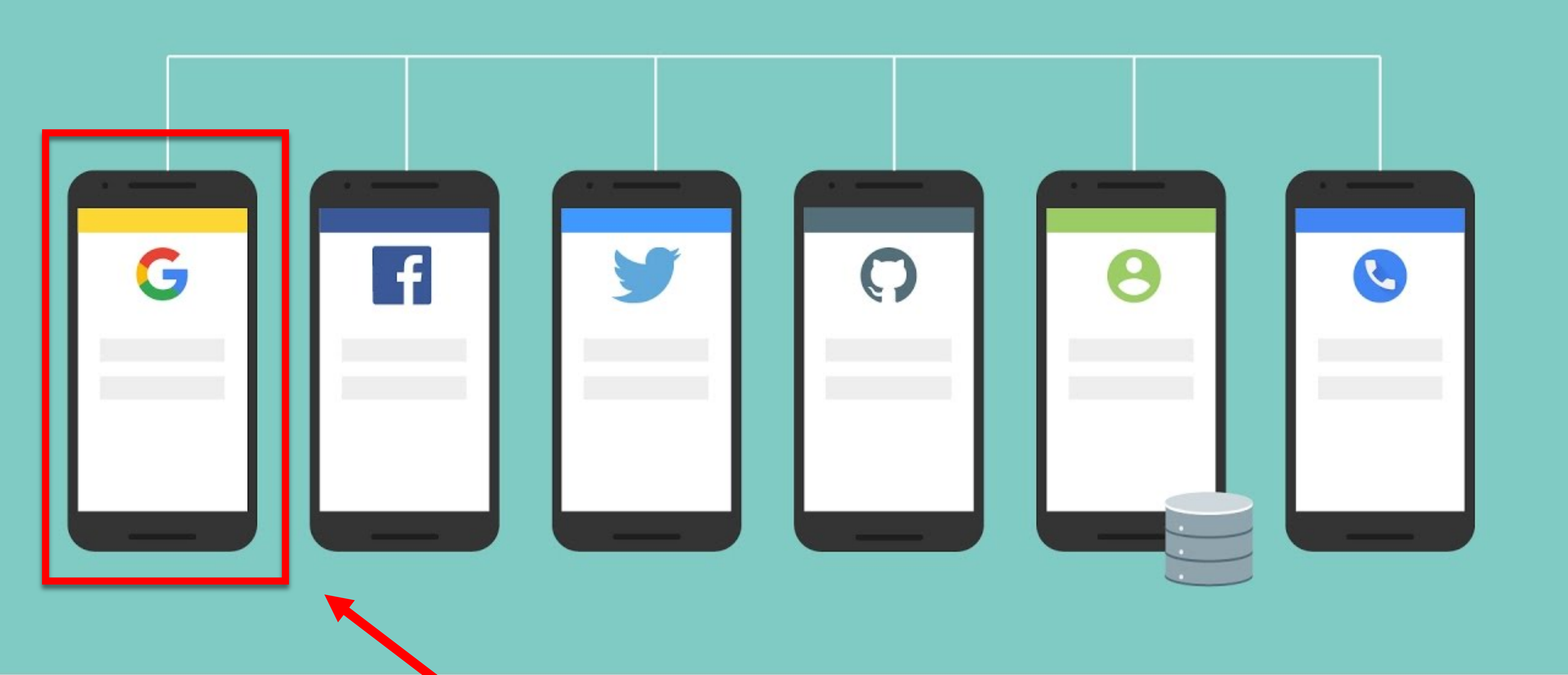
# Mobile Application Development

Produced by David Drohan (ddrohan@wit.ie)

Department of Computing & Mathematics
Waterford Institute of Technology
http://www.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE



# Authentication

## Overview (Recap)

- ❑ Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices
- ❑ Firebase Authentication provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers and popular federated identity providers
- ❑ You can let your users authenticate with Firebase using their Google Accounts by integrating Google Sign-In into your app.

## Overview

- ❑ With the Google Identity Platform for Android, you can allow application users to sign in with their existing Google accounts.
- ❑ It helps you in knowing your end users and providing them with a better enriched experience in your application.
- ❑ As soon as a allow the user to use Google Sign In, you can easily get info about that user such as email, display name etc.
- ❑ You can also get (for example) access to the users Google Drive account, allow them add an event to Calendar, share with their Google Contacts etc.

Overall, it is quick and easy way to engage end users in your application.

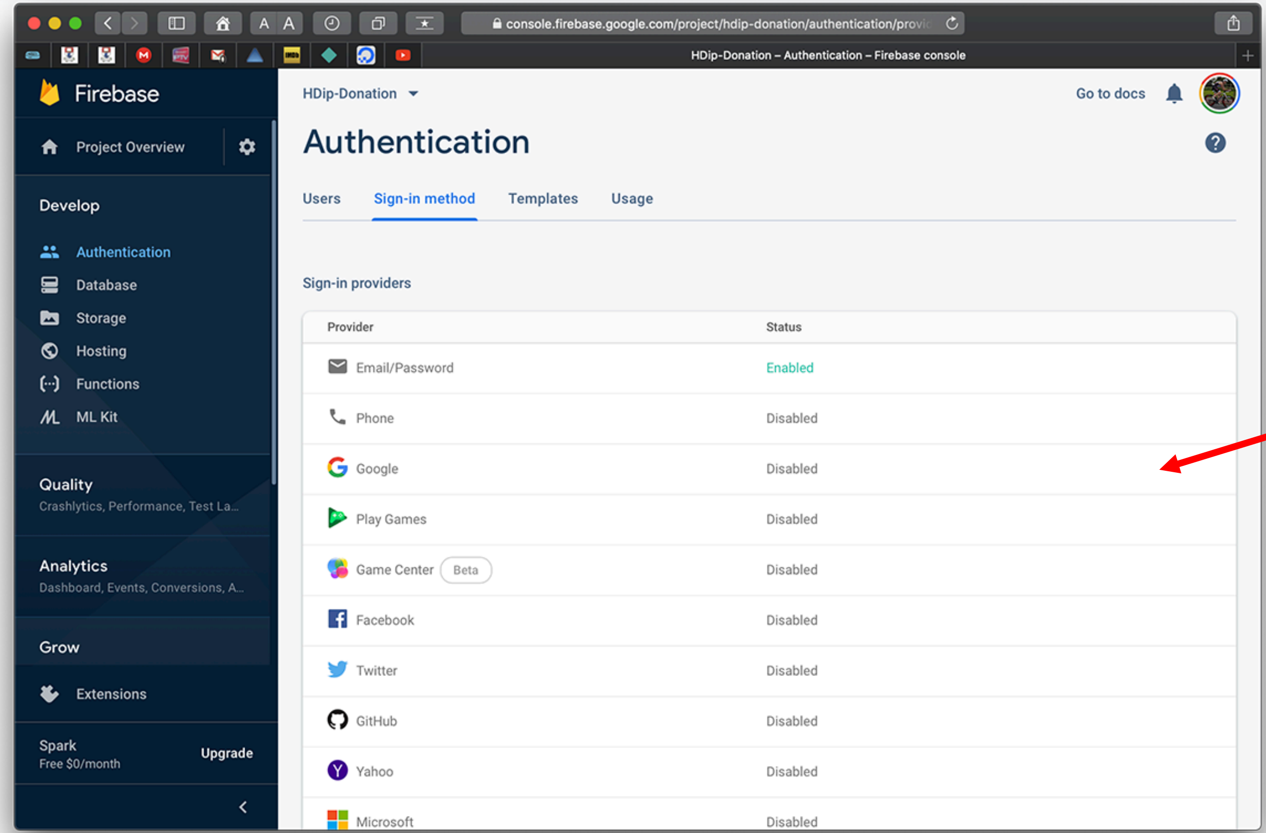
## Integrating Google Sign-In into Your Android App

