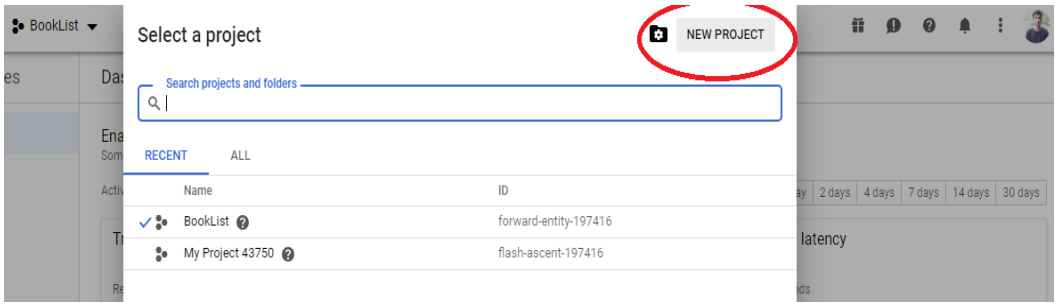
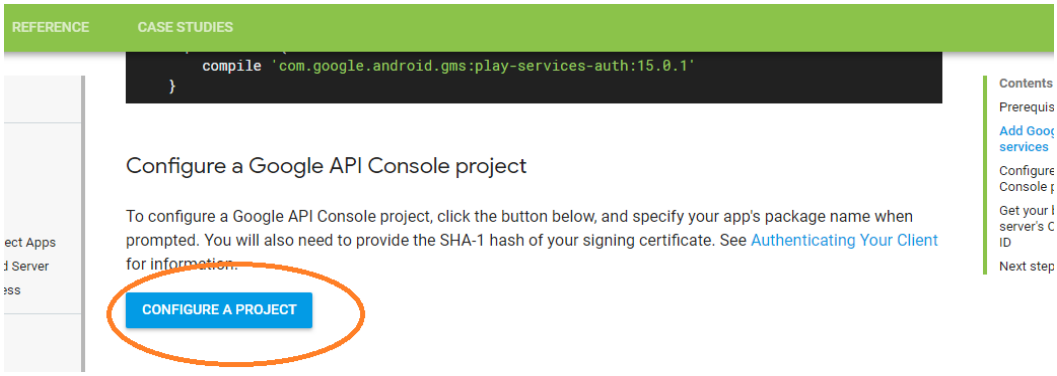
Google Sign in in an Android app

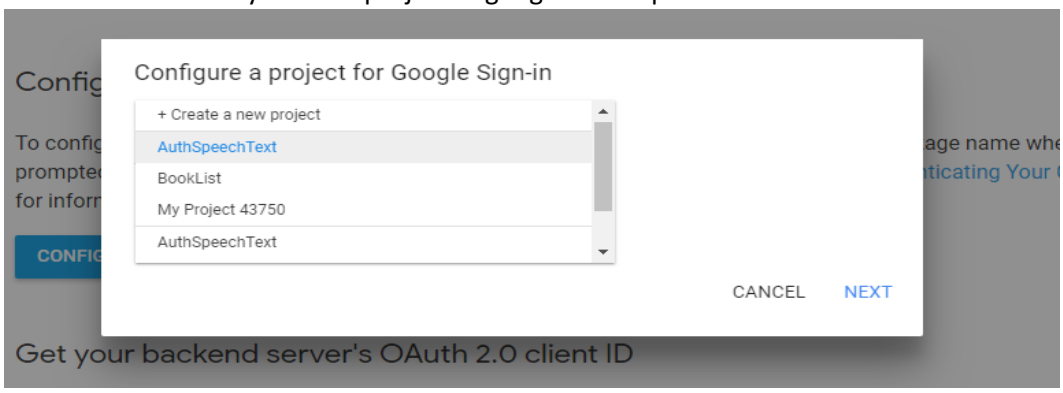
1. In your Android project Add internet permission
   <uses-permission android:name="android.permission.INTERNET"></uses-permission>
2. Go to google developer console.
   https://console.developers.google.com/apis/dashboard?project=forward-entity-197416&folder&organizationId&duration=PT1H
3. Create a new Project
   give it same name to your android project (Not mandatory).

