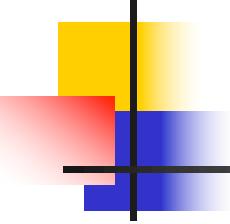


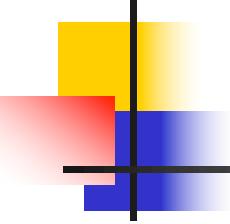
# In-class project: Android App and Firebase



# Plan of This In-class Project

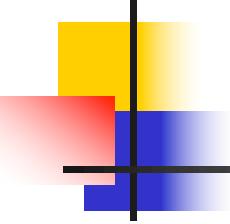
---

- 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



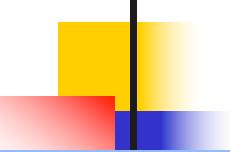
# Firebase: Create a New Project

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

<https://console.firebaseio.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



Your Firebase projects



+

Add project

JobsTracker  
jobstracker-acc89



---

Screenshot of Firebase console at [console.firebaseio.google.com](https://console.firebaseio.google.com)  
Click on **Add project** (JobsTracker has already been created)

# Firebase: Create a New Project

× Create a project (Step 1 of 3)

Let's start with a name for your project <sup>?</sup>

Project name

SuperApp

superapp-a62e1

Continue

Enter project name

Then, click Continue

# Firebase: Create a New Project

## Google Analytics for your Firebase project

Google Analytics is a free and unlimited analytics solution that enables targeting, reporting, and more in Firebase Crashlytics, Cloud Messaging, In-App Messaging, Remote Config, A/B Testing, and Cloud Functions.

Turn off Google Analytics for now

Google Analytics enabled

X A/B testing (?)

X Event-based Cloud Functions triggers (?)

X User segmentation across Firebase products (?)

X Free unlimited reporting (?)

X Crash reporting for users (?)

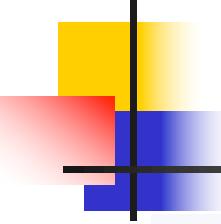
Enable Google Analytics for this project  
Recommended

This may take a moment...

Previous

Create project

Then, click Create project, which may take a moment



# Firebase: Create a New Project



SuperApp

✓ Your new project is ready

[Continue](#)



Once the new project is ready, click Continue

# Firebase: Add an Android App

The screenshot shows the Firebase console interface for adding an Android app to a project named "SuperApp". A green callout box highlights the "Spark plan" button above the main instructions. Another green callout box at the bottom encourages clicking to add an Android app. Two 3D-rendered characters, a Black man in a hoodie and a white woman in a yellow shirt, are standing on either side of the interface, gesturing towards it. The Firebase logo is visible in the background.

SuperApp ▾

SuperApp

Spark plan

Make sure you have the Spark plan

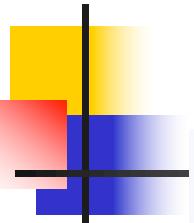
Get started by adding Firebase to your app

Android

iOS+ </> |

Add an app to get started

Click to add an Android app to this project



# Firebase: Add an Android App

## **× Add Firebase to your Android app**

## 1 Register app

Android package name [?](#)

edu.uga.cs.superapp

App nickname (optional) [?](#)

# My Android App

Debug signing certificate SHA-1 (optional) [?](#)

- Required for Dynamic Links, and Google Sign-In or phone number sign-in. Edit SHA-1s in Settings.

## Register app

Then click here

## 2 Download and then add config file

# 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](#)

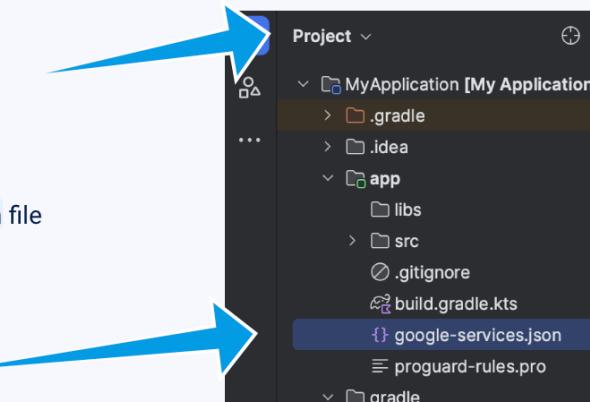
Switch to the Project view in Android Studio to see your project root directory.

