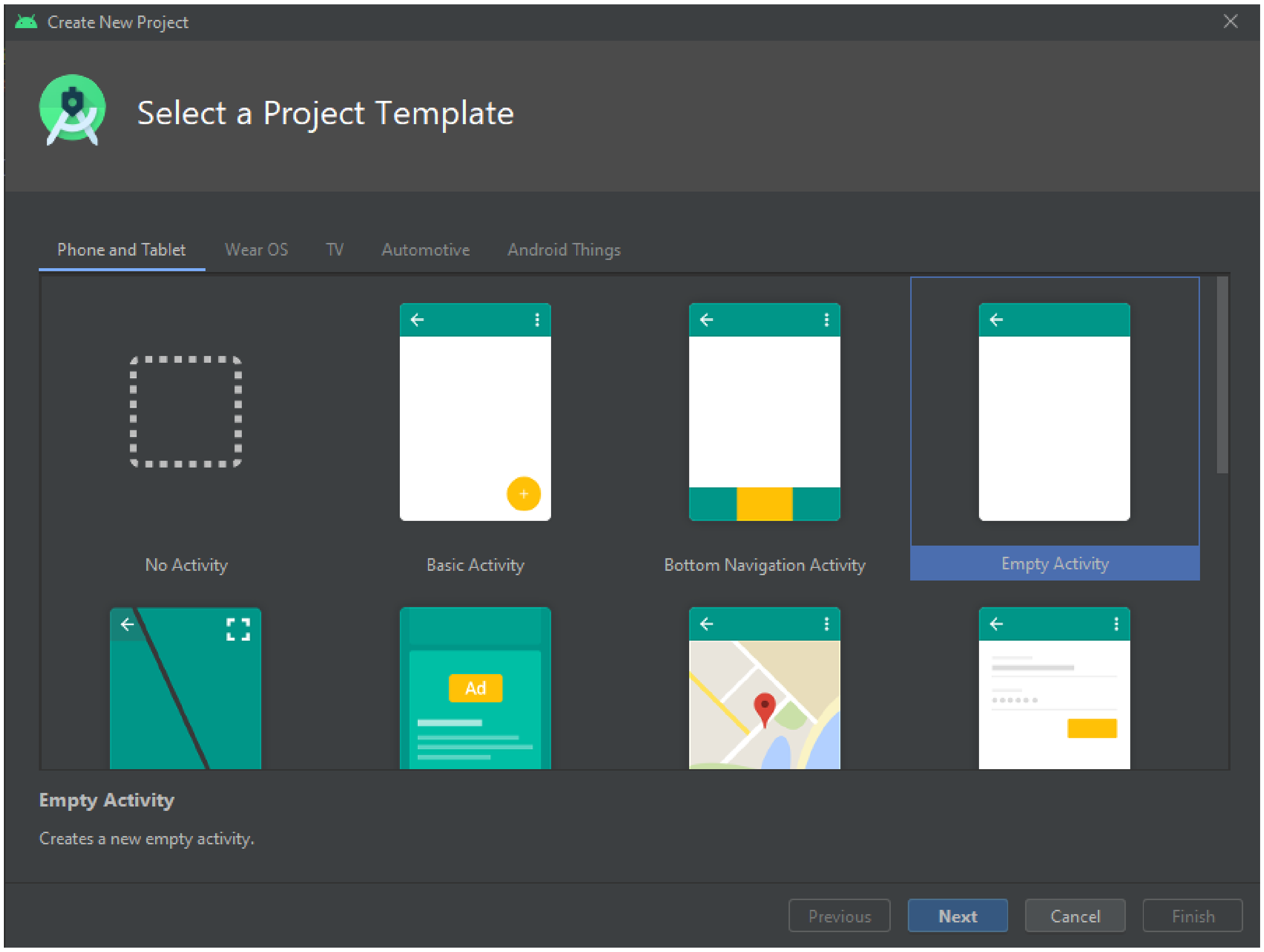
Firebase Cloud Messaging Tutorial

In this tutorial I will explain the steps needed to enable firebase cloud messaging and link it to an Android project, as well as how to register an app to receive notifications for a cloud messaging 'topic' they are subscribed to.

Initial Android project setup

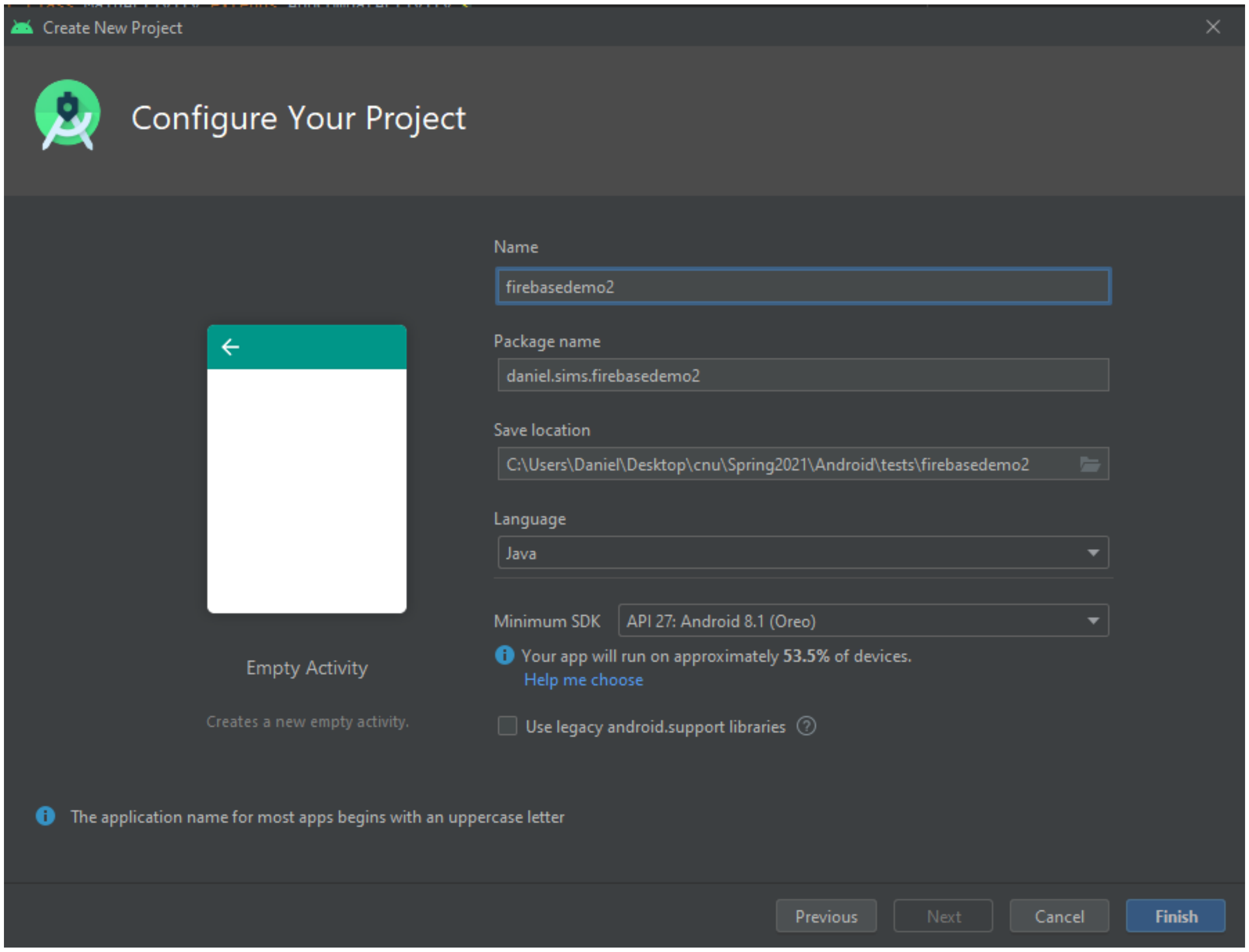
Step 1) Create a new Android project with an Empty Activity.



