

Mobile Application Development

Produced
by

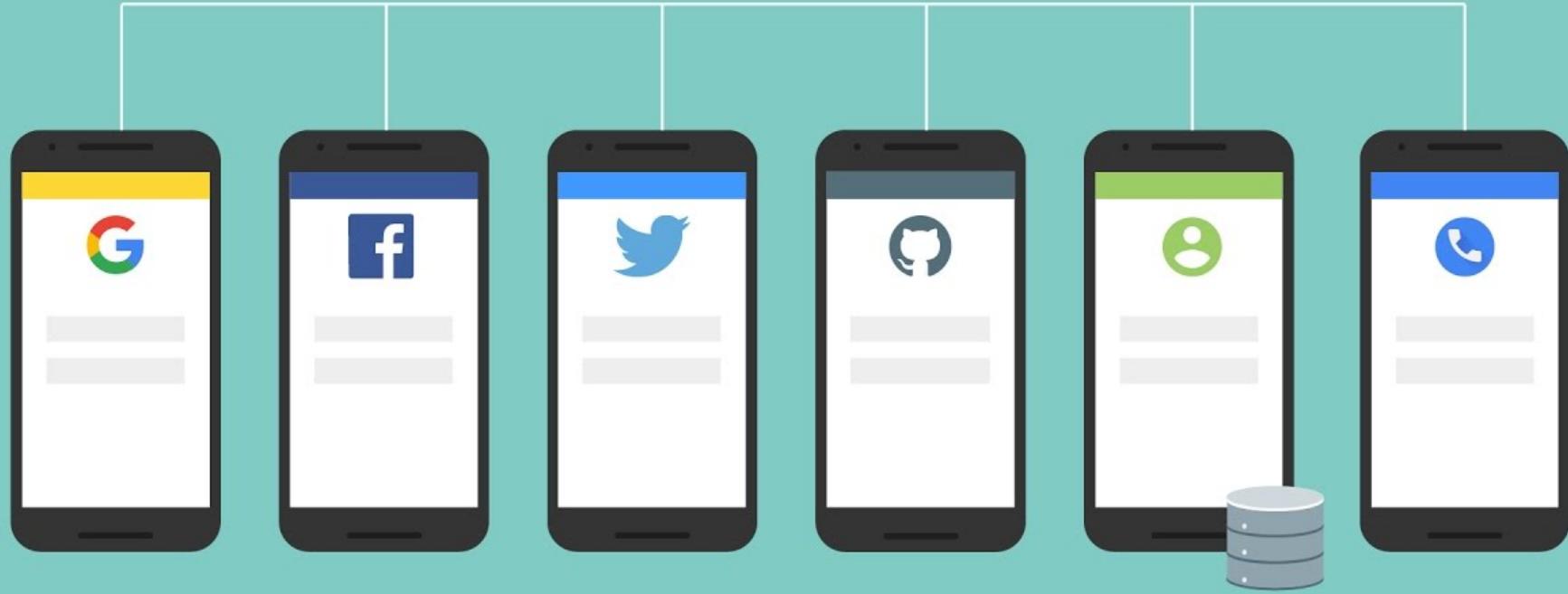
David Drohan (ddrohan@wit.ie)

