Firebase › Docs › Guides

Authenticate with Firebase Anonymously on Android

Contents

You can use Firebase Authentication to create and use temporary anonymous accounts to authenticate with Firebase. These temporary anonymous accounts can be used to allow users who haven't yet signed up to your app to work with data protected by security rules. If an anonymous user decides to sign up to your app, you can link their sign-in credentials to the anonymous account so that they can continue to work with their protected data in future sessions.

Before you begin

1. Add Firebase to your Android project.
2. Add the dependency for Firebase Authentication to your app-level `build.gradle` file:

   ```
   implementation 'com.google.firebase:firebase-auth:16.0.2'
   ```

3. If you haven't yet connected your app to your Firebase project, do so from the Firebase console.
4. Enable anonymous auth:
   a. In the Firebase console, open the **Auth** section.
   b. On the **Sign-in Methods** page, enable the **Anonymous** sign-in method.

Authenticate with Firebase anonymously

When a signed-out user uses an app feature that requires authentication with Firebase, sign in the user anonymously by completing the following steps:

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

```
private FirebaseAuth mAuth;
// ...
mAuth = FirebaseAuth.getInstance();
```

AnonymousAuthActivity.java

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

```
@Override
public void onStart() {
    super.onStart();
    // Check if user is signed in (non-null) and update UI accordingly.
    FirebaseUser currentUser = mAuth.getCurrentUser();
    updateUI(currentUser);
}
```

AnonymousAuthActivity.java

3. Finally, call `signInAnonymously` to sign in as an anonymous user:

```
mAuth.signInAnonymously()
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    // Sign in success, update UI with the signed-in user's
                    Log.d(TAG, "signInAnonymously:success");
                    FirebaseUser user = mAuth.getCurrentUser();
                    updateUI(user);
                } else {
                    // If sign in fails, display a message to the user.
                    Log.w(TAG, "signInAnonymously:failure", task.getExceptio
                    Toast.makeText(AnonymousAuthActivity.this, "Authenticati
                            Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }

                // ...
            }
        });
```

AnonymousAuthActivity.java

If sign-in succeeds you can use the `getCurrentUser` method to get the user's account data.

★ To protect your project from abuse, Firebase limits the number of new email/password and anonymous sign-ups that your application can have from the same IP address in a short period of time. You can request and schedule temporary changes to this quota from the Firebase console.

Convert an anonymous account to a permanent account

When an anonymous user signs up to your app, you might want to allow them to continue their work with their new account—for example, you might want to make the items the user added to their shopping cart before they signed up available in their new account's shopping cart. To do so, complete the following steps:

1. When the user signs up, complete the sign-in flow for the user's authentication provider up to, but not including, calling one of the `FirebaseAuth.signInWith` methods. For example, get the user's Google ID token, Facebook access token, or email address and password.
2. Get an `AuthCredential` for the new authentication provider:

Google Sign-In

```
AuthCredential credential = GoogleAuthProvider.getCredential(googleIdToken,
```

Facebook Login

```
AuthCredential credential = FacebookAuthProvider.getCredential(token.getToke
```

Email-password sign-in

```
AuthCredential credential = EmailAuthProvider.getCredential(email, password)
```

3. Pass the `AuthCredential` object to the sign-in user's `linkWithCredential` method:

```
mAuth.getCurrentUser().linkWithCredential(credential)
        .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if (task.isSuccessful()) {
                    Log.d(TAG, "linkWithCredential:success");
                    FirebaseUser user = task.getResult().getUser();
                    updateUI(user);
                } else {
                    Log.w(TAG, "linkWithCredential:failure", task.getExcepti
                    Toast.makeText(AnonymousAuthActivity.this, "Authenticati
                            Toast.LENGTH_SHORT).show();
                    updateUI(null);
                }

                // ...
            }
        });
```

AnonymousAuthActivity.java

If the call to `linkWithCredential` succeeds, the user's new account can access the anonymous account's Firebase data.

★ This technique can also be used to link any two accounts.

Next steps

Now that users can authenticate with Firebase, you can control their access to data in your Firebase database using Firebase rules.

Last updated June 28, 2018.

Guides

Reference

Samples

Libraries

GitHub

Stay Connected

Blog

Firebase Summit

Facebook

Google+

Twitter

YouTube

Support

Contact Support

Brand Guidelines

Stack Overflow

Google Group

Release Notes

FAQs

Legacy Documentation

Google Developers

Android

Chrome

Firebase