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



google-services.json

[Next](#)



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

# Firebase: Configure Project

## 3 Add Firebase SDK

Instructions for Gradle | [Unity](#) [C++](#)

★ Are you still using the `buildscript syntax` to manage [add Firebase plugins](#) using that syntax.

Select Groovy  
(`build.gradle`)

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

Kotlin DSL (`build.gradle.kts`)  Groovy (`build.gradle`)

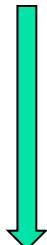
Add the plugin as a dependency to your **project-level** `build.gradle` file:

**Root-level (project-level) Gradle file** (`<project>/build.gradle`):

```
plugins {
    // ...

    // Add the dependency for the Google services Gradle plugin
    id 'com.google.gms.google-services' version '4.4.1' apply false
}
```

and scroll down

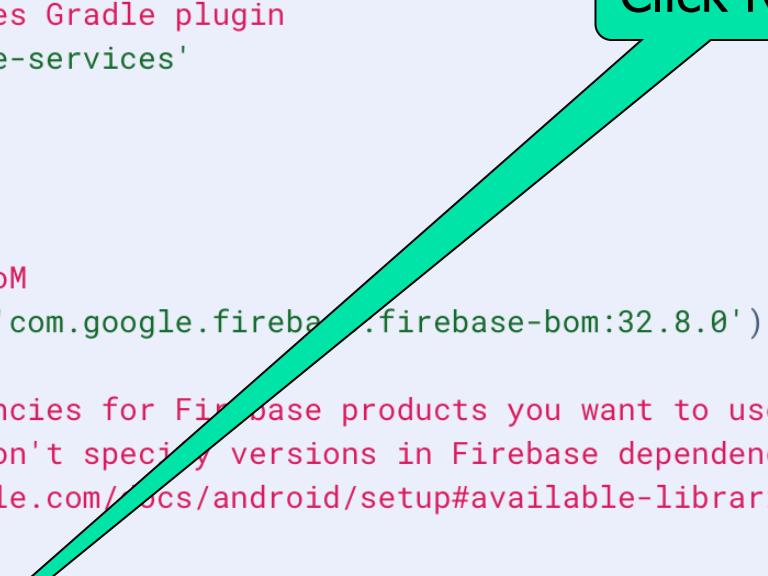


# Firebase: Configure Project

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  
}
```



**Click Next**

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**

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

# Firebase: Configure Project

## × Add Firebase to your Android app

- ✓ Register app  
Android package name: edu.uga.cs.superapp
- ✍ Download and then add config file
- ✍ Add Firebase SDK
- 4 Next steps

Click Continue to console

You're all set!

Make sure to check out the [documentation](#) to learn how to get started with each Firebase product that you want to use in your app.

You can also explore [sample Firebase apps](#).

Or, continue to the console to explore Firebase.

Previous

Continue to console

# Firebase: Configure Project

The screenshot shows the Firebase Project Overview interface for a project named "SuperApp". The left sidebar includes links for "Project Overview", "What's new", "Extensions (NEW)", "Release Monitor (NEW)", "Product categories", "Build", "Release & Monitor", "Analytics", "Engage", and "All products". A message at the bottom says "Customize your nav! You can now focus your console". The main area displays the "SuperApp" name, a "Spark plan" badge, and a list of apps: "edu.uga.cs.supera...". It features a call-to-action "Choose a product to add to your app" and two cards: "Authentication" (with a user icon) and "Cloud Firestore" (with a database icon). A large green arrow points from the text "Click here" in a teal box on the right towards the "See all Build features" link at the bottom right of the "Cloud Firestore" card.

Click here

SuperApp Spark plan

edu.uga.cs.supera... + Add app

Choose a product to add to your app

Store and sync app data in milliseconds

Authentication

Authenticate and manage users

Cloud Firestore