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



Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE





Authentication

Overview

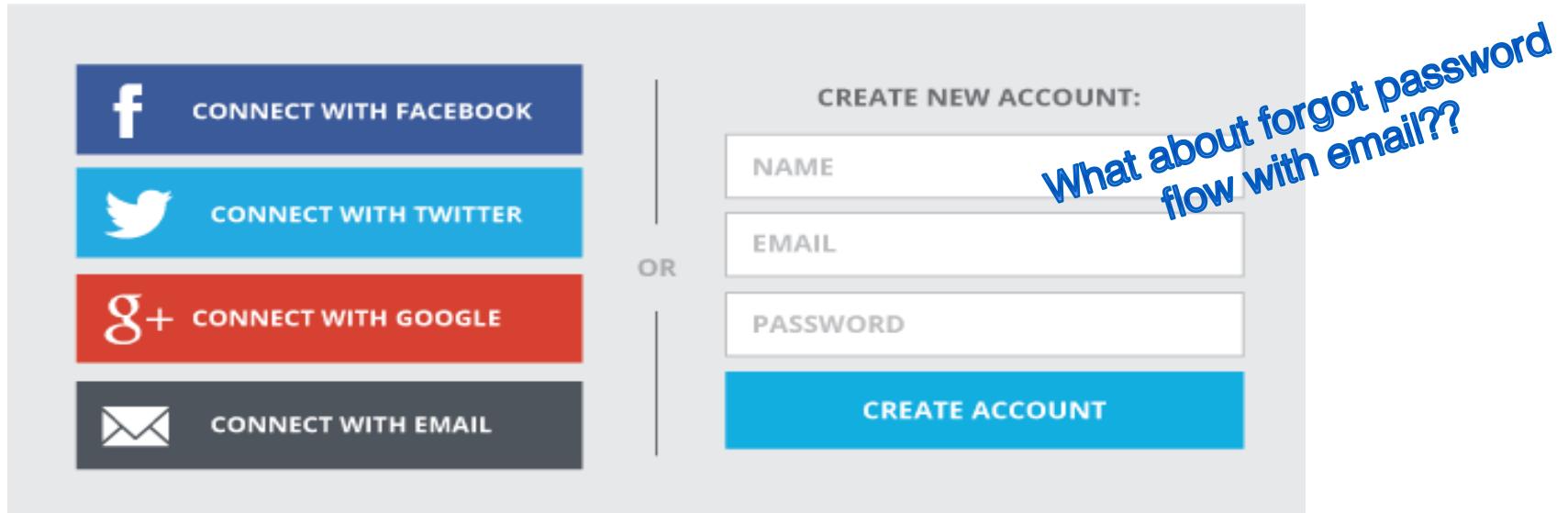


- ❑ Most apps need to know the identity of a user. Knowing a user's identity allows an app to securely save user data in the cloud and provide the same personalized experience across all of the user's devices
- ❑ **Firebase Authentication** provides backend services, easy-to-use SDKs, and ready-made UI libraries to authenticate users to your app. It supports authentication using passwords, phone numbers and popular federated identity providers (later)



Overview

- ❑ A type of screen present in almost all apps these days...



- ❑ How long would it take you to develop this???



Overview

- ❑ Integrate easily with popular identity providers like Google, Facebook, Twitter and more





Key Capabilities

- ❑ You can sign in users to your Firebase app either by using **FirebaseUI** as a complete drop-in auth solution or by using the **Firebase Authentication SDK** to manually integrate one or several sign-in methods into your app
- ❑ The **Firebase Authentication SDK** provides methods to create and manage users that use their email addresses and passwords to sign in
- ❑ Firebase Authentication also handles sending password reset emails



Firebase Authentication

- ❑ Of course you will need to register your app with individual service providers.
- ❑ Minimal client side handling, integrates seamlessly with firebase
- ❑ Ready made ‘forgot password’ flow with customizable email template

Firebase Authentication





1. Create your Firebase Project

The screenshot shows the official Firebase website at firebase.google.com. The top navigation bar includes links for Products, Use Cases, Pricing, Docs, Support, Language selection, and two buttons: "Go to console" and "Sign in". A large central image features two people interacting with floating blue squares against a blue background. Text on the left side reads: "Firebase helps mobile and web app teams succeed". Below this are two buttons: "Get started" and "Watch the video". At the bottom, three statements are listed: "Build apps fast, without managing infrastructure", "Backed by Google, trusted by top apps", and "One platform, with products that work better together".

Firebase helps mobile and web app teams succeed

Get started Watch the video

Build apps fast, without managing infrastructure

Backed by Google, trusted by top apps

One platform, with products that work better together



1. Create your Firebase Project

A screenshot of a web browser displaying the Firebase console at console.firebaseio.google.com. The page has a blue header with the title "Firebase console". Below the header, there's a "Recent projects" section featuring a woman icon and a "Recent projects" heading. A red box highlights the "Add project" button, which is a white button with a blue plus sign and the text "Add project" below it. Below this button is a link "Explore a demo project". To the right of the "Add project" button are three project cards: "HDip-Donation" (hdip-donation), "tutors" (tutors-177ed), and "CoffeeMate FullFat Server" (coffeemate-fullfat-serve-46ff4). Further down the page, there's a section titled "All Firebase projects" with three more cards: "CoffeeMateFBI" (coffeematefbi), "DonationWeb" (donationweb-vue), and another partially visible card.



