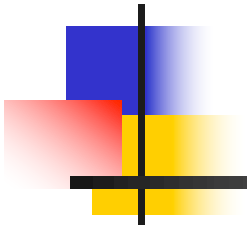
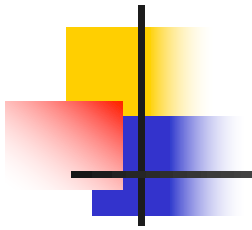


In-class project: Android App and Firebase

Plan of This In-class Project

- You will learn how to create and configure a Firebase project
- Create an initial user
- Create a simple Android Studio project and configure it for Firebase use
- Create an app which will connect to your Firebase database and read and display simple string message



Firebase Basic Workflow

- Basic steps to use Firebase are:
  - Create a project in the Firebase console
  - Configure database rules (authentication)
  - Add the Firebase SDK to the AS project
  - Implement reading/writing in the app

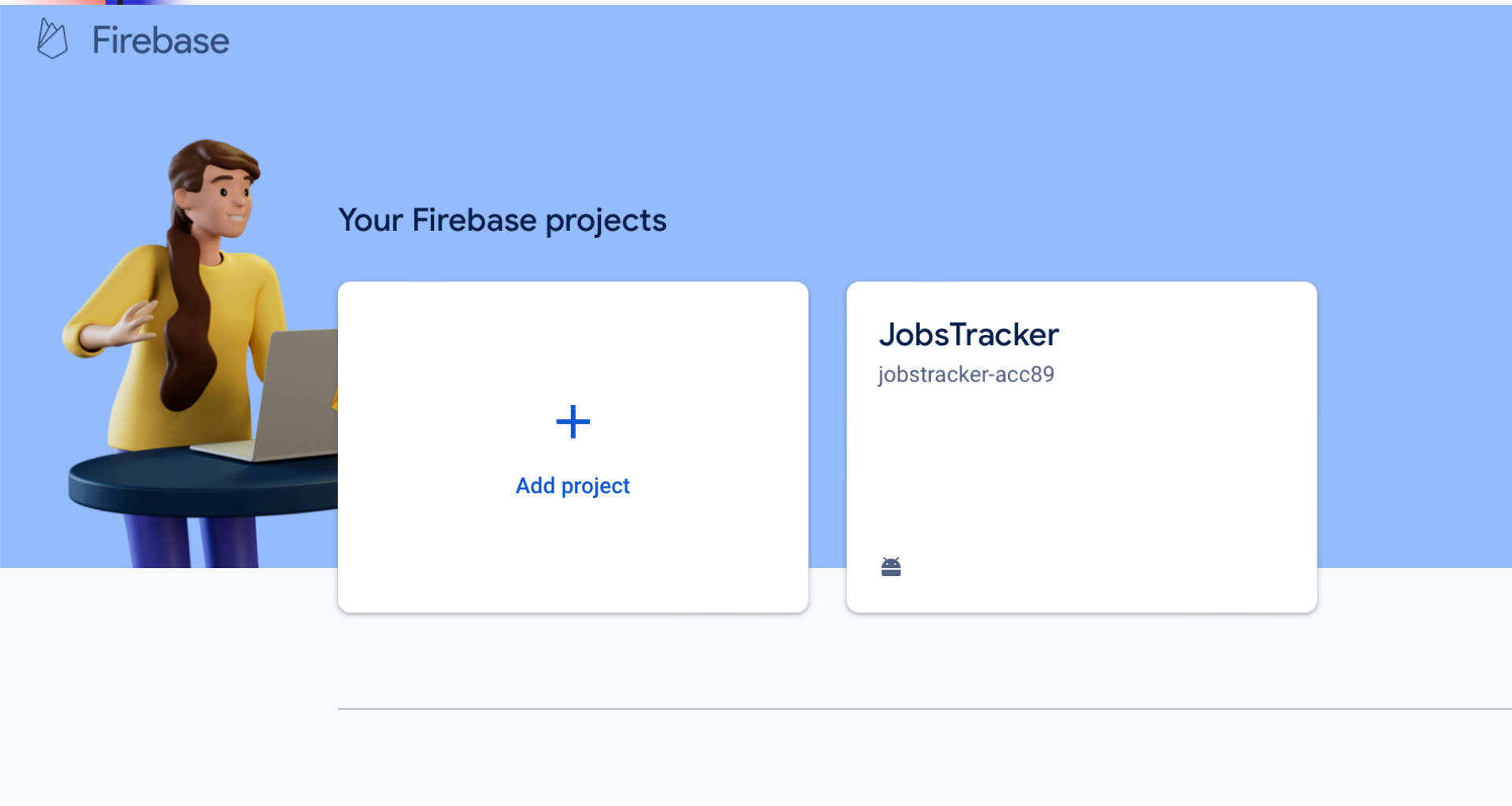
Firebase: Create a New Project

1. Sign-in to Google, if not signed-in yet, and open the Firebase console at:

   https://console.firebase.google.com

2. Create and then configure a project in the Firebase console:
   - Click on `Add project` and follow the steps, as shown in the following *multiple* slides
   - We will create a `SuperApp` Firebase project to work with our `SuperApp` Android app

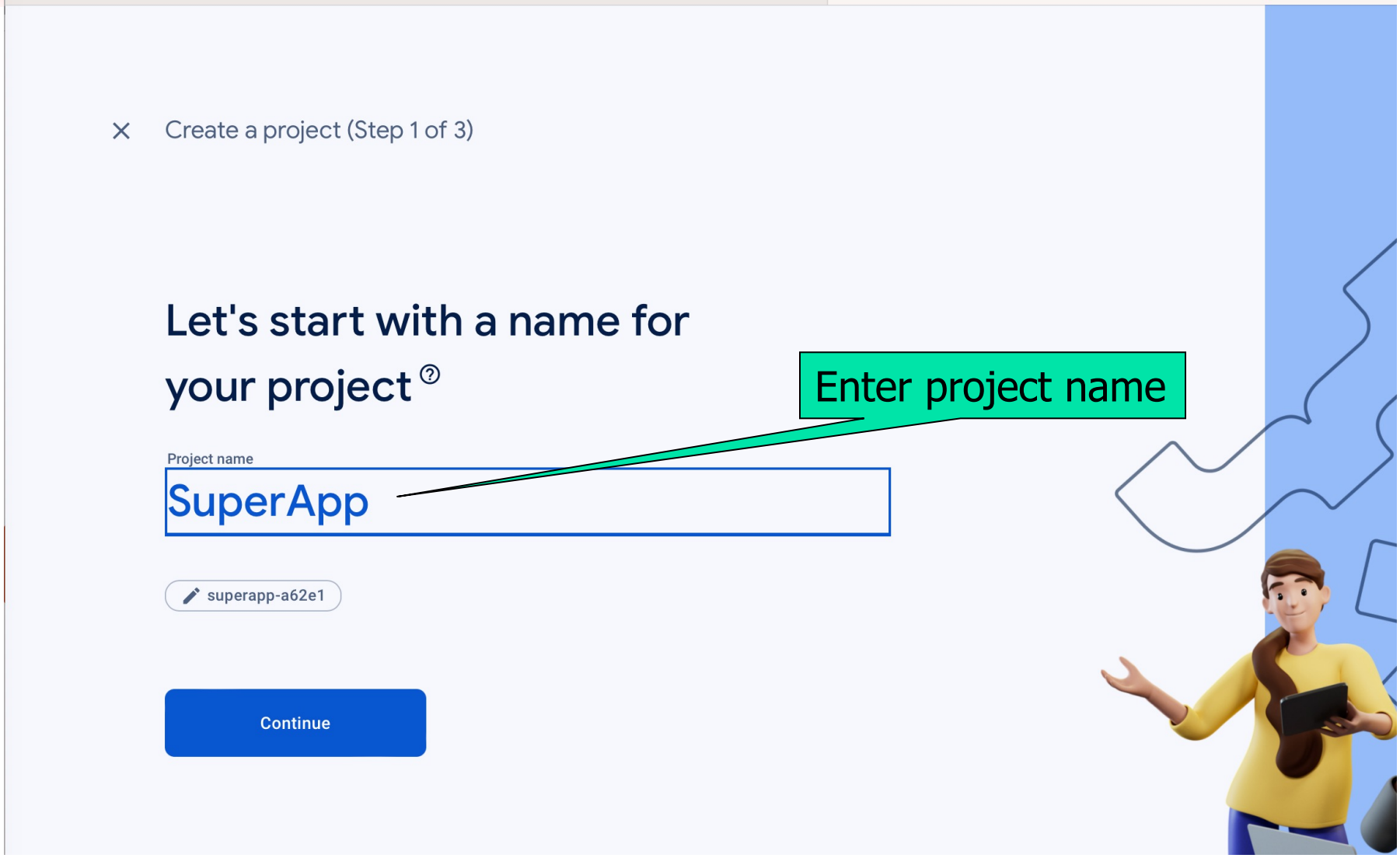
Firebase: Create a New Project



Screenshot of Firebase console at console.firebase.google.com

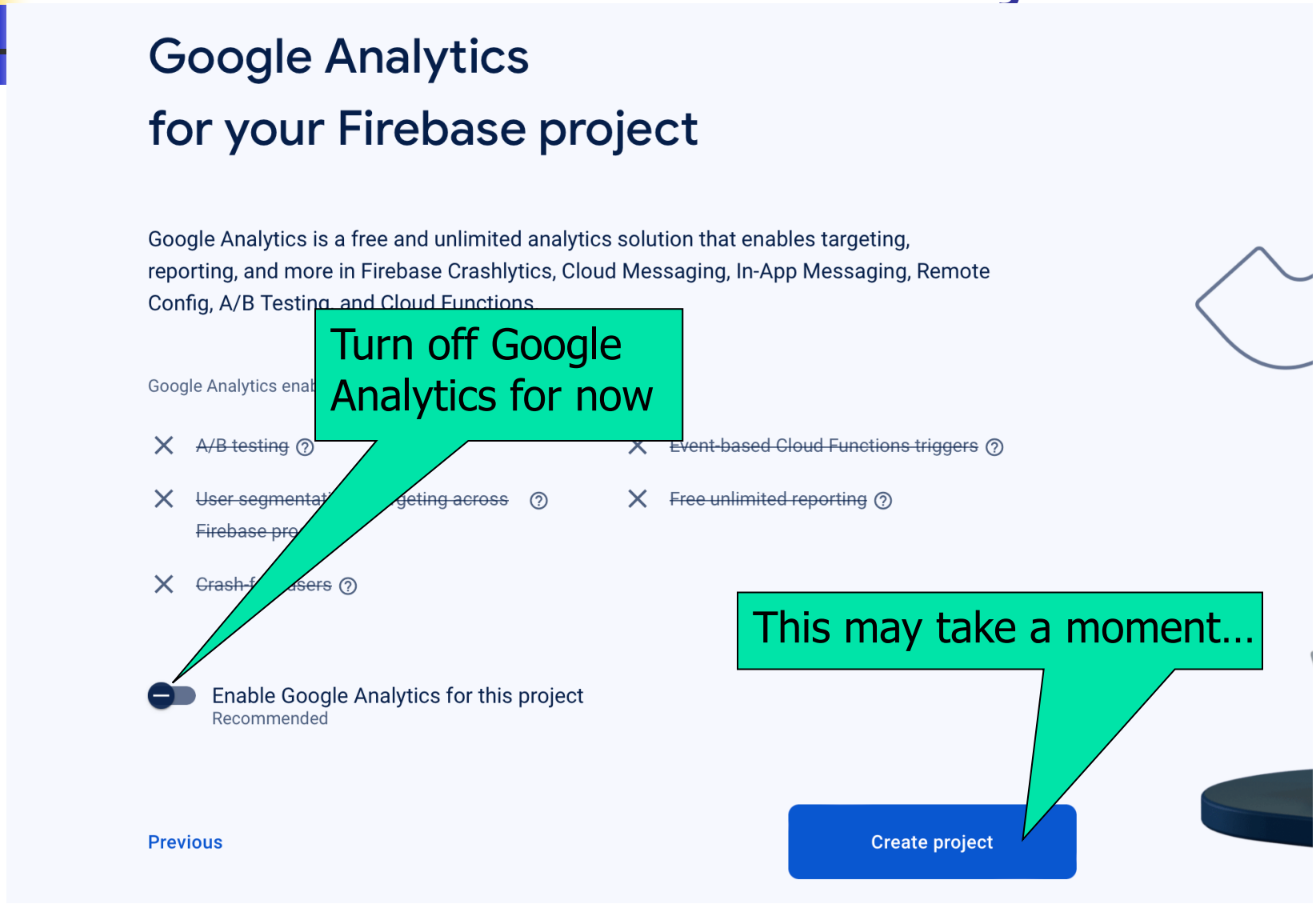
Click on **Add project** (JobsTracker has already been created)