Realtime updates, powerful queries, and automatic scaling

See all Build features

Click on **See all Build features** and then select Realtime Database

# Firebase: Configure Project

The screenshot shows the Firebase Realtime Database configuration interface. On the left, a sidebar menu includes 'Project Overview', 'Realtime Database' (which is selected and highlighted in blue), 'What's new', 'Extensions (NEW)', 'Release Monit...', 'Product categories', 'Build', 'Release & Monitor', 'Analytics', and 'Engage'. The main content area is titled 'Realtime Database' and features the sub-headline 'Store and sync data in real time'. A large blue checkmark icon is positioned next to a stack of three blue server racks. Below the server racks, there are three circular user icons connected by lines, representing a network. At the bottom of the main area, a white callout bar contains the text 'Is Realtime Database right for you?' followed by a blue 'Compare Databases' button.

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

# Firebase: Configure Project

The screenshot shows a modal dialog titled "Set up database". It has two tabs at the top: "Database options" (selected) and "Security rules". Below the tabs, a message says "Your location setting is where your Realtime Database data will be stored." A dropdown menu labeled "Realtime Database location" contains the option "United States (us-central1)". At the bottom right of the dialog are "Cancel" and "Next" buttons. The footer of the dialog bar includes the text "Introducing Firebase Realtime Database" and the Firebase logo.

Set up database

1 Database options — 2 Security rules

Your location setting is where your Realtime Database data will be stored.

Realtime Database location

United States (us-central1)

Cancel Next

Introducing Firebase Realtime Database

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

# Firebase: Configure Project

Realtime Database

## Set up database

 Database options —  2 Security rules

Select “Start in test mode”

Once you have defined your data structure **you will need to write rules to secure your data.**

[Learn more](#)

Start in **locked mode**  
Your data is private by default. Client read/write access will only be granted as specified by your security rules.

Start in **test mode**  
Your data is open by default to enable quick setup. However, you must update your security rules within 30 days to enable long-term client read/write access.

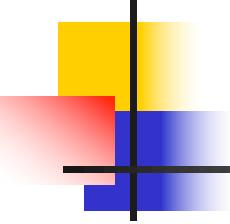
```
{  
  "rules": {  
    ".read": "now < 1715227200000", // 2024-5-9  
    ".write": "now < 1715227200000", // 2024-5-9  
  }  
}
```