Empty Activity

Creates a new empty activity.

Step 2) Ensure your minimum sdk version is greater than API level 16 (Jelly Bean). I target API level 27 in this tutorial to be safe.



Name: firebasedemo2

Package name: daniel.sims.firebasedemo2

Save location: C:\Users\Daniel\Desktop\cnu\Spring2021\Android\tests\firebasedemo2

Language: Java

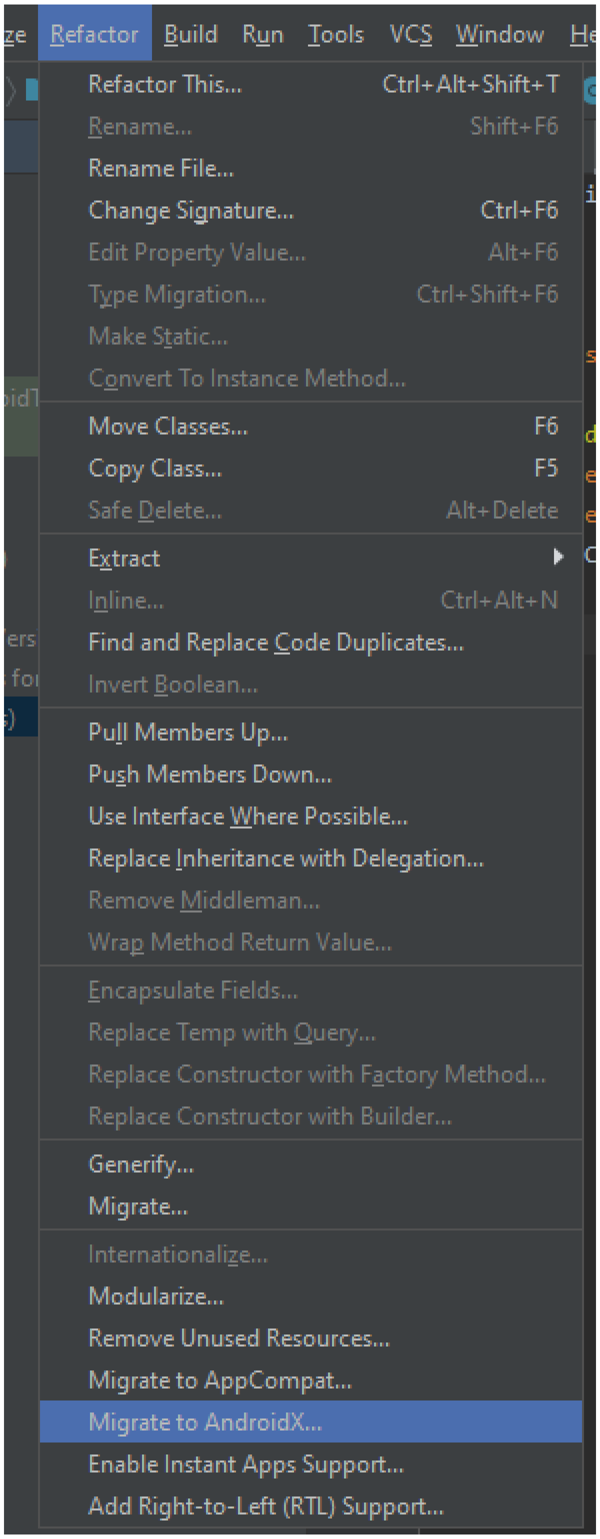
Minimum SDK: API 27: Android 8.1 (Oreo)

Your app will run on approximately **53.5%** of devices.
Help me choose

Use legacy android.support libraries

The application name for most apps begins with an uppercase letter

Step 3) In the dropdown menu **Refactor** at the top, press **Migrate to AndroidX** to ensure you have AndroidX enabled.