https://developers.google.com/identity/sign-in/android/sign-in

## Google Authentication (in Firebase)

## 1. Create your Firebase Project (set up in previous labs)

## 2. Setup your Sign-In Method



console.firebase.google.com/project/hdip-donation/authentication/provid

HDip-Donation – Authentication – Firebase console

Firebase

Project Overview

Develop

- Authentication
- Database
- Storage
- Hosting
- Functions
- ML Kit

Quality
Crashlytics, Performance, Test La...

Analytics
Dashboard, Events, Conversions, A...

Grow

- Extensions

Spark
Free $0/month
Upgrade

HDip-Donation

Go to docs

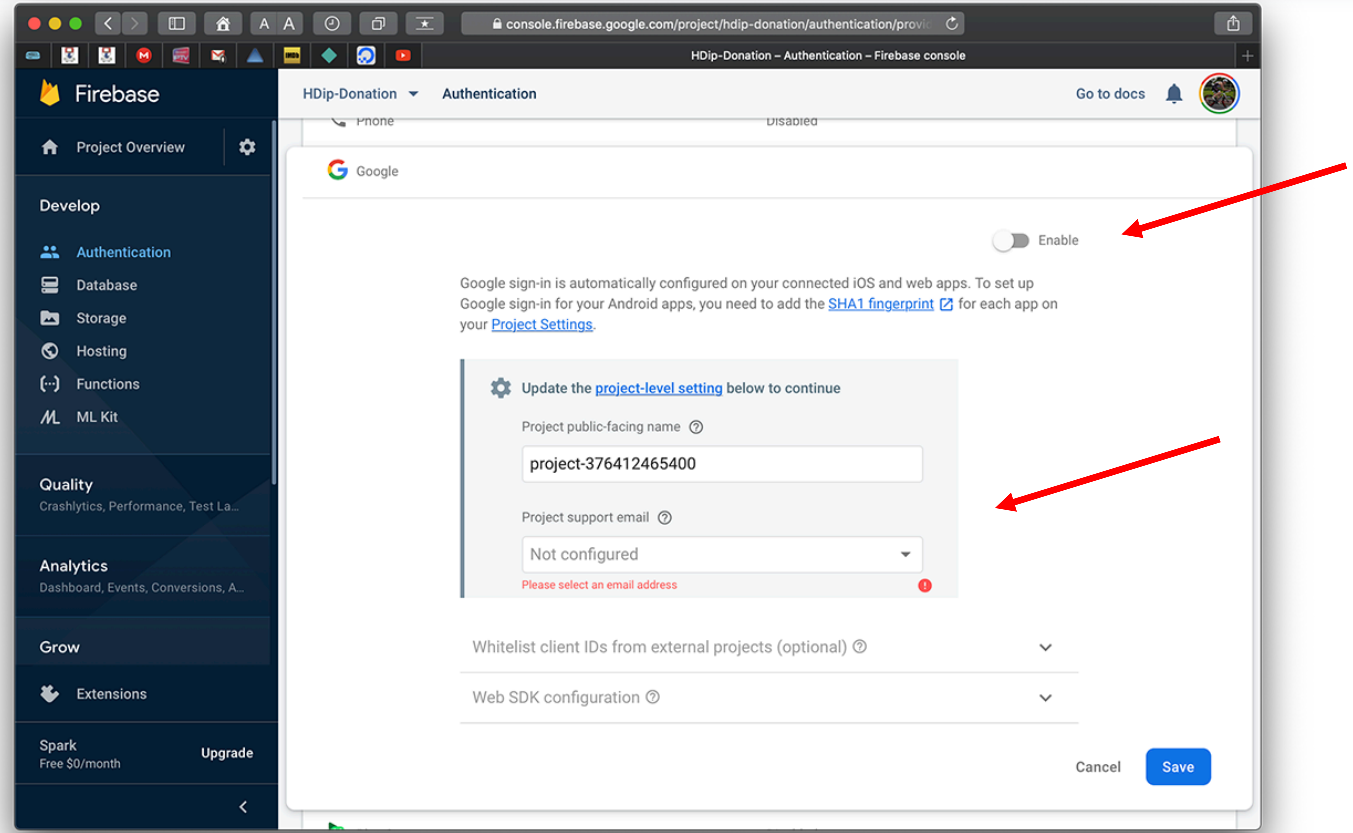
Authentication

Users | Sign-in method | Templates | Usage

Sign-in providers

| Provider | Status |
|---|---|
| Email/Password | Enabled |
| Phone | Disabled |
| Google | Disabled |
| Play Games | Disabled |
| Game Center Beta | Disabled |
| Facebook | Disabled |
| Twitter | Disabled |
| GitHub | Disabled |
| Yahoo | Disabled |
| Microsoft | Disabled |

# 2. Setup your Sign-In Method



console.firebase.google.com/project/hdip-donation/authentication/provic

HDip-Donation – Authentication – Firebase console

Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app on your Project Settings.