! The default security rules for test mode allow anyone with your database reference to view, edit and delete all data in your database for the next 30 days

Cancel  Enable

 View the docs

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

SuperApp ▾



## Realtime Database

Data

Rules

Backups

Usage

Extensions



Protect your Realtime Database resources from abuse, such as billing fraud or phishing

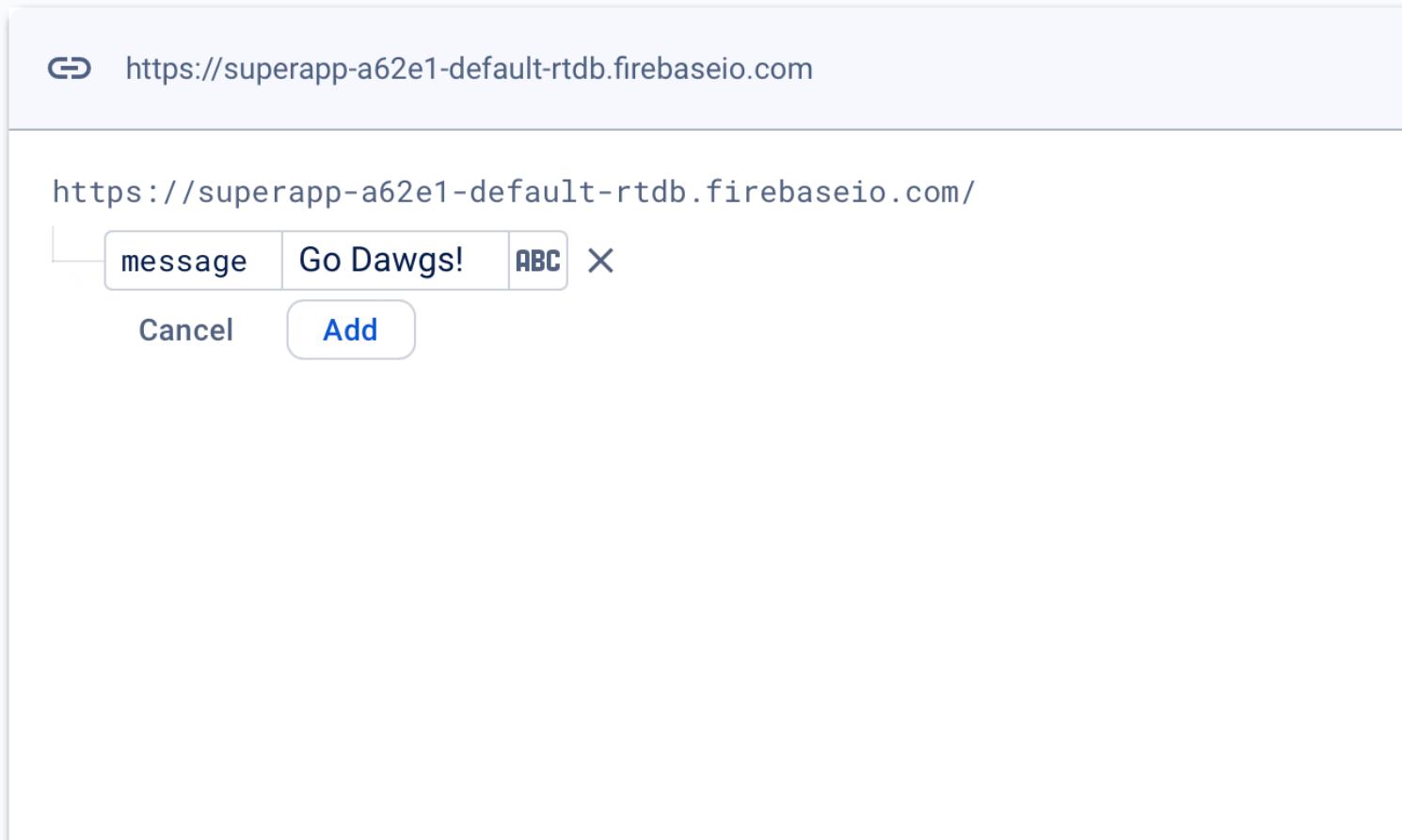
Configure App Check

🔗 https://superapp-a62e1-default-rtdb.firebaseio.com

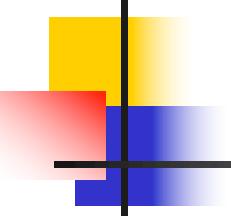
https://superapp-a62e1-default-rtdb.firebaseio.com/: null

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

Product categories

- Build
- Release & Monitor
- Analytics
- Engage

All products

Open Build

<https://superapp-a62e1-default.firebaseio.com>

<https://superapp-a62e1-default.firebaseio.com/>

message: "Go Dawgs!"

# Firebase: Configure Authentication

Product categories

Build

Authentication

App Check

Firestore Database

Realtime Database

Extensions

Storage

Hosting

Functions

Machine Learning

Remote Config

Release & Monitor

Analytics

Engage

Click Authentication

https://superapp-a62e1-default-rtbd.firebaseio.com

https://superapp-a62e1-default-rtbd.firebaseio.com/

message: "Go Dawgs!"

# Firebase: Configure Authentication

The screenshot shows the left sidebar of the Firebase console. At the top is the 'Project Overview' section. Below it are 'Project shortcuts', 'Realtime Database' (selected), and 'Authentication' (highlighted with a blue background). Further down are sections for 'Build', 'Release & Monitor', 'Analytics', and 'Engage'. At the bottom of the sidebar are buttons for 'All products', 'Customize your nav!', and links to 'Learn more' and 'Got it'.

The main page title is 'Authentication'. A sub-headline says 'Authenticate and manage users from a variety of providers without server-side code'. A large 'Get started' button is centered. To the right is a cartoon illustration of a yellow badge with a photo of a person, a blue asterisk, and a blue padlock with a checkmark. Below this are sections for 'Learn more' and 'Introducing Firebase Authentication' featuring icons for Google, Facebook, Twitter, GitHub, and a user profile.

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

# Firebase: Configure Authentication

The screenshot shows the Firebase console's Authentication page. On the left, there's a sidebar with project navigation and product categories like Build, Release & Monitor, Analytics, Engage, and All products. A callout bubble in the top right corner says "Select ‘Email/Password’". A large green arrow points from this bubble to the "Email/Password" provider in the "Native providers" section. The main page has tabs for Users, Sign-in method (which is selected), Templates, Usage, Settings, and Extensions.