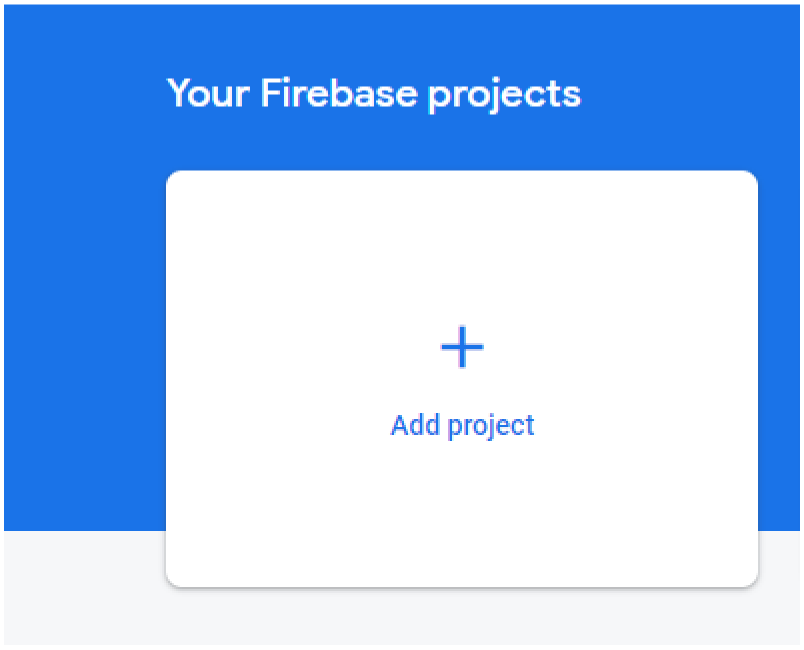

Refactor
Build
Run
Tools
VCS
Window
Refactor This... Ctrl+Alt+Shift+T
Rename... Shift+F6
Rename File...
Change Signature... Ctrl+F6
Edit Property Value... Alt+F6
Type Migration... Ctrl+Shift+F6
Make Static...
Convert To Instance Method...
Move Classes... F6
Copy Class... F5
Safe Delete... Alt+Delete
Extract
Inline... Ctrl+Alt+N
Find and Replace Code Duplicates...
Invert Boolean...
Pull Members Up...
Push Members Down...
Use Interface Where Possible...
Replace Inheritance with Delegation...
Remove Middleman...
Wrap Method Return Value...
Encapsulate Fields...
Replace Temp with Query...
Replace Constructor with Factory Method...
Replace Constructor with Builder...
Generify...
Migrate...
Internationalize...
Modularize...
Remove Unused Resources...
Migrate to AppCompat...
Migrate to AndroidX...
Enable Instant Apps Support...
Add Right-to-Left (RTL) Support...

Step 4) Do initial Firebase console setup

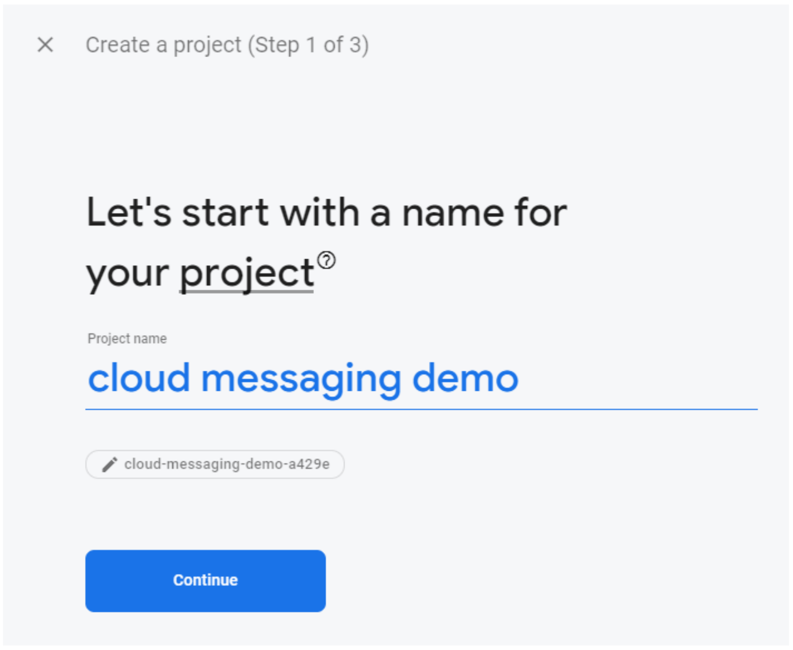
Firebase console setup

To utilize any of the tools in the Firebase platform, you need to create a project in the firebase console.

