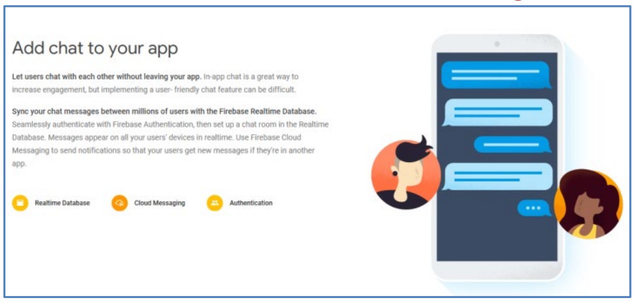
Copyright:
Author: Daniel DANG
School of Computing (SoC), Eastern Institute of Technology (EIT)
Email: daniel.dang.nz@gmail.com

Lab 5: Create a "Group Chat App" using Firebase

Objectives

Mobile applications need a **back-end server** in order to perform tasks such as **authenticating users** and **synchronizing user data across multiple devices**. Creating such a server, however, requires a skill set that most independent app developers lack. Fortunately, there are **several back-end** as a **service platforms**, often called **BaaS**, you can use today. Google's Firebase is one such platform.
Firebase offers **essential services** such as **analytics**, **crash reporting**, **user authentication**, and **cloud messaging** at no cost. Its freemium services include a **real-time NoSQL database**, **file hosting**, and **static website hosting**.

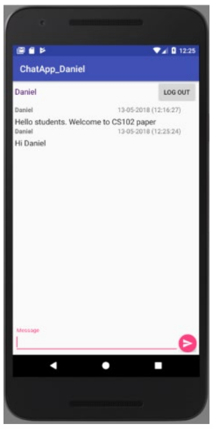


Add chat to your app

Let users chat with each other without leaving your app. In-app chat is a great way to increase engagement, but implementing a user- friendly chat feature can be difficult.

Sync your chat messages between millions of users with the Firebase Realtime Database. Seamlessly authenticate with Firebase Authentication, then set up a chat room in the Realtime Database. Messages appear on all your users' devices in realtime. Use Firebase Cloud Messaging to send notifications so that your users get new messages if they're in another app.

In this tutorial, students learn how to use Firebase UI to create a **group chat app (a real-time social application)** you can share with your friends. It's going to be a very **simple app** with just one chat room, which is open to all users.
As you might have guessed, the app will depend on **Firebase Auth** to manage **user registration and sign in**. It will also use Firebase's real-time database to store the group chat messages. You also learn how to add **user authentication**, **analytics**, and **remote real-time cloud data storage** to your Android app using **Firebase**.

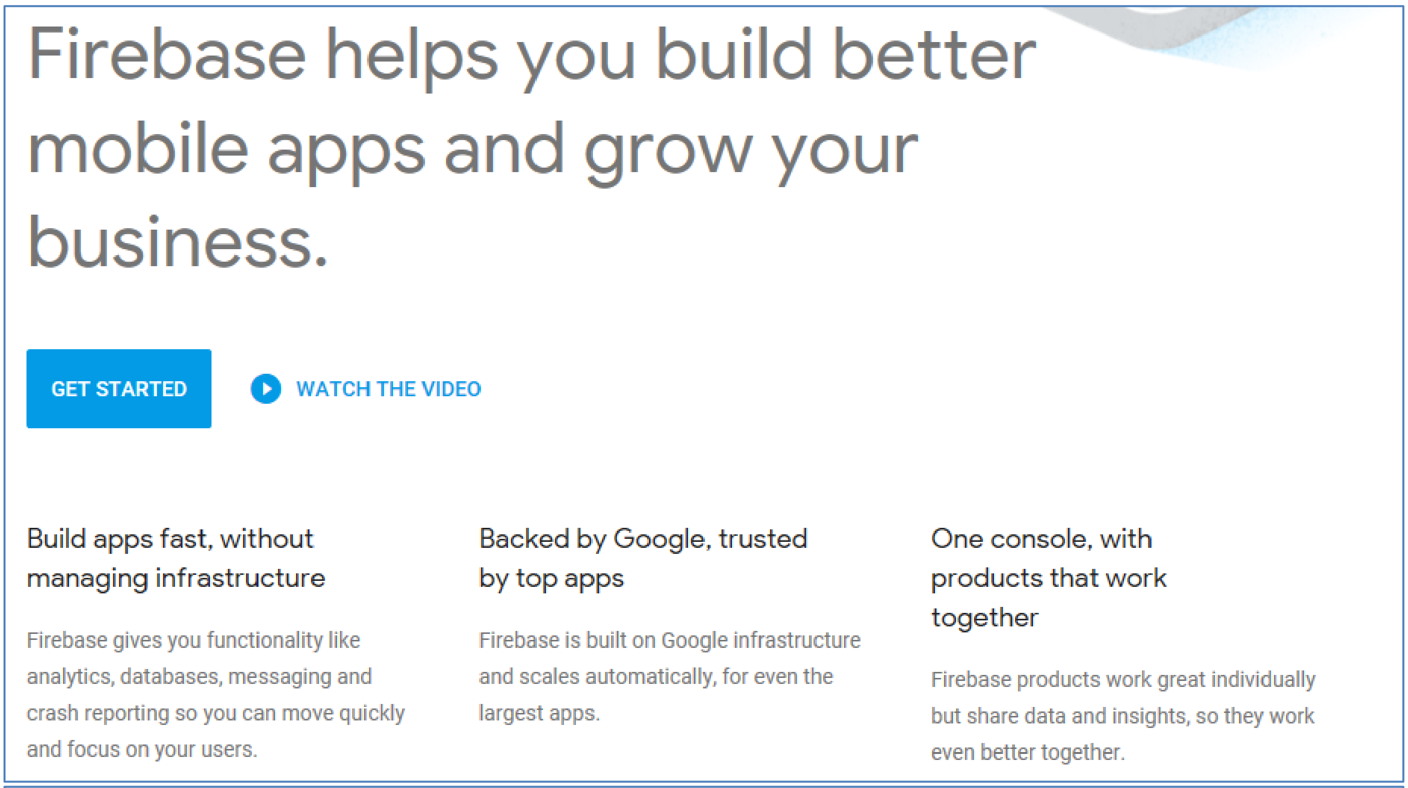


Daniel

Daniel 13-05-2018 (12:16:27)
Hello students. Welcome to CS102 paper
Daniel 13-05-2018 (12:25:24)
Hi Daniel

Sources:
[1]. Firebase tutorial: https://www.tutorialspoint.com/firebase/index.htm
[2]. Firebase for Android: https://code.tutsplus.com/tutorials/get-started-with-firebase-for-android--cms-27248
[3]. Firebase: https://firebase.google.com/
[4]. How to Create an Android Chat App Using Firebase: https://code.tutsplus.com/tutorials/how-to-create-an-android-chat-app-using-firebase--cms-27397
[5]. App chat: https://codelabs.developers.google.com/codelabs/firebase-android/#0

Firebase: https://firebase.google.com/



Firebase helps you build better mobile apps and grow your business.

Build apps fast, without managing infrastructure

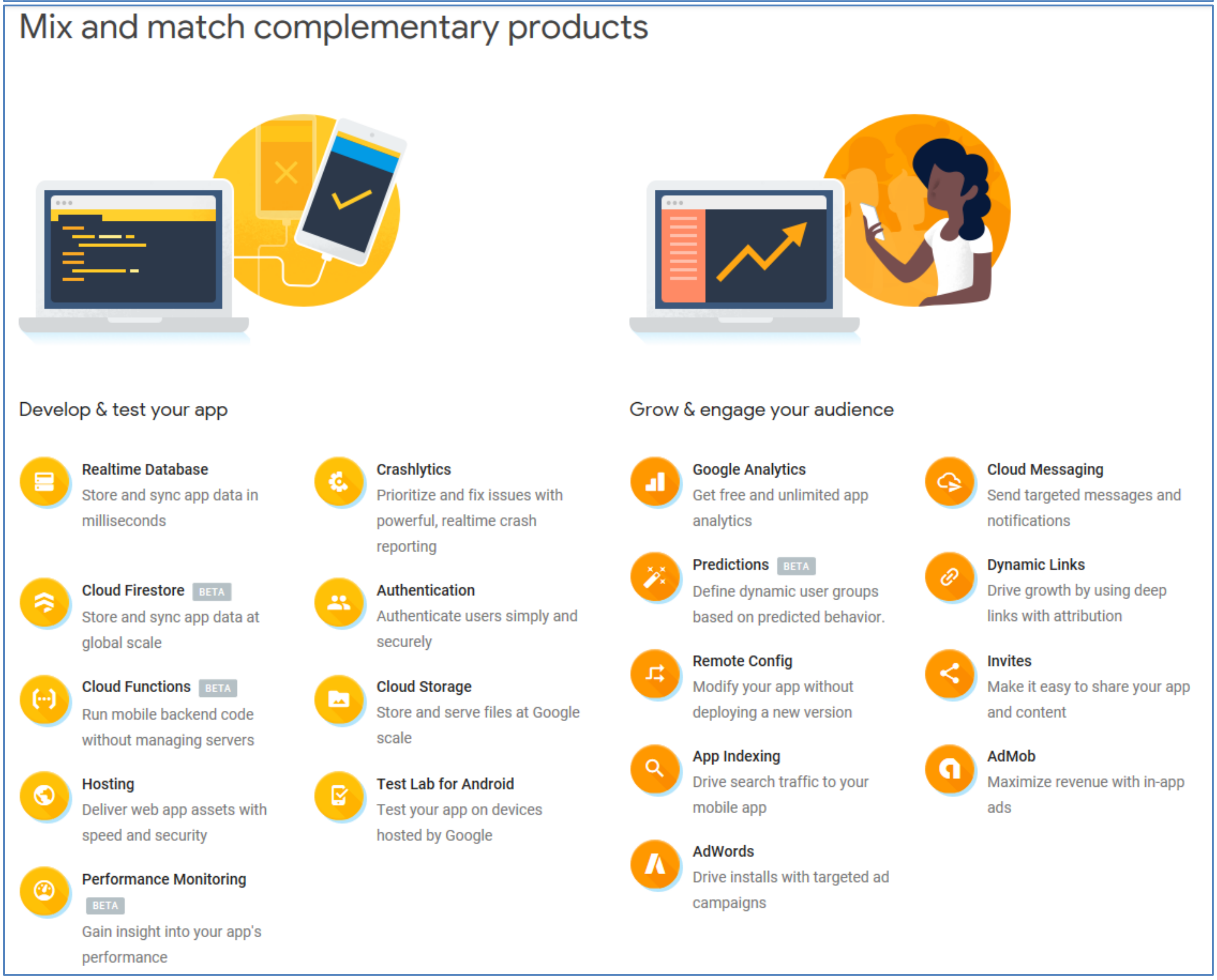
Firebase gives you functionality like analytics, databases, messaging and crash reporting so you can move quickly and focus on your users.

Backed by Google, trusted by top apps

Firebase is built on Google infrastructure and scales automatically, for even the largest apps.

One console, with products that work together

Firebase products work great individually but share data and insights, so they work even better together.



Mix and match complementary products

Develop & test your app

Realtime Database Store and sync app data in milliseconds

Cloud Firestore BETA Store and sync app data at global scale

Cloud Functions BETA Run mobile backend code without managing servers

Hosting Deliver web app assets with speed and security

Performance Monitoring BETA Gain insight into your app's performance

Crashlytics Prioritize and fix issues with powerful, realtime crash reporting

Authentication Authenticate users simply and securely

Cloud Storage Store and serve files at Google scale

Test Lab for Android Test your app on devices hosted by Google

Grow & engage your audience

Google Analytics Get free and unlimited app analytics

Predictions BETA Define dynamic user groups based on predicted behavior.