Firebase: Create a New Project



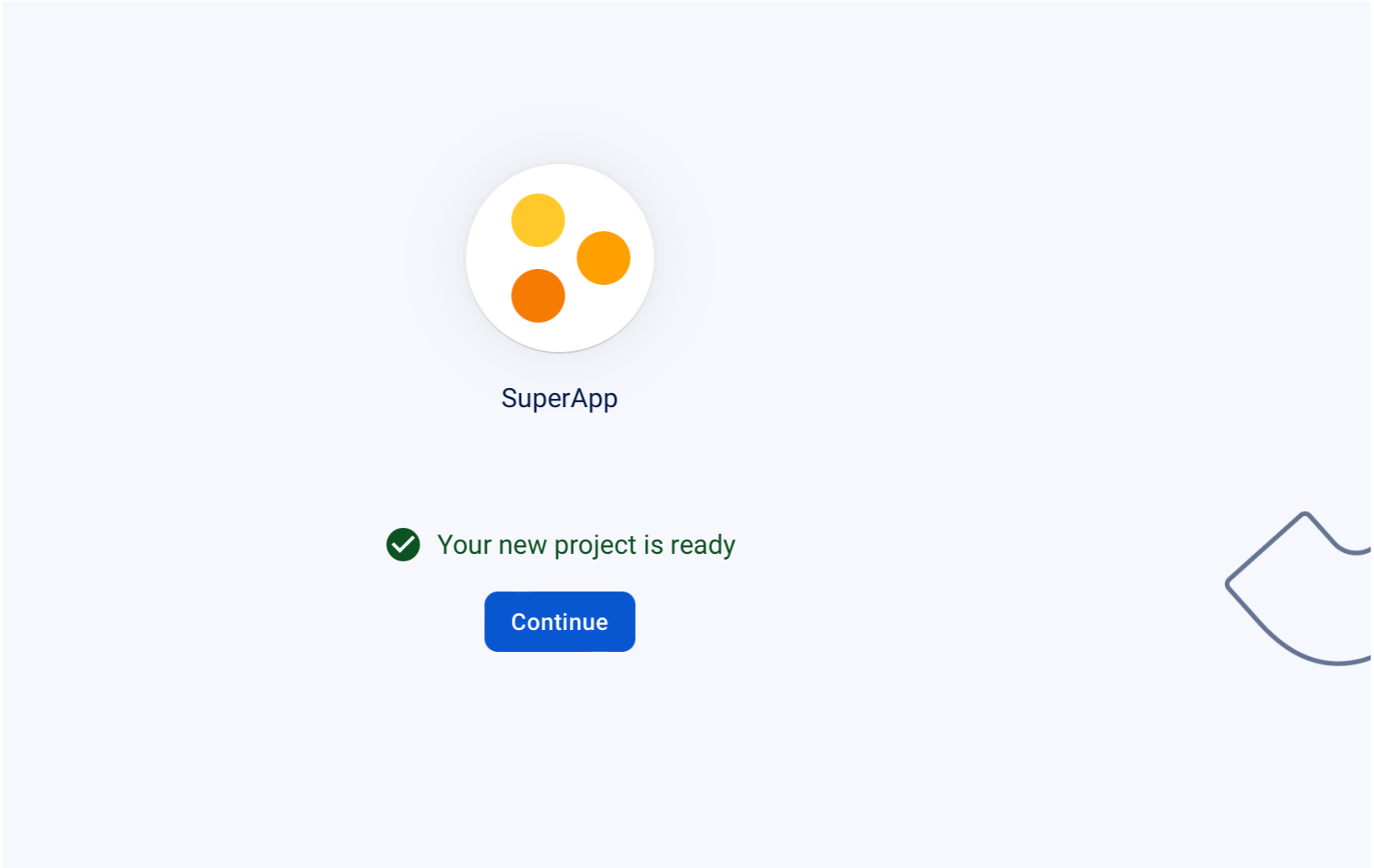
Then, click Continue

Firebase: Create a New Project



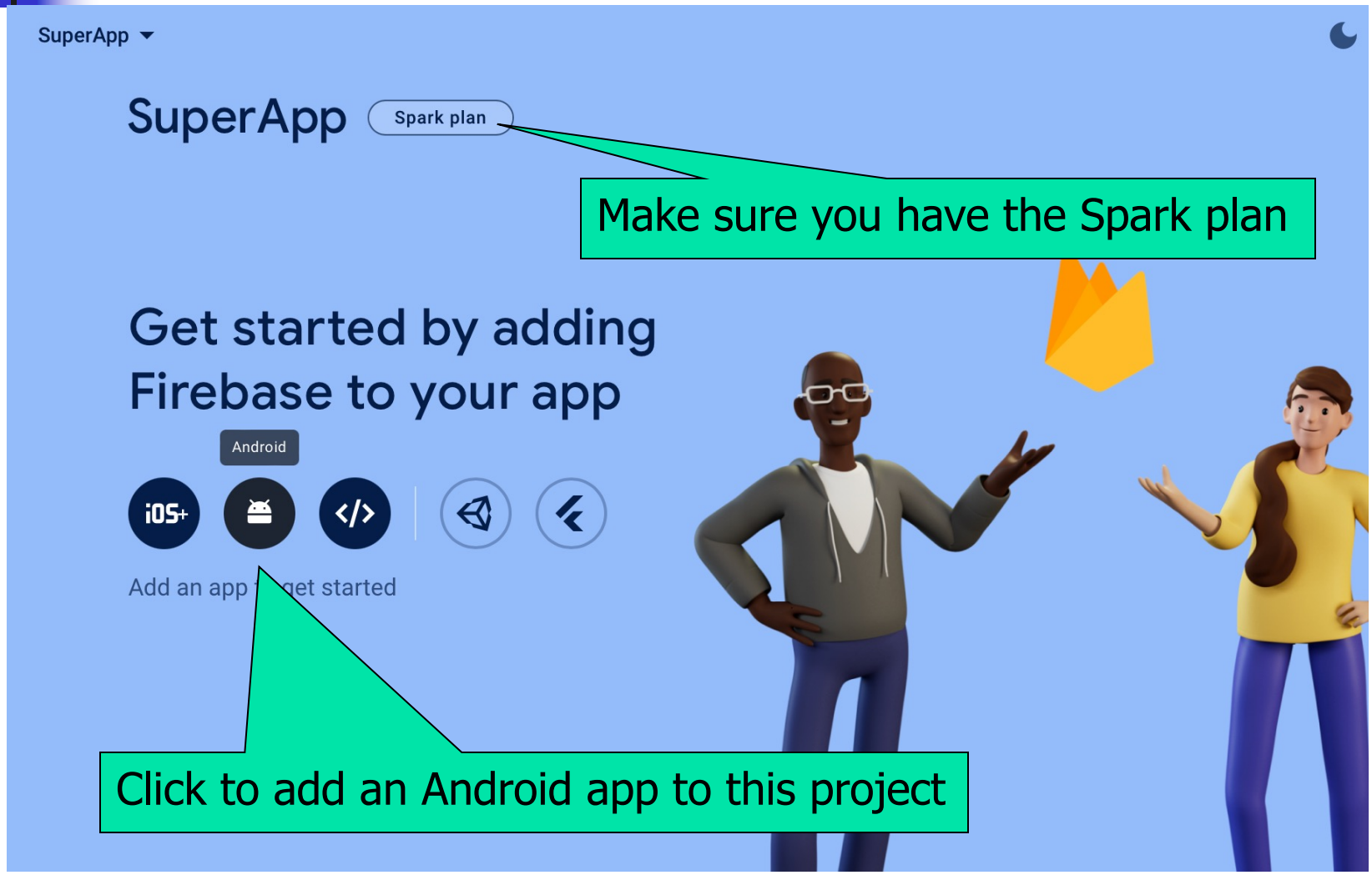
Then, click Create project, which may take a moment

Firebase: Create a New Project

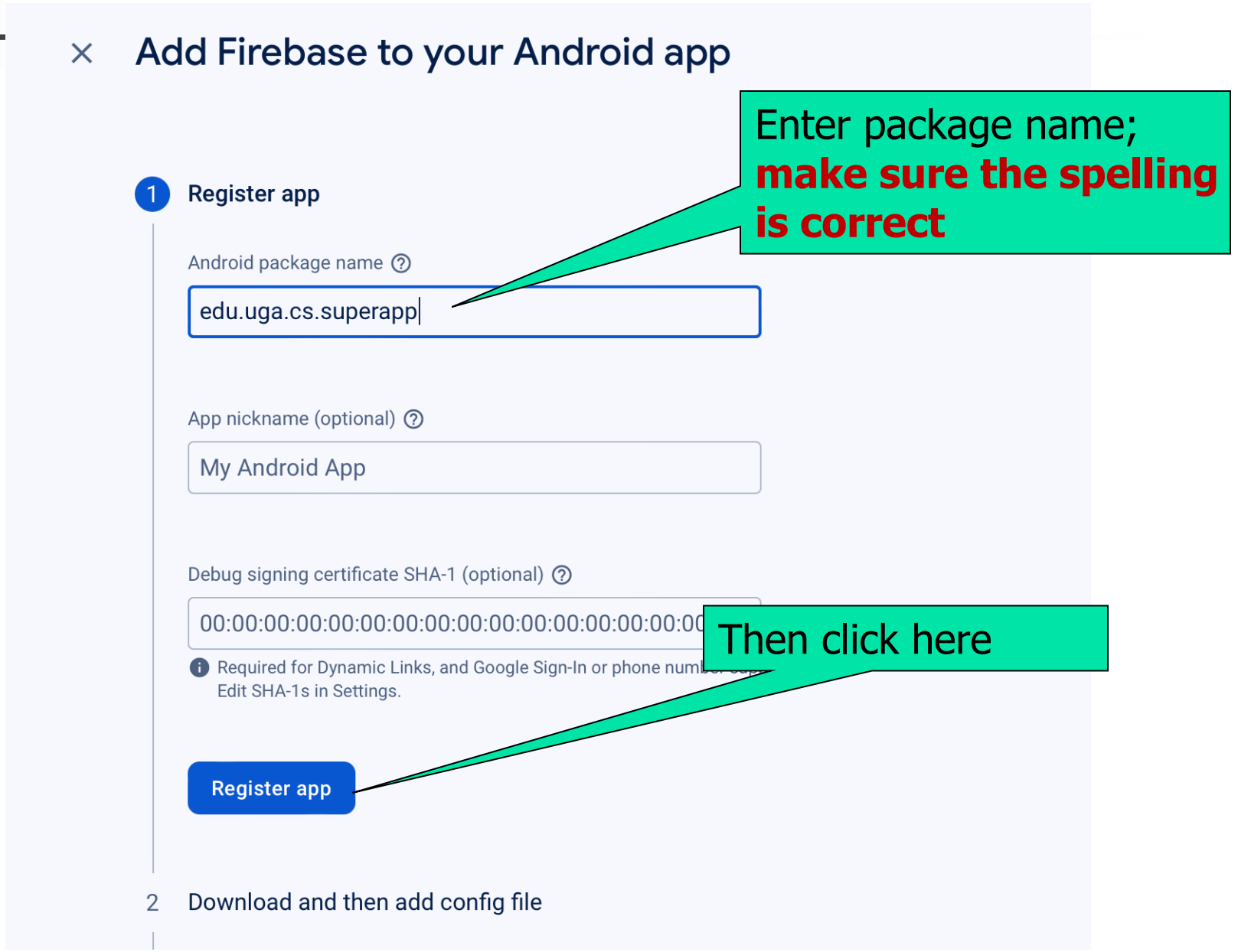


Once the new project is ready, click Continue

Firebase: Add an Android App



Firebase: Add an Android App



Firebase: Add an Android App

Add Firebase to your Android app

✓ Register app

Android package name: edu.uga.cs.superapp

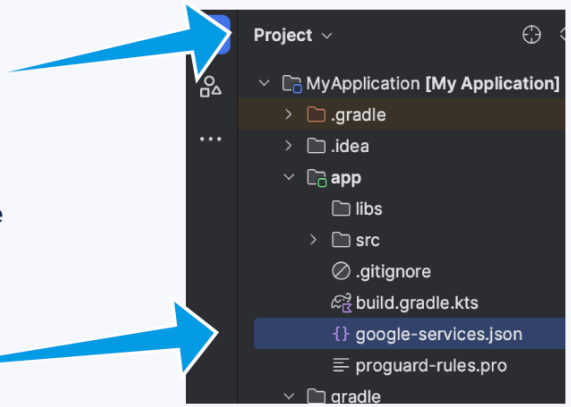
2 **Download and then add config file**

Instructions for Android Studio below | Unity C++

Download google-services.json

Switch to the **Project** view in Android Studio to see your project root directory.

Move your downloaded `google-services.json` file into your module (app-level) root directory.

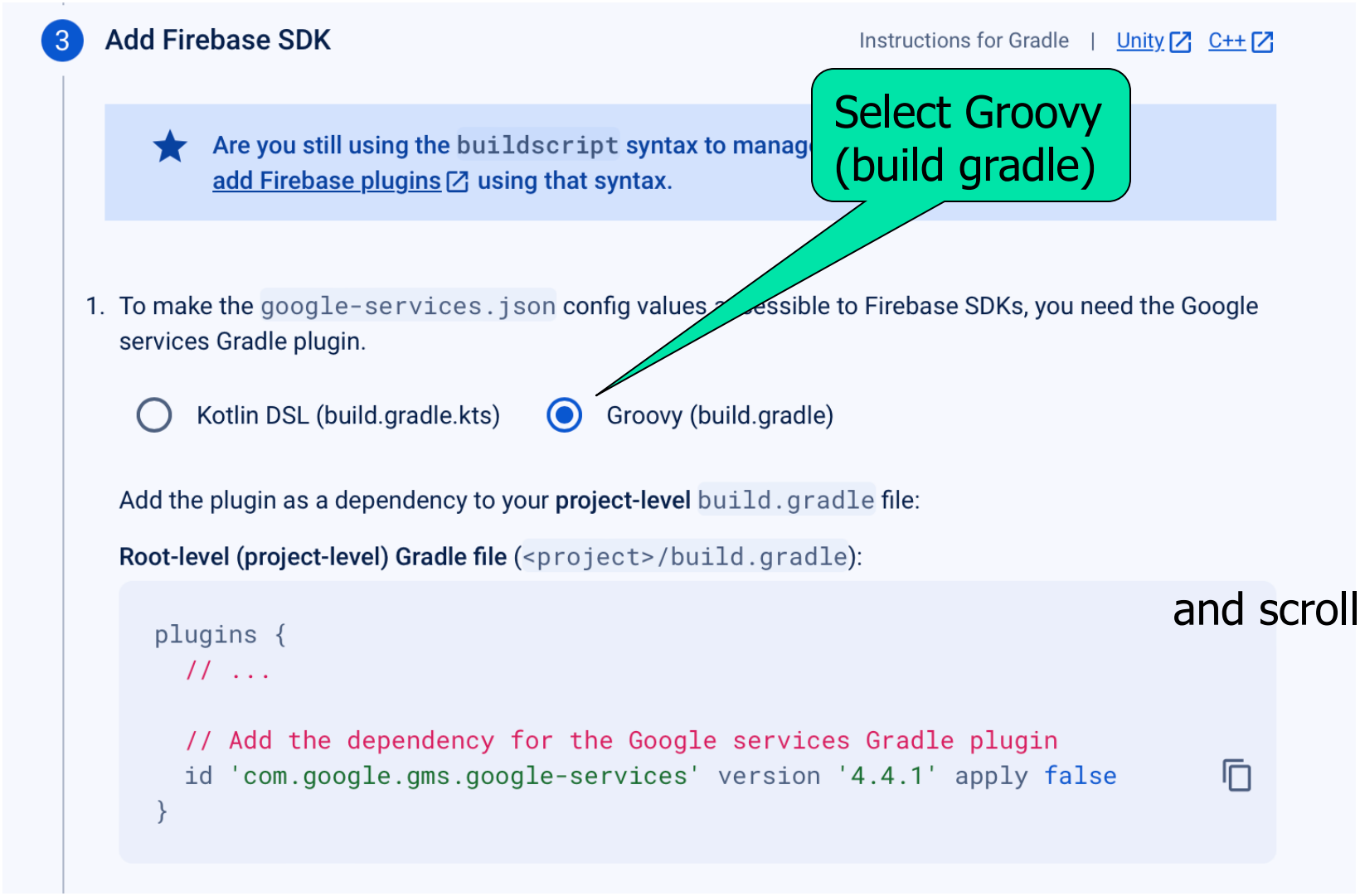


Next

3 Add Firebase SDK

Do not download the google-services file yet -- just click **Next**

Firebase: Configure Project



Firebase: Configure Project

2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.8.0')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