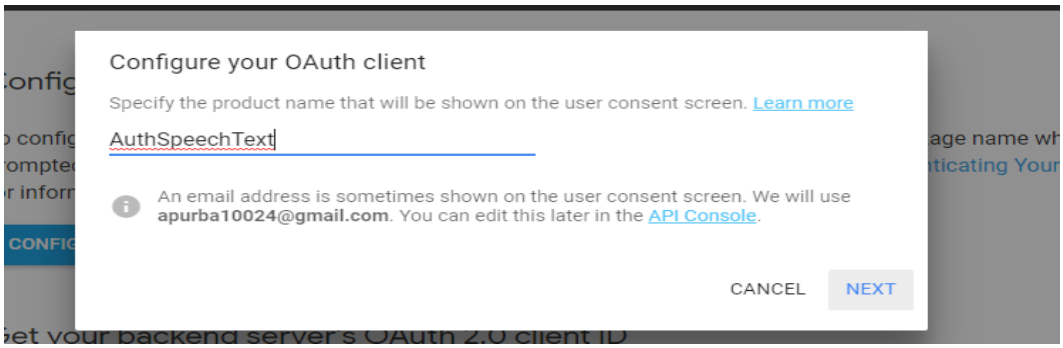


4. After creating the project, go to google sign in official documentation.
   https://developers.google.com/identity/sign-in/android/start-integrating
5. Then click on the following button

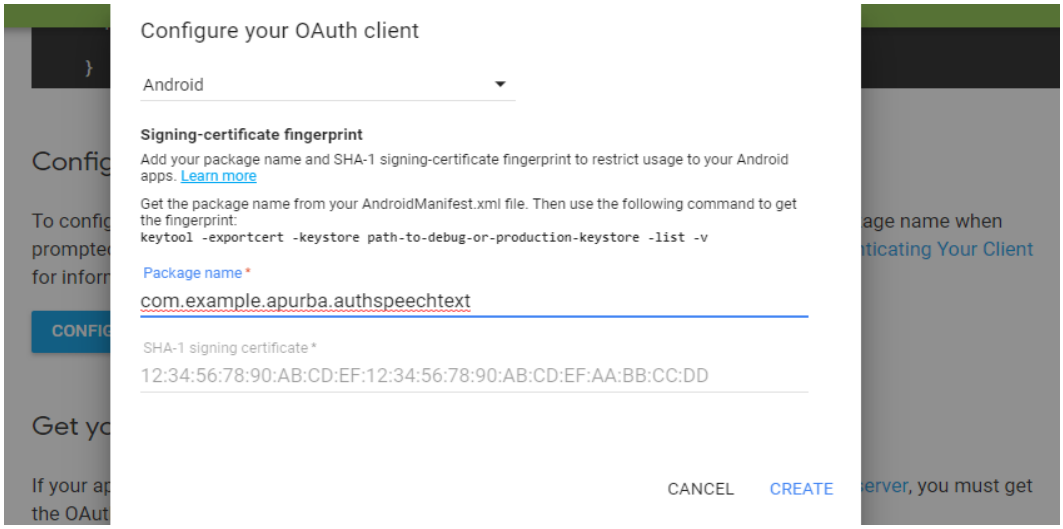


Now select the newly created project in google developer console as follows and click next