Remote Config Modify your app without deploying a new version

App Indexing Drive search traffic to your mobile app

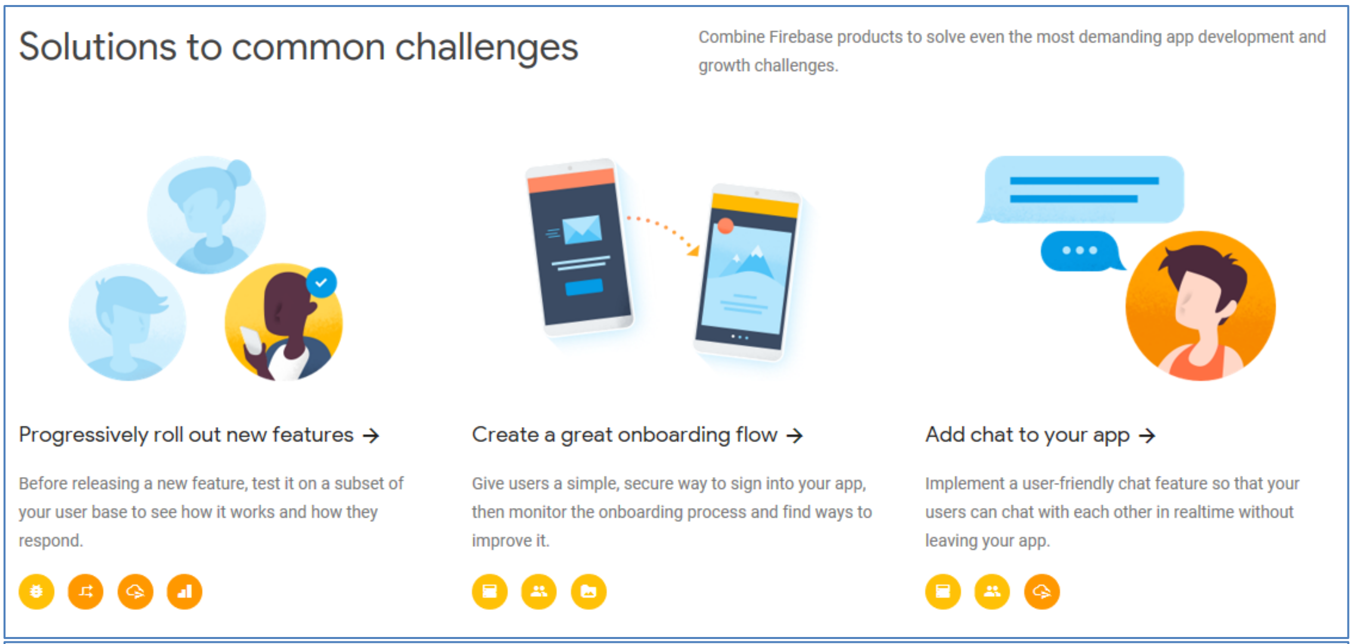
AdWords Drive installs with targeted ad campaigns

Cloud Messaging Send targeted messages and notifications

Dynamic Links Drive growth by using deep links with attribution

Invites Make it easy to share your app and content

AdMob Maximize revenue with in-app ads



Solutions to common challenges

Combine Firebase products to solve even the most demanding app development and growth challenges.

Progressively roll out new features →

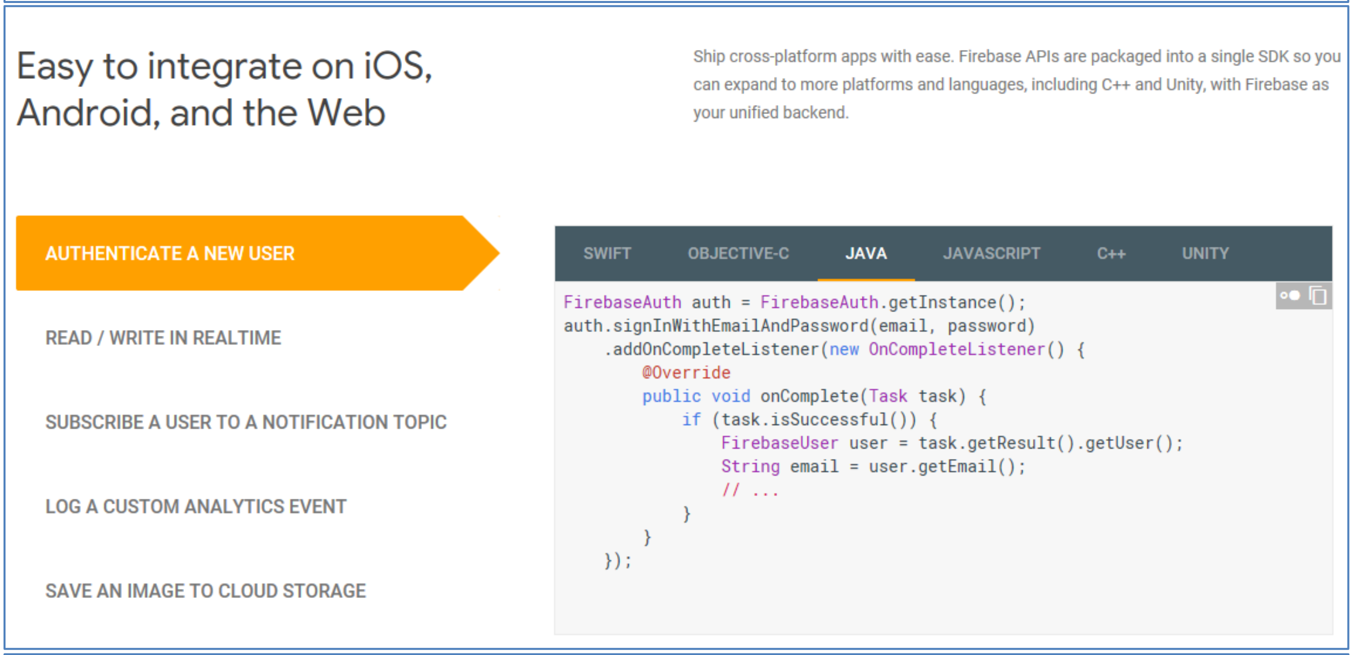
Before releasing a new feature, test it on a subset of your user base to see how it works and how they respond.

Create a great onboarding flow →

Give users a simple, secure way to sign into your app, then monitor the onboarding process and find ways to improve it.

Add chat to your app →

Implement a user-friendly chat feature so that your users can chat with each other in realtime without leaving your app.

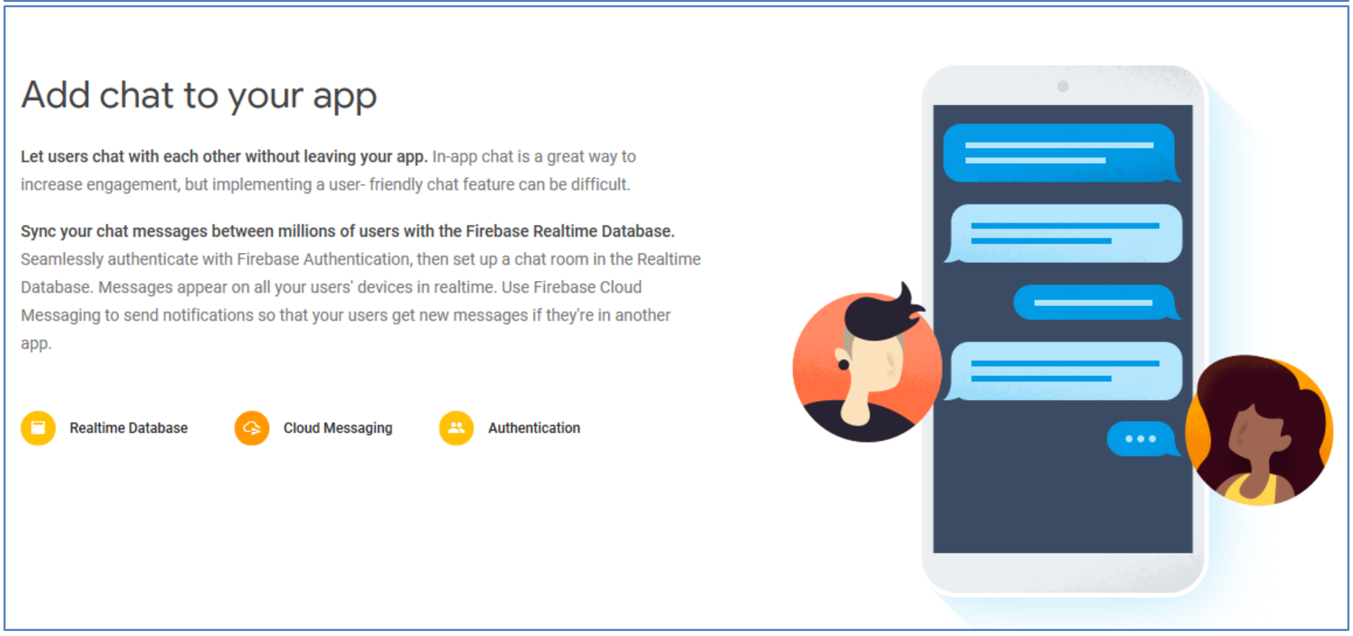


Easy to integrate on iOS, Android, and the Web

Ship cross-platform apps with ease. Firebase APIs are packaged into a single SDK so you can expand to more platforms and languages, including C++ and Unity, with Firebase as your unified backend.

AUTHENTICATE A NEW USER
READ / WRITE IN REALTIME
SUBSCRIBE A USER TO A NOTIFICATION TOPIC
LOG A CUSTOM ANALYTICS EVENT
SAVE AN IMAGE TO CLOUD STORAGE
SWIFT
OBJECTIVE-C
JAVA
JAVASCRIPT
C++
UNITY

```java
FirebaseAuth auth = FirebaseAuth.getInstance();
auth.signInWithEmailAndPassword(email, password)
    .addOnCompleteListener(new OnCompleteListener() {
        @Override
        public void onComplete(Task task) {
            if (task.isSuccessful()) {
                FirebaseUser user = task.getResult().getUser();
                String email = user.getEmail();
                // ...
            }
        }
    });
```



Add chat to your app

Let users chat with each other without leaving your app. In-app chat is a great way to increase engagement, but implementing a user- friendly chat feature can be difficult.

Sync your chat messages between millions of users with the Firebase Realtime Database. Seamlessly authenticate with Firebase Authentication, then set up a chat room in the Realtime Database. Messages appear on all your users' devices in realtime. Use Firebase Cloud Messaging to send notifications so that your users get new messages if they're in another app.

PART 1: Create Android Studio Project

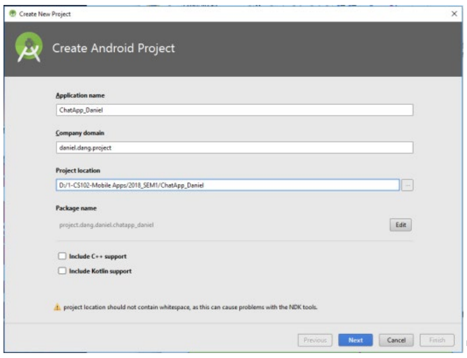
Fire up Android Studio and create a new project with an **empty activity** called **MainActivity**.

1. Create an Android Studio Project

Create a new Android Project in Android Studio IDE:

- Application name: **Lab_ChatApp _[yourname]**
- Company Domain: **ac.ames.project.[yourname]**
- Target Android devices: **Phone and Tablet**;
  - Minimum SDK: **API 21: Android 5.0**
- Add an Activity to Mobile: **Empty Activity**;
  - Activity name: **MainActivity**
  - Layout name: **activity_main**

+ Target devices: run on 71.3% devices
_Smartphone & Tablet
_Android KitKat (5.0) & API21 (min)

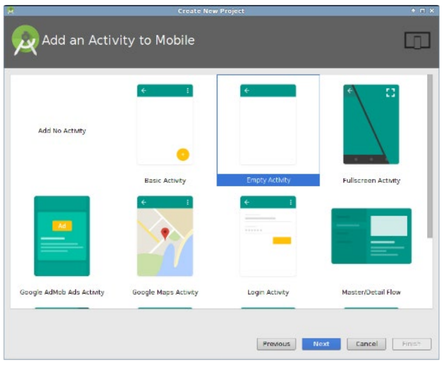


Create New Project

Create Android Project

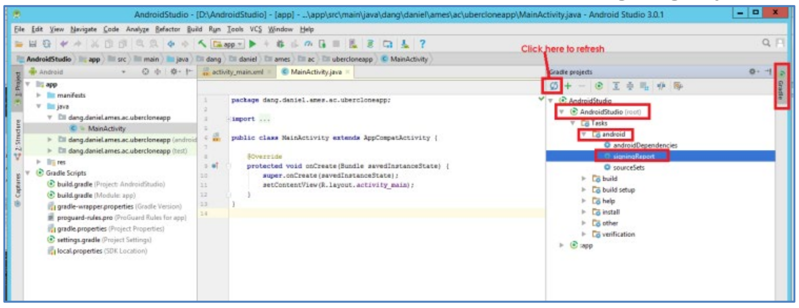
Application name: ChatApp_Daniel
Company domain: daniel.dang.project
Project location: D:/1-CS102-Mobile Apps/2018_SEM1/ChatApp_Daniel
Package name: project.dang.daniel.chatapp_daniel
Include C++ support [ ]
Include Kotlin support [ ]

project location should not contain whitespace, as this can cause problems with the NDK tools.