   By using the Firebase Android BoM, your app will always use compatible Firebase library versions. Learn more

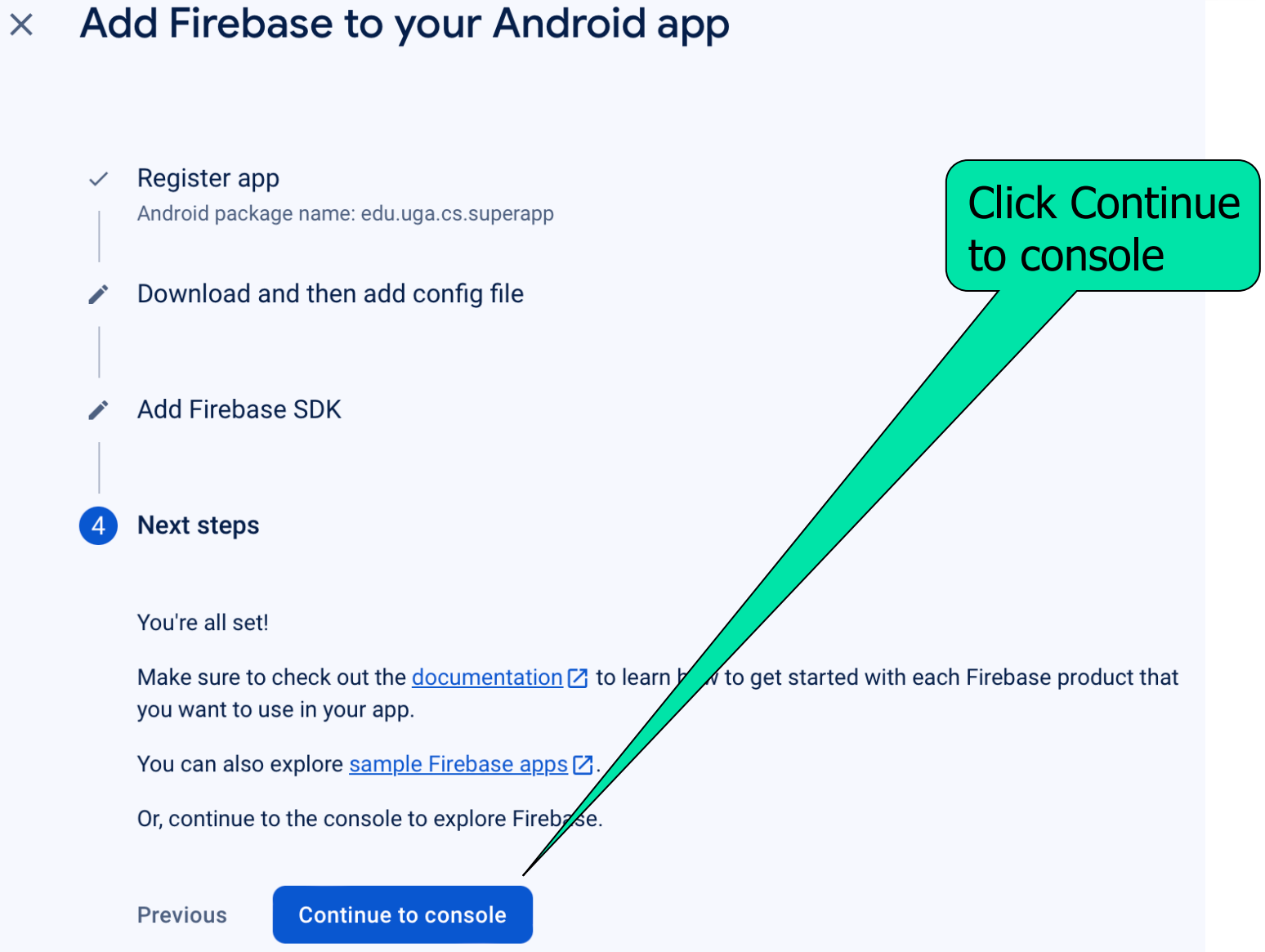
3. After adding the plugin and the desired SDKs, sync your Android project with Gradle files.