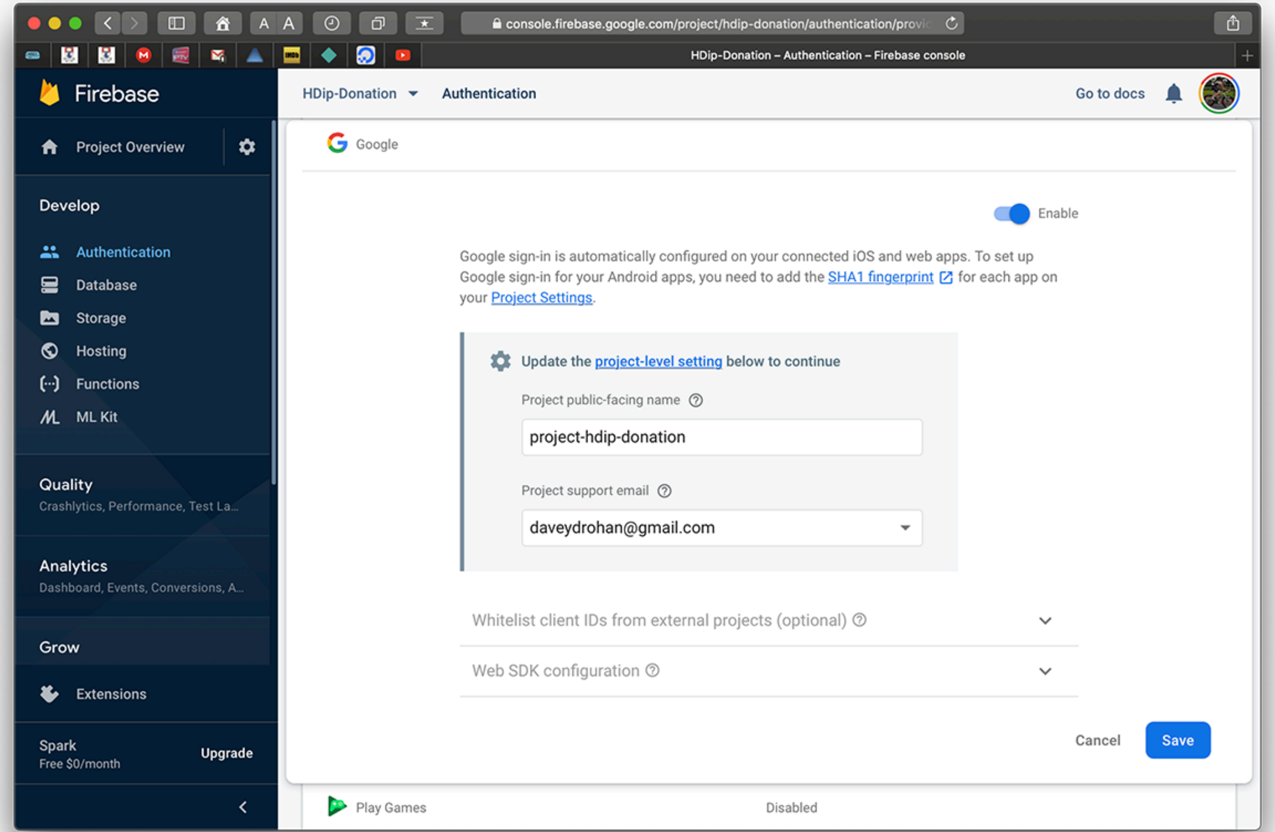
Update the project-level setting below to continue

Project public-facing name: project-376412465400

Project support email: Not configured

Please select an email address

## 2. Setup your Sign-In Method


console.firebase.google.com/project/hdip-donation/authentication/provid
HDip-Donation – Authentication – Firebase console
Firebase
HDip-Donation
Authentication
Go to docs
Project Overview
Develop
Authentication
Database
Storage
Hosting
Functions
ML Kit
Quality
Crashlytics, Performance, Test La...
Analytics
Dashboard, Events, Conversions, A...
Grow
Extensions
Spark
Free $0/month
Upgrade
Google
Enable

Google sign-in is automatically configured on your connected iOS and web apps. To set up Google sign-in for your Android apps, you need to add the SHA1 fingerprint for each app on your Project Settings.

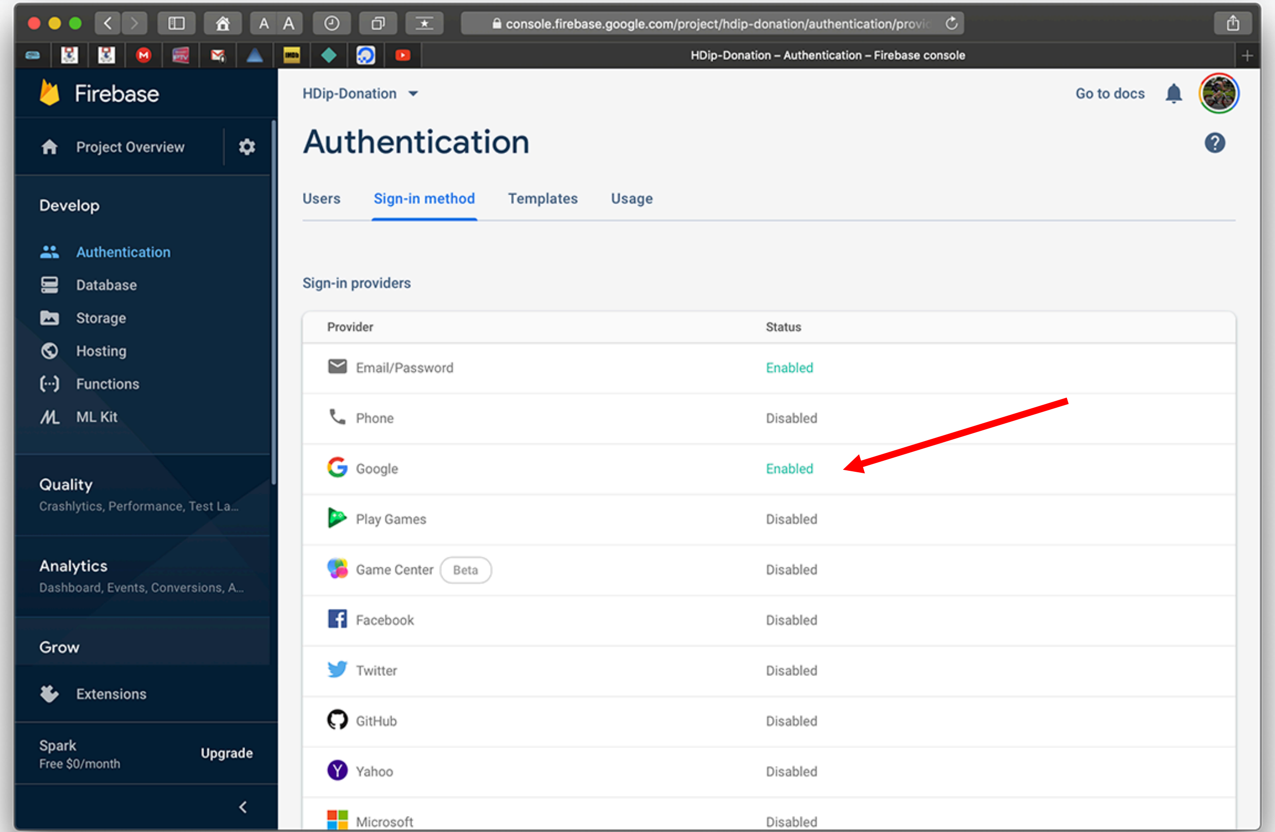
Update the project-level setting below to continue