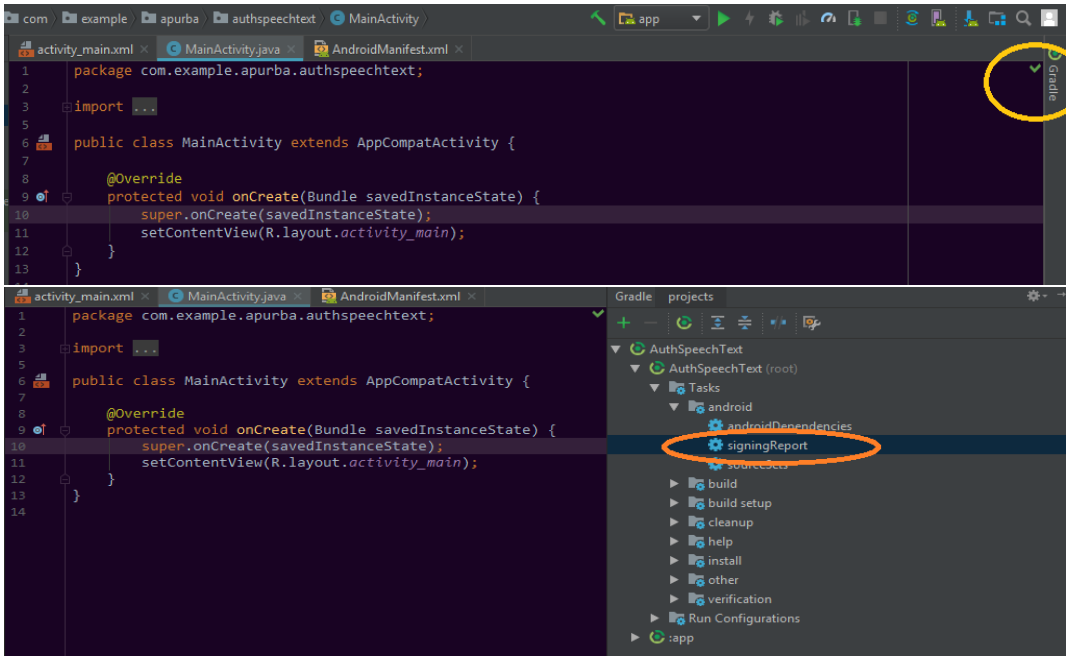


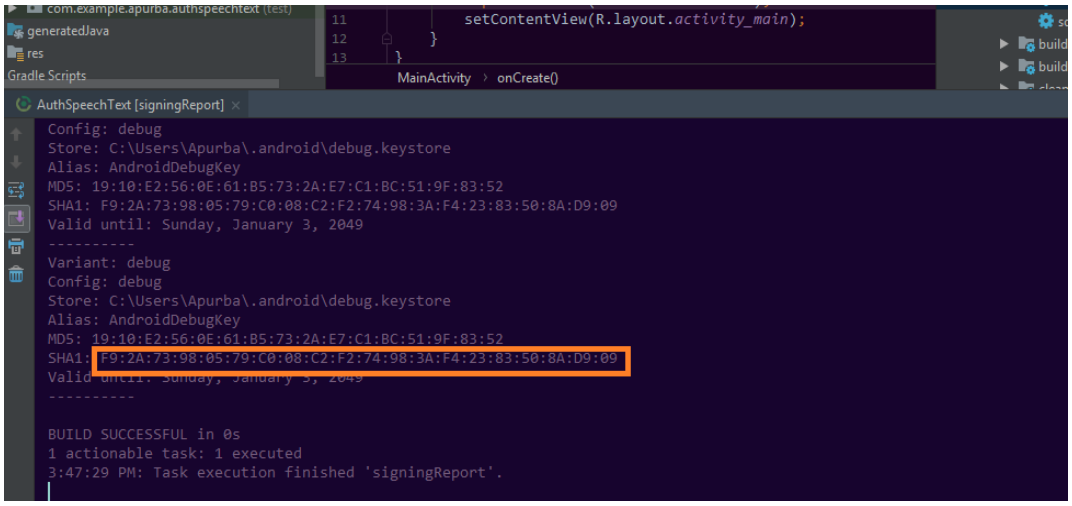


Now Paste your android project package name in following section

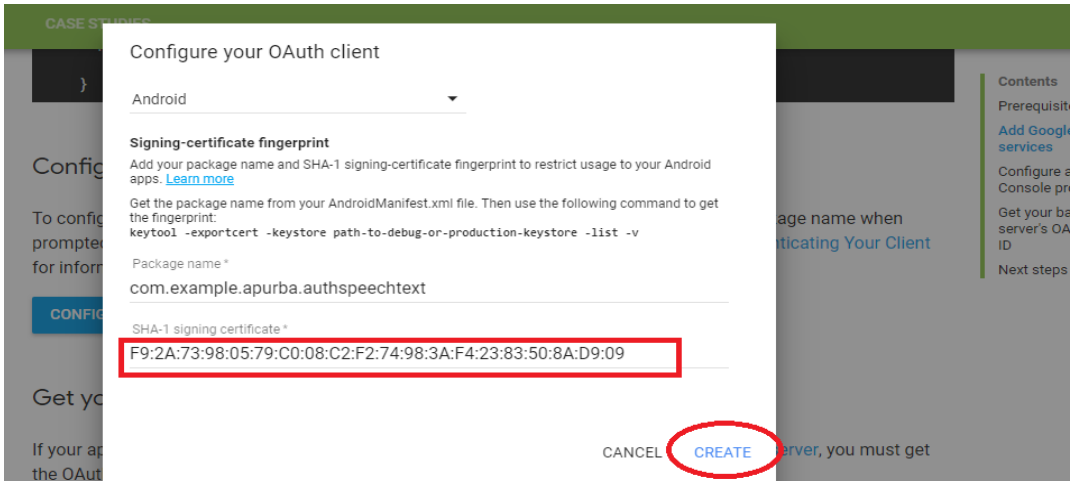


Now you need to create your android project SHA-1 fingerprint certificate. You can get your SHA-1 fingerprint in many different way such as “Key tool”, “Android debug” e.t.c. But its very easy to get SHA-1 fingerprint using android studio. Follow these following steps.