1. Create your Firebase Project

The screenshot shows a web browser window for 'console.firebaseio.google.com'. The title bar says 'Create a project(Step 1 of 3)'. The main content area has a heading 'Let's start with a name for your project' and a 'Project name' input field containing 'HDip-Donation'. A red box highlights the top right corner of the browser window. Below the input field is a small button with a pencil icon labeled 'hdip-donation'. At the bottom is a large blue 'Continue' button.



1. Create your Firebase Project

The screenshot shows the Firebase console interface. On the left, a sidebar menu is open under the 'Develop' section, which is highlighted with a red box. The menu items include Authentication, Database, Storage, Hosting, Functions, and ML Kit. Below this, other sections like Quality, Analytics, Grow, and Extensions are visible. At the bottom of the sidebar, it says 'Spark Free \$0/month' and has an 'Upgrade' button. The main content area is titled 'HDip-Donation' and shows a 'Spark plan'. It features a large blue background with the text 'Get started by adding Firebase to your app' and icons for iOS, Android, and web development. A call-to-action 'Add an app to get started' is present. At the bottom, there's a banner with the text 'Store and sync app data in milliseconds' and some small icons.



1. Create your Firebase Project

The screenshot shows the Firebase console interface for the project "HDip-Donation". A modal window titled "Firebase pricing plans" is displayed, listing three plan options: Spark, Flame, and Blaze. The "Spark" plan is highlighted with a red border and is described as "Free \$0/month". It includes usage quotas for various services and is included in all plans. The "Flame" plan is described as "Fixed \$25/month" and includes increased database and storage space, as well as pay-as-you-go billing. The "Blaze" plan is described as "Pay as you go" and includes free usage calculated daily. Each plan has a "See full plan details" link and a "Select plan" button.

Firebase pricing plans

Spark
Free \$0/month

Usage quotas for Database, Firestore, Storage, Functions, Phone Auth, Hosting and Test Lab

Ability to extend your project with Google Cloud Platform

Included in all plans

Analytics, Notifications, Crash Reporting, support and more

[See full plan details](#)

Flame
Fixed \$25/month

Increased Database, Firestore, Storage, Phone Auth, Hosting and Test Lab space. Outbound connections for Functions.

Ability to extend your project with Google Cloud Platform

Included in all plans

Analytics, Notifications, Crash Reporting, support and more

[See full plan details](#)

Blaze
Pay as you go

Includes free usage, calculated daily. After, pay only for what your project uses.

Ability to extend your project with Google Cloud Platform

Included in all plans

Analytics, Notifications, Crash Reporting, support and more

[See full plan details](#)

Current Plan

Select plan



2. Setup your Sign-In Method

The screenshot shows the Firebase console interface for a project named "HDip-Donation". The left sidebar has a red box around the "Authentication" section under the "Develop" heading. The main content area is titled "Authentication" and includes tabs for "Users", "Sign-in method", "Templates", and "Usage". A search bar at the top allows searching by email address, phone number, or user UID. Below the search bar is a table header with columns: Identifier, Providers, Created, Signed In, and User UID (sorted by User UID). A large central callout box contains an icon of a person's ID badge and the text: "Authenticate and manage users from a variety of providers without server-side code". It also includes links to "Learn more" and "View the docs", and a blue button labeled "Set up sign-in method". At the bottom right, there is a link to "Need help setting up your app? iOS" and a "Get started" button.

Open "<https://console.firebaseio.google.com/project/hdip-donation/authentication/users>" in a new tab behind the current one



2. Setup your Sign-In Method

The screenshot shows the Firebase Authentication console interface. On the left, there's a sidebar with 'Develop' and 'Quality' sections. The 'Authentication' item under 'Develop' is selected. The main area is titled 'Authentication' and has tabs for 'Users', 'Sign-in method' (which is highlighted with a red border), 'Templates', and 'Usage'. Below this, a table lists various sign-in providers: Email/Password (disabled), Phone, Google, Play Games, Game Center (Beta), Facebook, Twitter, GitHub, Yahoo, and Microsoft. Each row shows the provider icon, name, and status (Disabled). A red box highlights the first row, 'Email/Password'.