Previous Next

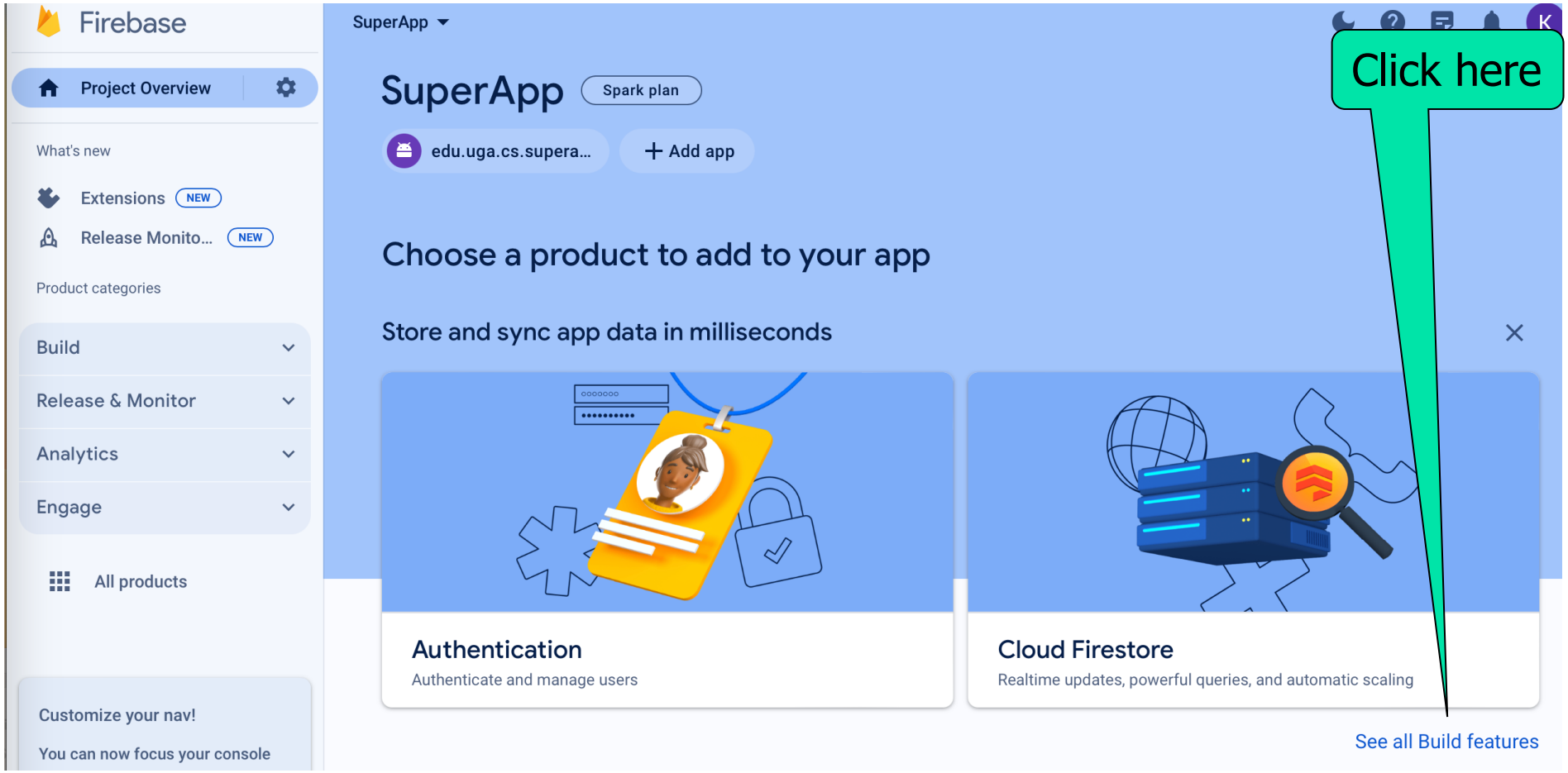
Click Next

Scroll past point "3. After adding the plugin ..."
-- and just click **Next**

Firebase: Configure Project

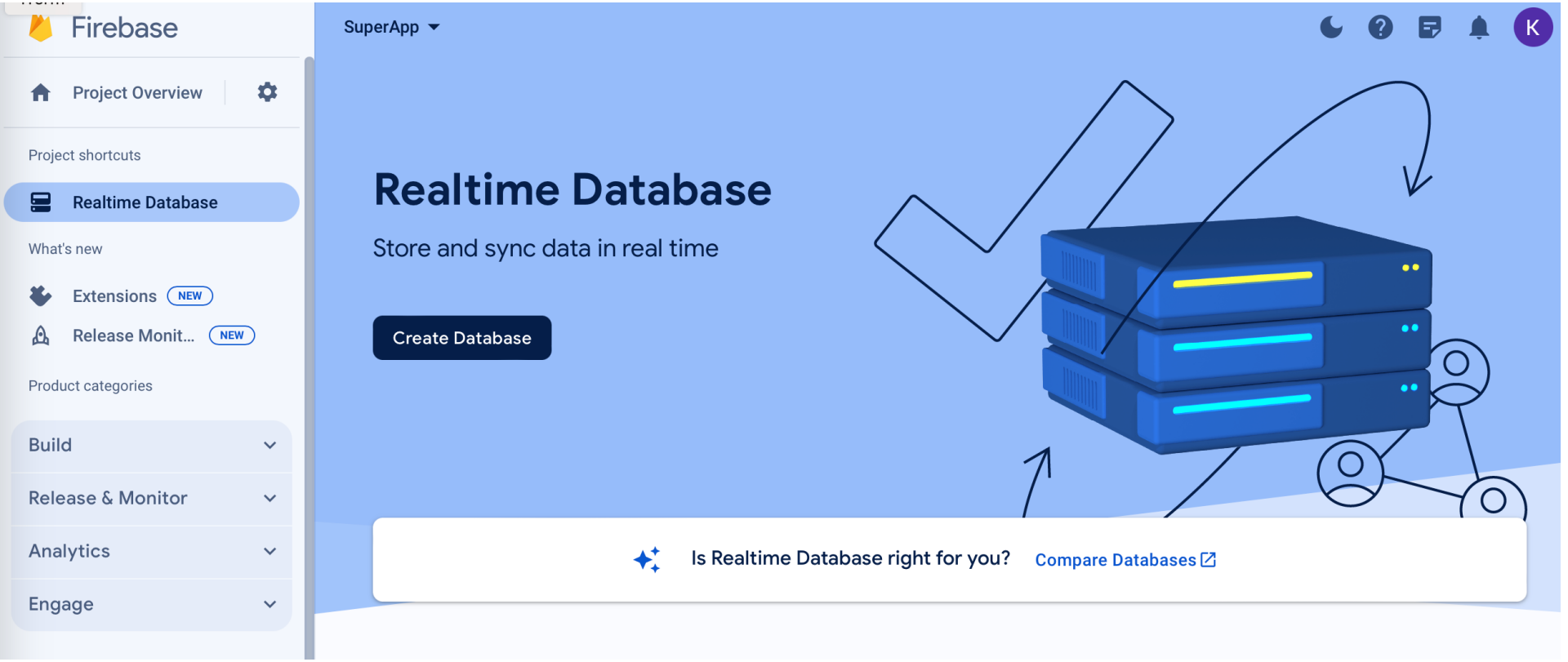


Firebase: Configure Project



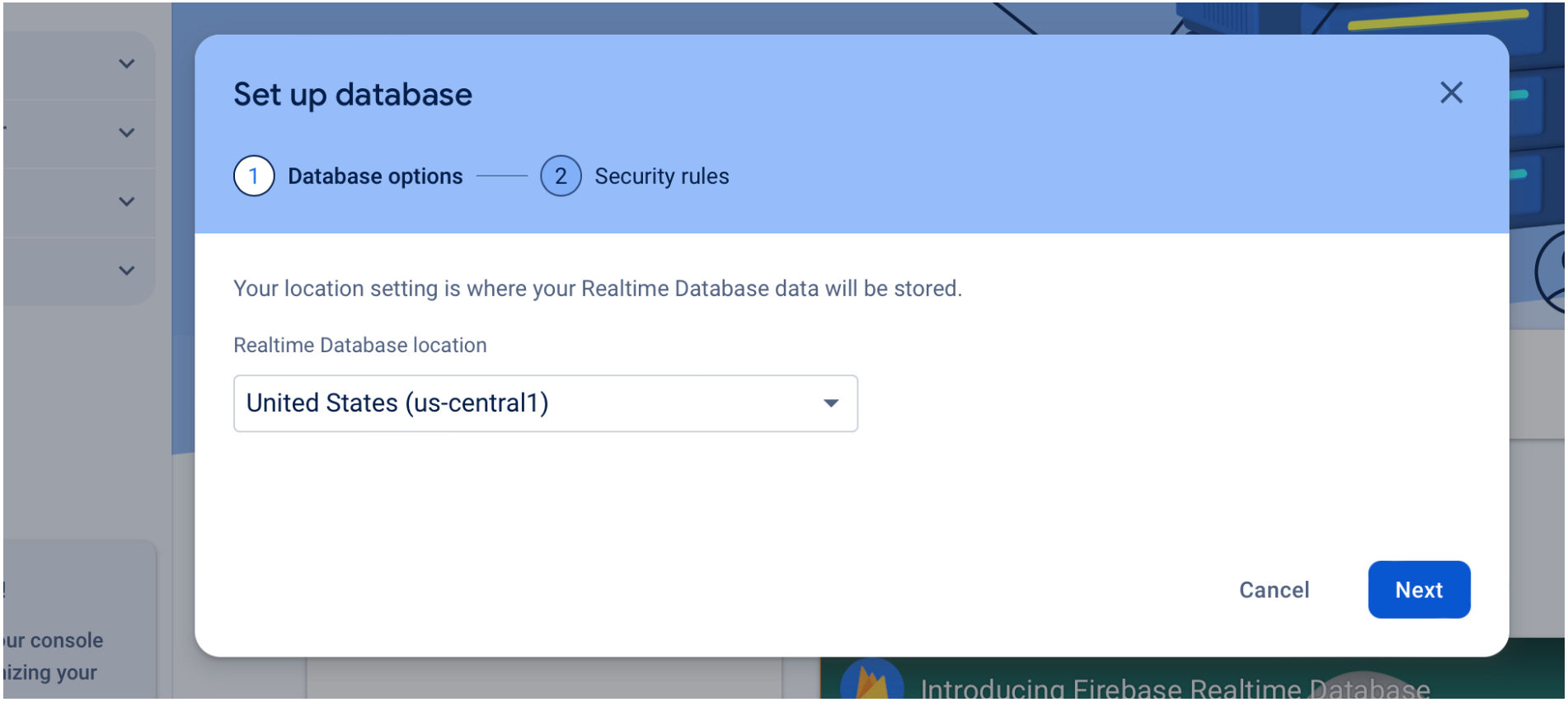
Click on See all Build features and then select Realtime Database

Firebase: Configure Project



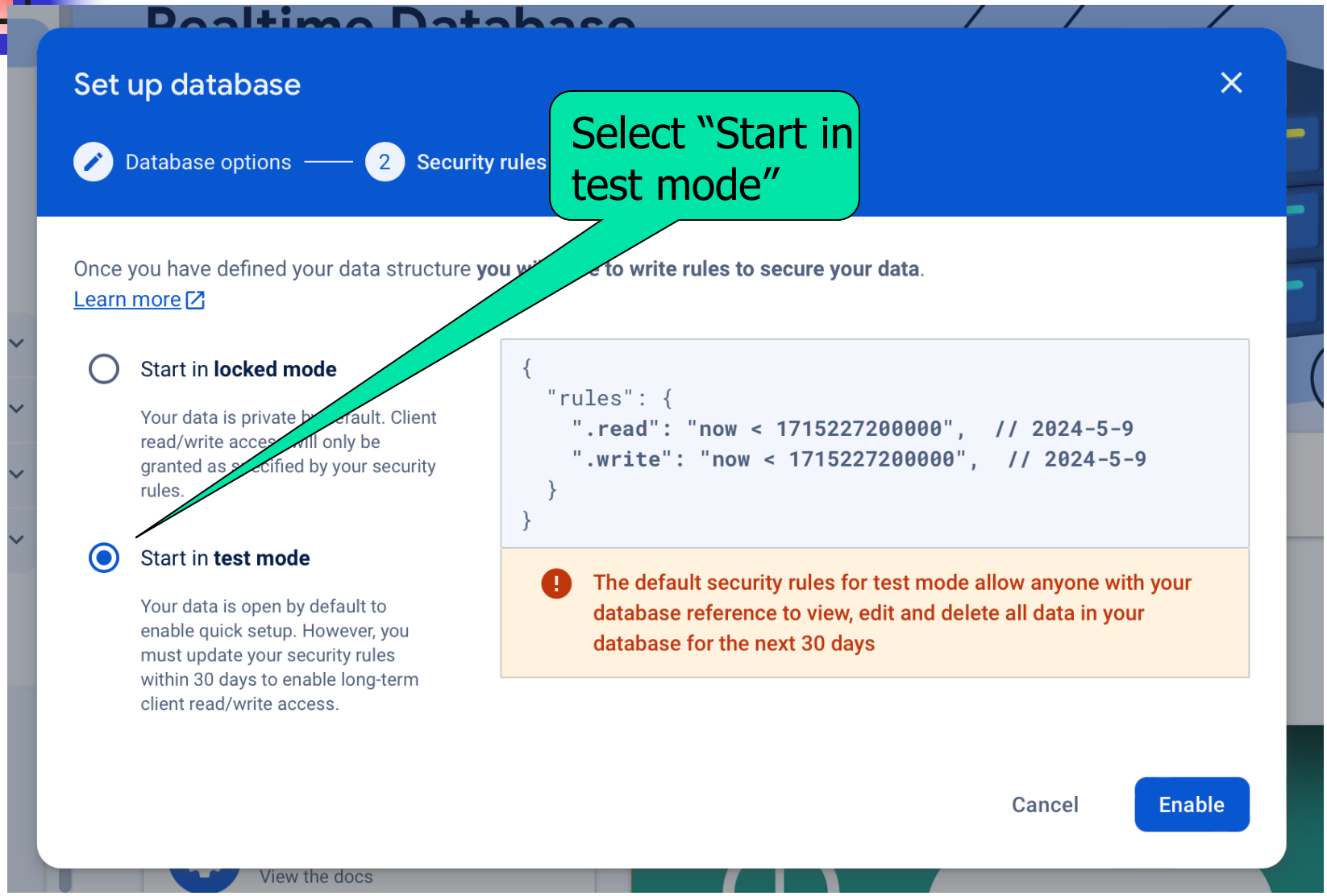
Scroll down to see Realtime Database and **click on its panel**. Then click to **Create Database**

Firebase: Configure Project



Click **Next** to accept the default US database location

Firebase: Configure Project



Select `Start in` **`test mode`**, for now, and click **Enable**

Firebase: Configure Project

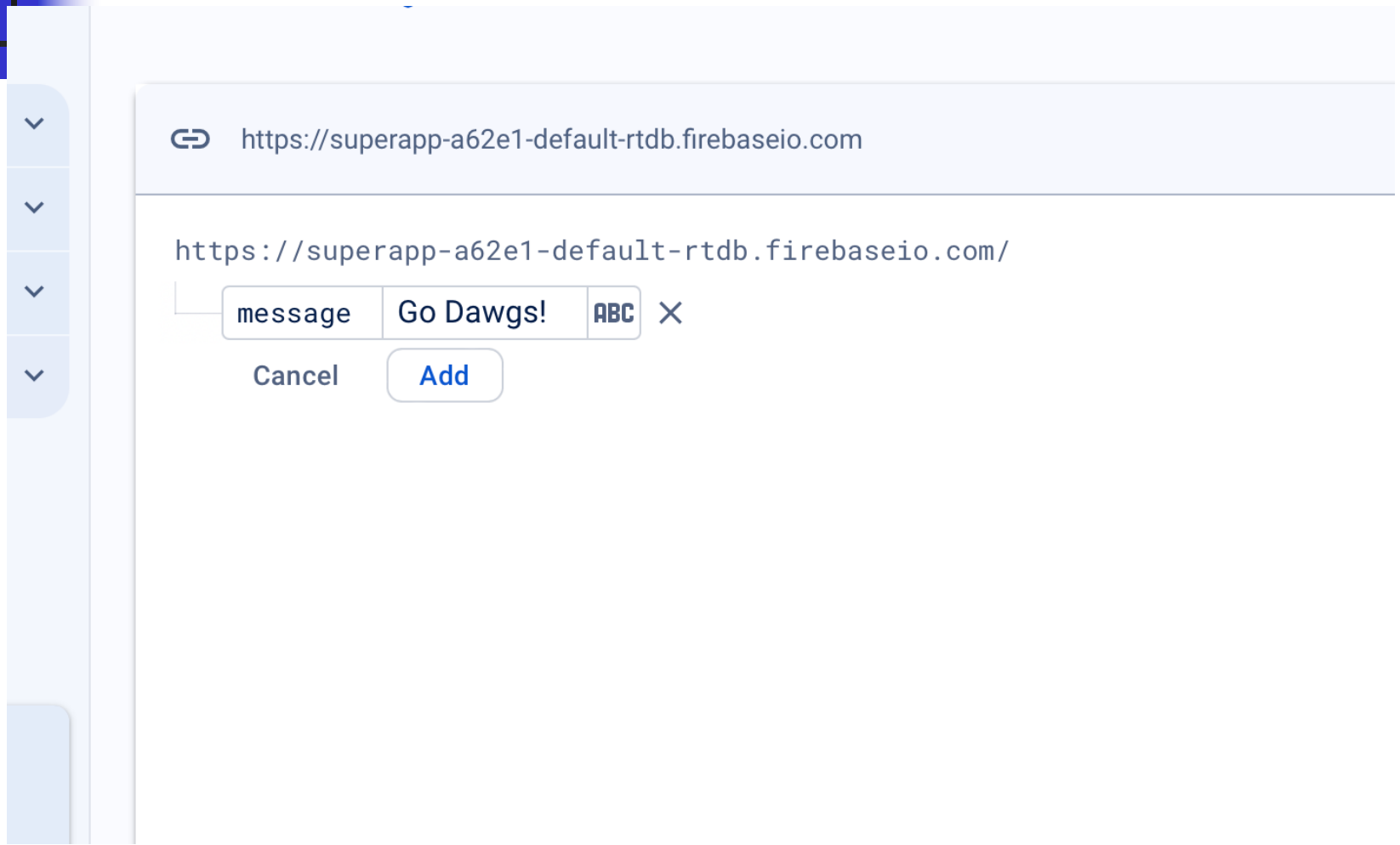
- Later, you should modify the access rules to disallow access to just anybody
- At the least, you should set up the rules to allow only authenticated (logged-in) users
- To do that, set the access rules to:

```
{
  "rules": {
    ".read": "auth != null",
    ".write": "auth != null"
  }
}
```

Firebase: Configure Project

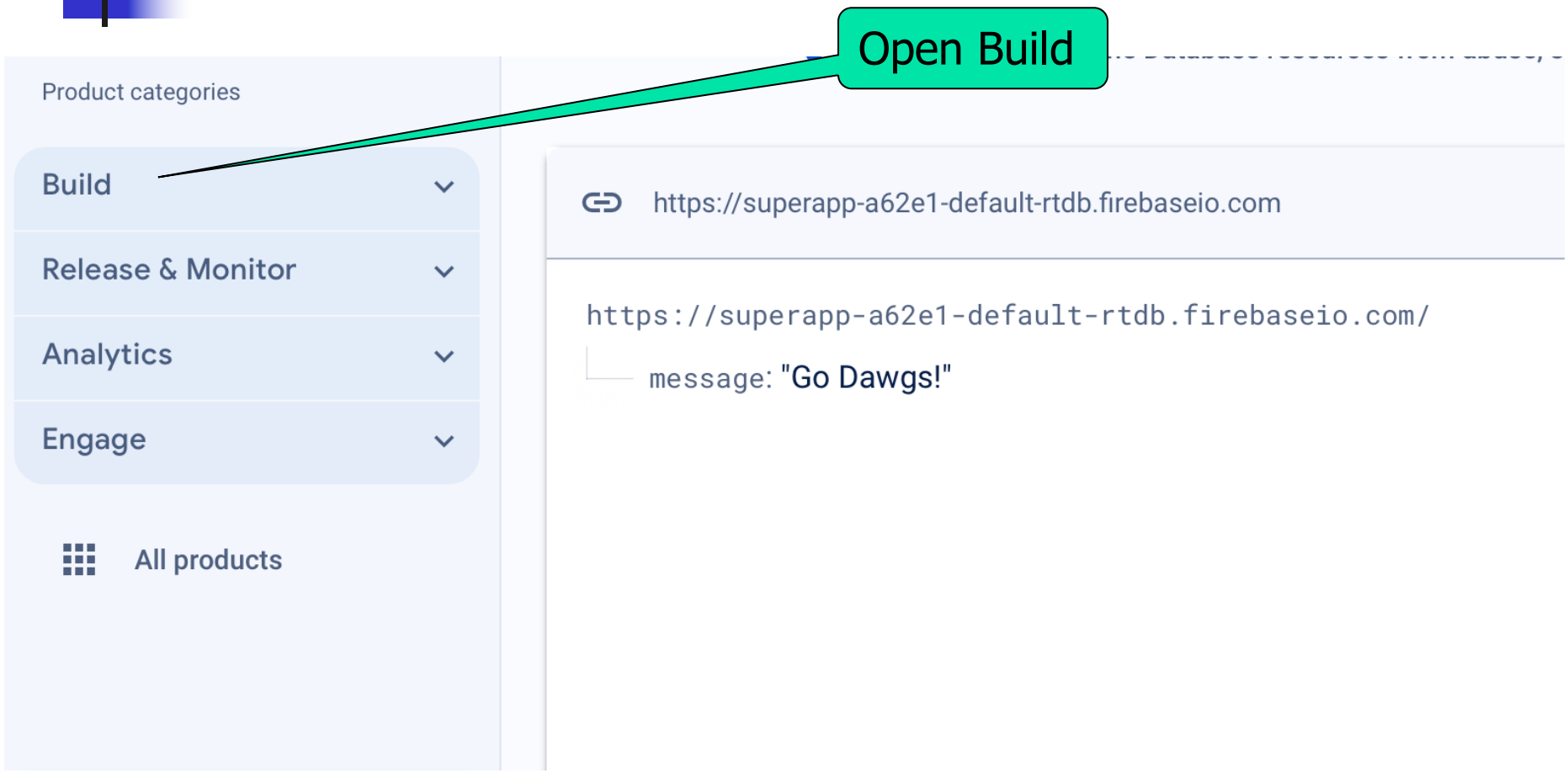
There is no data yet, so the top-level JSON object (`https://superapp-a62e1-default-rtdb.firebaseio.com/`) is null.
(Your object id will be different.)