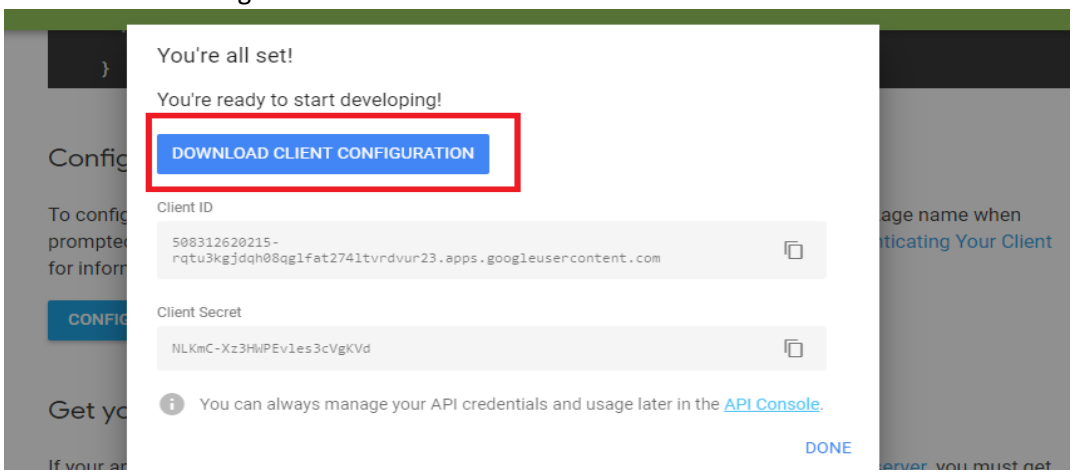




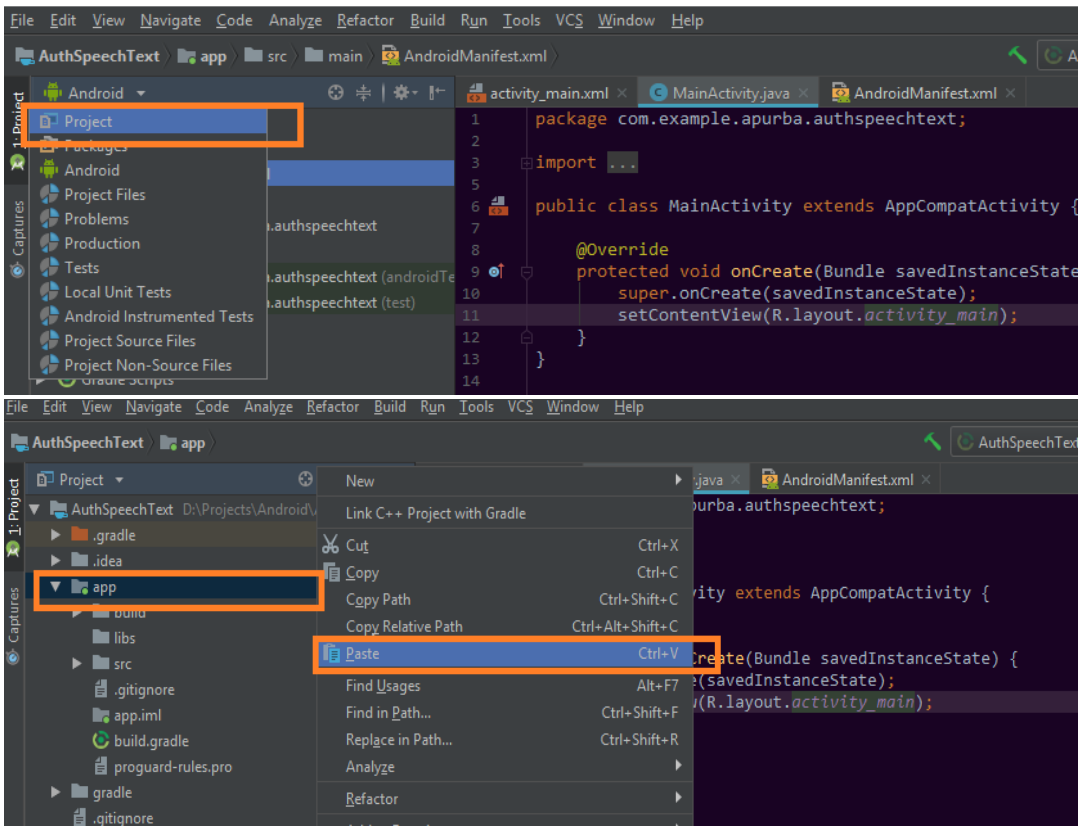
Ok now you got your SHA-1 fingerprint, copy and paste it on that previous section