SuperApp ▾

## Authentication

Users    **Sign-in method**    Templates    Usage    Settings    Extensions

Sign-in providers

Get started with Firebase Auth by adding your first sign-in method

Native providers	Additional providers	Custom providers
Email/Password	Google	Play Games
Phone	Facebook	Github
Anonymous	Game Center	Microsoft
	Apple	Twitter
		Yahoo

Advanced

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

# Firebase: Configure Authentication

The screenshot shows the Firebase console's Authentication screen for a project named "SuperApp". The left sidebar contains navigation links for Project Overview, Authentication (which is selected and highlighted in blue), and other products like Realtime Database and Analytics. The main content area is titled "Authentication" and includes tabs for Users, Sign-in method (which is selected), Templates, Usage, Settings, and Extensions. Under the "Sign-in providers" section, two options are listed: "Email/Password" (enabled) and "Email link (passwordless sign-in)" (disabled). A "Save" button is located at the bottom right.

SuperApp ▾

## Authentication

Users    **Sign-in method**    Templates    Usage    Settings    Extensions

### Sign-in providers

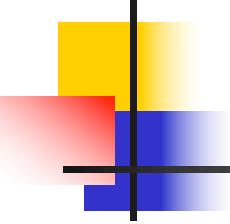
Email/Password Enable

Allow users to sign up using their email address and password. Our SDKs also provide email address verification, password recovery, and email address change primitives. [Learn more](#)

Email link (passwordless sign-in) Disable

Cancel Save

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



# Firebase: Configure Authentication

## Authentication

Users    Sign-in method    Templates    Usage    Settings     Extensions

Identifier	Providers	Created ↓	Signed In	User UID	  
No users for this project yet					

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

Users    Sign-in method    Templates    Usage    Settings    🌐 Extensions



Search by email address, phone number, or user UID

Add user



Identifier	Providers	Created ↓	Signed In	User UID
dawg@mail.com	✉️	Apr 9, 2024		CppJOFyNALgZGCiJImEp6rDE...

Rows per page:

50 ▾

1 – 1 of 1



Authentication screen with the 'dawg@mail.com' user added

# Firebase: Get the Configuration File

- Switch to the project settings:

The screenshot shows the Firebase Project Overview page for a project named "SuperApp". On the left, there's a sidebar with navigation links: Project Overview, Authentication (which is currently selected), Realtime Database, Cloud Firestore, Cloud Functions, Cloud Storage, Cloud ML Engine, Cloud Pub/Sub, Cloud Scheduler, Cloud Run, Cloud Build, Cloud Deployment, Cloud Trace, Cloud Metrics, Cloud Logging, Cloud Monitoring, Cloud App Check, and Cloud Run Webhooks. Below the sidebar, there are sections for Product categories (Build, Release & Monitor, Analytics, Engage) and a list of users (Identifier: dawg@mail.com, Providers: Email). A green callout box points to the "Project Overview" link with the text "Open the Project Overview". Another green callout box points to the "Project settings" option in the dropdown menu with the text "And select Project settings".

Open the Project Overview

And select Project settings

# Firebase: Get the Configuration File

SuperApp ▾

## Project settings

General

Cloud Messaging

Integrations

Service accounts

Data privacy

Users and permissions

Your project

Project name

SuperApp 

Project ID  ②

superapp-a62e1

Project number  ②

614734430702

Default GCP resource location  ② Not yet selected 

Web API Key

AlzaSyCaw-WVHvM0c7W8bdB515RN86VwtmedSmc

Environment

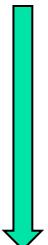
Scroll down

This setting customizes your project for different stages of the app lifecycle

Environment type

Unspecified 

Screenshot of the general info about SuperApp



# Firebase: Get the Configuration File

Your apps

Add app

Android apps



edu.uga.cs.superapp

## SDK setup and configuration

Need to reconfigure the Firebase SDKs for your app? Revisit the SDK setup instructions or just download the configuration file containing keys and identifiers for your app.

[See SDK instructions](#)

[google-services.json](#)

App ID [?](#)

1:614734430702:android:f5e4e9a00428f95ff

App nickname

Add a nickname

Package name

edu.uga.cs.superapp

SHA certificate fingerprints [?](#)

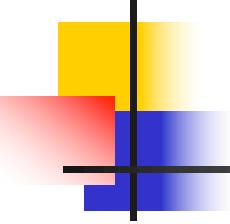
Type [?](#)

[Add fingerprint](#)

Download the app  
Configuration file

[Remove this app](#)

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

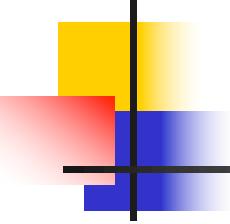


# Create a New Android Studio Project

You may be presented with a popup saying:

Do you want to allow downloads on  
“console.firebaseio.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:

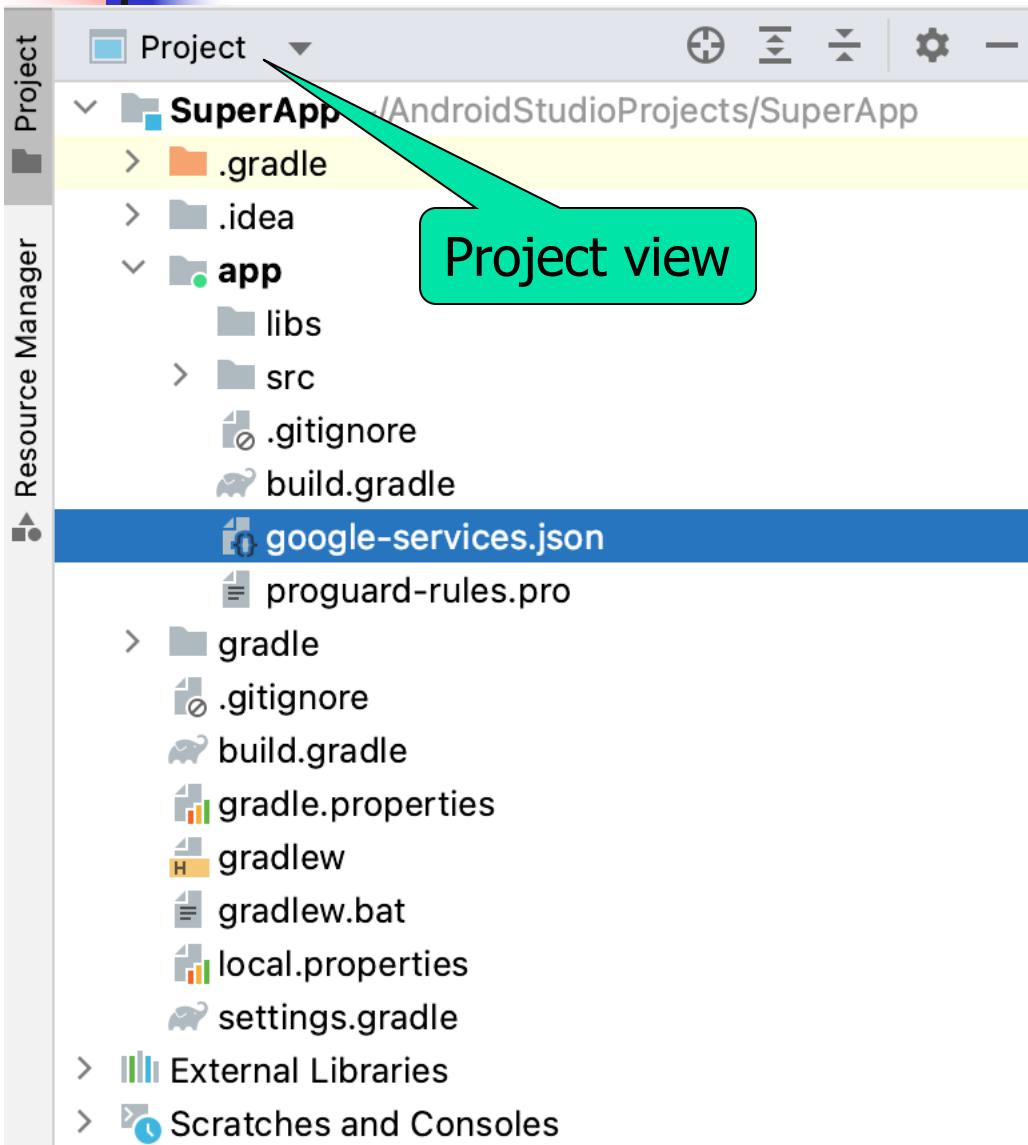
Select the Empty Views Activity and make sure the package name is (**exactly as specified when creating a Firebase project**)

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.**

Add this line

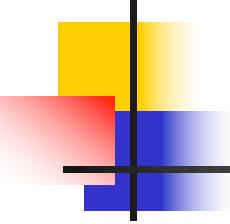
```
// Top-level build file where you can add configuration options common to all sub-projects/modules.  
plugins {  
    id 'com.google.gms.google-services' version '4.4.1' apply false  
    id 'com.android.application' version '8.2.1' apply false  
}
```

# Firebase: configuring the app

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