Step 1) Go to console.firebase.google.com and click **Add project.**

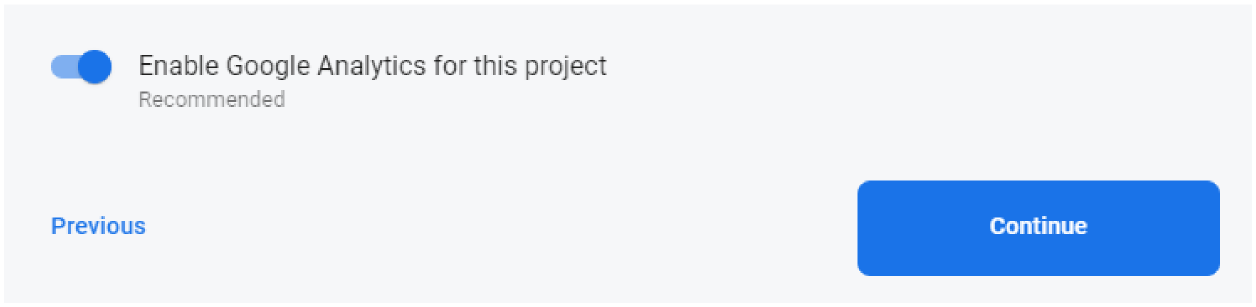


Step 2) Give your project a unique name then click **continue.**


Create a project (Step 1 of 3)
Let's start with a name for your project
Continue

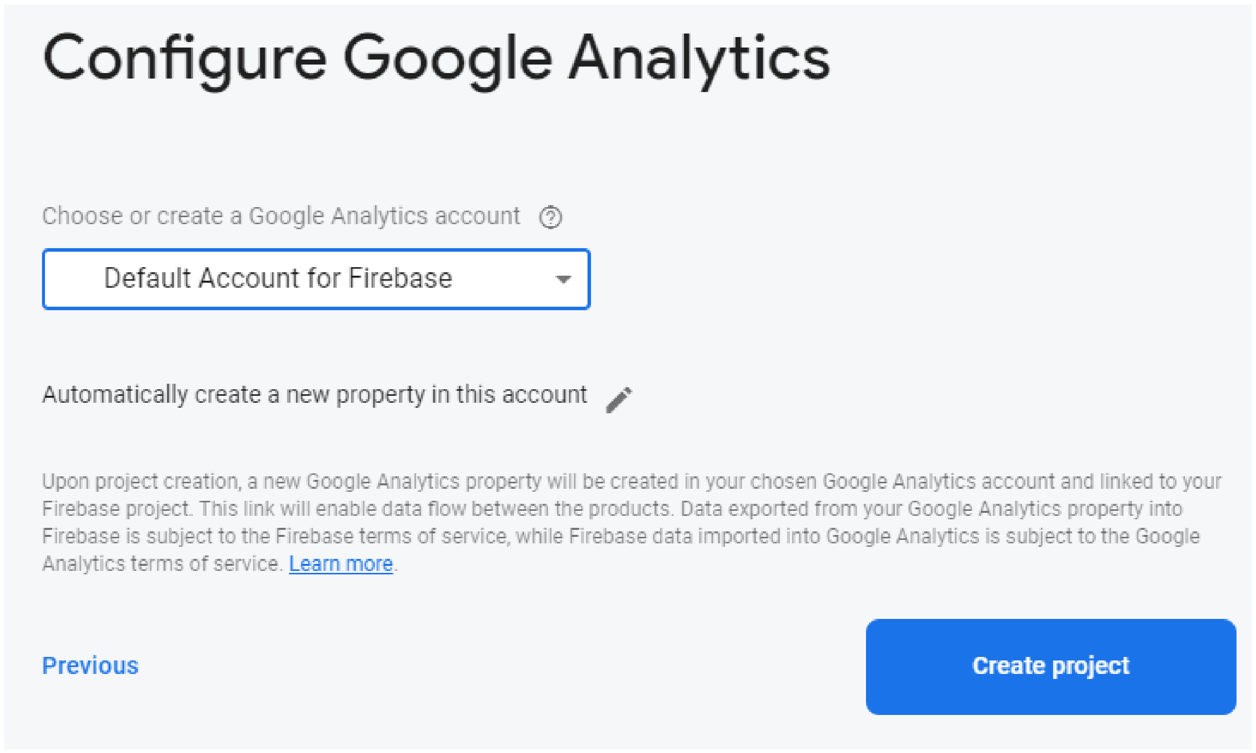
Project name
cloud messaging demo
cloud-messaging-demo-a429e

Step 3) **Enable Google Analytics** and press **Continue** (Analytics are needed for cloud messaging).



Enable Google Analytics for this project
Recommended

Step 4) Select **Default Account for Firebase** in Google Analytics account selection dropdown, then click **Create project.** (Note this will take up to a few minutes)



Configure Google Analytics

Choose or create a Google Analytics account

Default Account for Firebase

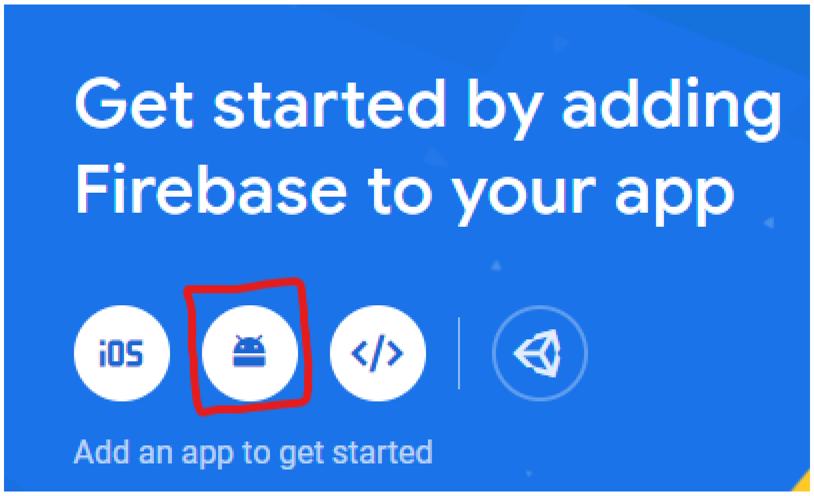
Automatically create a new property in this account

Upon project creation, a new Google Analytics property will be created in your chosen Google Analytics account and linked to your Firebase project. This link will enable data flow between the products. Data exported from your Google Analytics property into Firebase is subject to the Firebase terms of service, while Firebase data imported into Google Analytics is subject to the Google Analytics terms of service. Learn more.

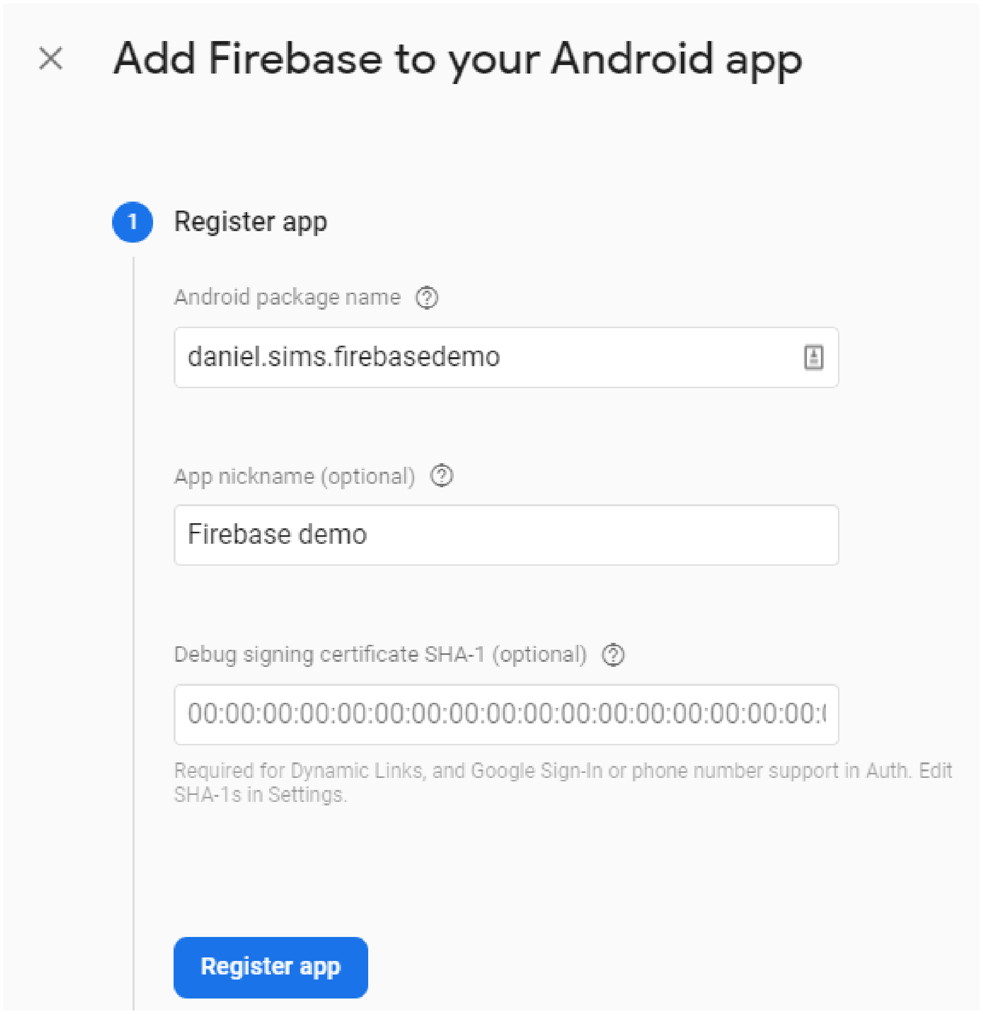
Associate Android App with Firebase Project