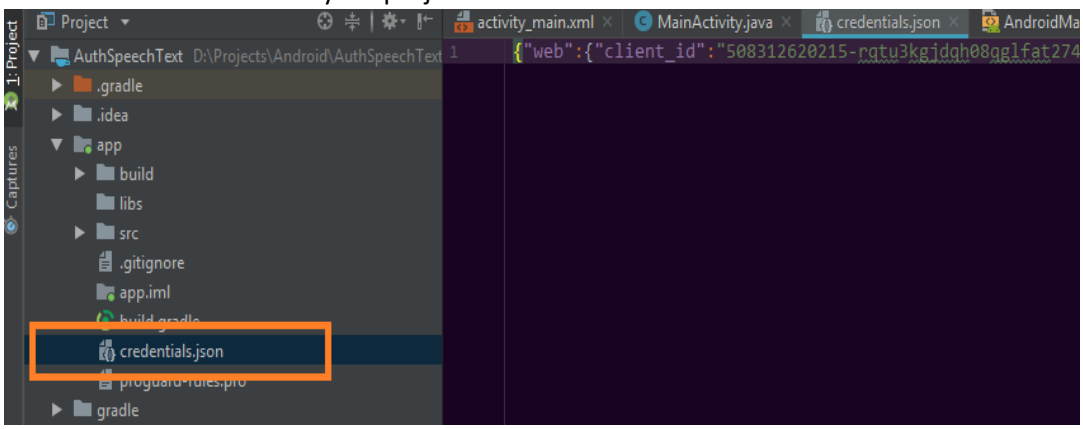
Download the configuration file



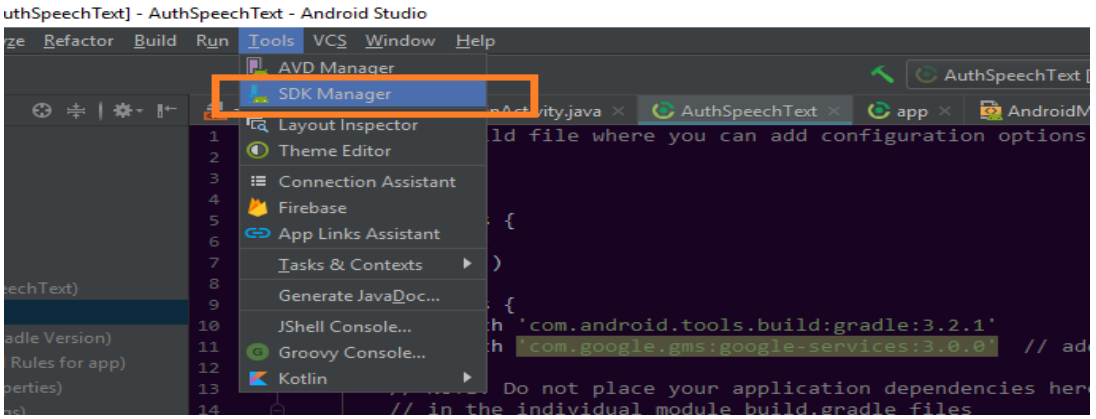
6. After downloading the “XXX.json” fie you need to add it to your android project. Do it as follows.(NOTE: When you add the json file in your project it must be named as “ google-services.json”).