Firebase: Initialize Data

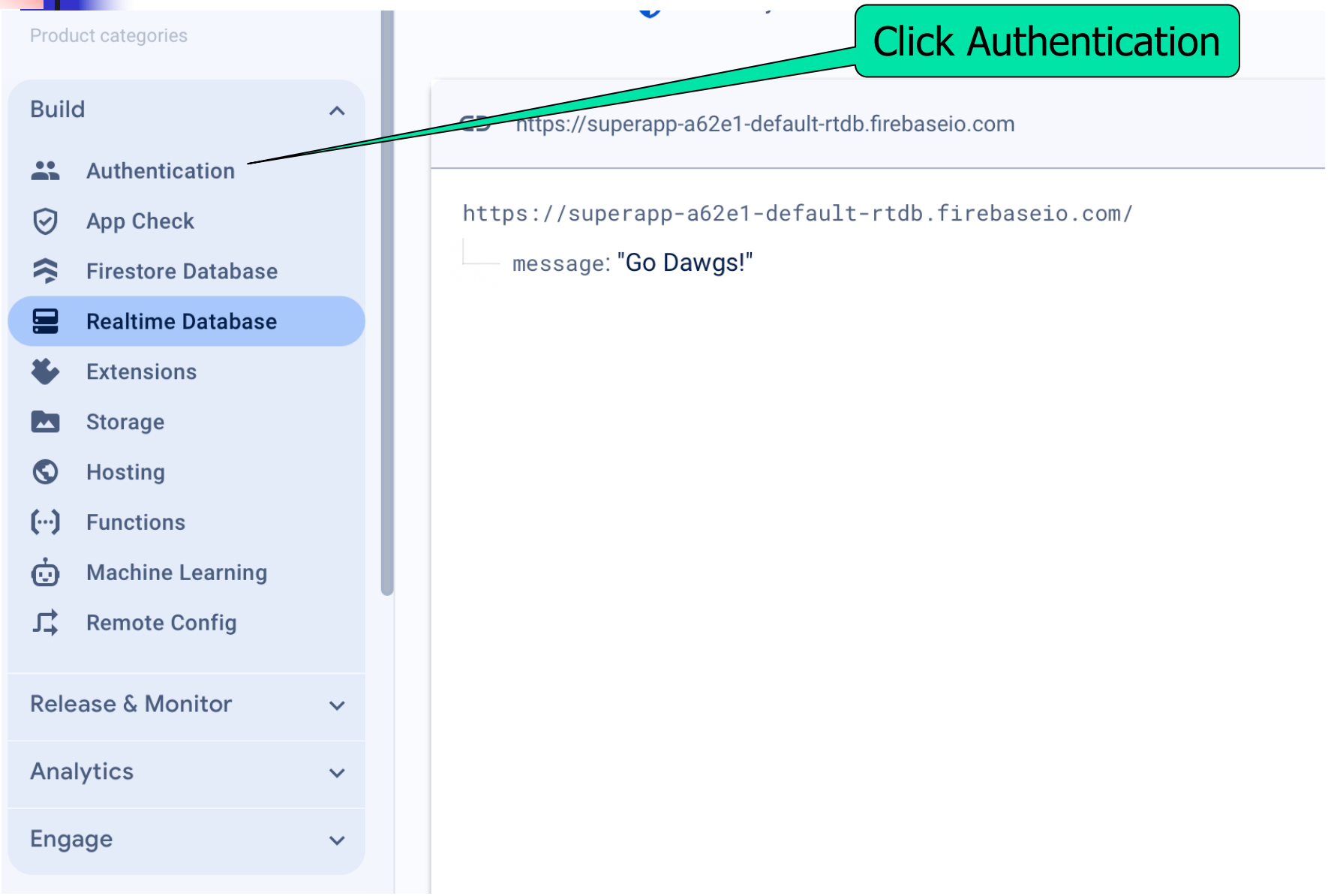


Hover over the `null`, then click on `+` and add a key: `message`, with a String value (select String from the drop-down) and enter the value of `Go Dawgs!` and click `Add`. You now have a test value in your Realtime Database!

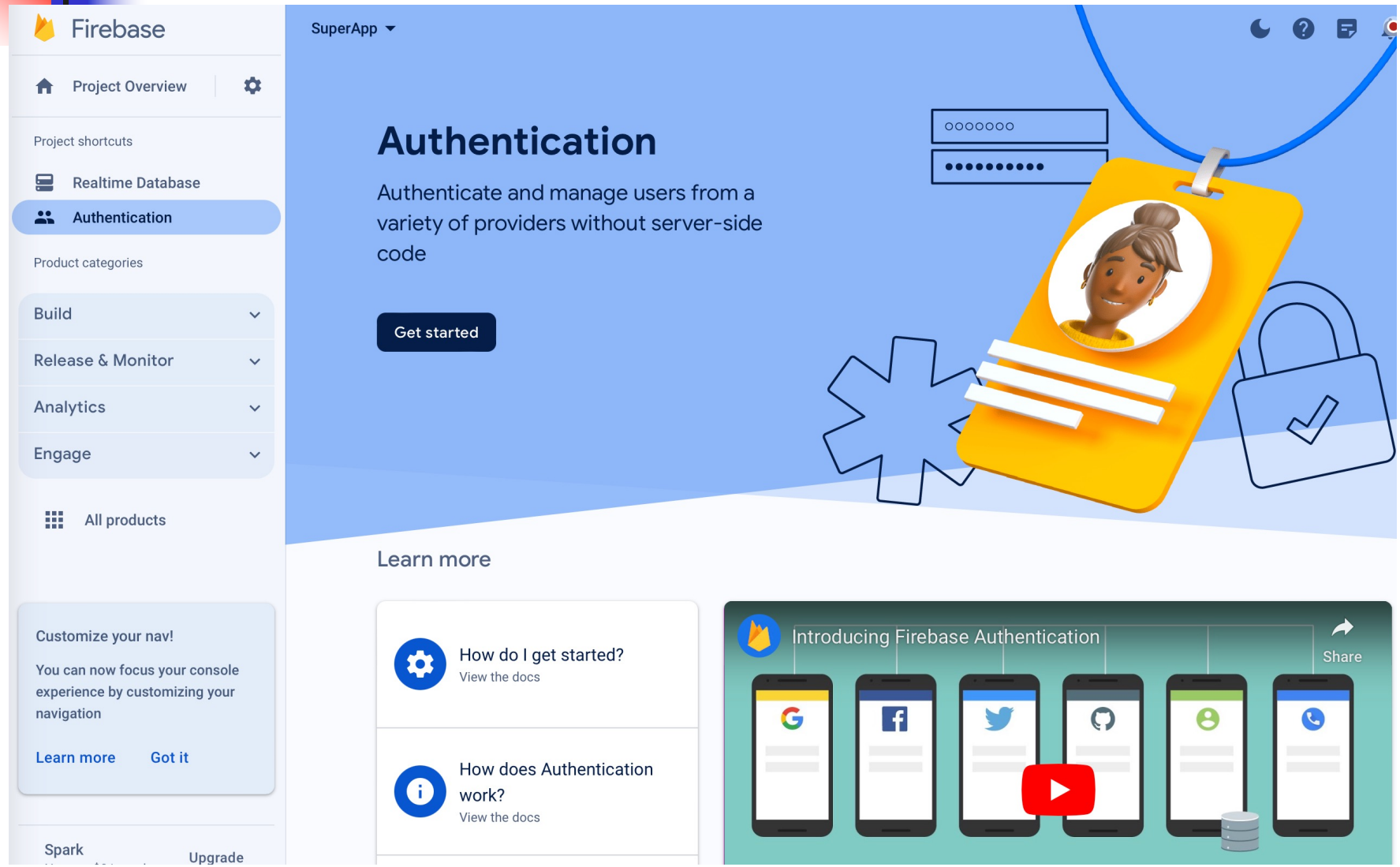
Firebase: Configure Authentication



Firebase: Configure Authentication

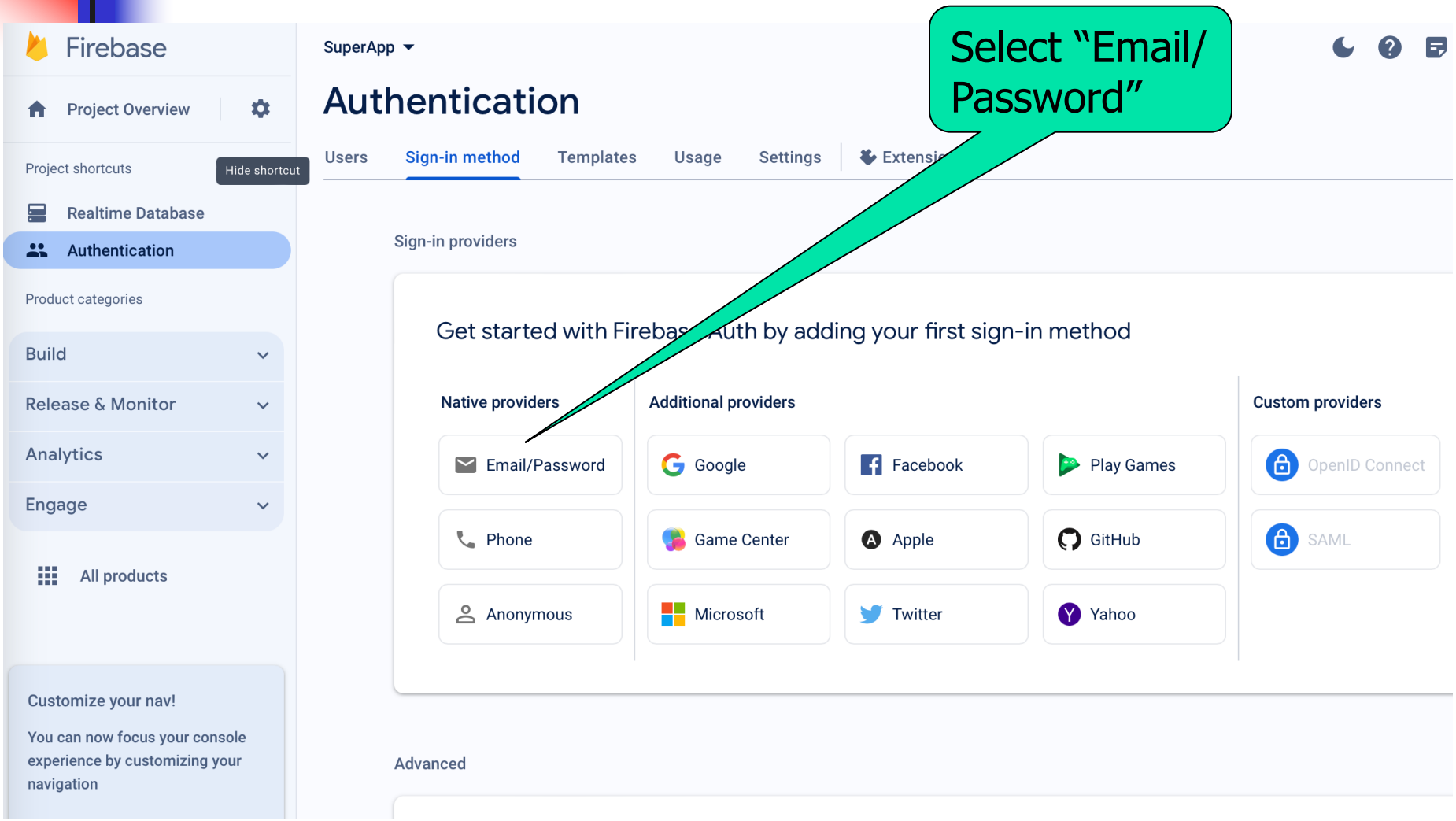


Firebase: Configure Authentication



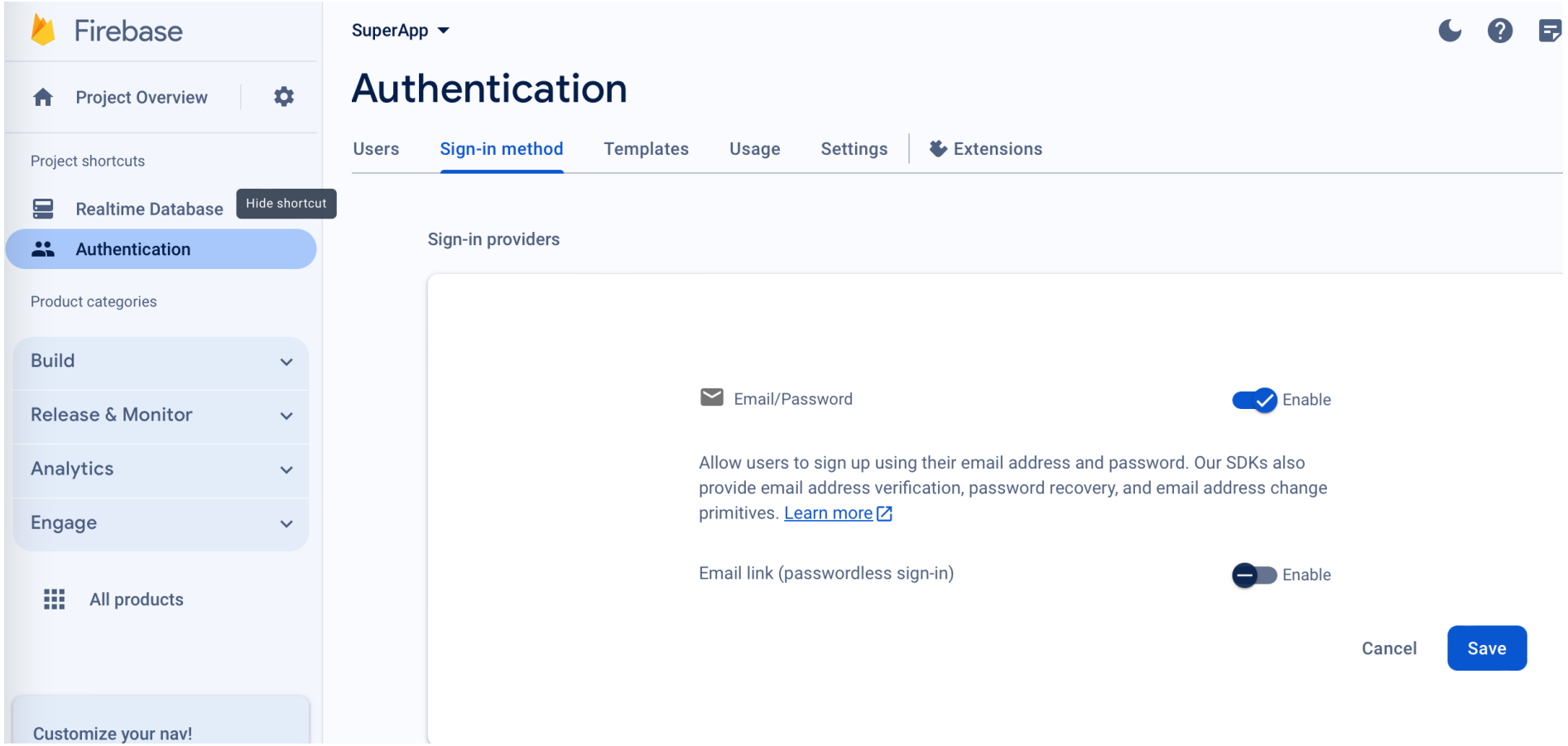
Click on `Get Started` and configure your sign-in method(s)