For your Android app to be able to communicate with Firebase and utilize the various tools and methods provided by the Firebase platform, you need to associate the Android package with your Firebase project to ensure that they communicate properly.

Step 1) Select the **android icon** to add an Android app


Get started by adding Firebase to your app
iOS
</>
Add an app to get started

Step 2) Fill in the **package name** of your Android project, and optionally a nickname and Debug signing SHA-1 key


Add Firebase to your Android app

1 Register app

Android package name

daniel.sims.firebasedemo

App nickname (optional)

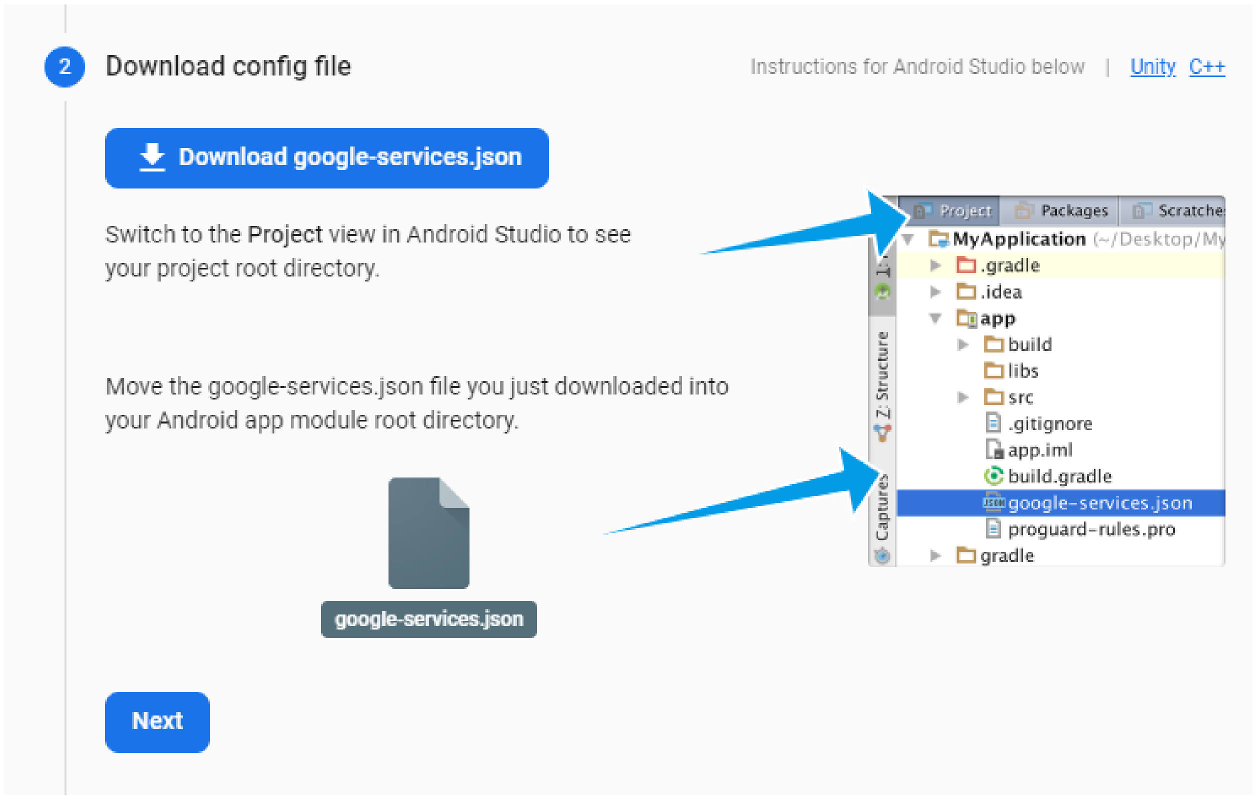
Firebase demo

Debug signing certificate SHA-1 (optional)

00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:0

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

Step 3) Click **Register app** and download the **google-services.json**. Place this json file in your project's root directory.



2 Download config file

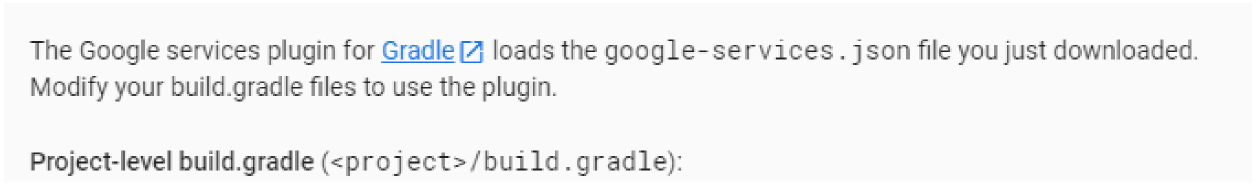
Instructions for Android Studio below | Unity C++

Download google-services.json

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

Move the google-services.json file you just downloaded into your Android app module root directory.