2. Open the Android Studio IDE to do "Android Sign Report" in order to generate a SHA1 code:

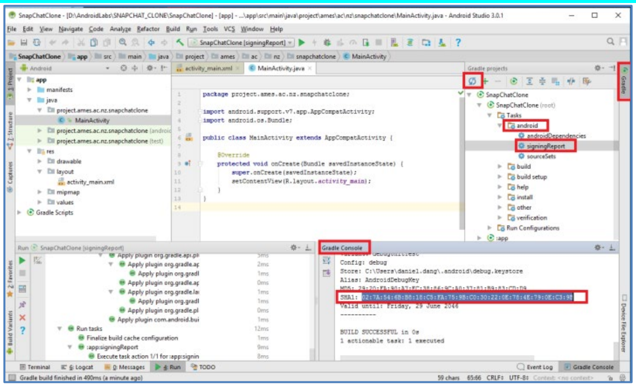
- Click "**Gradle**" → "**Refresh**" → "**Android Studio**" → "**Android**" → "**SigningReport**"



AndroidStudio - [D:\AndroidStudio] - [app] - ...\app\src\main\java\dang\daniel\ames\ac\ubercloneapp\MainActivity.java - Android Studio 3.0.1

```java
package dang.daniel.ames.ac.ubercloneapp;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

- Upon completing "**signingReport**", open up "**Gradle console**" at the bottom right corner to obtain the SHA1 code ➔ SHA1 = "***A6:66:31:8B:EE:AE:46:2B:FD:16:37:34:EC:C3:F2:71:03:F7:32:7F***"



SnapChatClone - [D:\AndroidLabs\SNAPCHAT_CLONE\SnapChatClone] - [app] - ...\app\src\main\java\project\ames\ac\nz\snapchatclone\MainActivity.java - Android Studio 3.0.1

```java
package project.ames.ac.nz.snapchatclone;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

```
Config: debug
Store: C:\Users\daniel.dang\.android\debug.keystore
Alias: AndroidDebugKey
MD5: 29:20:FA:90:A7:EC:38:86:9C:A0:37:81:B9:83:CD:D9
SHA1: 02:7A:54:6B:B8:18:C3:FA:75:9B:C0:30:22:0E:78:4E:79:0E:C3:9B
Valid until: Friday, 29 June 2046
----------

BUILD SUCCESSFUL in 0s
1 actionable task: 1 executed
```

PART 2: Create Firebase database and integrate it to Android Project

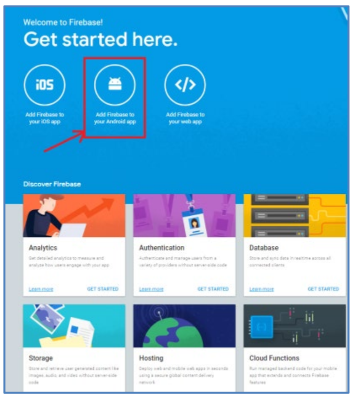
Mobile applications need a **back-end server** in order to perform tasks such as **authenticating users** and **synchronizing user data across multiple devices**. Creating such a server, however, requires a skill set that most independent app developers lack. Fortunately, there are **several back-end** as a **service platforms**, often called **BaaS**, you can use today. Google's Firebase is one such platform.

Firebase offers **essential services** such as **analytics**, **crash reporting**, **user authentication**, and **cloud messaging** at no cost. Its freemium services include a **real-time NoSQL database**, **file hosting**, and **static website hosting**.
In this tutorial, you learn how to add **user authentication**, **analytics**, and **remote data storage** to your Android app using **Firebase**.

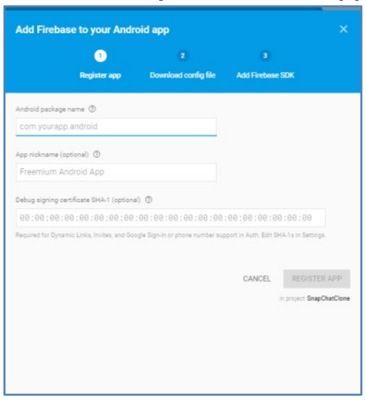
Firebase Analytics is one of the most popular mobile app analytics solutions available today. By using it, you can get a precise understanding of who your users are and how they are using your app.

Step 1: Add Firebase to your Android app

Open the **firebase console at website**: https://console.firebase.google.com/
Since we want to add Firebase to Android app, so click "**Add FireBase to your Android app**" option:


Welcome to Firebase!
Get started here.
iOS
Add Firebase to your iOS app
Add Firebase to your Android app
Add Firebase to your web app
Discover Firebase
Analytics
Get detailed analytics to measure and analyze how users engage with your app
Learn more
GET STARTED
Authentication
Authenticate and manage users from a variety of providers without server-side code
Learn more
GET STARTED
Database
Store and sync data in realtime across all connected clients
Learn more
GET STARTED
Storage
Store and retrieve user generated content like images, audio, and video without server-side code
Hosting
Deploy web and mobile web apps in seconds using a secure global content delivery network
Cloud Functions
Run managed backend code for your mobile app that extends and connects Firebase features

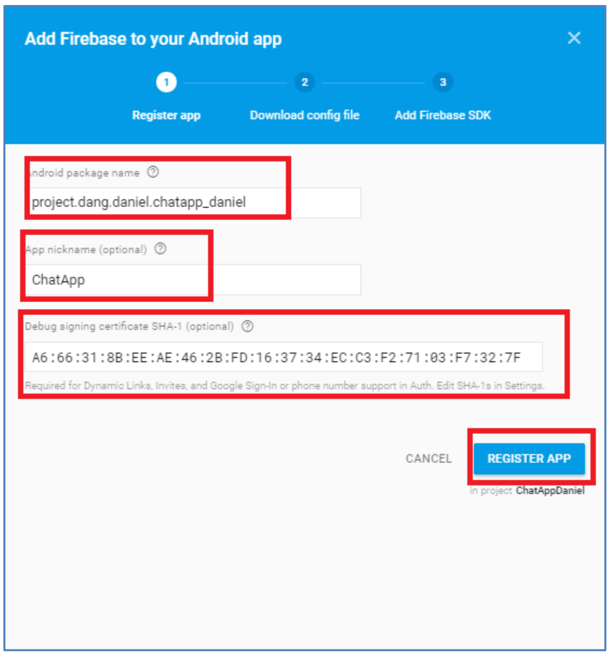
➔



a. Register app:

Change this info to your own Android project

- Enter the **Android package name**: ***project.dang.daniel.chatapp_daniel***
- App nickname: ***ChatApp***
- Debug signing certificate SHA-1: ***A6:66:31:8B:EE:AE:46:2B:FD:16:37:34:EC:C3:F2:71:03:F7:32:7F***
- Click **"Register app"** button



Android package name
project.dang.daniel.chatapp_daniel

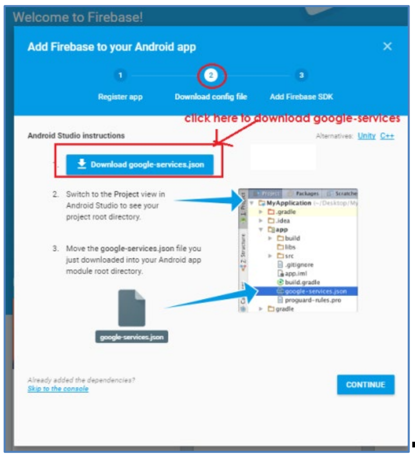
App nickname (optional)
ChatApp

Debug signing certificate SHA-1 (optional)
A6:66:31:8B:EE:AE:46:2B:FD:16:37:34:EC:C3:F2:71:03:F7:32:7F

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

in project ChatAppDaniel

b. Download config file (*google-services.json*) to your computer by clicking "download google-services.json" button:



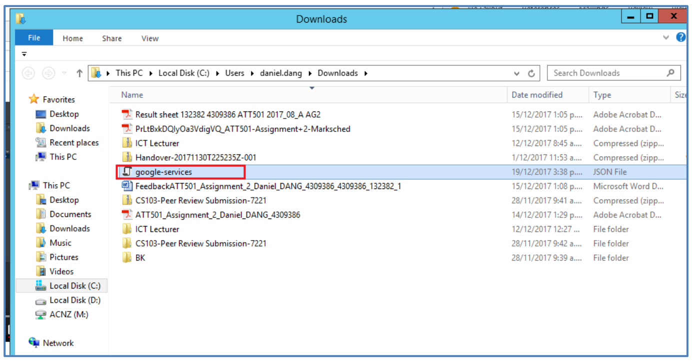
Android Studio instructions

Alternatives: Unity C++

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

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

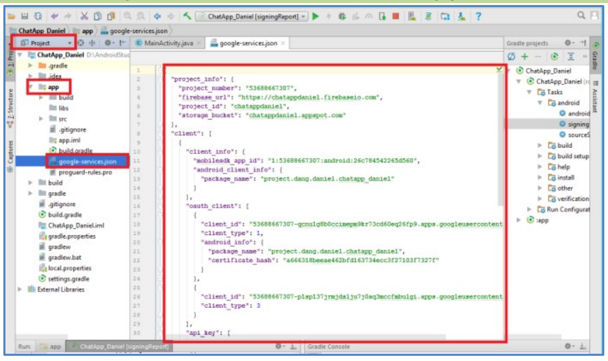
Already added the dependencies?
Skip to the console



Name	Date modified	Type
Result sheet 132382 4309386 ATT501 2017_08_A AG2	15/12/2017 1:05 p....	Adobe Acrobat D...
PrLtBxkDQlyOa3VdigVQ_ATT501-Assignment+2-Marksched	15/12/2017 1:05 p....	Adobe Acrobat D...
ICT Lecturer	12/12/2017 8:45 a....	Compressed (zipp...
Handover-20171130T225235Z-001	1/12/2017 11:53 a....	Compressed (zipp...
google-services	19/12/2017 3:38 p....	JSON File
FeedbackATT501_Assignment_2_Daniel_DANG_4309386_4309386_132382_1	15/12/2017 1:08 p....	Microsoft Word D...
CS103-Peer Review Submission-7221	28/11/2017 9:41 a....	Compressed (zipp...
ATT501_Assignment_2_Daniel_DANG_4309386	14/12/2017 1:29 p....	Adobe Acrobat D...
ICT Lecturer	12/12/2017 12:27 ...	File folder
CS103-Peer Review Submission-7221	28/11/2017 9:42 a....	File folder
BK	28/11/2017 9:39 a....	File folder

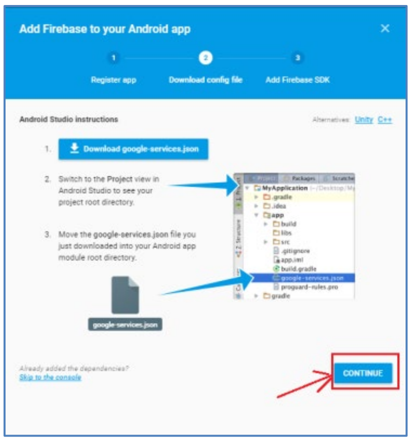
c. Copy and paste the config file (*google-services.json*) into "app" folder in Android Studio IDE:

- Go back to **Android Studio IDE**;
- Switch to the "**Project**" view;
- Copy and paste the *google-services.json* file into the folder "**app**";



```
{
  "project_info": {
    "project_number": "53688667307",
    "firebase_url": "https://chatappdaniel.firebaseio.com",
    "project_id": "chatappdaniel",
    "storage_bucket": "chatappdaniel.appspot.com"
  },
  "client": [
    {
      "client_info": {
        "mobilesdk_app_id": "1:53688667307:android:26c784542265d560",
        "android_client_info": {
          "package_name": "project.dang.daniel.chatapp_daniel"
        }
      },
      "oauth_client": [
        {
          "client_id": "53688667307-qcnulg8b0ccimepm9kr73cd60eq26fp9.apps.googleusercontent
          "client_type": 1,
          "android_info": {
            "package_name": "project.dang.daniel.chatapp_daniel",
            "certificate_hash": "a666318beeae462bfd163734ecc3f27103f7327f"
          }
        },
        {
          "client_id": "53688667307-plsp137jrmjdslju7j0aq3mccfmbulgi.apps.googleusercontent
          "client_type": 3
        }
      ],
      "api_key": [
```

d. Go back to Firebase console (website) and click "Continue" button to go on:


Add Firebase to your Android app

1 Register app — 2 Download config file — 3 Add Firebase SDK

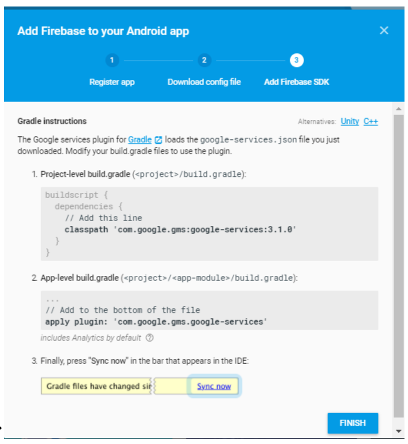
Android Studio instructions — Alternatives: Unity C++

1. Download google-services.json
2. Switch to the Project view in Android Studio to see your project root directory.
3. Move the google-services.json file you just downloaded into your Android app module root directory.

