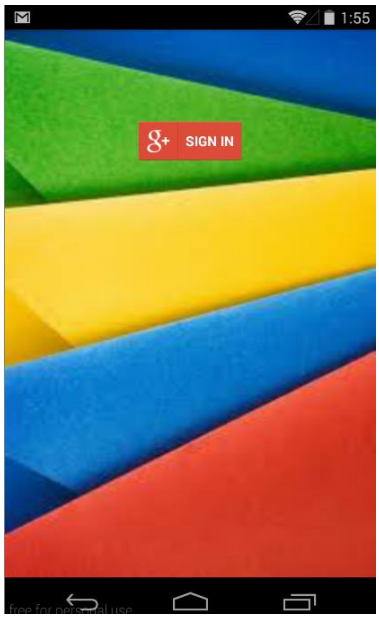
Create a new Activity which will have an EditText widget and a share button which will share a text on Google+ stream. In the xml file of the second activity, add two widgets EditText and a button. EditText widget will contain the text to be shared and by clicking the button it will share the post.

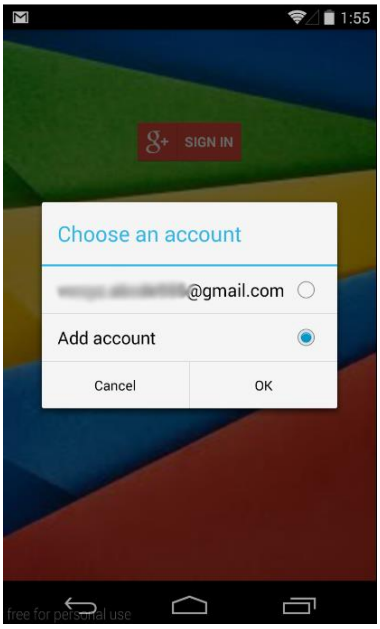
```
<EditText android:text="Write Your Post!" android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/editText"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:background="#fff" />

<Button
    android:id="@+id/share_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Share on Google+"
    android:layout_marginTop="87dp"
    android:layout_below="@+id/editText"
    android:textColor="#fff"
    android:background="#f00"
    android:layout_alignEnd="@+id/editText" />
```

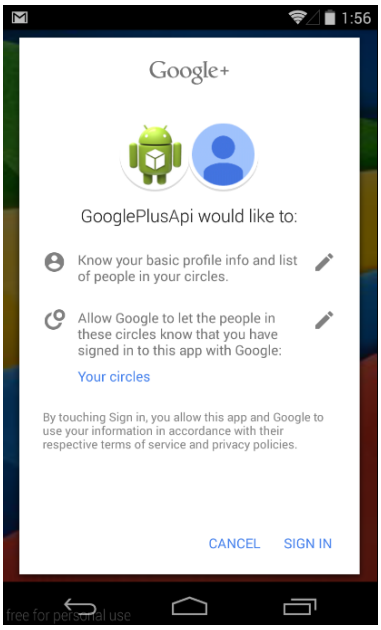
In the second activity, set an onClickListener on the share button. The below code will create a google dialog box which will have attributes about with whom to share your post. You can share it publicly, or to your circles or to a specific person.

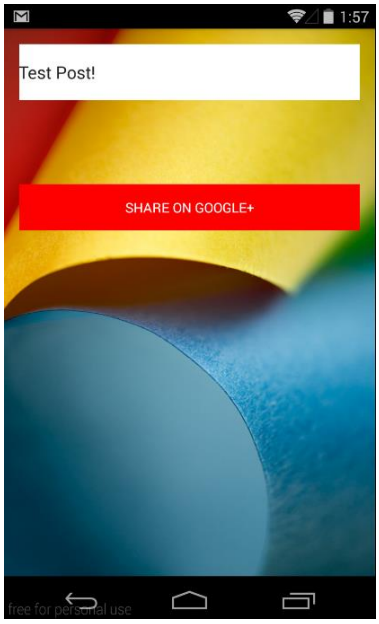
```
shareButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent shareIntent = new PlusShare.Builder(v.getContext())
                .setType("text/plain")
                .setText(((EditText) findViewById(R.id.editText)).getText())
                .getIntent();

        startActivityForResult(shareIntent, 0);
    }
});
```

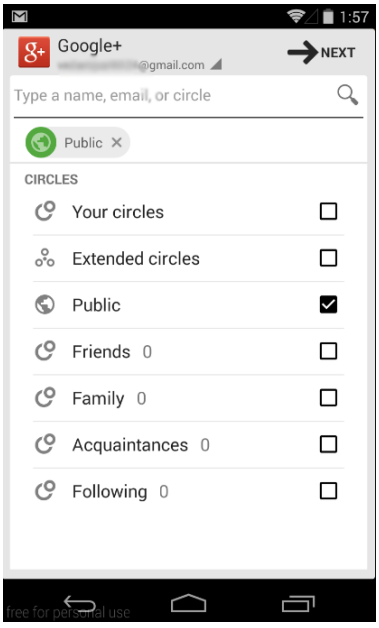
UI SCREENSHOTS OF THE GOOGLE+ INTEGRATION DEMO

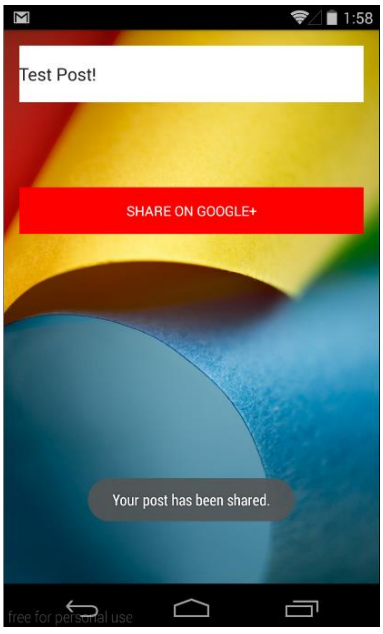












REFERENCES:

- **http://developers.google.com**
- **http://stackoverflow.com**
- **http://Wikipedia.org**
- **http://developer.android.com**