Project public-facing name: project-hdip-donation

Project support email: daveydrohan@gmail.com

Whitelist client IDs from external projects (optional)
Web SDK configuration
Cancel
Save
Play Games
Disabled

# 2. Setup your Sign-In Method


console.firebase.google.com/project/hdip-donation/authentication/provi

HDip-Donation – Authentication – Firebase console

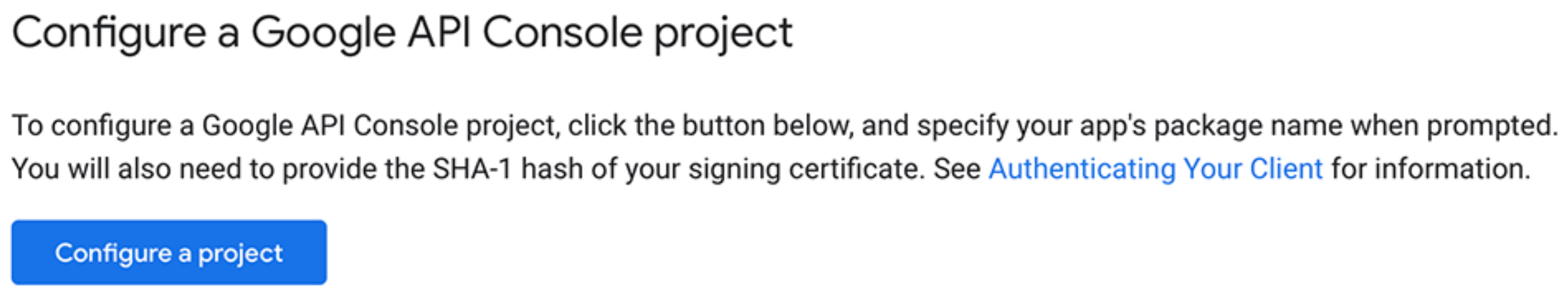
## Authentication

Users | Sign-in method | Templates | Usage

Sign-in providers

| Provider | Status |
|---|---|
| Email/Password | Enabled |
| Phone | Disabled |
| Google | Enabled |
| Play Games | Disabled |
| Game Center Beta | Disabled |
| Facebook | Disabled |
| Twitter | Disabled |
| GitHub | Disabled |
| Yahoo | Disabled |
| Microsoft | Disabled |

## 3. Setup your Sign-In Method (via Web Link)

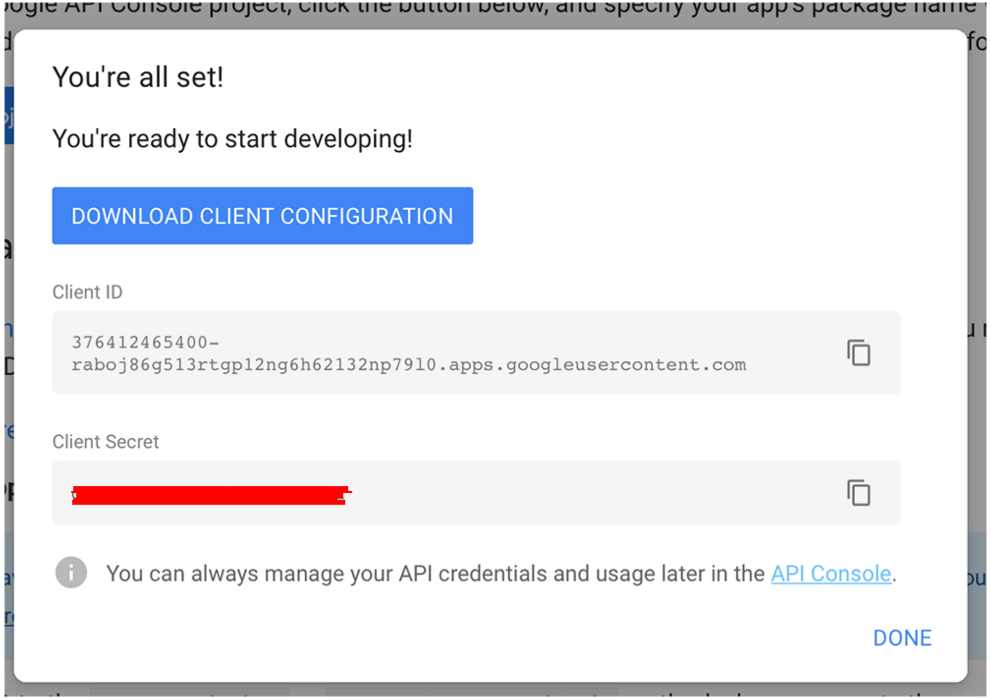


Configure a Google API Console project

To configure a Google API Console project, click the button below, and specify your app's package name when prompted. You will also need to provide the SHA-1 hash of your signing certificate. See Authenticating Your Client for information.

(Detailed Instructions in the Lab)

# 3. Setup your Sign-In Method (via Web Link)



You're all set!

You're ready to start developing!

DOWNLOAD CLIENT CONFIGURATION

Client ID

```
376412465400-
raboj86g513rtgp12ng6h62132np7910.apps.googleusercontent.com
```