Already added the dependencies?
Skip to the console

CONTINUE

→


Add Firebase to your Android app

1 Register app — 2 Download config file — 3 Add Firebase SDK

Gradle instructions — Alternatives: Unity C++

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

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

```
buildscript {
  dependencies {
    // Add this line
    classpath 'com.google.gms:google-services:3.1.0'
  }
}
```

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

```
...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'
```

includes Analytics by default

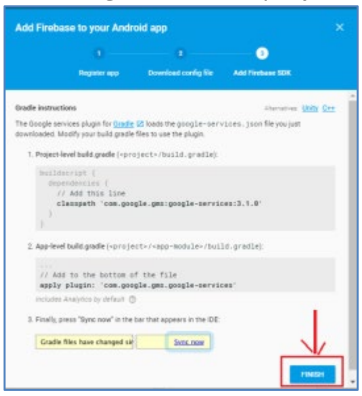
3. Finally, press "Sync now" in the bar that appears in the IDE:

Gradle files have changed si… Sync now

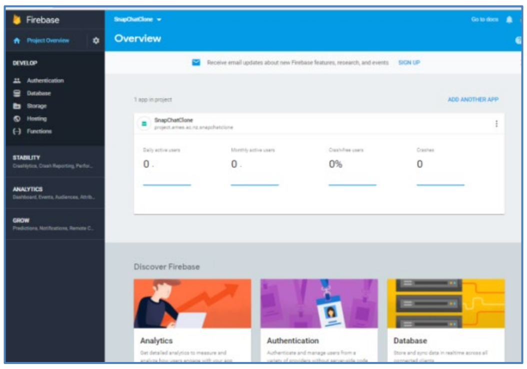
FINISH

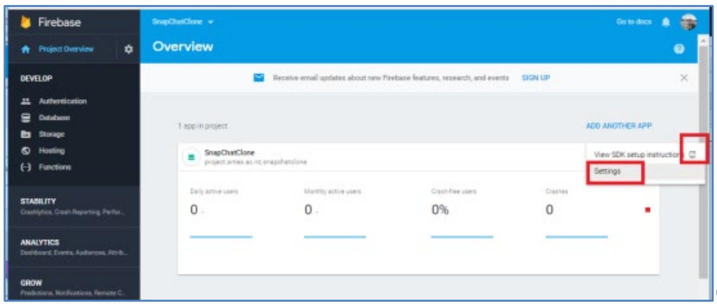
e. Click "FINISH" button:

- Click "**Project Setting**" to see the project information:

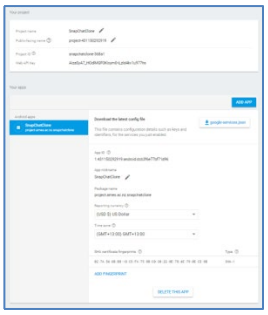


→





→

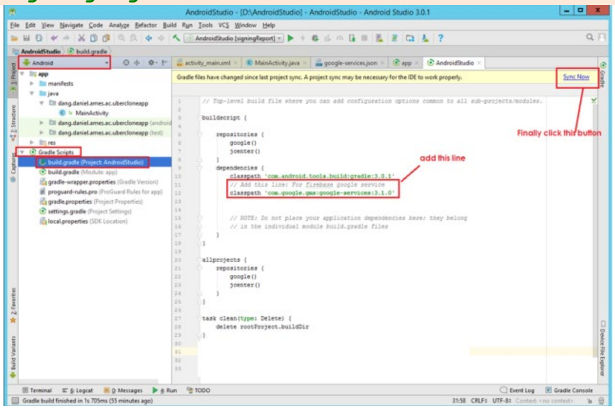


f. Add Firebase to your Android app:

Open Android Studio IDE, modify the build.gradle and sync it:

- Open Android Studio IDE, switch to **"Android" view**;
- Open the **build.gradle file (Project: Android Studio)** and add the below line:

```
// Add this line: For firebase google service
classpath 'com.google.gms:google-services:3.1.0'
```


AndroidStudio - [D:\AndroidStudio] - AndroidStudio - Android Studio 3.0.1

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Android
app
manifests
java
dang.daniel.ames.ac.ubercloneapp
MainActivity
res
Gradle Scripts
build.gradle (Project: AndroidStudio)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MainActivity.java
google-services.json
app
AndroidStudio
Sync Now
Finally click this button
add this line
Terminal
Logcat
Messages
Run
TODO
Event Log
Gradle Console

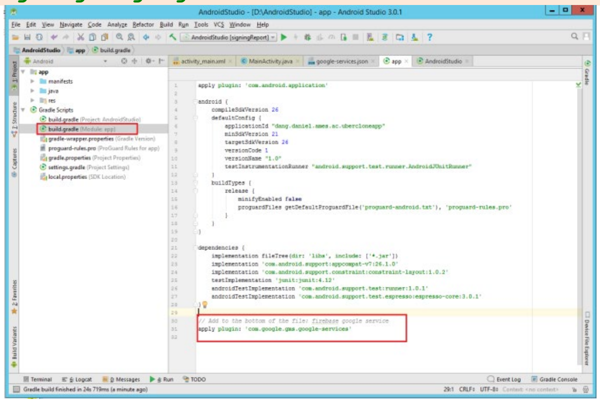
Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {

    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.1'
        // Add this line: For firebase google service
        classpath 'com.google.gms:google-services:3.1.0'


        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Gradle build finished in 1s 705ms (55 minutes ago)

- Open the **build.gradle file (Module: app)** and add the line at the bottom:

```
// Add to the bottom of the file: firebase google service
apply plugin: 'com.google.gms.google-services'
```



AndroidStudio - [D:\AndroidStudio] - app - Android Studio 3.0.1

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "dang.daniel.ames.ac.ubercloneapp"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}

// Add to the bottom of the file: firebase google service
apply plugin: 'com.google.gms.google-services'
```

app
manifests
java
res
Gradle Scripts
build.gradle (Project: AndroidStudio)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version)
proguard-rules.pro (ProGuard Rules for app)
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
activity_main.xml
MainActivity.java
google-services.json
app
AndroidStudio
Terminal
Logcat
Messages
Run
TODO
Event Log
Gradle Console
Gradle build finished in 24s 719ms (a minute ago)

- Click "**SYNC NOW**" button to synchronise the project.

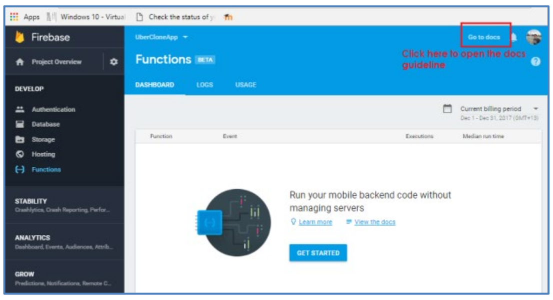
<u>**Step 2**</u>: Integrate **Firebase libraries** to your app:

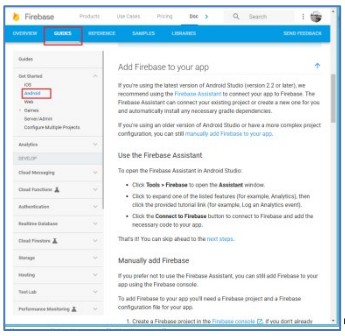
Open the **Firebase console at website**: https://console.firebase.google.com/

+ Click the "**Go to docs**" button on the top right corner of the firebase web;

+ Select "**Guides**" and "**Android**" to see the guidelines how to add Firebase libraries to your app → Scroll down to see the "**available libraries**" for various **Firebase features**:

- *com.google.firebase:firebase-core:11.8.0*
- *com.google.firebase:firebase-database:11.8.0*
- *com.google.firebase:firebase-auth:11.8.0*





Add Firebase to your app

If you're using the latest version of Android Studio (version 2.2 or later), we recommend using the Firebase Assistant to connect your app to Firebase. The Firebase Assistant can connect your existing project or create a new one for you and automatically install any necessary gradle dependencies.

If you're using an older version of Android Studio or have a more complex project configuration, you can still manually add Firebase to your app.

Use the Firebase Assistant

To open the Firebase Assistant in Android Studio:

- Click **Tools > Firebase** to open the **Assistant** window.
- Click to expand one of the listed features (for example, Analytics), then click the provided tutorial link (for example, Log an Analytics event).
- Click the **Connect to Firebase** button to connect to Firebase and add the necessary code to your app.

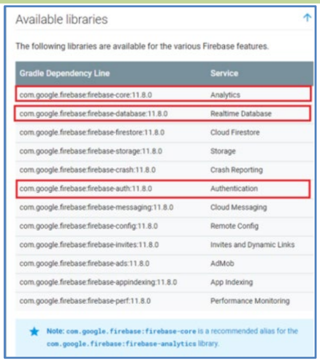
That's it! You can skip ahead to the next steps.

Manually add Firebase

If you prefer not to use the Firebase Assistant, you can still add Firebase to your app using the Firebase console.

To add Firebase to your app you'll need a Firebase project and a Firebase configuration file for your app.

1. Create a Firebase project in the Firebase console. If you don't already

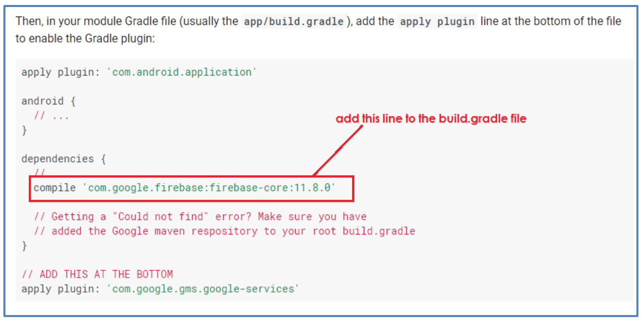


Available libraries

The following libraries are available for the various Firebase features.

Gradle Dependency Line	Service
com.google.firebase:firebase-core:11.8.0	Analytics
com.google.firebase:firebase-database:11.8.0	Realtime Database
com.google.firebase:firebase-firestore:11.8.0	Cloud Firestore
com.google.firebase:firebase-storage:11.8.0	Storage
com.google.firebase:firebase-crash:11.8.0	Crash Reporting
com.google.firebase:firebase-auth:11.8.0	Authentication
com.google.firebase:firebase-messaging:11.8.0	Cloud Messaging
com.google.firebase:firebase-config:11.8.0	Remote Config
com.google.firebase:firebase-invites:11.8.0	Invites and Dynamic Links
com.google.firebase:firebase-ads:11.8.0	AdMob
com.google.firebase:firebase-appindexing:11.8.0	App Indexing
com.google.firebase:firebase-perf:11.8.0	Performance Monitoring

Note: `com.google.firebase:firebase-core` is a recommended alias for the `com.google.firebase:firebase-analytics` library.



Then, in your module Gradle file (usually the `app/build.gradle`), add the `apply plugin` line at the bottom of the file to enable the Gradle plugin:

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  compile 'com.google.firebase:firebase-core:11.8.0'

  // Getting a "Could not find" error? Make sure you have
  // added the Google maven respository to your root build.gradle
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

+ Open **Android Studio IDE**, open the **build.gradle** file and add the lines to **integrate 3 below libraries**:

```
//Integrate 3 Firebase libraries to the app
implementation 'com.google.firebase:firebase-core:11.8.0'//For "Analytics" service
implementation 'com.google.firebase:firebase-database:11.8.0'//For "Real-time database" service
implementation 'com.google.firebase:firebase-auth:11.8.0'//For "Authentication " service
```

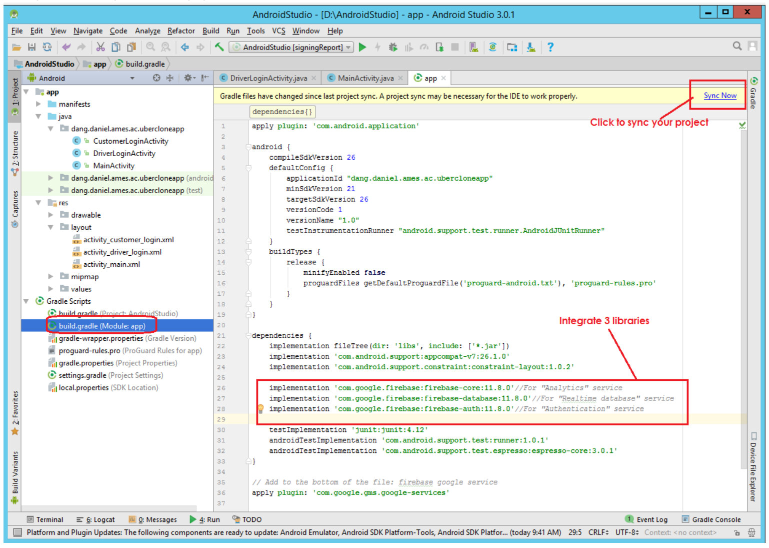
+ We'll be using **one more library** in this project: **Android design support library**:

- Read more about "**Android design support library**" at https://developer.android.com/topic/libraries/support-library/revisions.html

Therefore, we add the following implementation dependencies to it.