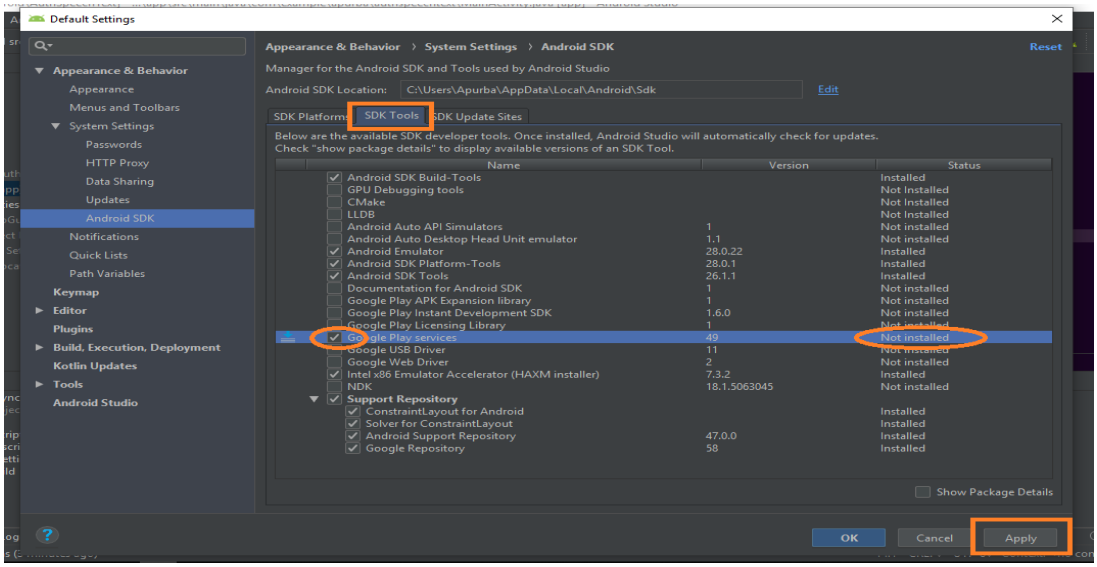


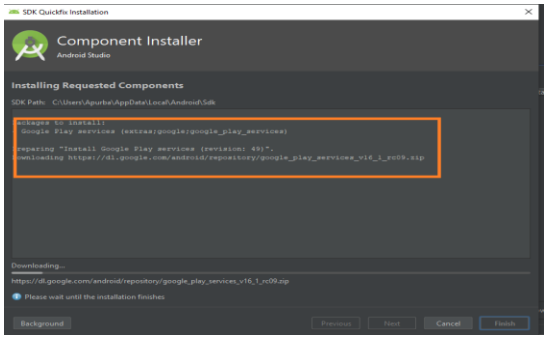
Now the file is available in your project.

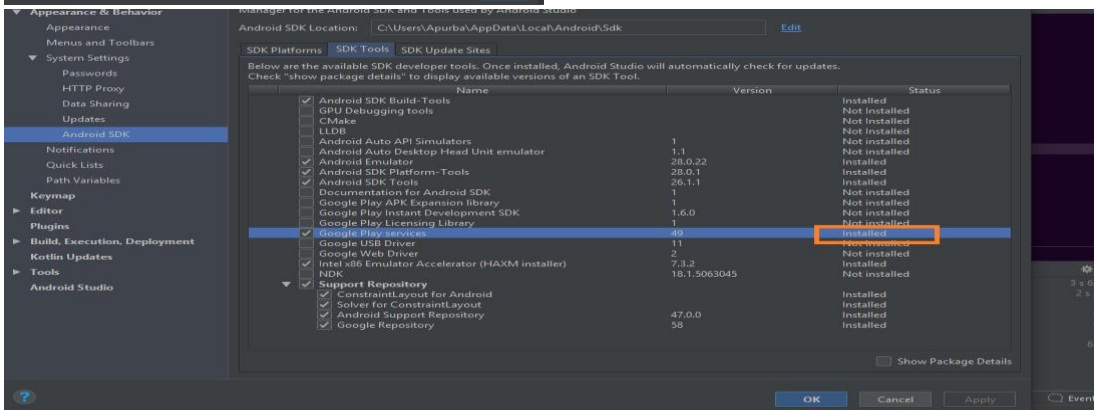


7. To enable google sign in you need to add some dependencies in your project.
   First make sure “google play service” is install in your project, to check this follow the step bellow.

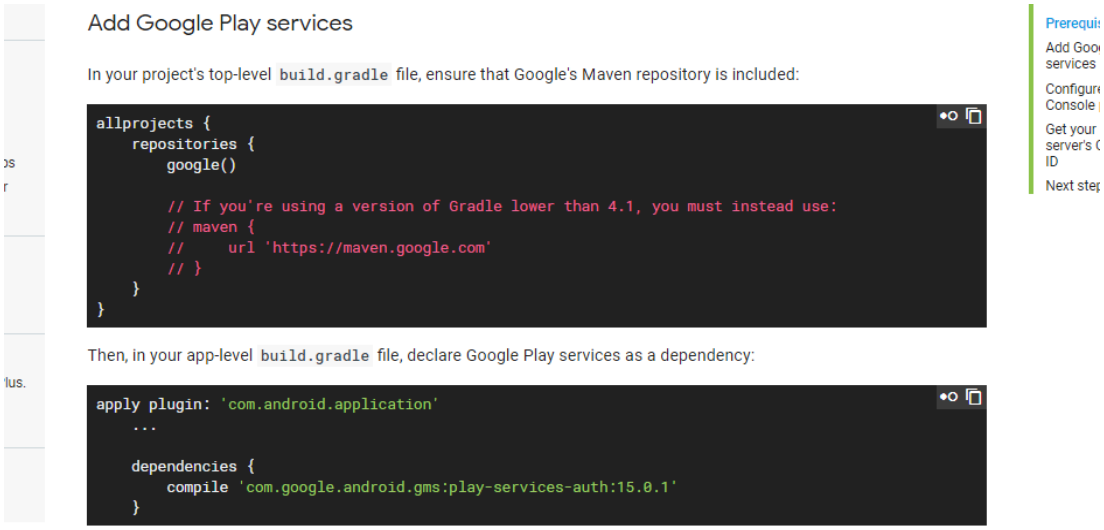

uthSpeechText] - AuthSpeechText - Android Studio
Refactor Build Run Tools VCS Window Help
AVD Manager
SDK Manager
Layout Inspector
Theme Editor
Connection Assistant
Firebase
App Links Assistant
Tasks & Contexts
Generate JavaDoc...
JShell Console...
Groovy Console...
Kotlin
AuthSpeechText
app
'com.android.tools.build:gradle:3.2.1'
'com.google.gms:google-services:3.0.0'