Step 4) Add Firebase SDK to `build.gradle` files.



The Google services plugin for Gradle loads the `google-services.json` file you just downloaded. Modify your build.gradle files to use the plugin.

Project-level build.gradle (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
  }
  dependencies {
    ...
    // Add this line
    classpath 'com.google.gms:google-services:4.3.5'
  }
}
```

```
allprojects {
  ...
  repositories {
    // Check that you have the following line (if not, add it):
    google()  // Google's Maven repository
    ...
  }
}
```

App-level build.gradle (`<project>/<app-module>/build.gradle`):

```
apply plugin: 'com.android.application'
// Add this line
apply plugin: 'com.google.gms.google-services'

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:27.0.0')

  // Add the dependency for the Firebase SDK for Google Analytics
  // When using the BoM, don't specify versions in Firebase dependencies
implementation 'com.google.android.material:material:1.3.0'
  implementation 'com.google.firebase:firebase-analytics'
  implementation 'com.google.firebase:firebase-messaging'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Step 5) Click **Next** then **Continue to console**, and you are ready to launch!

Creating an app that can receive Firebase Cloud Messages (FCMs)

1) Create a service called **MyFirebaseMessagingService** which extends **FirebaseMessagingService** then create the two methods we will implement later

```
import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {
  @Override
  public void onMessageReceived(RemoteMessage remoteMessage) {

  }

  @Override
  public void onNewToken(String token) {

  }
}
```

2) Open **AndroidManifest.xml** and add the following service declaration within the <**application>** tag, but below the **<activity>** tag for your MainActivity.

```
<service
  android:name=".MyFirebaseMessagingService"
  android:exported="false">
  <intent-filter>
    <action android:name="com.google.firebase.MESSAGING_EVENT" />
  </intent-filter>
</service>
```

3) Add some ui to your **activity_main.xml** that you will interact with later

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  tools:context=".MainActivity">

  <Button
    android:id="@+id/subscribeButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Subscribe"
    app:layout_constraintBottom_toTopOf="@+id/tokenTextView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/topicEditText" />

  <TextView
```

```xml
        android:id="@+id/tokenTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/logButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/subscribeButton" />

    <TextView
        android:id="@+id/resultsTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/topicEditText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/topicEditText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="Topic to subscribe to"
        android:inputType="textPersonName"
        app:layout_constraintBottom_toTopOf="@+id/subscribeButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/resultsTextView" />

    <Button
        android:id="@+id/logButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Log Token"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tokenTextView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4) Initialize all UI elements in MainActivity

```java
private Button subscribeButton, logButton;
private TextView resultsTextView, tokenTextView;
private EditText topicEditText;
```

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  subscribeButton = findViewById(R.id.subscribeButton);
  logButton = findViewById(R.id.logButton);
  resultsTextView = findViewById(R.id.resultsTextView);
  tokenTextView = findViewById(R.id.tokenTextView);
  topicEditText = findViewById(R.id.topicEditText);
}
```

5) Add the ability to subscribe to a topic of your choice

```
subscribeButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
     String topicToSubscribeTo = topicEditText.getText().toString();
     // Check to make sure we always have a valid topic
     if (topicToSubscribeTo.equals("")) {
        topicToSubscribeTo = "empty_topic";
     }
     topicToSubscribeTo = topicToSubscribeTo.toLowerCase().replace(" ", "");


FirebaseMessaging.getInstance().subscribeToTopic(topicToSubscribeTo).addOnCompleteListener(new
OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {
           String message = "Topic subscription failed!";
           if (task.isSuccessful()) {
              message = "Topic subscription succeeded!";
           }
           Toast.makeText(MainActivity.this, message, Toast.LENGTH_SHORT).show();
        }
     });
  }
});
```

6) Add the ability to log and show a the current user **Token**

```
logButton.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View view) {
     FirebaseMessaging.getInstance().getToken()
           .addOnCompleteListener(new OnCompleteListener<String>() {
              @Override
```

```
            public void onComplete(@NonNull Task<String> task) {
                if (!task.isSuccessful()) {
                    return;
                }
                // Get new FCM registration token
                String token = task.getResult();
                // Log and toast
                String message = "FCM registration token: " + token;
                tokenTextView.setText(message);
            }
        });
    }
});
```

7) **Now that you have done this, launch the app and make sure you can subscribe to a topic of your choice, as well as view your FCM key**

Adding push notifications when a messages received

To be able to see the messages that are received on the device, we need to show a push notification.

1) **Create notification channel** in MainActivity onCreate() method

```
private static final String CHANNEL_NAME = "Firebase Cloud Messaging notifications";
private static final String CHANNEL_ID = "fb_notifications_channel";