Firebase: Configure Authentication



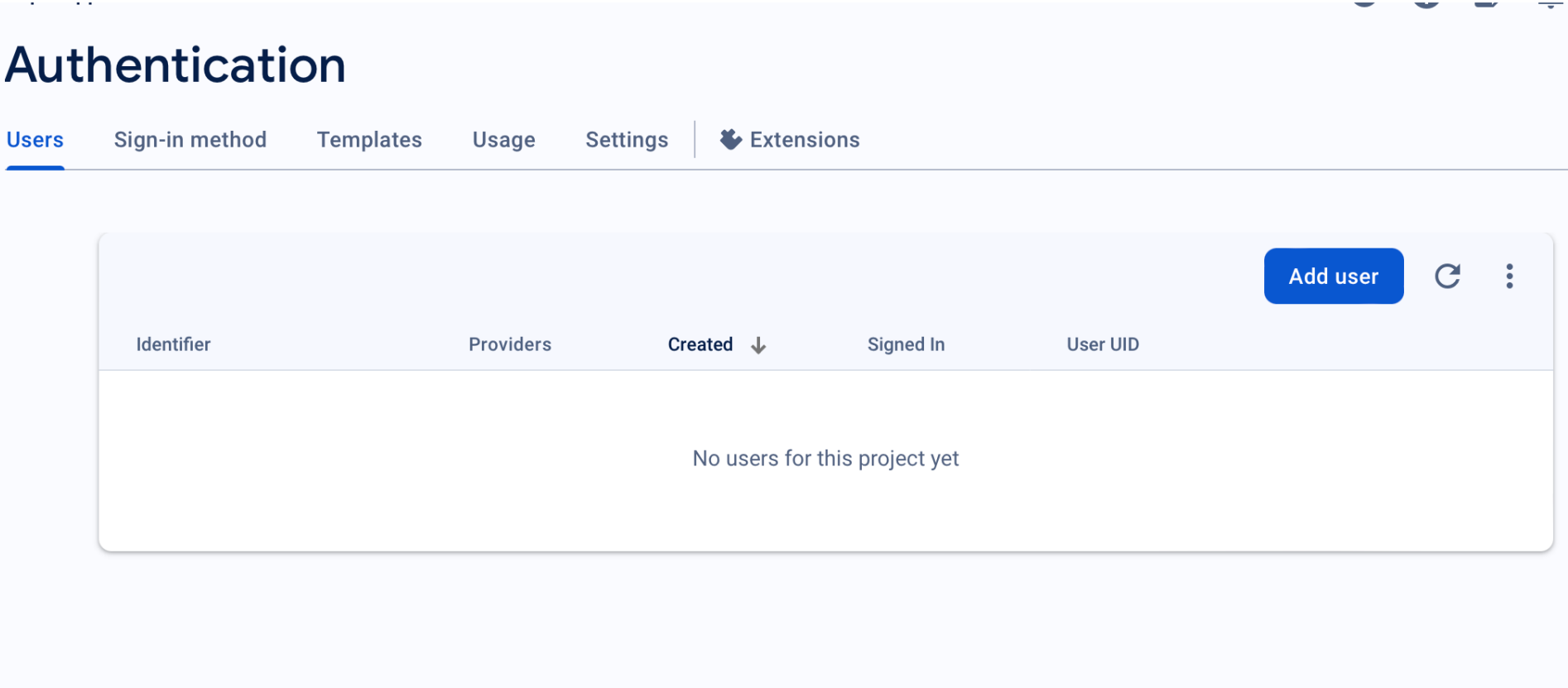
Click on `Email/Password` to enable it.
Later, you can use other methods.

Firebase: Configure Authentication



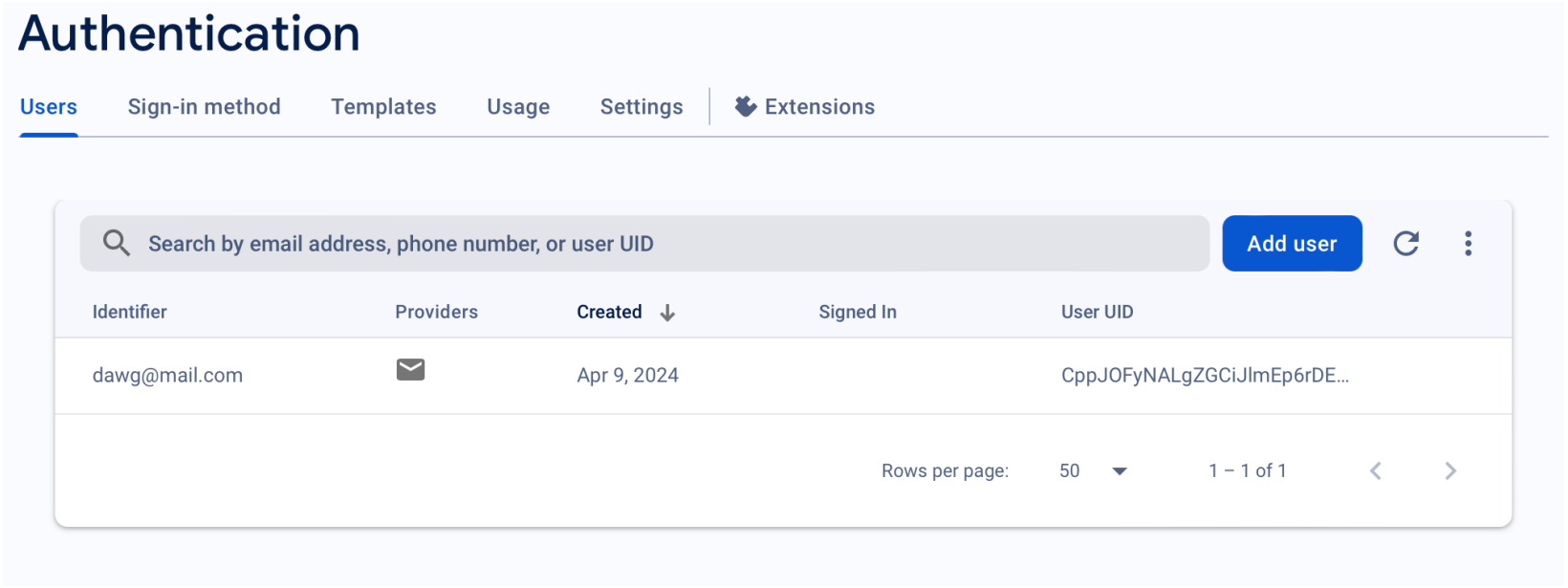
Toggle to `Enable` Email/Password for now and Click Save.

Firebase: Configure Authentication



Switch to the `Users` tab and add a new user (click **Add user**).
Enter email: `dawg@mail.com` and password: "`password`"
(make sure the spelling is correct!)