```
//Add two more libraries being used for our project
implementation 'com.android.support:design:26.1.0'
```

Click “**Sync Now**” button to update your app;

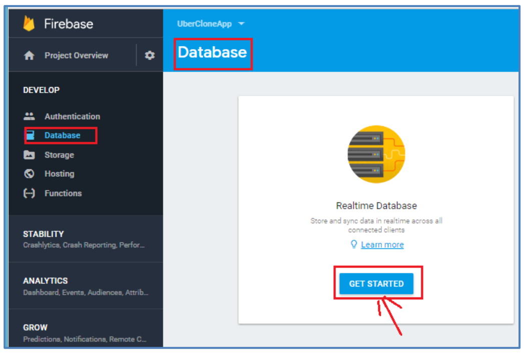

AndroidStudio - [D:\AndroidStudio] - app - Android Studio 3.0.1
Sync Now
Click to sync your project
Integrate 3 libraries
build.gradle (Module: app)

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

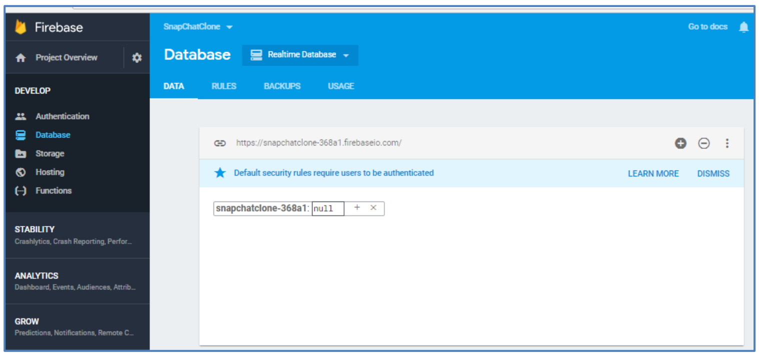
```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "dang.daniel.ames.ac.ubercloneapp"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'

    implementation 'com.google.firebase:firebase-core:11.8.0'//For "Analytics" service
    implementation 'com.google.firebase:firebase-database:11.8.0'//For "Realtime database" service
    implementation 'com.google.firebase:firebase-auth:11.8.0'//For "Authentication" service

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.1'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
}

// Add to the bottom of the file: firebase google service
apply plugin: 'com.google.gms.google-services'
```

Step 3: Create Real-time database for the project

- Open **Firebase console**;
- **Go to “Database” tab and click “Get started” button to create a “null” database;**


Firebase
UberCloneApp
Project Overview
Database
DEVELOP
Authentication
Database
Storage
Hosting
Functions
STABILITY
Crashlytics, Crash Reporting, Perfor...
ANALYTICS
Dashboard, Events, Audiences, Attrib...
GROW
Predictions, Notifications, Remote C...
Realtime Database
Store and sync data in realtime across all connected clients
Learn more
GET STARTED

➔



https://snapchatclone-368a1.firebaseio.com/

Default security rules require users to be authenticated

snapchatclone-368a1: null

PART 3: Use Firebase Auth to handle user login and registration processes

Source: http://www.andrious.com/tutorials/android-chat-application-using-firebase/

User authentication is an important requirement for most Android apps today. By being able to securely authenticate your users—and thus uniquely identify them—you can offer a customized experience to them based on their interests and preferences. You can also make sure that they have no problems accessing their private data while using your app from multiple devices.
Firebase Auth allows you to authenticate your users using a variety of **sign-in mechanisms**. In this tutorial, I'll show you how to allow your users to sign in to your app **using their email addresses and passwords**.

Step 1: Configure Your Android Project

+ To be able to use **Firebase Auth** in your Android Studio project, you must add the following **compile dependency** to the **app module's** build.gradle file:

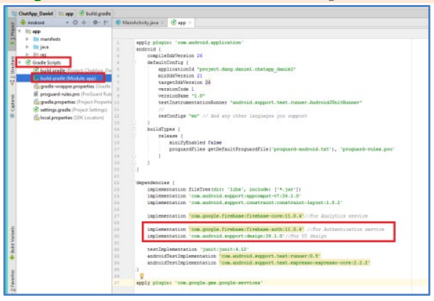
```
implementation 'com.google.firebase:firebase-auth:11.0.4' //For Authentication service
```

+ Since we need to connect to the Network add the Internet permission in **AndroidManifest.xml file** :

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

+ Add one more dependency "*com.android.support:design*" to **gradle file** as well; since we will use few widgets in this library for UI design later on:

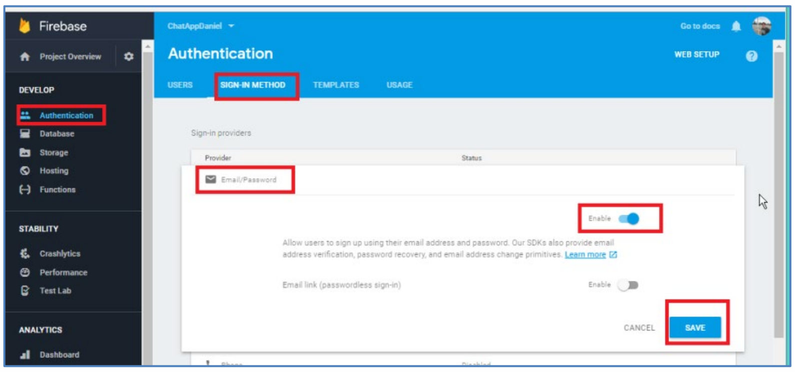
```
implementation 'com.android.support:design:26.1.0'//For UI design
```



```
apply plugin: 'com.android.application'
android {
    compileSdkVersion 26
    defaultConfig {
        applicationId "project.dang.daniel.chatapp_daniel"
        minSdkVersion 21
        targetSdkVersion 26
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        //
        resConfigs "en" // And any other languages you support
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'

    implementation 'com.google.firebase:firebase-core:11.0.4'//For Analytics service

    implementation 'com.google.firebase:firebase-auth:11.0.4' //For Authentication service
    implementation 'com.android.support:design:26.1.0'//For UI design

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:0.5'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:2.2.2'
}

apply plugin: 'com.google.gms.google-services'
```

Step 2: Enable Password-Based Authentication on Firebase console

By default, Firebase doesn't allow user authentication. Therefore, you must manually **enable password-based user authentication** in the **Firebase console**. To do so, navigate to the **Authentication** section, and press the **Set up sign-in method** button. You'll now see a list of all the sign-in providers available, choose **Email/Password**, enable it, and press **Save**:



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

Note that it is your responsibility to **validate email addresses** and **make sure that the users use strong passwords**.

We will make separate layouts for **login process** and **registration process**. Furthermore, the app has a separate "**chat layout**" as well. So first of all, we will add **3 more Activities** to the project:

- The first one is LoginActivity,
- The second one is RegistrationActivity,
- The third one is ChatActivity.

Step 3: Add 3 new Activities to the project

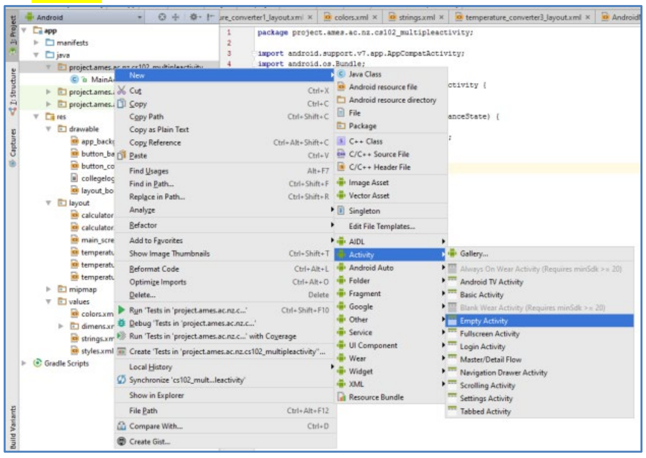
+ Add a new Activity, called "LoginActivity" and its associated "login_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **LoginActivity***

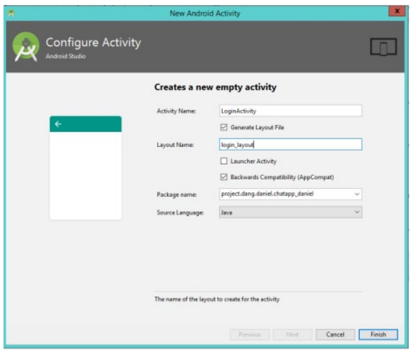
*_Enter the "Layout Name": **login_layout***

*_Click "**Finish**"*



```
package project.ames.ac.nz.cs102_multipleactivity;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

➔


New Android Activity

Configure Activity

Android Studio

Creates a new empty activity

Activity Name: LoginActivity

[✓] Generate Layout File

Layout Name: login_layout

[ ] Launcher Activity

[✓] Backwards Compatibility (AppCompat)

Package name: project.dang.daniel.chatapp_daniel

Source Language: Java

The name of the layout to create for the activity

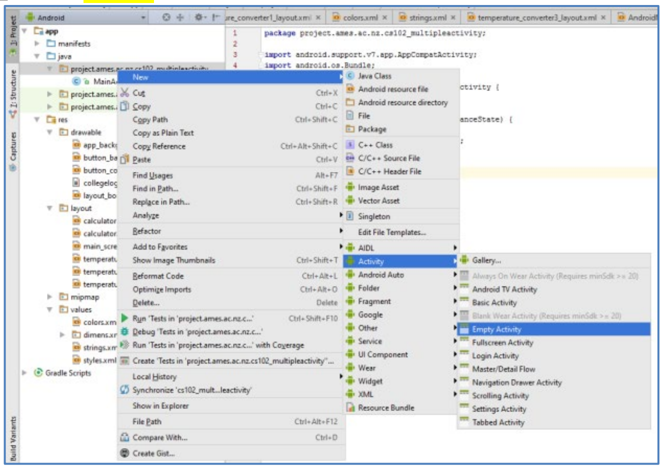
+ Add another new Activity, called "RegistrationActivity" and its associated "registration_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

*_Enter the "Activity Name": **RegistrationActivity***

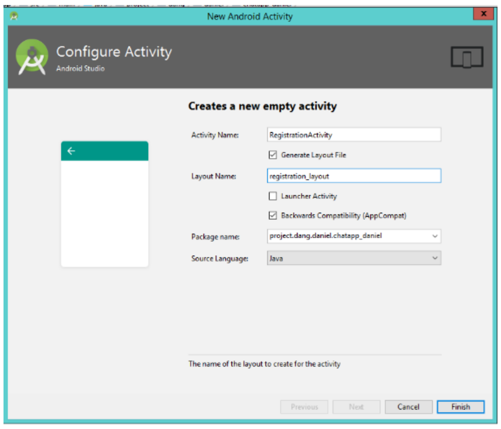
*_Enter the "Layout Name": **registration_layout***

*_Click "**Finish**"*



```
package project.ames.ac.nz.cs102_multipleactivity;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```

➔


New Android Activity

Configure Activity

Android Studio

Creates a new empty activity

Activity Name: RegistrationActivity

[✓] Generate Layout File

Layout Name: registration_layout

[ ] Launcher Activity

[✓] Backwards Compatibility (AppCompat)

Package name: project.dang.daniel.chatapp_daniel

Source Language: Java

The name of the layout to create for the activity

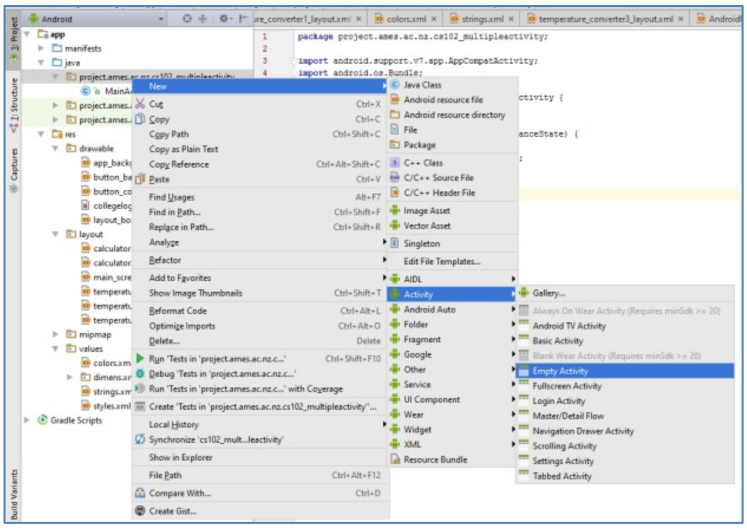
+ Add one more new Activity, called "ChatActivity" and its associated "chat_layout"

*_Right click on the folder containing **MainActivity** → "**New**" → "**Activity**" → "**Empty Activity**" → a "Configure Acitivy" window is appeared:*

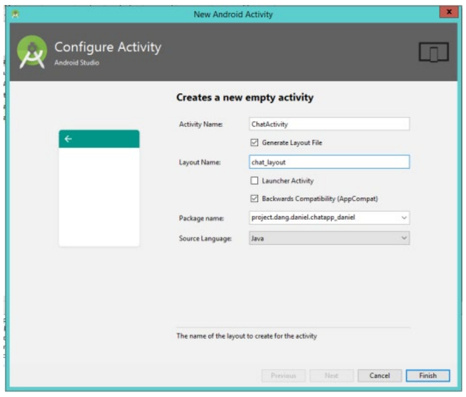
*_Enter the "Activity Name": **ChatActivity***

*_Enter the "Layout Name": **chat_layout***

*_Click "**Finish**"*