```
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.6.1'  
    implementation 'com.google.android.material:material:1.11.  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'  
  
    implementation platform('com.google.firebase:firebase-bom:32.8.0')  
    implementation 'com.google.firebase:firebase-auth'  
    implementation 'com.google.firebaseio.firebaseio-database'  
    implementation 'com.firebaseioui:firebase-ui-auth:7.2.0'  
}
```

Add these four lines



# 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
}
```

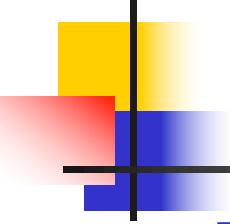
# Firebase documentation

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"
```



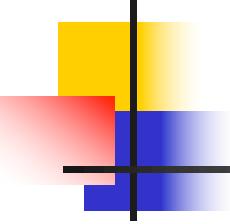
# Create a new Android Studio Project

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;
```



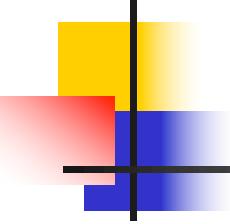
# Create a new Android Studio Project

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.

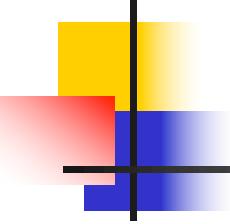
You will need to add imports to several  
other classes after the next few slides.



# 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

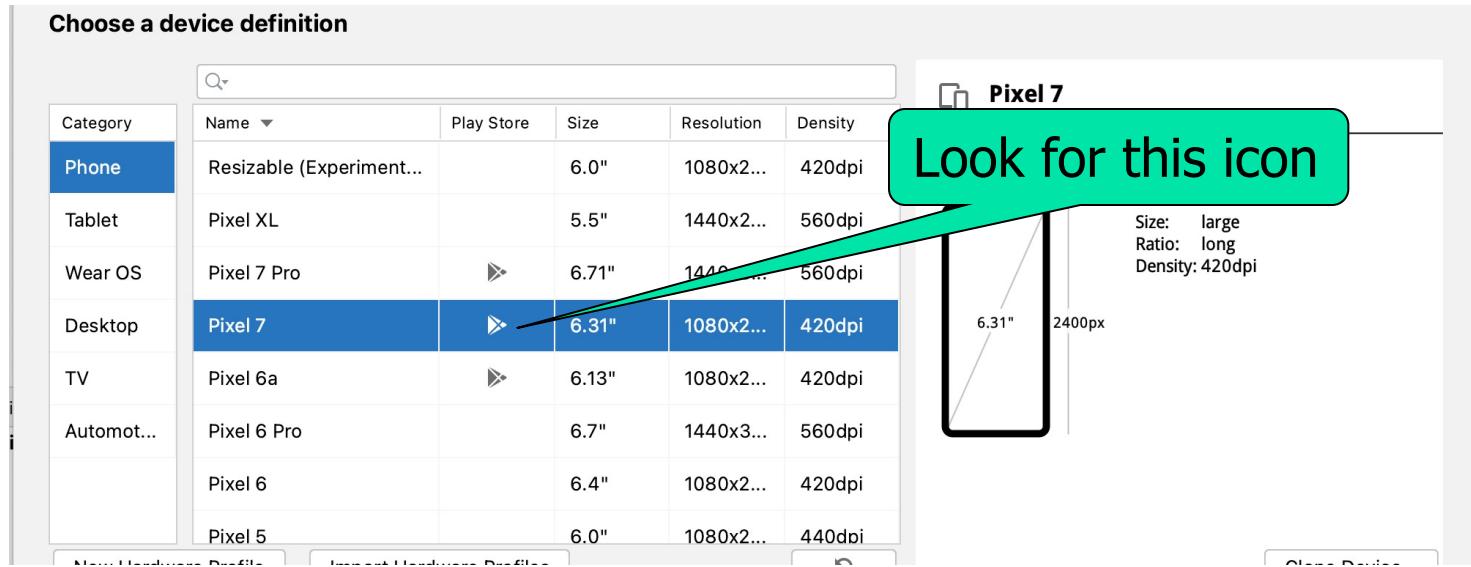
Add any missing imports.