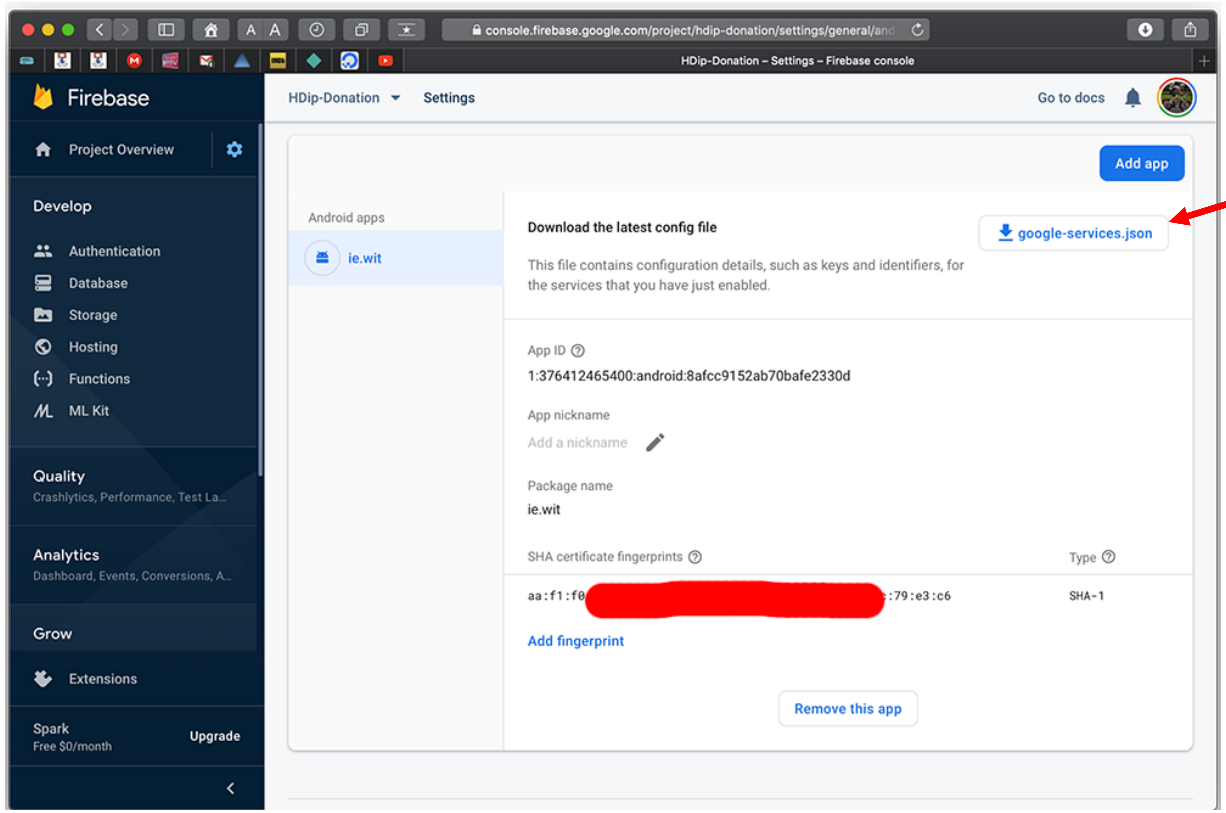
Client Secret

[redacted]

You can always manage your API credentials and usage later in the API Console.

DONE

## 3. Setup your Sign-In Method (via Web Link)



**Download the latest config file**

This file contains configuration details, such as keys and identifiers, for the services that you have just enabled.

App ID
1:376412465400:android:8afcc9152ab70bafe2330d

App nickname
Add a nickname

Package name
ie.wit

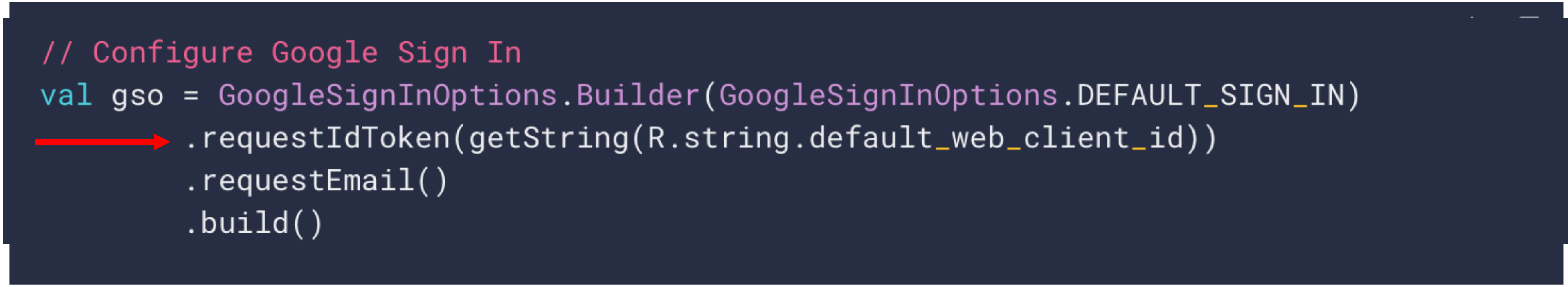
| SHA certificate fingerprints | Type |
|---|---|
| aa:f1:f0[redacted]:79:e3:c6 | SHA-1 |

## 4. Setup Authentication Flow & UI

https://github.com/firebase/quickstart-android

# 4. Configure Google Sign-in & GoogleSignInClient

In your sign-in activity's `onCreate` method, configure Google Sign-In to request the user data required by your app. For example, to configure Google Sign-In to request users' ID and basic profile information, create a `GoogleSignInOptions` object with the `DEFAULT_SIGN_IN` parameter. To request users' email addresses as well, create the `GoogleSignInOptions` object with the `requestEmail` option.



```
// Configure Google Sign In
val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .build()
```

## 4. Configure Google Sign-in & GoogleSignInClient

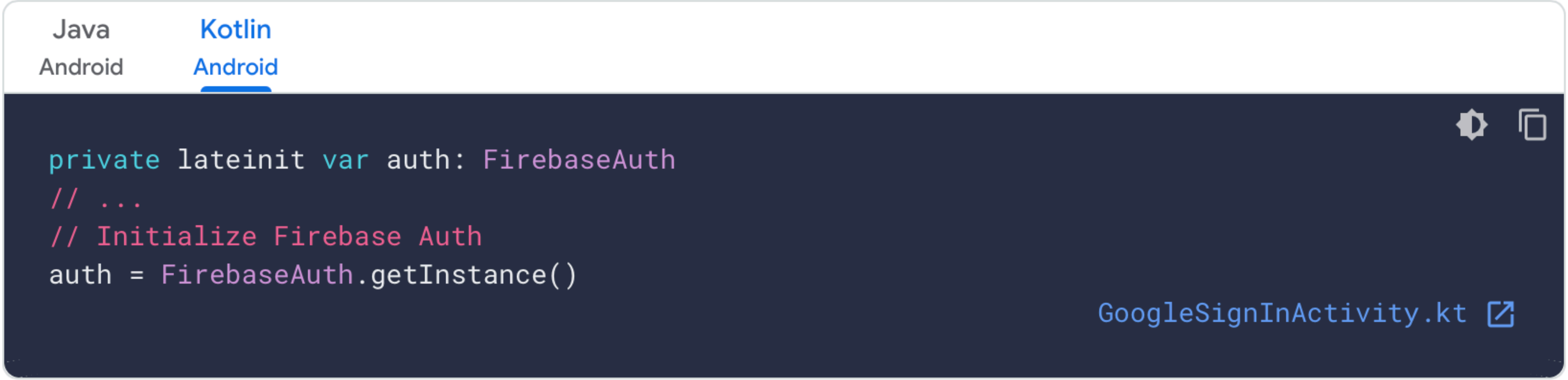
If you need to request additional scopes to access Google APIs, specify them with `requestScopes`. For the best user experience, on sign-in, only request the scopes that are required for your app to minimally function. Request any additional scopes only when you need them, so that your users see the consent screen in the context of an action they performed. See Requesting Additional Scopes.