```
package project.ames.ac.nz.cs102_multipleactivity;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
```



New Android Activity

Configure Activity

Creates a new empty activity

Activity Name: ChatActivity

[✓] Generate Layout File

Layout Name: chat_layout

[ ] Launcher Activity

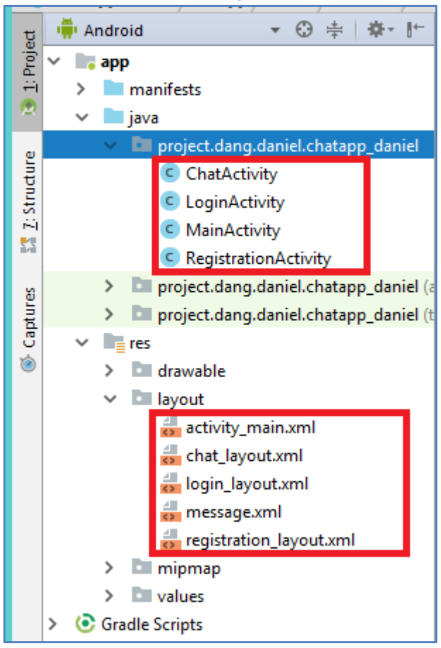
[✓] Backwards Compatibility (AppCompat)

Package name: project.dang.daniel.chatapp_daniel

Source Language: Java

The name of the layout to create for the activity

+ Now, you'll see that your project has 4 Activities and 4 layouts:



Step 4: Edit MainActivity.java and design activity_main.xml layout to become a Splash Screen

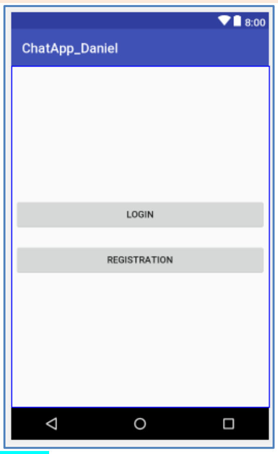
When users open the app, there are two scenarios:

- First scenario: If users haven't logged in, the app will display 2 buttons: "Login" button and "Register" button so that users can either do a login or do a registration.
- Second scenario: If users have been logged in, the app will skip the Login/Registration process and open up the **ChatActivity** (chat layout).

In order to do it, we need to access to **Authentication (Firebase database)** and check the user account. Furthermore, we create a **variable "started"** to track down if users already logged in or not.

+ Open activity_main.xml layout and edit it as below:

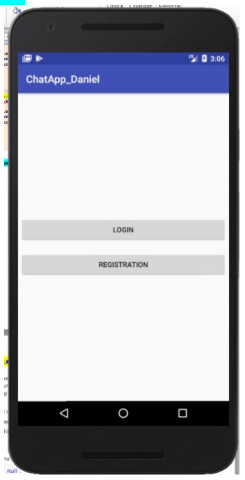
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="5dp"
    tools:context=".MainActivity">

    <!--Login button-->
    <Button
        android:id="@+id/login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Login" />

    <!--Registration button-->
    <Button
        android:id="@+id/registration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="Registration" />
</LinearLayout>
```



+ Open MainActivity.java file and edit it as below:

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.google.firebase.auth.FirebaseAuth;

public class MainActivity extends AppCompatActivity {
    ////////////////////////////////////////////////////////////////////////////////////
    //1: Declare variables
    public static Boolean started = false;
    private FirebaseAuth mAuth;
    private Button mLogin, mRegistration;

    ////////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ///////////////////////////////////////////////////////////////////////////////
        //2: Get userID in Firebase Authentication database
        mAuth = FirebaseAuth.getInstance();

        ///////////////////////////////////////////////////////////////////////////////
        //2: Find reference and do casting for "mLogin"  & "mRegistration" buttons
        mLogin = (Button) findViewById(R.id.login);
        mRegistration = (Button) findViewById(R.id.registration);
        //At the beginning, make these two button invisible
        mLogin.setVisibility(View.INVISIBLE);
        mRegistration.setVisibility(View.INVISIBLE);

        ///////////////////////////////////////////////////////////////////////////////
        //3: Check whether the userID exists in the Authentication database. If yes, go straight to
        //"ChatActivity". If not, display "Login" and "Registration" buttons for users options
        if (mAuth.getCurrentUser() != null) {
            //User already logged in yet, go straight to chat layout
            Intent intent = new Intent(getApplication(), ChatActivity.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
            finish();
            return;

        } else {
            //User hasn't logged in yet, make two buttons visible
            mLogin.setVisibility(View.VISIBLE);
            mRegistration.setVisibility(View.VISIBLE);
        }

        ///////////////////////////////////////////////////////////////////////////////
        //4: Set listener for 'Login" button, open LoginActivity when clicking it
        mLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplication(), LoginActivity.class);
                startActivity(intent);
                return;
            }
        });

        ///////////////////////////////////////////////////////////////////////////////
        //4: Set listener for "Registration" button, open RegistrationActivity when clicking it
        mRegistration.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(getApplication(), RegistrationActivity.class);
                startActivity(intent);
                return;
            }
        });
    }
}
```

+ Build and run you app on AVD to see the result:



Step 5: Design registration_layout.xml and edit RegistrationActivity.java to handle Registration process

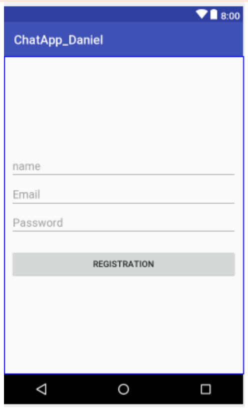
+ Support User Registration

With password-based authentication, new users must register themselves by **providing a unique email address and a password**. To add this functionality to your app, you can use the **createUserWithEmailAndPassword()** **method** of the **FirebaseAuth** **class**. As its name suggests, the method expects an email address and a password as its arguments.

To be able to determine the result of the **createUserWithEmailAndPassword()** **method**, you must add an **OnCompleteListener** to it using the **addOnCompleteListener()** **method**.

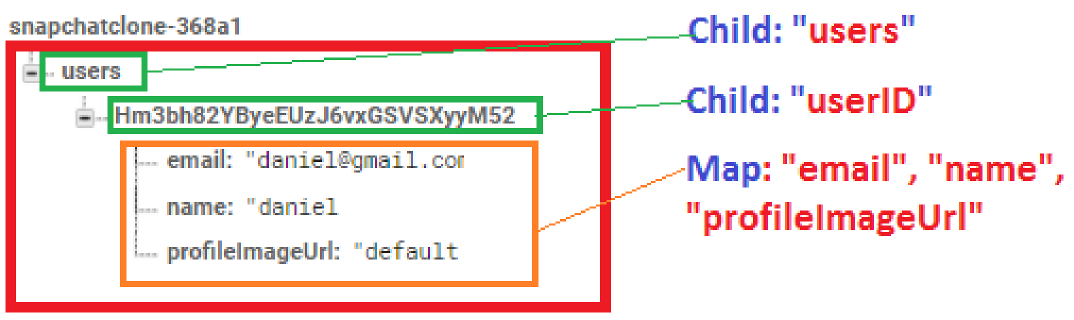
+ Open registration_layout.xml file and edit it as below:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".RegistrationActivity">

    <!--EditText: enter user name-->
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="name" />

    <!--EditText: enter email address-->
    <EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />

    <!--EditText: enter password-->
    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />

    <!--Button Registration to create a new account-->
    <Button
        android:id="@+id/registration"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="registration" />
</LinearLayout>
```



+ Open RegistrationActivity.java file and edit it as below:

```java
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;
import com.google.firebase.auth.UserProfileChangeRequest;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;
import java.util.Map;

public class RegistrationActivity extends AppCompatActivity {
    ///////////////////////////////////////////////////////////////////////////////////
    //1: Declare variables
    private Button mRegistration;
    private EditText mEmail, mPassword, mName;
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener firebaseAuthStateListener;//Firebase Authentication listener

    ///////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.registration_layout);

        ///////////////////////////////////////////////////////////////////////////
        //2: Listen the Authentication database. If the database exists, go back to MainActivity
        firebaseAuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
                if (user != null) {
                    Intent intent = new Intent(getApplication(), MainActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(intent);
                    finish();
                    return;
                }
            }
        };

        //////////////////////////////////////////////////////////////////////////
        //3: Get the entered log-in information ==> create a user in Authentication with those information
        mAuth = FirebaseAuth.getInstance();
        //
        mRegistration = findViewById(R.id.registration);
        mName = findViewById(R.id.name);
        mEmail = findViewById(R.id.email);
        mPassword = findViewById(R.id.password);
        //
        mRegistration.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Get the entered user information
                final String name = mName.getText().toString();
                final String email = mEmail.getText().toString();
                final String password = mPassword.getText().toString();
                //Check if name, email and password is empty or not. If being empty, do nothing
                if (name.isEmpty() || email.isEmpty() || password.isEmpty()) {
                    //ERROR, pop up a message
                    Toast.makeText(getApplication(), "Please enter all your name, email and password",
                            Toast.LENGTH_SHORT).show();
                    return;
                }
                //
                mAuth.createUserWithEmailAndPassword(email, password).addOnCompleteListener(
                        RegistrationActivity.this, new OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult> task) {
                                if (!task.isSuccessful()) {
                                    //ERROR, pop up a message
                                    Toast.makeText(getApplication(), "ERROR. EMAIL IS INVALID",
                                            Toast.LENGTH_SHORT).show();
                                } else {
                                    //SUCCESSFULLY create a new user with those information
                                    //1: setDisplayName of user while creating user in Authentication
                                    FirebaseUser user = mAuth.getCurrentUser();
                                    UserProfileChangeRequest profileUpdates = new
UserProfileChangeRequest.Builder().setDisplayName(name).build();
                                    user.updateProfile(profileUpdates);
                                    //2: Add a "user" child into the Real-time Database
                                    String userId = mAuth.getCurrentUser().getUid();
                                    DatabaseReference currentUserDb = FirebaseDatabase
                                            .getInstance()
                                            .getReference()
                                            .child("users")
                                            .child(userId);
                                    //3: Use "Map" type to add user profile
                                    Map userInfo = new HashMap();
                                    userInfo.put("email", email);
                                    userInfo.put("name", name);
                                    userInfo.put("profileImageUrl", "default");
                                    //Add new data to Real-time Database
                                    currentUserDb.updateChildren(userInfo);
                                    //4: Pop up a Toast message
                                    Toast.makeText(getApplication(), "REGISTERED SUCCESSFULLY",
                                            Toast.LENGTH_SHORT).show();
                                }
                            }
                        });
                }
            });

    }

    ///////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onStart() {
        super.onStart();
        //Add "Firebase Authentication listener" whenever start or start again the Activity
        mAuth.addAuthStateListener(firebaseAuthStateListener);
    }

    ///////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onResume() {
        super.onResume();
        //Add "Firebase Authentication listener" whenever start or start again the Activity
        mAuth.addAuthStateListener(firebaseAuthStateListener);
    }

    ///////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onStop() {
        super.onStop();
        //When activity is stopped, remove "Firebase Authentication listener"
        mAuth.removeAuthStateListener(firebaseAuthStateListener);
    }
}
```


snapchatclone-368a1
users
Hm3bh82YByeEUzJ6vxGSVSXyyM52
email: "daniel@gmail.com
name: "daniel
profileImageUrl: "default
Child: "users"
Child: "userID"
Map: "email", "name", "profileImageUrl"

Step 6: Design login_layout.xml and edit LoginActivity.java to handle Login process

+ Support User Sign-In

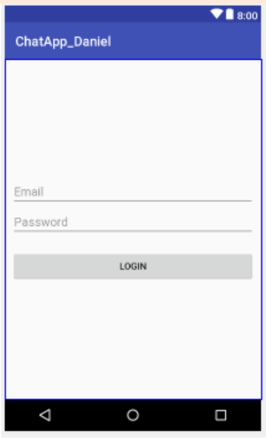
After a successful registration, the user is automatically signed in. The user will continue to stay signed in even if your app restarts. Nevertheless, your app must include code that **allows users to manually sign in** using the email address and password they specified during the registration process.

To sign a user in manually, you must use the `signInWithEmailAndPassword()` method of the `FirebaseAuth` class. The method expects an email address and a password as its only arguments, and returns a `Task<AuthResult>` object. By adding an `OnCompleteListener` to it, you can check if the sign-in was completed successfully.

In order to avoid signing in a user who's already signed in, you must call the `signInWithEmailAndPassword()` method only if the current `FirebaseUser` is `null`.