Default Settings
Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Updates
Android SDK
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin Updates
Tools
Android Studio
Appearance & Behavior › System Settings › Android SDK
Reset
Manager for the Android SDK and Tools used by Android Studio
Android SDK Location: C:\Users\Apurba\AppData\Local\Android\Sdk
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools
GPU Debugging tools
CMake
LLDB
Android Auto API Simulators
Android Auto Desktop Head Unit emulator
Android Emulator
Android SDK Platform-Tools
Android SDK Tools
Documentation for Android SDK
Google Play APK Expansion library
Google Play Instant Development SDK
Google Play Licensing Library
Google Play services
Google USB Driver
Google Web Driver
Intel x86 Emulator Accelerator (HAXM installer)
NDK
Support Repository
ConstraintLayout for Android
Solver for ConstraintLayout
Android Support Repository
Google Repository
Show Package Details
OK
Cancel
Apply


SDK Quickfix Installation
Component Installer
Android Studio
Installing Requested Components
SDK Path: C:\Users\Apurba\AppData\Local\Android\Sdk
Please wait until the installation finishes
Background
Previous
Next
Cancel
Finish


Appearance & Behavior
Appearance
Menus and Toolbars
System Settings
Passwords
HTTP Proxy
Data Sharing
Updates
Android SDK
Notifications
Quick Lists
Path Variables
Keymap
Editor
Plugins
Build, Execution, Deployment
Kotlin Updates
Tools
Android Studio
Android SDK Location: C:\Users\Apurba\AppData\Local\Android\Sdk
Edit
SDK Platforms
SDK Tools
SDK Update Sites
Below are the available SDK developer tools. Once installed, Android Studio will automatically check for updates.
Check "show package details" to display available versions of an SDK Tool.
Name
Version
Status
Android SDK Build-Tools
GPU Debugging tools
CMake
LLDB
Android Auto API Simulators
Android Auto Desktop Head Unit emulator
Android Emulator
Android SDK Platform-Tools
Android SDK Tools
Documentation for Android SDK
Google Play APK Expansion library
Google Play Instant Development SDK
Google Play Licensing Library
Google Play services
Google USB Driver
Google Web Driver
Intel x86 Emulator Accelerator (HAXM installer)
NDK
Support Repository
ConstraintLayout for Android
Solver for ConstraintLayout
Android Support Repository
Google Repository
Show Package Details
OK
Cancel
Apply

Now google play service is installed in your project. Now add some dependencies.
Go to https://developers.google.com/identity/sign-in/android/start-integrating to see latest instruction in the bellow section.



Add Google Play services

In your project's top-level `build.gradle` file, ensure that Google's Maven repository is included:

```
allprojects {
    repositories {
        google()

        // If you're using a version of Gradle lower than 4.1, you must instead use:
        // maven {
        //     url 'https://maven.google.com'
        // }
    }
}
```

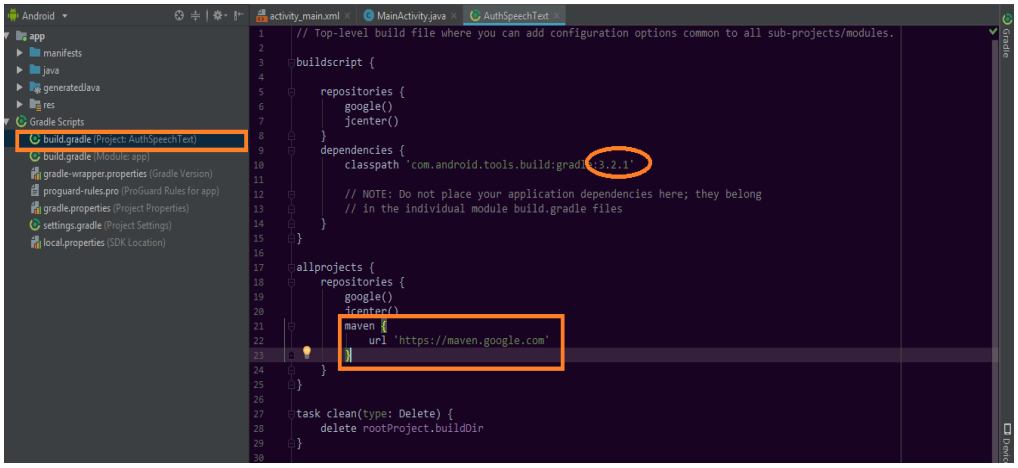
Then, in your app-level `build.gradle` file, declare Google Play services as a dependency:

```
apply plugin: 'com.android.application'
    ...

    dependencies {
        compile 'com.google.android.gms:play-services-auth:15.0.1'
    }
```