Then, also in your sign-in activity's `onCreate` method, create a `GoogleSignInClient` object with the options you specified.

```
// Build a GoogleSignInClient with the options specified by gso.
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```

## 4. Get Firebase Auth Shared Instance

In your sign-in activity's `onCreate` method, get the shared instance of the `FirebaseAuth` object:



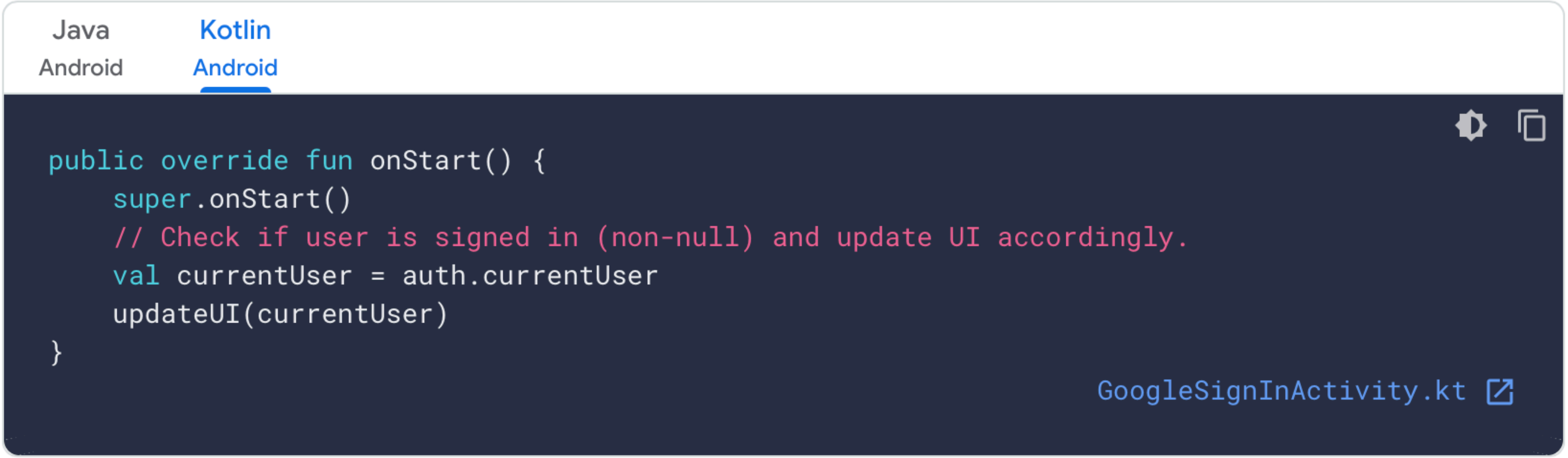
Java Android | Kotlin Android

```kotlin
private lateinit var auth: FirebaseAuth
// ...
// Initialize Firebase Auth
auth = FirebaseAuth.getInstance()
```

GoogleSignInActivity.kt

## 4. Check for Existing Signed In User

When initializing your Activity, check to see if the user is currently signed in:



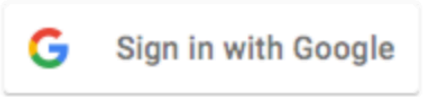
Java Android | Kotlin Android

```kotlin
public override fun onStart() {
    super.onStart()
    // Check if user is signed in (non-null) and update UI accordingly.
    val currentUser = auth.currentUser
    updateUI(currentUser)
}
```

GoogleSignInActivity.kt

## 4. Add the Google Sign-in button to your app

1. Add the `SignInButton` in your application's layout:



```xml
<com.google.android.gms.common.SignInButton
 android:id="@+id/sign_in_button"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content" />
```

2. **Optional**: If you are using the default sign-in button graphic instead of providing your own sign-in button assets, you can customize the button's size with the `setSize` method.

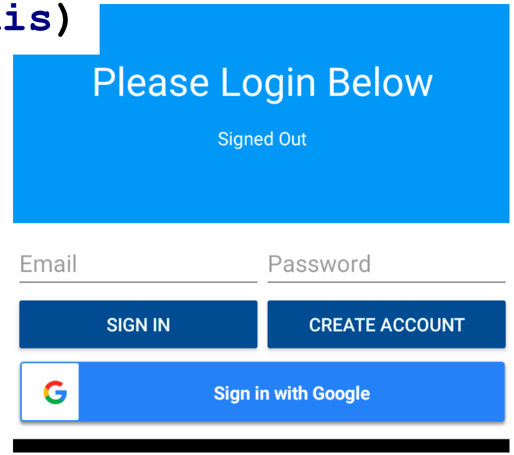
```java
// Set the dimensions of the sign-in button.
SignInButton signInButton = findViewById(R.id.sign_in_button);
signInButton.setSize(SignInButton.SIZE_STANDARD);
```

# 4. Add the Google Sign-in button to your app

3. In the Android activity (for example, in the `onCreate` method), register your button's `OnClickListener` to sign in the user when clicked:

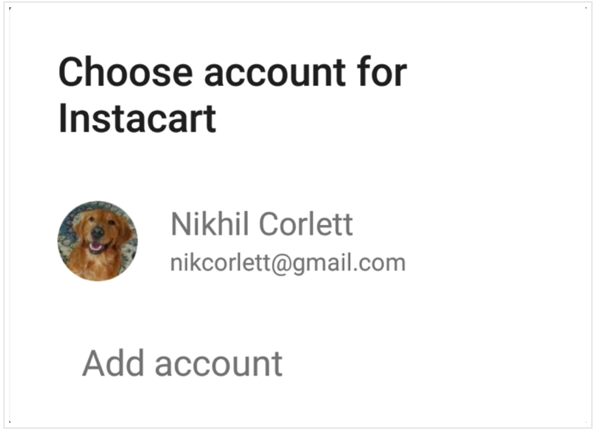
```
findViewById(R.id.sign_in_button).setOnClickListener(this);
```

```
sign_in_button.setOnClickListener(this)
```



## 4. Start the Sign-in Flow

1. In the activity's `onClick` method, handle sign-in button taps by creating a sign-in intent with the `getSignInIntent` method, and starting the intent with `startActivityForResult`.


Choose account for Instacart

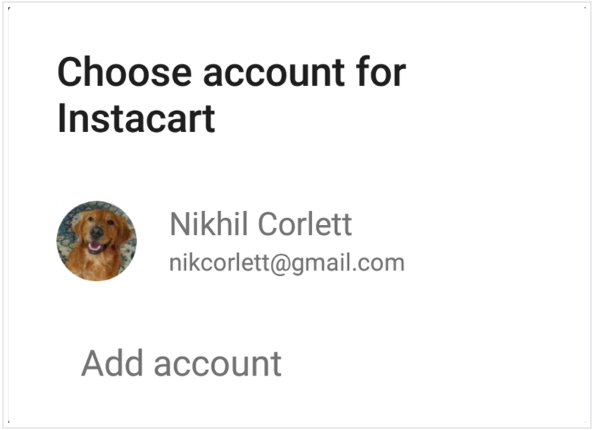
Nikhil Corlett
nikcorlett@gmail.com

Add account

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.sign_in_button:
            signIn();
            break;
        // ...
    }
}
```

## 4. Start the Sign-in Flow

1. In the activity's `onClick` method, handle sign-in button taps by creating a sign-in intent with the `getSignInIntent` method, and starting the intent with `startActivityForResult`.