Provider	Status
Email/Password	Disabled
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center <small>Beta</small>	Disabled
Facebook	Disabled
Twitter	Disabled
Github	Disabled
Yahoo	Disabled
Microsoft	Disabled



2. Setup your Sign-In Method

The screenshot shows the Firebase console interface for setting up authentication methods. The left sidebar includes sections for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, A...), and Grow (Extensions). The main content area is titled 'Authentication' under the project 'HDip-Donation'. The 'Sign-in method' tab is highlighted with a red box. Below it, the 'Sign-in providers' section lists several options: Email/Password (status: Enabled), Email link (passwordless sign-in) (status: Disabled), Phone (status: Disabled), Google (status: Disabled), Play Games (status: Disabled), and Game Center (status: Beta, Disabled). A 'Save' button is visible at the bottom right of the provider list.

Provider	Status
Email/Password	Enable
Email link (passwordless sign-in)	Disable
Phone	Disabled
Google	Disabled
Play Games	Disabled
Game Center	Beta Disabled



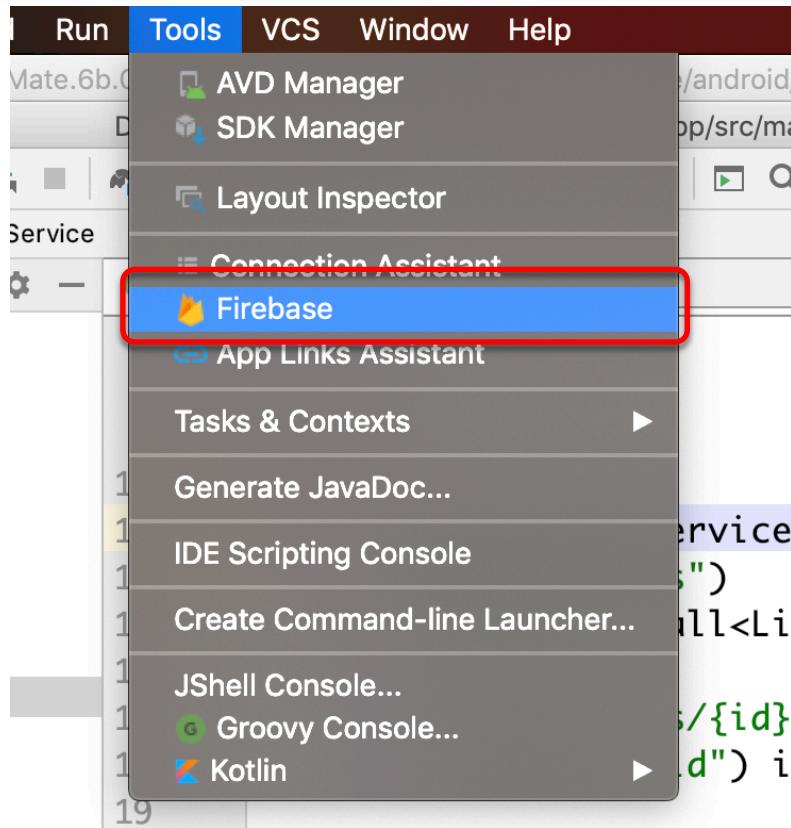
2. Setup your Sign-In Method

The screenshot shows the Firebase Authentication console interface. The left sidebar includes sections for Project Overview, Develop (Authentication, Database, Storage, Hosting, Functions, ML Kit), Quality (Crashlytics, Performance, Test Lab), Analytics (Dashboard, Events, Conversions, A/B Testing), Grow, Extensions, and Spark (Free \$0/month, Upgrade). The main content area is titled "Authentication" under "HDip-Donation". The navigation bar at the top has tabs for Users, Sign-in method (which is selected and highlighted with a red border), Templates, and Usage. The "Sign-in providers" section lists various authentication methods: Email/Password (Enabled, highlighted with a red box), Phone, Google, Play Games, Game Center (Beta), Facebook, Twitter, GitHub, Yahoo, and Microsoft. All other providers are listed as Disabled.