In your Project level `build.gradle` file add following line

```
allprojects {
    repositories {
        google()
        // If you're using a version of Gradle lower than 4.1, you must instead use:
        // maven {
        //     url 'https://maven.google.com'
        // }
    }
}
```



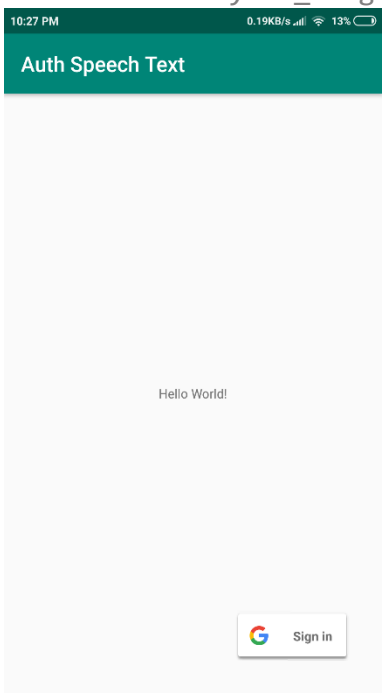
In your app level `build.gradle` file add following line

```
dependencies {
    implementation 'com.google.android.gms:play-services-auth:16.0.1'
}
```

8. Now you are ready to use google paly service.

Sign in Button(optional):

```
<!-- res/layout/activity_?.xml -->
<com.google.android.gms.common.SignInButton
    android:id="@+id/sign_in_button"
    android:layout_width="wrap_content"
    android:layout_margin="30dp"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentBottom="true"/>
```



Note: Can't listen to events using onClick (must attach in Java code)

Initialize Google API:

In your Login Activity onCreate() method

```
SignInButton signInButton = findViewById(R.id.sign_in_button);
signInButton.setOnClickListener(new View.OnClickListener() {
        @Override
         public void onClick(View v) {
            signInClicked();
       }
});

// request the user's ID, email address, and basic profile
GoogleSignInOptions options = new GoogleSignInOptions.Builder(
            GoogleSignInOptions.DEFAULT_SIGN_IN)
            .requestEmail()
            .build();
```

```
// build API client with access to Sign-In API and options above
google = new GoogleApiClient.Builder(this)
        .enableAutoManage(this, this)//“this” will get Error For now
        .addApi(Auth.GOOGLE_SIGN_IN_API, options)
        .addConnectionCallbacks(this)
        .build();
```

Activity Events and Callbacks:

in order to pass this to API client builder .enableAutoManage call, you must implement some event listening methods in your activity:

```
public class MainActivity extends AppCompatActivity implements
        GoogleApiClient.OnConnectionFailedListener,
        GoogleApiClient.ConnectionCallbacks {
    ...

    @Override
    public void onConnected(@Nullable Bundle bundle) {
        Log.v("MainActivity", "*********onConnected*********");
    }

    @Override
    public void onConnectionSuspended(int i) {
        Log.v("MainActivity",
        "*********onConnectionSuspended*********");
    }

    @Override
    public void onConnectionFailed(@NonNull ConnectionResult
    connectionResult) {
        Log.v("MainActivity",
        "*********onConnectionFailed*********");
    }
}
```

Implement the signInClicked() method:

to do a login, start an activity using an intent created as follows:

```
private void signInClicked(){
    Toast.makeText(this, "Sign in clicked", Toast.LENGTH_SHORT).show();

    // connect to Google server to log in
    Intent intent = Auth.GoogleSignInApi.getSignInIntent(google);
    startActivityForResult(intent, REQ_CODE_GOOGLE_SIGNIN);
}
```

when the activity returns, examine the result:

```
/*
 * This method is called when Google Sign-in comes back to my activity.
 * We grab the sign-in results and display the user's name and email address.
 */
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);

    if (requestCode == REQ_CODE_GOOGLE_SIGNIN) {
        TextView outPutTextView = findViewById(R.id.tv_Output);
        // google sign-in has returned
        GoogleSignInResult result
  Auth.GoogleSignInApi.getSignInResultFromIntent(intent);
        if (result.isSuccess()) {
            // yay; user logged in successfully
            GoogleSignInAccount acct = result.getSignInAccount();

            outPutTextView.setText("You signed in as: " + acct.getDisplayName() +
                    " "
                    + acct.getEmail());
        } else {
            outPutTextView.setText("Login fail. :(");
        }
    }
}
```

Done