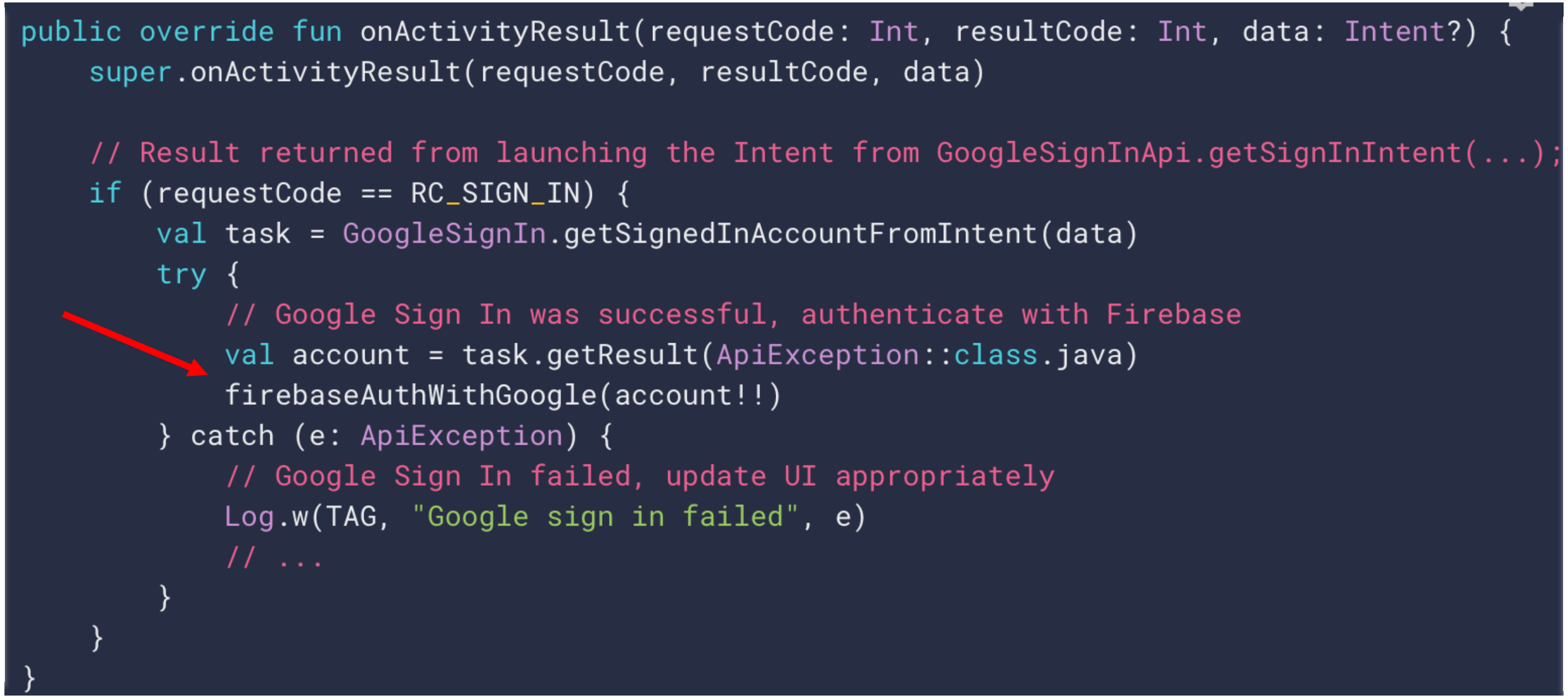


```
private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

Starting the intent prompts the user to select a Google account to sign in with. If you requested scopes beyond `profile`, `email`, and `openid`, the user is also prompted to grant access to the requested resources.

## 4. Start the Sign-in Flow

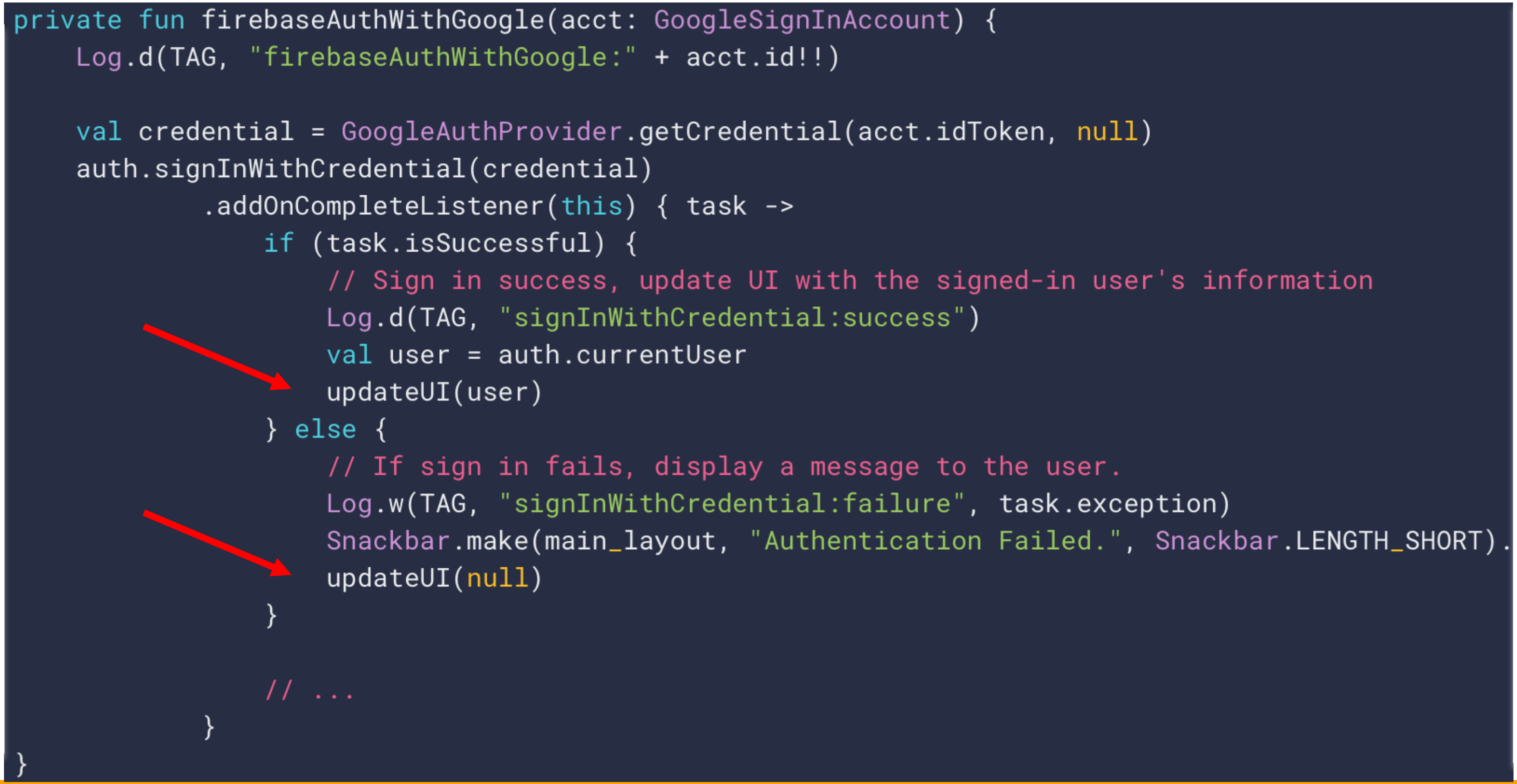
2. After the user signs in, you can get a `GoogleSignInAccount` object for the user in the activity's `onActivityResult` method.