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



3. Setup Firebase in your App





3. Setup Firebase in your App

The screenshot shows the Android Studio interface with the 'Assistant' tab selected. On the left, there's a code editor with some Java code related to a donation model. On the right, the 'Assistant' panel lists various Firebase services. The 'Authentication' service is highlighted with a red box. Other services listed include Analytics, Cloud Messaging, Realtime Database, Storage, Remote Config, Test Lab, App Indexing, and Dynamic Links. Each service has a brief description and a 'More info' link.

```
atationModel>>
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}

String): Call<DonationModel>
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}

String): Call<DonationModel>
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}

String,
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}

//donationweb-hdip<
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}

Service {
    @Override
    public void onCallResult(Call<DonationModel> call, CallResult result) {
        if (result.isSuccess()) {
            DonationModel donationModel = call.body();
            Log.d("Donation", "Received donation: " + donationModel);
        } else {
            Log.e("Donation", "Error receiving donation: " + result.error());
        }
    }
}
```

Assistant Firebase

Firebase

Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

- ▶ **Analytics**
Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)
- ▶ **Cloud Messaging**
Deliver and receive messages and notifications reliably across cloud and device. [More info](#)
- ▶ **Authentication**
Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)
- ▶ **Realtime Database**
Store and sync data in realtime across all connected clients. [More info](#)
- ▶ **Storage**
Store and retrieve large files like images, audio, and video without writing server-side code. [More info](#)
- ▶ **Remote Config**
Customize and experiment with app behavior using cloud-based configuration parameters. [More info](#)
- ▶ **Test Lab**
Test your apps against a wide range of physical devices hosted in Google's cloud. [More info](#)
- ▶ **App Indexing**
Get your app content into Google Search. [More info](#)
- ▶ **Dynamic Links**
Create web URLs that can be shared to drive app installs and deep-linked into relevant content of your app. [More info](#)



3. Setup Firebase in your App

The screenshot shows the Firebase console interface. At the top, there are tabs for 'Assistant' and 'Firebase'. Below the tabs, the word 'Analytics' is visible. A red box highlights the 'Authentication' section, which includes sub-options for 'Email and password authentication', 'Realtime Database', 'Storage', 'Remote Config', 'Test Lab', 'App Indexing', and 'Dynamic Links'. Each option has a brief description and a 'More info' link.

- ▶ **Analytics**
Measure user activity and engagement with free, easy, and unlimited analytics.
[More info](#)
- ▶ **Cloud Messaging**
Deliver and receive messages and notifications reliably across cloud and device.
[More info](#)
- ▶ **Authentication**
Sign in and manage users with ease, accepting emails, Google Sign-in, Facebook and other login providers.
[More info](#)
 - **Email and password authentication**
 - **Realtime Database**
Store and sync data in realtime across all connected clients.
[More info](#)
 - **Storage**
Store and retrieve large files like images, audio, and video without writing server-side code.
[More info](#)
 - **Remote Config**
Customize and experiment with app behavior using cloud-based configuration parameters.
[More info](#)
 - **Test Lab**
Test your apps against a wide range of physical devices hosted in Google's cloud.
[More info](#)
 - **App Indexing**
Get your app content into Google Search.
[More info](#)
 - **Dynamic Links**



3. Setup Firebase in your App

The screenshot shows the 'Email and password authentication' section of the Firebase documentation. A red circle highlights the second step, 'Add Firebase Authentication to your app'. This step includes a 'Launch in browser' button and a 'Connect to Firebase' button. Below it, there's a note about enabling authentication in the Firebase console and a link to the Sign-in Method page. The rest of the page contains steps for connecting the app to Firebase and checking the current auth state.

Email and password authentication

You can use Firebase Authentication to let your users sign in with their email addresses and passwords, and to manage your app's password-based accounts. This tutorial helps you set up an email and password system and then access information about the user.