+ Open login_layout.xml file and edit it as below:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="10dp"
    tools:context=".LoginActivity">

    <!--EditText: enter email address-->
    <EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:inputType="textEmailAddress" />

    <!--EditText: enter password-->
    <EditText
        android:id="@+id/password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Password"
        android:inputType="textPassword" />

    <!--Button: login with above account-->
    <Button
        android:id="@+id/login"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:text="login" />
</LinearLayout>
```



+ Open LoginActivity.java file and edit it as below:

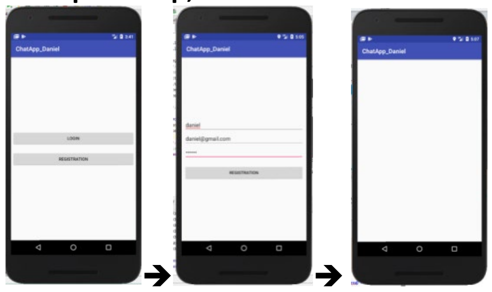
```java
import android.content.Intent;
import android.support.annotation.NonNull;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

public class LoginActivity extends AppCompatActivity {
    ///////////////////////////////////////////////////////////////////////////////////
    //1: Declare variables
    private Button mLogin;
    private EditText mEmail, mPassword;
    //
    private FirebaseAuth mAuth;
    private FirebaseAuth.AuthStateListener firebaseAuthStateListener;//Listener of Authentication database

    ///////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.login_layout);

        ///////////////////////////////////////////////////////////////////////////////
        //2: Listen the Authentication database. If the database exists, get the user (ID) account
        firebaseAuthStateListener = new FirebaseAuth.AuthStateListener() {
            @Override
            public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
                FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();//Get user ID
                //Check whether the user exists in the database. If yes, go to MainActivity
                if (user != null) {
                    Intent intent = new Intent(getApplication(), MainActivity.class);
                    intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                    startActivity(intent);
                    finish();
                    return;
                }
            }
        };

        ///////////////////////////////////////////////////////////////////////////////
        //3: Get the entered log-in information and sign-in Authentication with those information:
        mAuth = FirebaseAuth.getInstance();
        mLogin = findViewById(R.id.login);
        mEmail = findViewById(R.id.email);
        mPassword = findViewById(R.id.password);

        //Set listener
        mLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                //Get entered data on email and password fields
                final String email = mEmail.getText().toString();
                final String password = mPassword.getText().toString();
                //Check if email and password is empty or not. If being empty, do nothing
                if (email.isEmpty() || password.isEmpty()) {
                    //ERROR, pop up a message
                    Toast.makeText(getApplication(), "Please enter all your email and password",
                            Toast.LENGTH_LONG).show();
                    return;
                }
                //Process the sign-in
                mAuth.signInWithEmailAndPassword(email, password).addOnCompleteListener(
                        LoginActivity.this, new OnCompleteListener<AuthResult>() {
                            @Override
                            public void onComplete(@NonNull Task<AuthResult> task) {
                                if (!task.isSuccessful()) {
                                    //Login unsuccessfully
                                    Toast.makeText(LoginActivity.this,
                                            "SIGN IN ERROR. WRONG EMAIL OR PASSWORD",
                                            Toast.LENGTH_LONG).show();

                                }
                                }
                            });
                    }
                });

    }

    ////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onStart() {
        super.onStart();
        //Add "listener" whenever start or start again the Activity
        mAuth.addAuthStateListener(firebaseAuthStateListener);
    }

    ////////////////////////////////////////////////////////////////////////////////
    @Override
    protected void onStop() {
        super.onStop();
        //When activity is stopped, remove "listener"
        mAuth.removeAuthStateListener(firebaseAuthStateListener);
    }
}
```

Step 7: Build and run the app on AVD:

+ On the mobile app side:

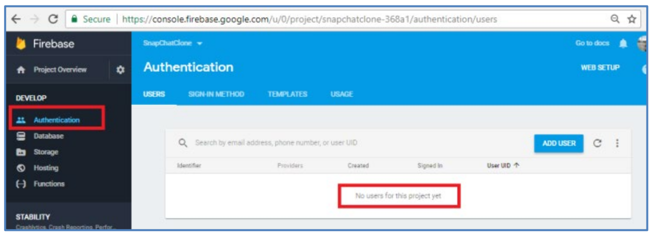
First of all, register a new user for testing by clicking "Registration" button:

- Name: Daniel
- Email: daniel@gmail.com
- Pass: 123456
- **Click "Registration" button to add a new account to database ➔ Now you will log in successfully and a black layout (chat_layout) will be opened up;**

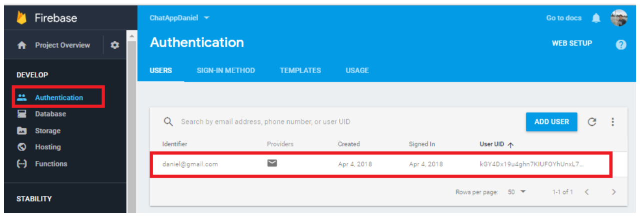


+ Open the Firebase console (website) to see the change:

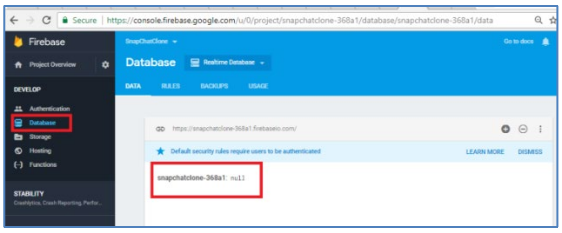
Go to Authentication & Database



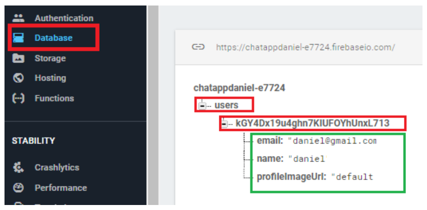
Before registration, no users in the Authentication



"Daniel" user has been added to Authentication


Firebase
Database
Realtime Database
DATA
RULES
BACKUPS
USAGE
Default security rules require users to be authenticated
snapchatclone-368a1: null

Before registration process, no "user" in database



After registration, a new user has been added to database

PART 4: Use Real-time Database to handle group chat

Using Firebase's Real-Time Database

One of the most powerful features of the Firebase platform is its **real-time database**, and it's named that for a good reason: all write operations performed on it are instantly available to all the clients that are observing it. With such a database, you can create applications that allow users to seamlessly switch from one device to another, and also instantly collaborate with other users.

Write Data

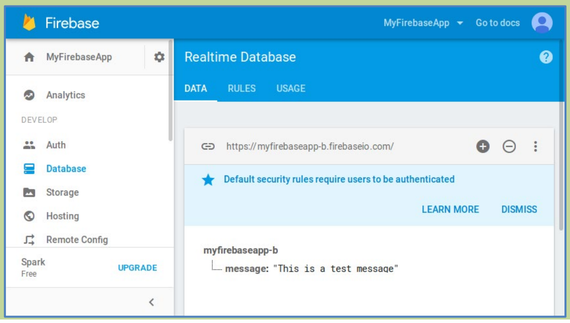
You can think of the real-time database as **a JSON object**. That means you can **add key-value pairs to it**, where the keys are always strings, and the values are either primitives, arrays, or other JSON objects.

Before you write to the real-time database, you must **get a reference** to it using the `getInstance()` method of the `FirebaseDatabase` class. You can then get a reference to any path inside the database using the `getReference()` method. The path can contain either a single key or several keys arranged hierarchically. Once you have a `DatabaseReference` to a path, you can **set its value** using the `setValue()` method.

The following code snippet shows you how to **add a simple key-value pair to the database**:

```
FirebaseDatabase db = FirebaseDatabase.getInstance();
DatabaseReference ref = db.getReference("message"); // Key
ref.setValue("This is a test message"); // Value
```

By going to the **Database** section of the Firebase console, you can view all the data that's inside the real-time database.



https://myfirebaseapp-b.firebaseio.com/

Default security rules require users to be authenticated

```
myfirebaseapp-b
└ message: "This is a test message"
```

Read Data

In order to **read a value from the real-time database**, you must attach an asynchronous observer to the associated key or path. More precisely, you must attach a `ValueEventListener` to a `DatabaseReference` object using the `addValueEventListener()` method.

The `onDataChange()` method of the `ValueEventListener` gives you access to a `DataSnapshot` object, whose `getValue()` method can be used to retrieve the **latest value of the key**.

For example, here's how you can retrieve the value we set in the previous step:

```
FirebaseDatabase db = FirebaseDatabase.getInstance();
DatabaseReference ref = db.getReference("message"); // Key

// Attach listener
ref.addValueEventListener(new ValueEventListener() {
   @Override
```

```
        public void onDataChange(DataSnapshot dataSnapshot) {
            // Retrieve latest value
            String message = dataSnapshot.getValue(String.class);
        }

        @Override
        public void onCancelled(DatabaseError databaseError) {
            // Error handling
        }
    });
```

Note that once a ValueEventListener is added, its onDataChange() method is triggered every time the value it is observing changes.

Step 1: Design chat_layout.xml

+ Copy the "send" icon image into drawable folder:

send_icon.png
(Source: https://cdnjs.loli.net/ajax/libs/material-design-icons/1.0.0/content/3x_ios/)

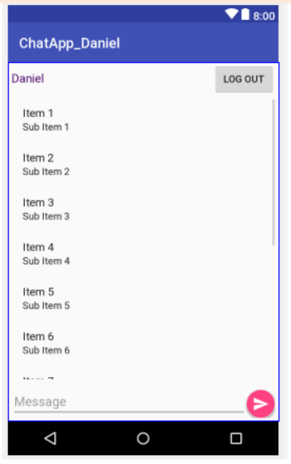
+ The **chat_layout.xml** file, which is already bound to ChatActivity, defines the contents of the chat messages sent by different users. In other words, it will represent the **chat room**. Like most other **group chat apps** available today, our app will have the **following UI elements**:

- **A list** that displays all the **group chat messages** in a chronological order
- **An input field** where the user can type in a new message
- **A button** the user can press to post the message

Therefore, **activity_main.xml** must have a ListView, an EditText, and a FloatingActionButton.

+ Open chat_layout.xml file and edit it as below:

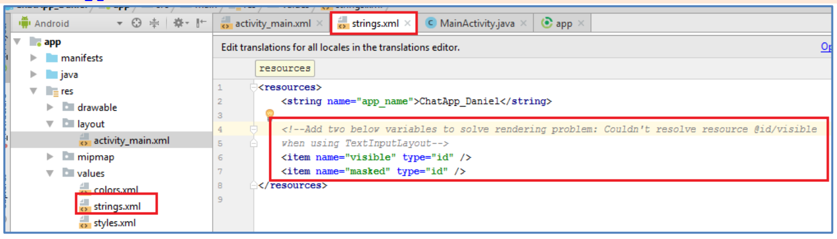
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".ChatActivity">

    <!--Top layout: contains user name & "logout" button-->
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:paddingLeft="5dp"
        android:paddingRight="5dp">

        <!--Add a TextView: display user name-->
        <TextView
            android:id="@+id/userName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_weight="1"
            android:gravity="left"
            android:text="Daniel"
            android:textColor="#4b086c"
            android:textSize="17sp" />

        <!--Add a Button: Logout to sign out-->
        <Button
            android:id="@+id/logoutBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="LOG OUT" />
    </LinearLayout>

    <!--This layout contain listview, editText and a button-->
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:padding="5dp">

        <!--Add a Button "send" at the bottom of screen: press to post the message-->
        <android.support.design.widget.FloatingActionButton
            android:id="@+id/sendBtn"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentEnd="true"
            android:clickable="true"
            android:src="@drawable/send_icon"
            android:tint="@android:color/white"
            app:fabSize="mini" />

        <!--Add a TextInputLayout containing an EditText-->
        <android.support.design.widget.TextInputLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentStart="true"
            android:layout_toLeftOf="@+id/sendBtn">

            <!--Add an EditText inside TextInputLayout for users entering their text message-->
            <EditText
                android:id="@+id/input"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:hint="Message" />
        </android.support.design.widget.TextInputLayout>

        <!--Add a listView: displays all the group chat messages in a chronological order-->
        <ListView
            android:id="@+id/list_of_messages"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_above="@+id/sendBtn"
            android:layout_alignParentStart="true"
            android:layout_alignParentTop="true"
            android:layout_marginBottom="15dp"
            android:divider="@android:color/transparent" />
    </RelativeLayout>
</LinearLayout>
```



+ Notes:

- We've placed the EditText widget inside a TextInputLayout widget. Doing so **adds a floating label** to the EditText, which is important if you want to adhere to the **guidelines of material design**.
- If you see the issue "Couldn't resolve resource @id/visible" on main screen, solve this problem as below: Open **strings.xml file** and add 2 variables:

```
<!--Add two below variables to solve rendering problem: Couldn't resolve resource @id/visible
when using TextInputLayout-->
<item name="visible" type="id" />
<item name="masked" type="id" />
```



```
<resources>
    <string name="app_name">ChatApp_Daniel</string>

    <!--Add two below variables to solve rendering problem: Couldn't resolve resource @id/visible
    when using TextInputLayout-->
    <item name="visible" type="id" />
    <item name="masked" type="id" />
</resources>
```

Step 2: Handle User Sign-Out

By default, Firebase uses Smart Lock for Passwords. Therefore, once the users sign in, they'll stay signed in even if the app is restarted. To allow the users to sign out, we'll now process the **sign-out option when users click "Log out" Button** on chat_layout.