// within onCreate method
  // Create channel to show notifications.
  NotificationManager notificationManager =
        getSystemService(NotificationManager.class);
  notificationManager.createNotificationChannel(new
NotificationChannel(CHANNEL_NAME,
        CHANNEL_ID, NotificationManager.IMPORTANCE_LOW));
```

2) Head over to the messaging service and implement **onMessageReceived()**

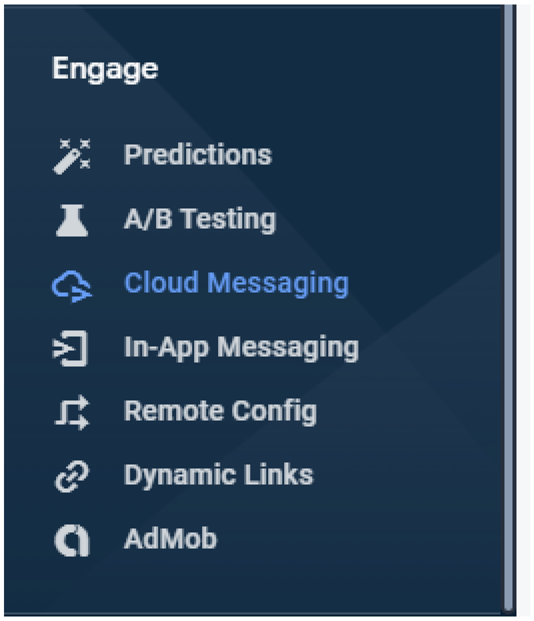
```
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
  String body = remoteMessage.getNotification().getBody();
  generatePushNotification(body);
}
```

3) Implement **generatePushNotification()** so that we can get push notifications on the app.

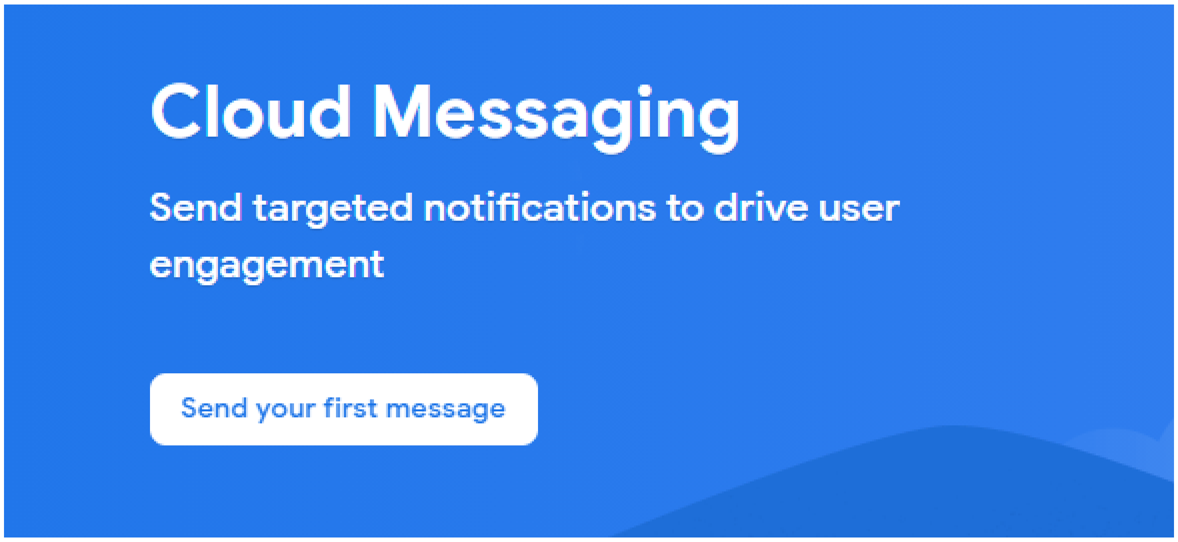
```
private void generatePushNotification(RemoteMessage.Notification fcmNotification) {
  // Ensure that body/title aren't null
  String body = fcmNotification.getBody();
  String title = fcmNotification.getTitle();
  if (body == null) {
      body = "";
  }
  if (title == null) {
      title = "";
  }
  Intent intent = new Intent(this, MainActivity.class);
  intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
  PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
          PendingIntent.FLAG_ONE_SHOT);

  String channelId = MainActivity.CHANNEL_ID;
  Uri defaultSoundUri =
RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
  NotificationCompat.Builder notificationBuilder =
          new NotificationCompat.Builder(this, channelId)
                  .setSmallIcon(R.drawable.ic_launcher_foreground)
                  .setContentTitle(title)
                  .setContentText(body)
                  .setAutoCancel(true)
                  .setSound(defaultSoundUri)
                  .setContentIntent(pendingIntent);

  NotificationManager notificationManager =
          (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