```
public override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        val task = GoogleSignIn.getSignedInAccountFromIntent(data)
        try {
            // Google Sign In was successful, authenticate with Firebase
            val account = task.getResult(ApiException::class.java)
            firebaseAuthWithGoogle(account!!)
        } catch (e: ApiException) {
            // Google Sign In failed, update UI appropriately
            Log.w(TAG, "Google sign in failed", e)
            // ...
        }
    }
}
```

# 4. Start the Sign-in Flow

The `GoogleSignInAccount` object contains information about the signed-in user, such as the user's name.

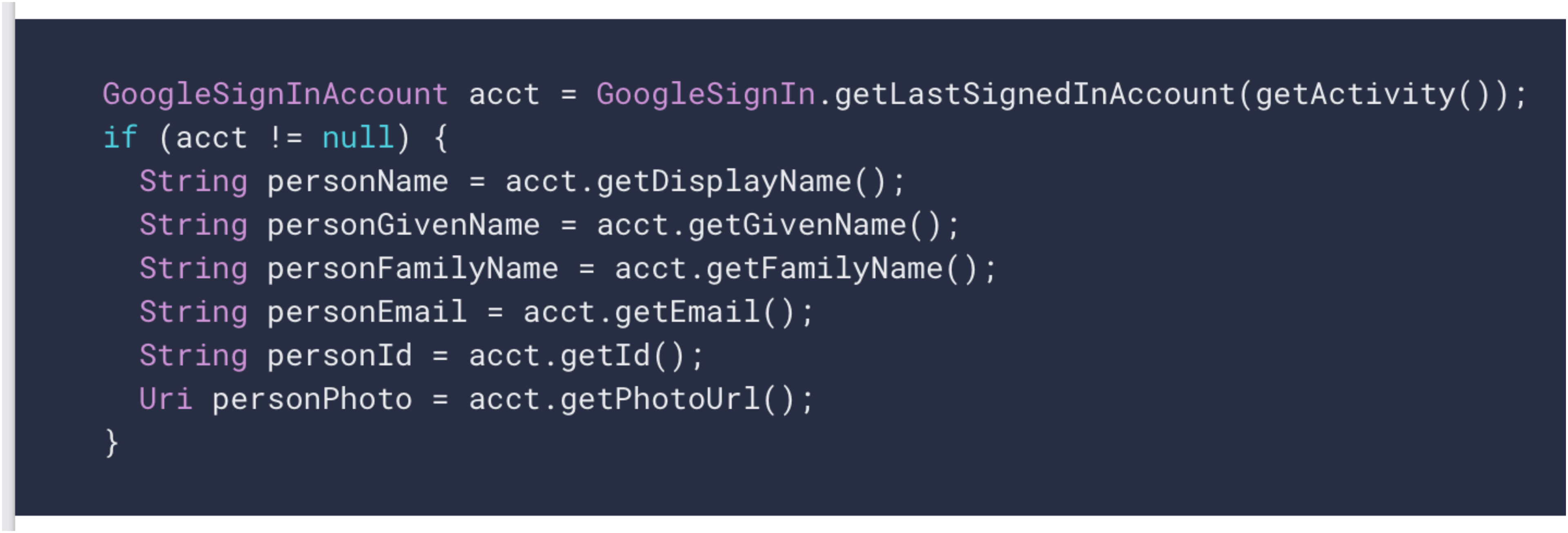


```
private fun firebaseAuthWithGoogle(acct: GoogleSignInAccount) {
    Log.d(TAG, "firebaseAuthWithGoogle:" + acct.id!!)

    val credential = GoogleAuthProvider.getCredential(acct.idToken, null)
    auth.signInWithCredential(credential)
            .addOnCompleteListener(this) { task ->
                if (task.isSuccessful) {
                    // Sign in success, update UI with the signed-in user's information
                    Log.d(TAG, "signInWithCredential:success")
                    val user = auth.currentUser
                    updateUI(user)
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "signInWithCredential:failure", task.exception)
                    Snackbar.make(main_layout, "Authentication Failed.", Snackbar.LENGTH_SHORT).
                    updateUI(null)
                }

                // ...
            }
}
```

## 4. Start the Sign-in Flow (Extra bits)

- ❑ You can also get the user's email address with **`getEmail()`**,
- ❑ The user's Google ID (for client-side use) with **`getId()`**, and
- ❑ An ID token for the user with **`getIdToken()`**
- ❑ If you need to pass the currently signed-in user to a backend server, send the ID token to your backend server and validate the token on the server

## 4. Start the Sign-in Flow (Extra bits)



```java
GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(getActivity());
if (acct != null) {
  String personName = acct.getDisplayName();
  String personGivenName = acct.getGivenName();
  String personFamilyName = acct.getFamilyName();
  String personEmail = acct.getEmail();
  String personId = acct.getId();
  Uri personPhoto = acct.getPhotoUrl();
}
```

We’ll look at what we do with the Google authenticated user in the Code walkthrough and the Lab

## References

https://console.firebase.google.com/
https://firebase.google.com/
https://github.com/firebase/quickstart-android
https://firebase.google.com/docs/auth/android/google-signin
https://developers.google.com/identity

Thanks.