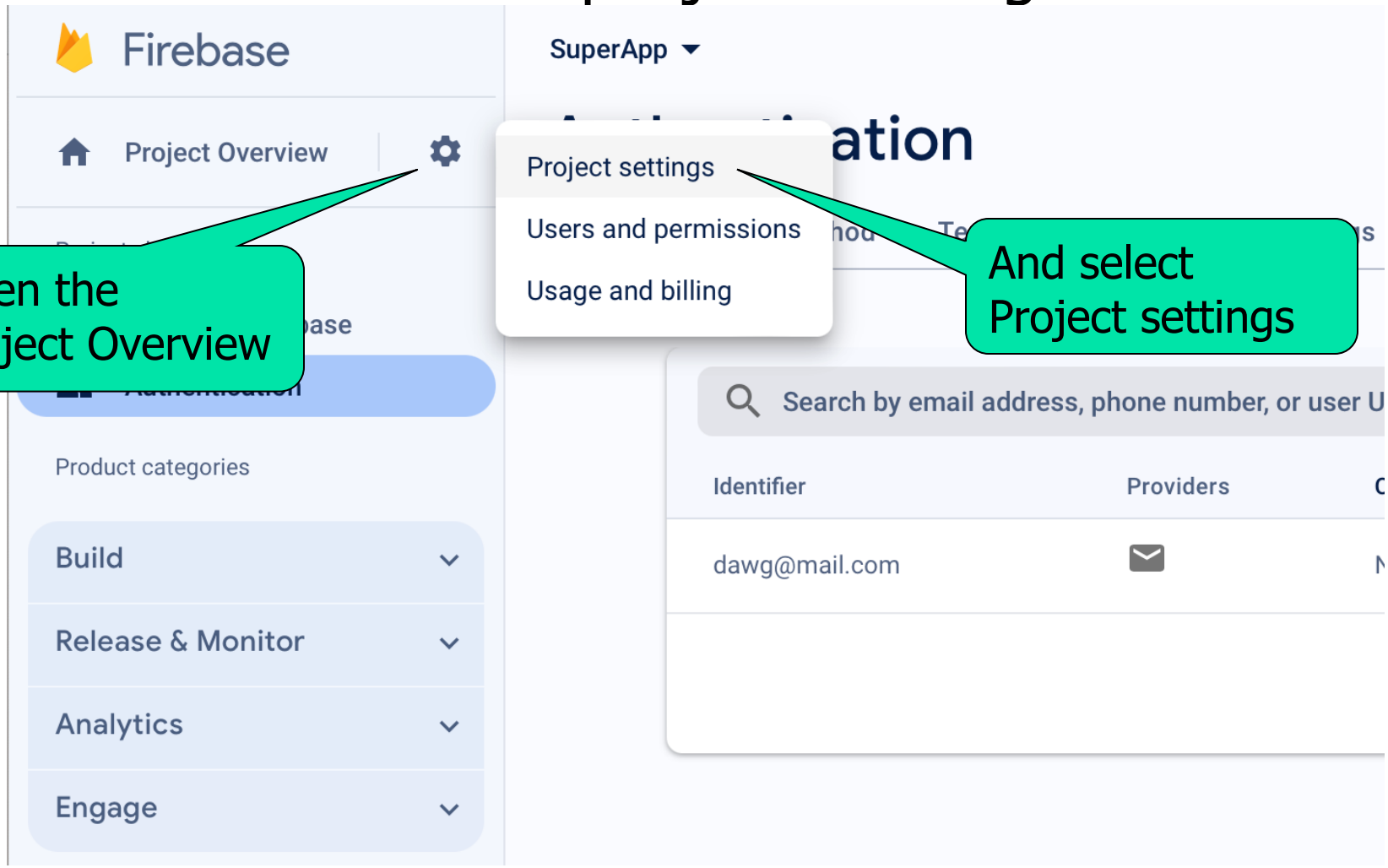
Firebase: Configure Authentication



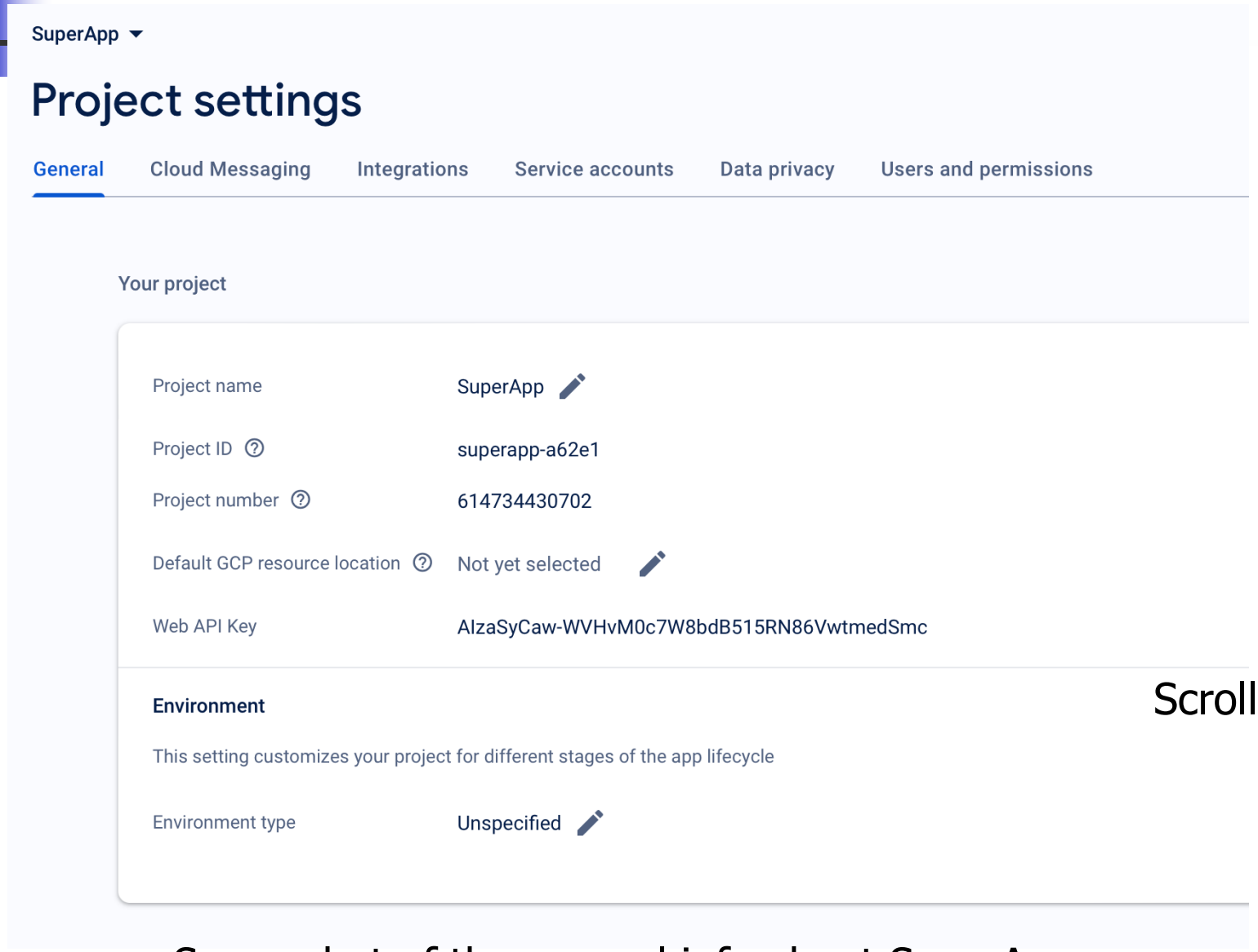
Authentication screen with the ‘dawg@mail.com’ user added

Firebase: Get the Configuration File

- Switch to the project settings:



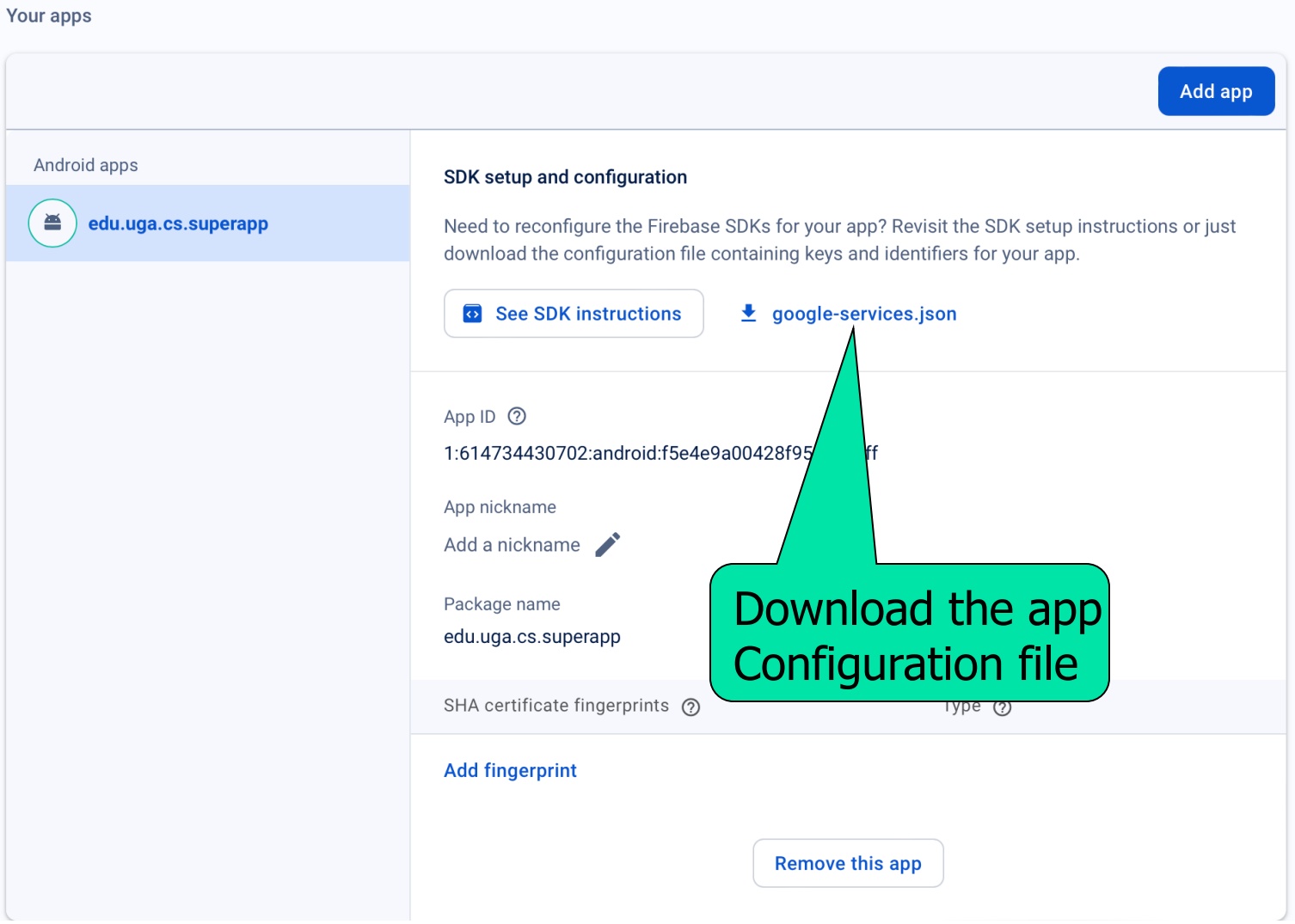
Firebase: Get the Configuration File



Scroll down

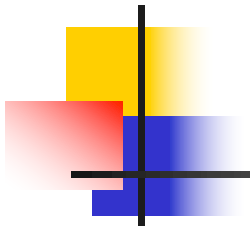
Screenshot of the general info about SuperApp

Firebase: Get the Configuration File



Scroll down to see the `google-services.json` link and download this file to your laptop.

Create a New Android Studio Project



You may be presented with a popup saying:

Do you want to allow downloads on “console.firebase.google.com”?

If so, allow it.

Create a New Android Studio Project

3. Open Android Studio and create a new project, called `SuperApp`, as usual:

   Select the `Empty Views Activity` and

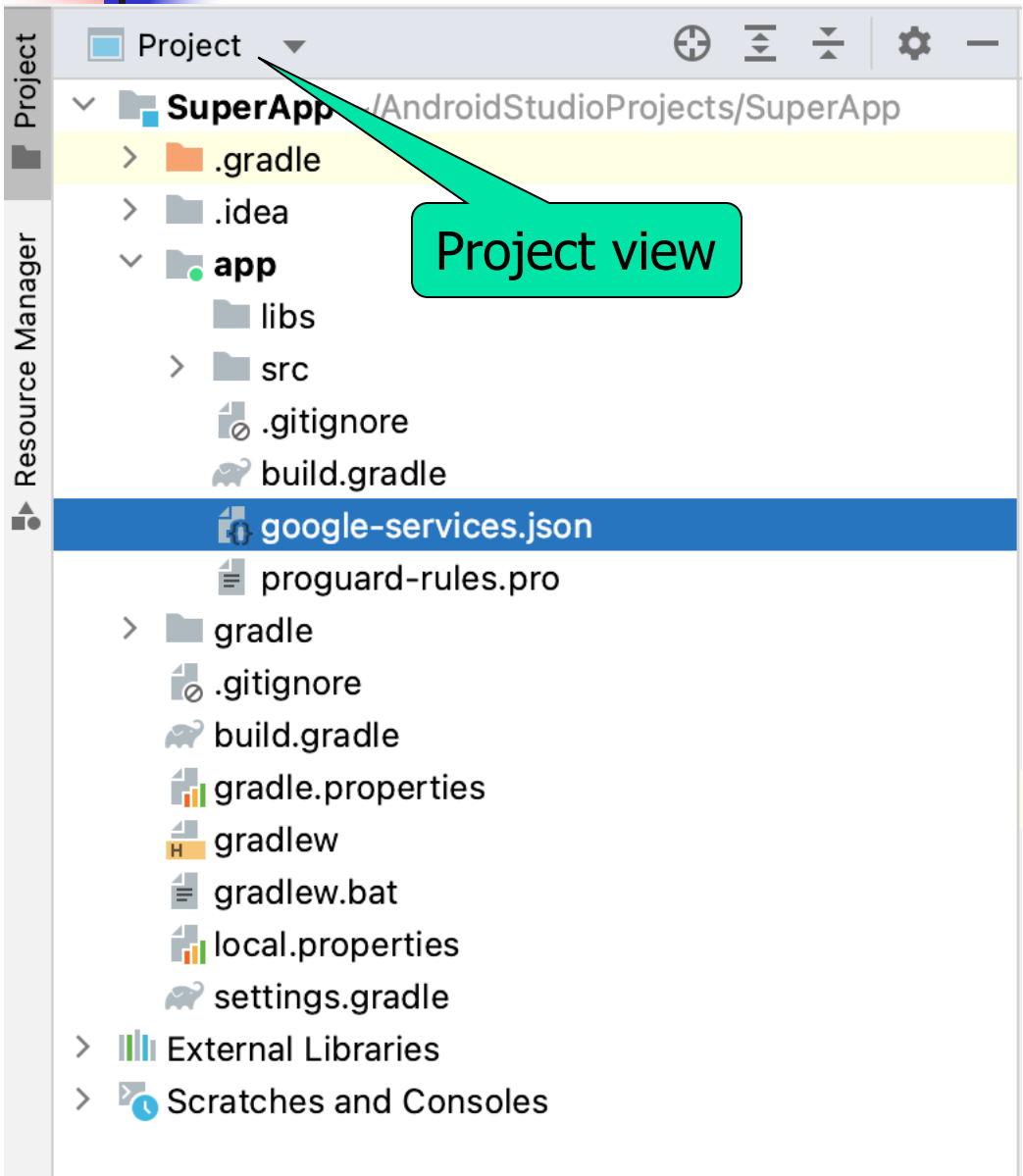
   make sure the package name is (exactly as specified when creating a Firebase project)

   ```
   edu.uga.cs.superapp
   ```

   And select Java, API 24 (Nougat) and Groovy DSL, as usual.

4. Configure the project, as shown on the following several slides

Firebase: Configure the Project



In the Project panel, switch from `Android` to `Project` view

Look for the `google-services.json` file in your `Downloads` directory.

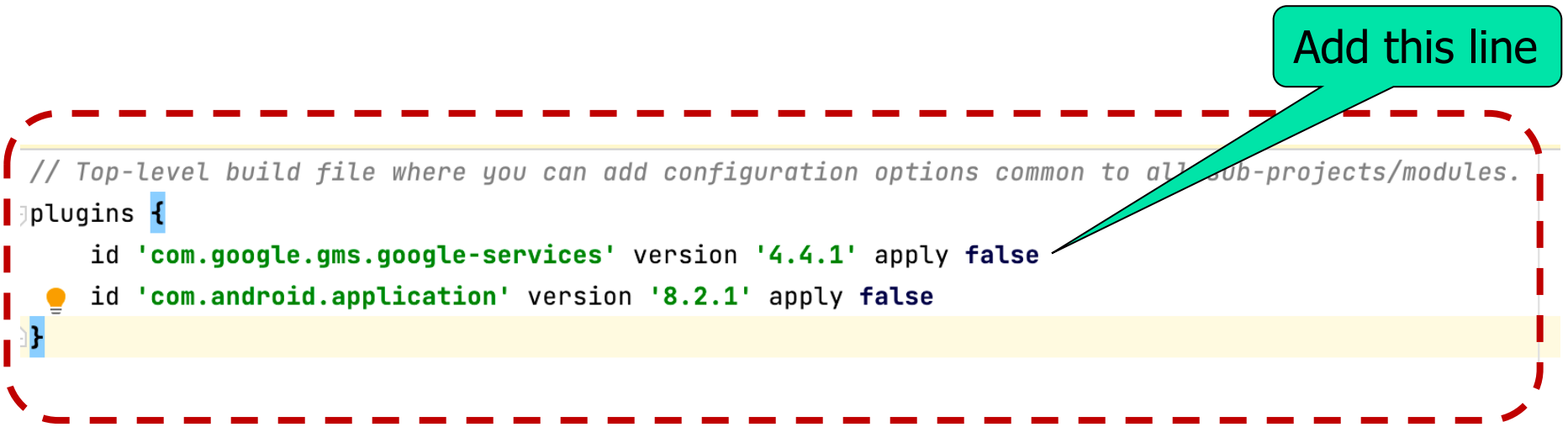
Copy the downloaded `google-services.json` file to your project's **app** directory, while in Project view.

Once you are done, switch back to Android view.