8. Make sure your emulator has Google Play available:

Create Device		Device	API	Size on disk	
<input type="checkbox"/>	Pixel_3a_API_33_arm64-v8a	Android 13.0 Google APIs   arm64	33	301 MB	
<input type="checkbox"/>	Pixel 4 API 30	Android 11.0 Google Play   arm64	30	13 GB	
<input type="checkbox"/>	Pixel 7 API 30	Android 11.0 Google Play   arm64	30	14 GB	
<input type="checkbox"/>	Samsung Tab S 8.4 API 27	Android 8.1 Google APIs   arm64	27	8.0 GB	
<input type="checkbox"/>	Samsung Tab S 8.4 API 27 2	Android 8.1 Google APIs   arm64	27	8.0 GB	

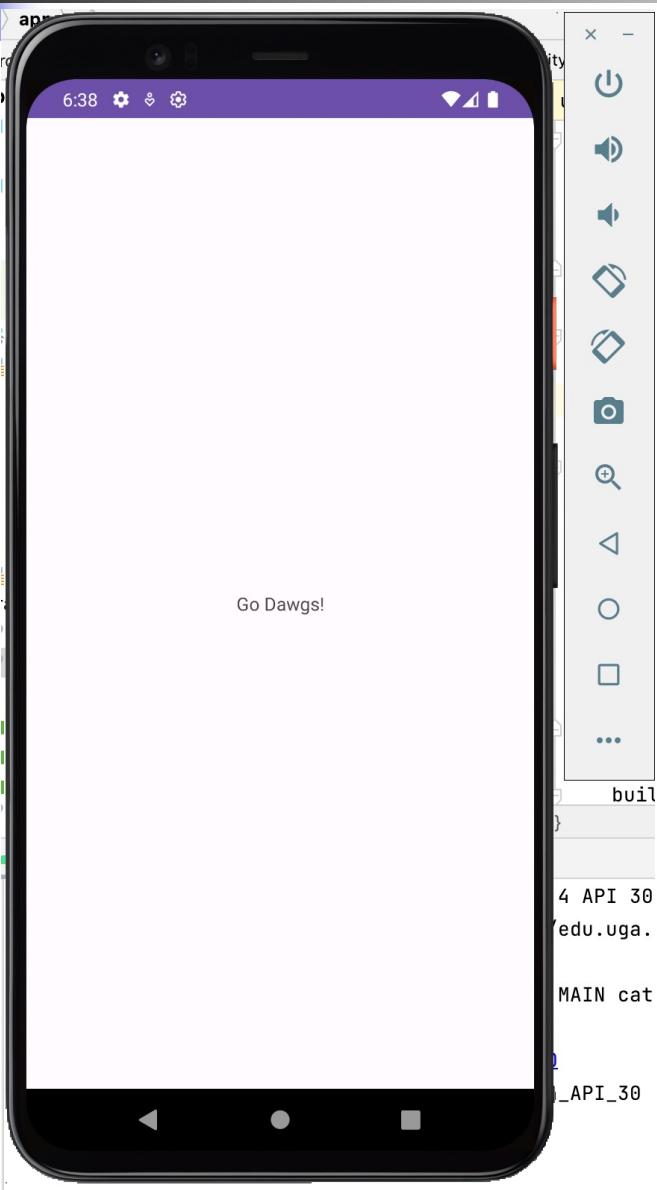
If not, create one with Google Play available

# Check Your Emulator for Google Play



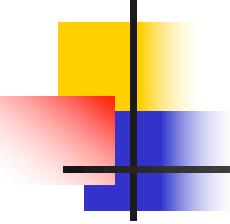
- A good choice is a Pixel 7, which is already predefined in Android Studio; it has a good “skin” available
- The API 31-32 is a good Android version for this new AVD

# 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.**