  NotificationChannel channel = new NotificationChannel(channelId,
          MainActivity.CHANNEL_NAME,
          NotificationManager.IMPORTANCE_HIGH);
  notificationManager.createNotificationChannel(channel);
  notificationManager.notify(1 /* ID of notification */, notificationBuilder.build());
}
```

Check that you can receive push notifications

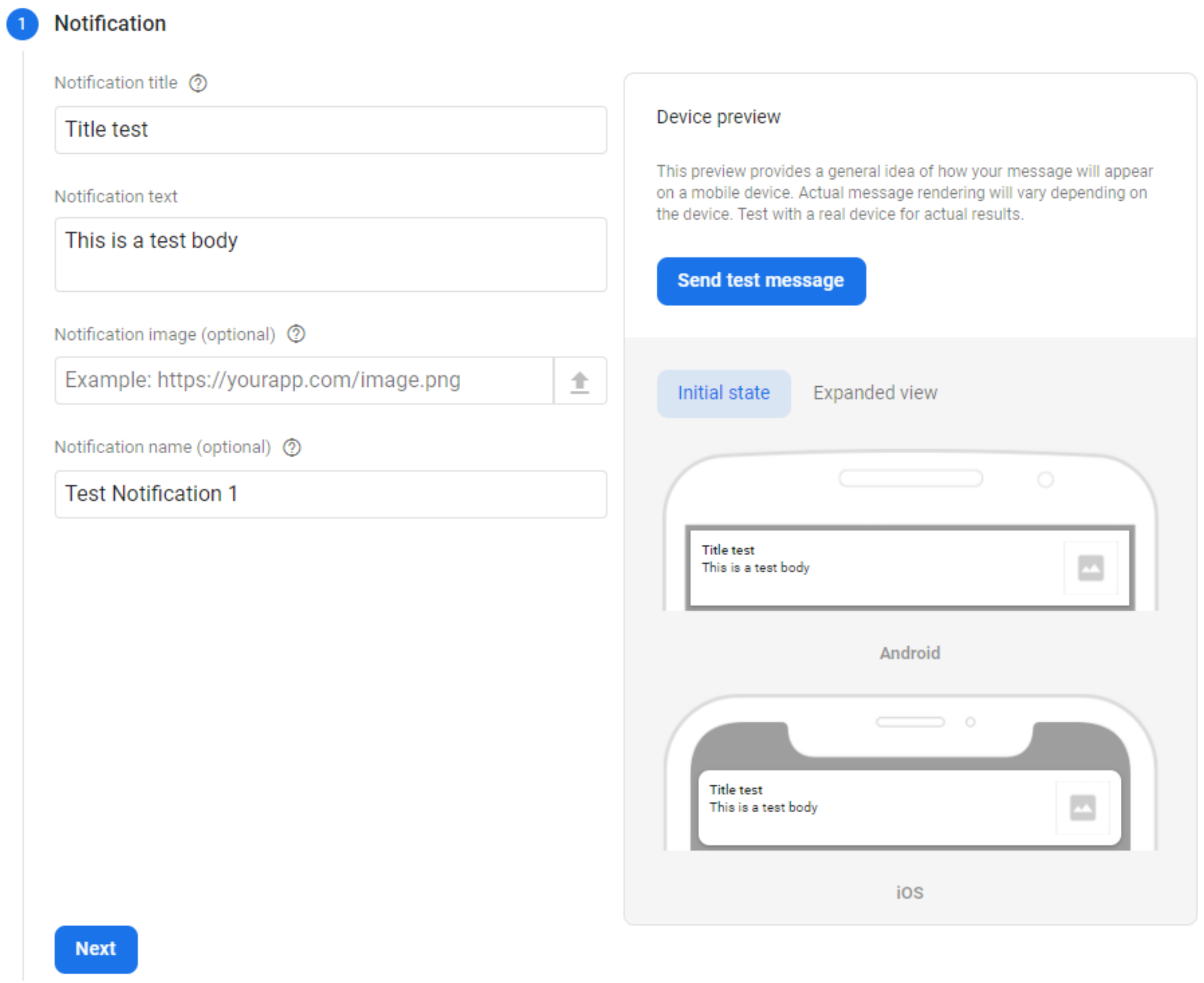
1) Go back to your firebase console for your project and click on **Cloud Messaging** in the bottom left sidebar



2) Click **Send your first message**



3) Enter a notification **name**, **title**, and **text,** then click **NEXT**



1 Notification

Notification title

Title test

Notification text

This is a test body

Notification image (optional)

Example: https://yourapp.com/image.png

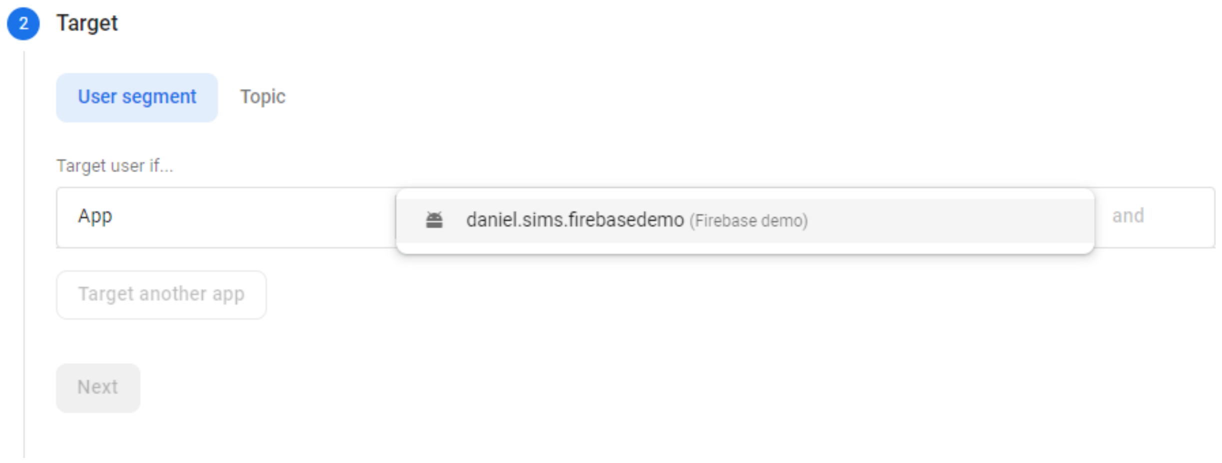
Notification name (optional)

Test Notification 1

Device preview

This preview provides a general idea of how your message will appear on a mobile device. Actual message rendering will vary depending on the device. Test with a real device for actual results.

4) Check to see if your topic has arrived in the **Topic** section and select one if it exists, if not, select your android package name from the **Select an app** dropdown.

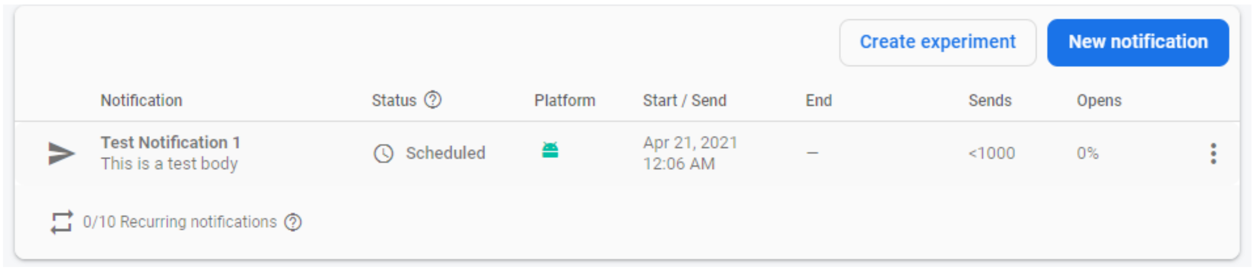


2 Target

Target user if...

App daniel.sims.firebasedemo (Firebase demo) and

5) Click **Next** then **Review** then **Publish,** This will show you the status of your sent notification.



Notification	Status	Platform	Start / Send	End	Sends	Opens
Test Notification 1 This is a test body	Scheduled		Apr 21, 2021 12:06 AM	–	<1000	0%

0/10 Recurring notifications

Displaying the contents of a received FCM on screen

Whenever we click a notification that is sent by the Firebase Cloud Messenger, it includes extras in the Intent that opens up the Main Activity. We can extract the contents of the notification through these extras.

1) In **MainActivity.java**, add the following code to the end of **onCreate()**

```
if (getIntent().getExtras() != null) {
    String results = String.format("Last Notification\nTitle: %s\nBody: %s",
getIntent().getStringExtra("title"), getIntent().getStringExtra("body"));
    resultsTextView.setText(results);
}
```

Congratulations!

You now have the ability to send and receive Firebase Cloud messages!