Firebase: configuring the app

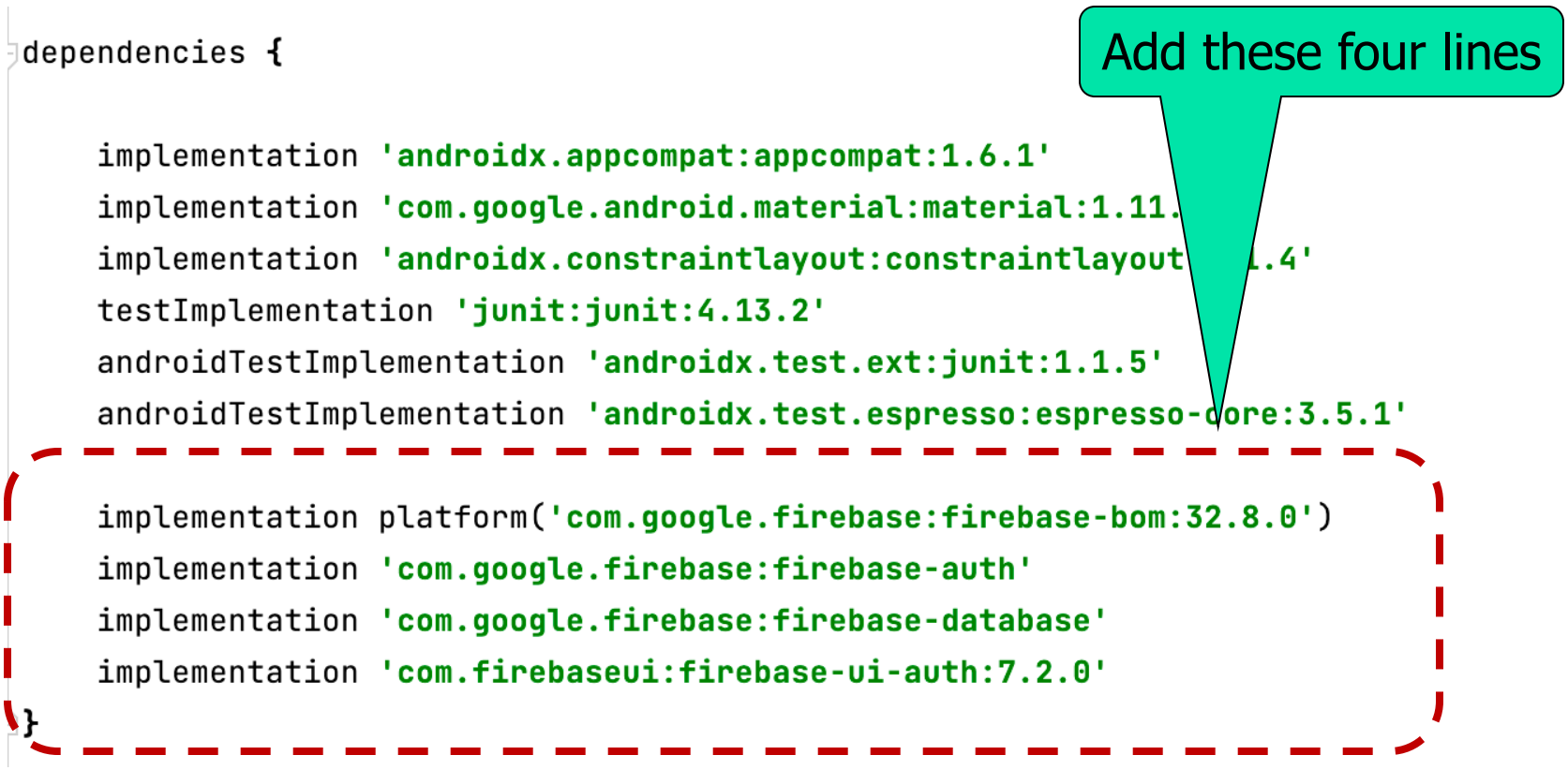
Open the `build.gradle (Project: …)` file and add the circled lines to your file, right above the `plugins` section (at the very top of the file).

Fragments of code for easy copy/paste are available on eLC in the Extra materials folder.



Firebase: configuring the app

Open the `build.gradle (Module: app)` file
and add the circled 'implementation' lines to your `dependencies` section, as shown below.



Firebase: configuring the app

In the same gradle file, in the `android` section, right above the `dependencies`, add the two `apply plugin` lines, as shown below:

```
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    apply plugin: 'com.android.application'
    apply plugin: 'com.google.gms.google-services'  // Google Play services Gradle plugin
}
```

Firebase documentation

Open the AndroidManifest.xml file and add the indicated `uses-permission` line to allow networking.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

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

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="SuperApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.SuperApp"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
```

Create a new Android Studio Project

5. Sync your project with gradle files
6. Open the `activity_main.xml` layout and add the identifier (id) `textView` to the automatically created TextView (when the project was created):

```
android:id="@+id/textView"
```

7. Open the `MainActivity.java` class and add the following code, right after the class declaration header:

```
public static final String TAG = "SuperApp";
private FirebaseAuth mAuth;
```

Create a new Android Studio Project

Right after the call to `setContentView` add:

```
TextView textView = findViewById( R.id.textView );

mAuth = FirebaseAuth.getInstance();
String email = "dawg@mail.com";
String password = "password";
```

Add the needed import for the FirebaseAuth class.

You will need to add imports to several other classes after the next few slides.

Create a new Android Studio Project

Right after that, add this code, to sign-in to Firebase (make sure email/password are ok):

```
mAuth.signInWithEmailAndPassword( email, password )
    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            if (task.isSuccessful()) {
                // Sign in success
                Log.d( TAG, "signInWithEmail:success" );
                FirebaseUser user = mAuth.getCurrentUser();
            }
            else {
                // If sign in fails
                Log.d( TAG, "signInWithEmail:failure", task.getException() );
            }
        }
    });
```

Create a new Android Studio Project

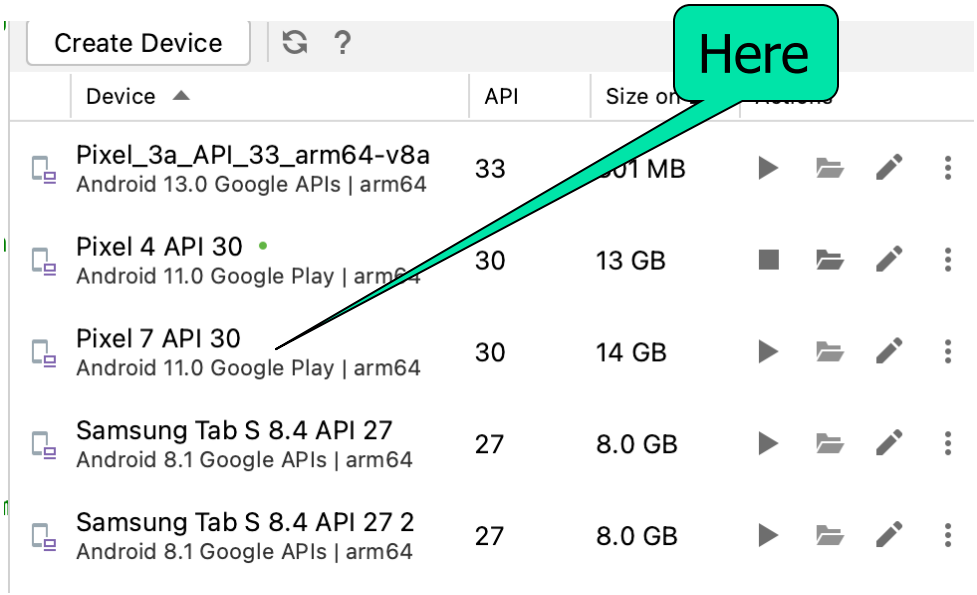
Finally, add this code, to read from Firebase:

```
FirebaseDatabase database = FirebaseDatabase.getInstance();
DatabaseReference myRef = database.getReference( "message" );

// Read from the database value for ”message”
myRef.addValueEventListener( new ValueEventListener() {
      @Override
      public void onDataChange(DataSnapshot dataSnapshot) {
         // This method is called once, initially, and when data is updated
         String message = dataSnapshot.getValue(String.class);
         Log.d( TAG, "Read message: " + message );
         textView.setText( message );
      }

      @Override
      public void onCancelled( DatabaseError error ) {
         // Failed to read value
         Log.d( TAG, "Failed to read value.", error.toException() );
      }
});
```

Check Your Emulator for Google Play

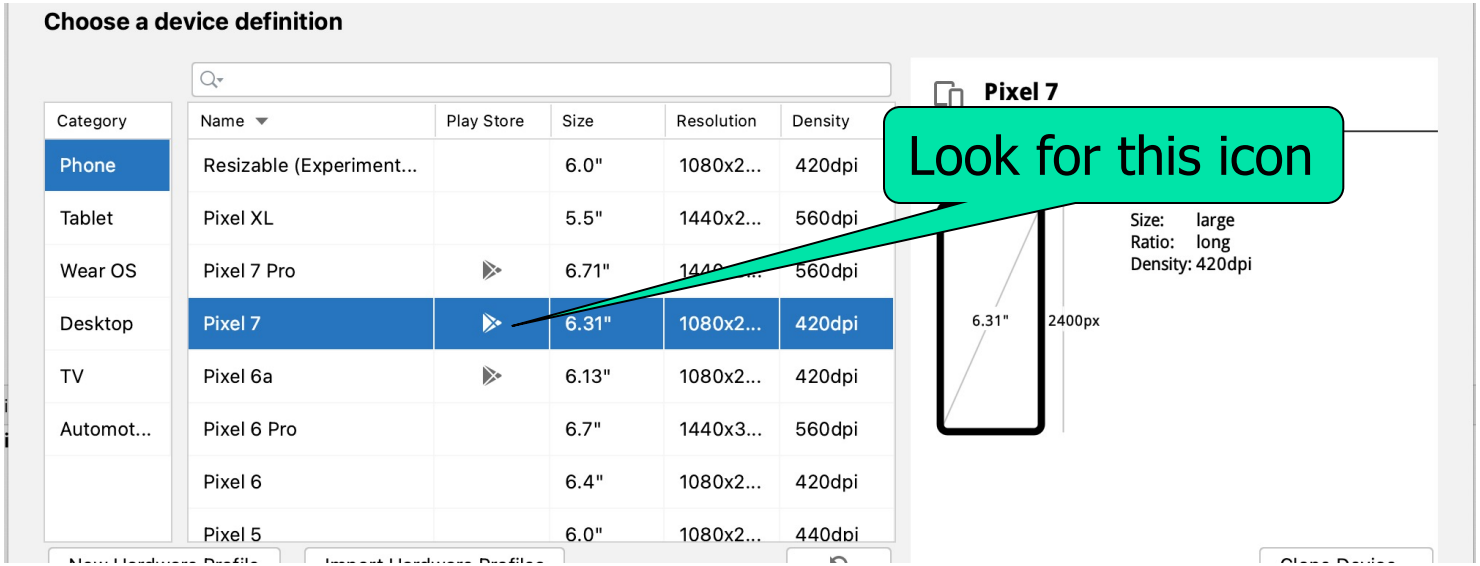
Add any missing imports.

8. Make sure your emulator has Google Play available:



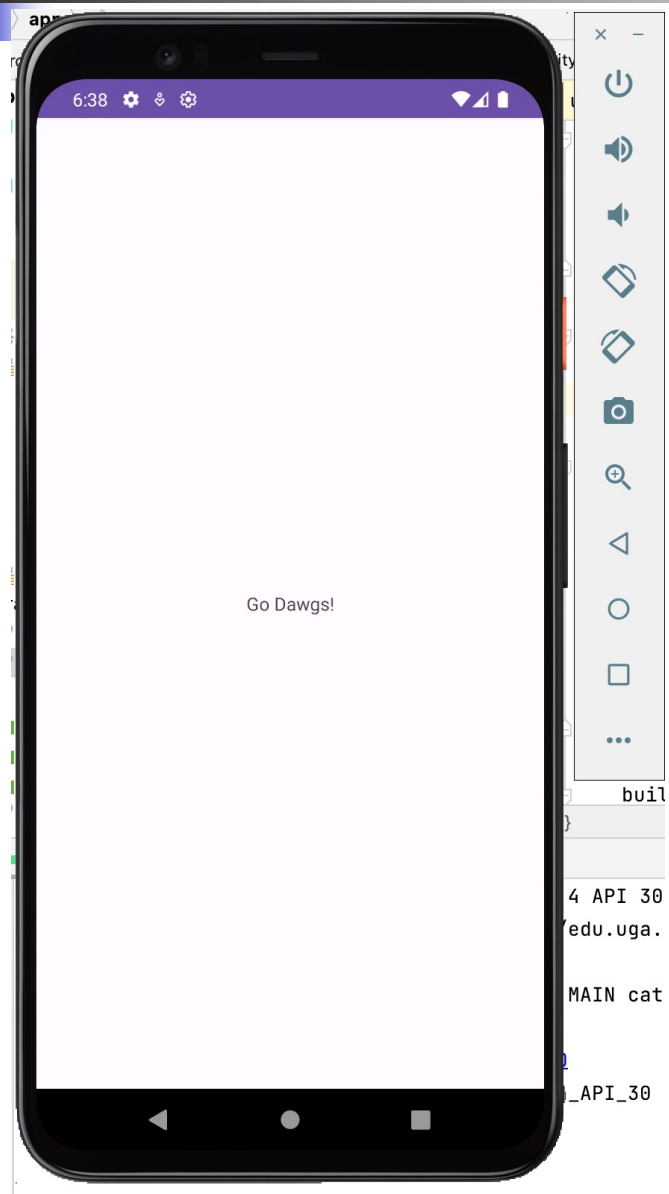
If not, create one with Google Play available

Check Your Emulator for Google Play



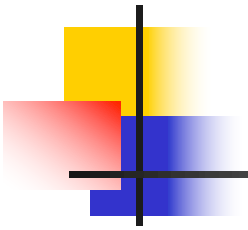
- A good choice is a Pixel 7, which is already predefined in Android Studio; it has a good "skin" available
- The API 31-32 is a good Android version for this new AVD

Run the App in Emulator



9. Run the app in your emulator

   You should see the Go Dawgs! message received from Firebase!



Submit the Screenshots

10. Take two screenshots of your project:
    - Your Firebase database content (with the defined message field)
    - Emulator window running your app and displaying the `Go Dawgs!` message
11. Submit the screenshots to the `Firebase App` assignment folder on eLC

    You must submit by today 11:59pm.