Launch in browser

① Connect your app to Firebase

② Add Firebase Authentication to your app

To use an authentication provider, you need to enable it in the [Firebase console](#). Go to the Sign-in Method page in the Firebase Authentication section to enable Email/Password sign-in and any other identity providers you want for your app.

③ Check current auth state

Declare an instance of FirebaseAuth

```
private FirebaseAuth mAuth;
```

In the onCreate() method, initialize the FirebaseAuth instance.

```
// Initialize Firebase Auth
mAuth = FirebaseAuth.getInstance();
```

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

```
@Override
protected void onStart() {
```



3. Setup Firebase in your App

Connect to Firebase

Firebase

Create new Firebase project [What's this?](#) Signed in as **daveydrohan@gmail.com** [Sign out](#)

Donation

Choose an existing Firebase or Google project

CoffeeMate FullFat Server	1 Android app(s) connected
CoffeeMateFBI	2 Android app(s) connected
DonationWeb	
DonationWeb-SSD	
HDip-Donation	
PaceMaker	
tutors	

[What's this?](#)

By default, your Firebase Analytics data will enhance other Firebase features and Google products. You can control how your Firebase Analytics data is shared in your settings at anytime.
[Learn more](#)

Cancel **Connect to Firebase**



3. Setup Firebase in your App

Add Firebase Authentication to your app

Performing this action will make the following changes to your project.

build.gradle (project-level)

```
Add Firebase Gradle buildscript dependency  
classpath 'com.google.gms:google-services:4.2.0'
```

app/build.gradle

```
Add Firebase plugin for Gradle  
apply plugin: 'com.google.gms.google-services'
```

build.gradle will include these new dependencies:
compile 'com.google.firebaseio:firebase-auth:16.0.5'

This will also enable the firebase-core library which includes Firebase Analytics. [Learn more](#)



4. Introduce Authentication Flow

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



4. Introduce Authentication Flow – Create Account

To create a new user account with a password, complete the following steps in your app's sign-in activity:

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

Java [Kotlin](#)
Android [Android](#)

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

EmailPasswordActivity



4. Introduce Authentication Flow – Create Account

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

Java

Android

Kotlin

Android

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

EmailPasswordActivity

- When a new user signs up using your app's sign-up form, complete any new account validation steps that your app requires, such as verifying that the new account's password was correctly typed and meets your complexity requirements.



4. Introduce Authentication Flow – Create Account

Java Kotlin
Android Android

```
auth.createUserWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) {
            // Sign in success, update UI with the signed-in user's information
            Log.d(TAG, "createUserWithEmail:success")
            val user = auth.currentUser
            updateUI(user)
        } else {
            // If sign in fails, display a message to the user.
            Log.w(TAG, "createUserWithEmail:failure", task.exception)
            Toast.makeText(baseContext, "Authentication failed.",
                Toast.LENGTH_SHORT).show()
            updateUI(null)
        }
    }
}
```

EmailPasswordActivity.kt

If the new account was created, the user is also signed in. In the callback, you can use the `getCurrentUser` method to get the user's account data.



4. Introduce Authentication Flow – Sign In a User

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

Java

Kotlin

Android

Android

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

EmailPasswordActivity



4. Introduce Authentication Flow – Sign In a User

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

Java

Kotlin

Android

Android

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

EmailPasswordActivity

3. When a user signs in to your app, pass the user's email address and password to `signInWithEmailAndPassword`:



Java **Kotlin**
Android Android

```
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(this) { task ->
        if (task.isSuccessful) {
            // Sign in success, update UI with the signed-in user's information
            Log.d(TAG, "signInWithEmailAndPassword:success")
            val user = auth.currentUser
            updateUI(user)
        } else {
            // If sign in fails, display a message to the user.
            Log.w(TAG, "signInWithEmailAndPassword:failure", task.exception)
            Toast.makeText(baseContext, "Authentication failed.",
                Toast.LENGTH_SHORT).show()
            updateUI(null)
        }
    }
}
```

EmailPasswordActivity.kt

If sign-in succeeded, you can use the returned `FirebaseUser` to proceed.



We'll look at what we do with the authenticated user in the Code walk through and the Lab



References

<https://console.firebaseio.google.com/>

<https://firebase.google.com/>

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

Thanks.