+ Open ChatActivity.java file and add the below code to process the log-out process:

_Right after the class declaration, declare a variable:

```
/////////////////////////////////////////////////////////////////////////////////
//Sign-out - 1: Declare variables
private TextView username;
private Button logoutBtn;
```

_Inside onCreate() method, find the reference to above variable:

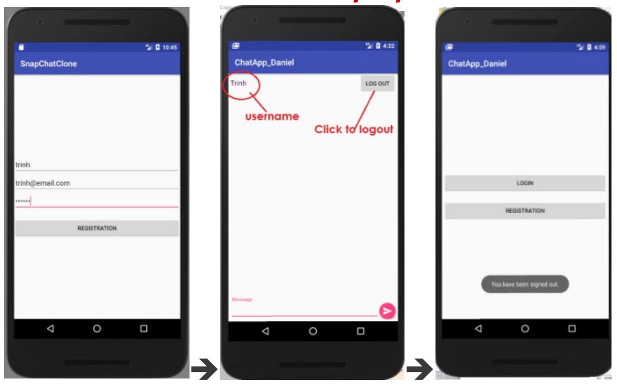
```
/////////////////////////////////////////////////////////////////////////////////
//Sign-out - 2: Find references for UI widgets
username = (TextView) findViewById(R.id.userName);
logoutBtn = (Button) findViewById(R.id.logoutBtn);
```

_Add click listener for **logoutBtn** to handle the log-out process by calling **signOut() method** of the **AuthUI class** to sign the user out:

```
/////////////////////////////////////////////////////////////////////////////////
//Sign-out - 3: Set click listener for the logoutBtn
logoutBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //1: Sign out the Firbase Authentication database
        FirebaseAuth.getInstance().signOut();
        //2: Go back to MainActivity
        Intent intent = new Intent(getApplicationContext(), MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        startActivity(intent);
        //3: Pop up a Toast message
        Toast.makeText(getApplicationContext(),
                "Bye Bye " + username.getText().toString(), Toast.LENGTH_LONG).show();
        //
        return;
    }
});
```

+ Build and run your app:

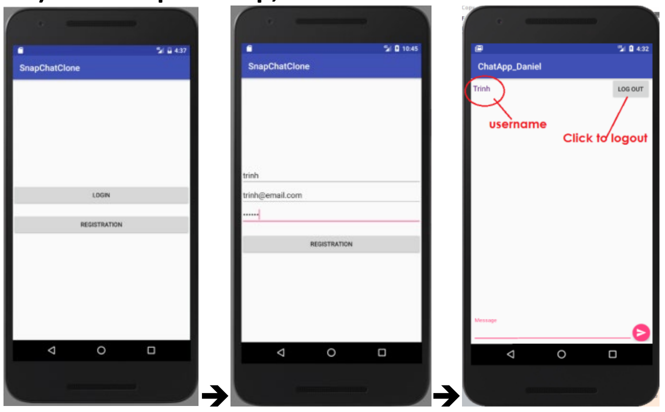
When you log in with your account, then click "Log out" button on ChatActivity, the app will log out and move to the MainActivity layout.



+ Register another user:

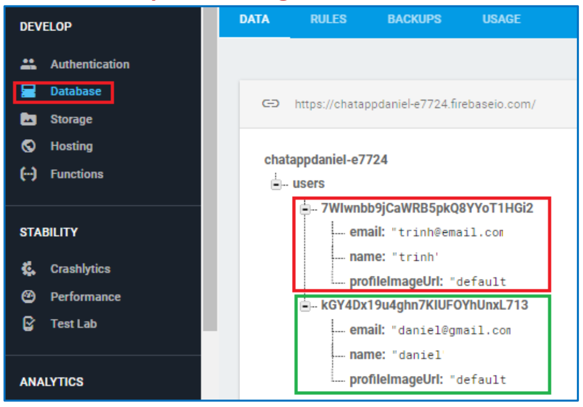
Now click "Registration" button to create a new account such as:

- Name: Trinh
- Email: trinh@email.com
- Pass: 123456
- **Click "Registration" button to add a new account to database ➔ Now you will log in successfully and a black layout (chat_layout) will be opened up;**



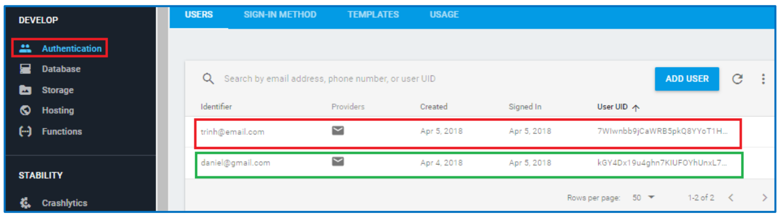
+ Open the Firebase console (website) to see the change:

Go to Authentication & Database sections: you will see that there are two accounts (one is Daniel and one is Trinh) in the Authentication. There are also two users in database containing 3 information: email, name and profileImageURL.



```
https://chatappdaniel-e7724.firebaseio.com/

chatappdaniel-e7724
 users
   7WIwnbb9jCaWRB5pkQ8YYoT1HGi2
     email: "trinh@email.com
     name: "trinh'
     profileImageUrl: "default
   kGY4Dx19u4ghn7KIUFOYhUnxL713
     email: "daniel@gmail.com
     name: "daniel
     profileImageUrl: "default
```

Database



Identifier	Providers	Created	Signed In	User UID
trinh@email.com	✉	Apr 5, 2018	Apr 5, 2018	7WIwnbb9jCaWRB5pkQ8YYoT1H...
daniel@gmail.com	✉	Apr 4, 2018	Apr 5, 2018	kGY4Dx19u4ghn7KIUFOYhUnxL7...

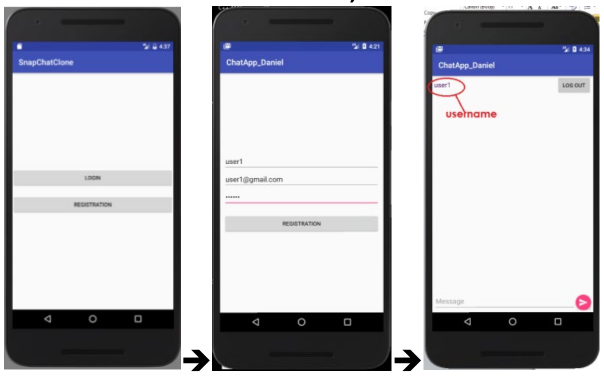
Rows per page: 50 1-2 of 2

Authentication

+ Now create a new account such as:

- Name: user1
- Email: user1@email.com
- Pass: 123456

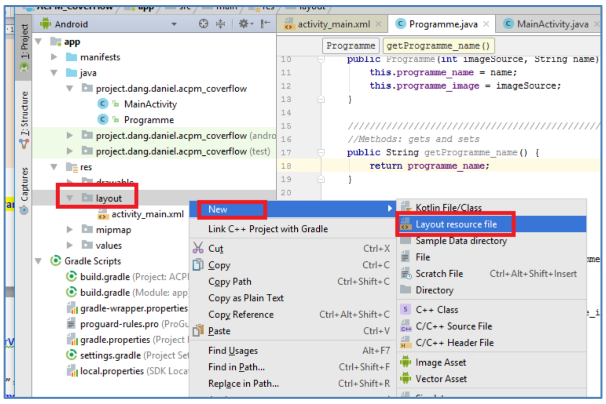
Click "Registration" button to add a new account to database;



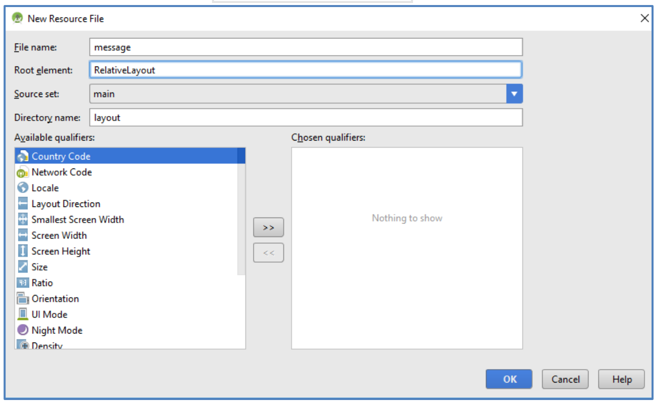
Step 3: Create layout for the chat messages

Now that the layout of the home screen is ready, we can move on to creating a **layout for the chat messages**, which will be items inside the **ListView** .

+ Create a new **layout XML file** called **message.xml,** whose **root element** is **RelativeLayout** :



```
public Programme(int imageSource, String name)
    this.programme_name = name;
    this.programme_image = imageSource;
}
//////////////////////////////////////////
//Methods: gets and sets
public String getProgramme_name() {
    return programme_name;
}
```



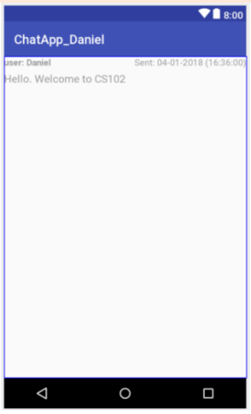
New Resource File

File name: message
Root element: RelativeLayout
Source set: main
Directory name: layout

+ The layout must have **TextView** widgets to display the chat message's text, the time it was sent, and its author.

Open the **message.xml file** and free to place widgets in any order:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <!--Add TextView: display user who sent the message-->
    <TextView
        android:id="@+id/message_user"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:hint="user: Daniel"
        android:textStyle="normal|bold" />

    <!--Add TextView: display time that message was sent-->
    <TextView
        android:id="@+id/message_time"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/message_user"
        android:layout_alignParentEnd="true"
        android:hint="Sent: 04-01-2018 (16:36:00)" />

    <!--Add TextView: display the message-->
    <TextView
        android:id="@+id/message_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/message_user"
        android:layout_marginTop="5dp"
        android:hint="Hello. Welcome to CS102"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1"
        android:textSize="17sp" />
</RelativeLayout>
```



ChatApp_Daniel

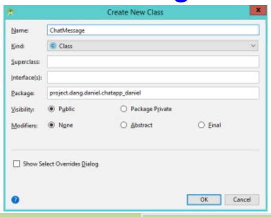
user: Daniel   Sent: 04-01-2018 (16:36:00)

Hello. Welcome to CS102

Step 4: Create a "Chat Message" Model

In order to store the chat messages in the **Firebase real-time database**, you must create a model for them. The layout of the chat message, which we created earlier in this tutorial, has **three views**. To be able to populate those views, **the model too must have at least three fields**.

+ Add anew Java class to your project, name it as ChatMessage:


Create New Class

Name: ChatMessage
Kind: Class
Superclass:
Interface(s):
Package: project.dang.daniel.chatapp_daniel
Visibility: [✓ Public] Package Private
Modifiers: [✓ None] Abstract Final
☐ Show Select Overrides Dialog

+ The **ChatMessage.java** has three member variables to it: messageText , messageUser , and messageTime . We also add a constructor to initialize those variables. To make the model compatible with **FirebaseUI**, you must also **add a default constructor** to it, along with **getters** and **setters** for all the **member variables**.

+ Open ChatMessage.java file and edit it as below:

```java
import java.util.Date;

public class ChatMessage {
    ////////////////////////////////////////////////////////////////////////////////////
    //Declare class properties/data
    private String messageText;
    private String messageUser;
    private long messageTime;

    ////////////////////////////////////////////////////////////////////////////////////
    //Constructor 1
    public ChatMessage(String messageText, String messageUser) {
        this.messageText = messageText;
        this.messageUser = messageUser;

        // Initialize to current time
        messageTime = new Date().getTime();
    }

    ////////////////////////////////////////////////////////////////////////////////////
    //Constructor 2
    public ChatMessage() {

    }

    ////////////////////////////////////////////////////////////////////////////////////
    //Getter and setters
    public String getMessageText() {
        return messageText;
    }

    //
    public void setMessageText(String messageText) {
        this.messageText = messageText;
    }

    //
    public String getMessageUser() {
        return messageUser;
    }

    public void setMessageUser(String messageUser) {
        this.messageUser = messageUser;
    }

    //
    public long getMessageTime() {
        return messageTime;
    }

    //
    public void setMessageTime(long messageTime) {
        this.messageTime = messageTime;
    }
}
```

Step 5: Edit ChatActivity.java to handle the group chat: Post a Chat Message

Now that the model is ready, we can easily **add new chat messages** to the **Firebase real-time database**.

To post a new message, the user will press the **FloatingActionButton**. Therefore, you must add an OnClickListener to it. Inside the listener, you must first get a DatabaseReference object using the getReference() method of the FirebaseDatabase class. You can then call the push() and setValue() methods to add new instances of the ChatMessage class to the real-time database.

The ChatMessage instances must, of course, be initialized using the contents of the EditText and the display name of the currently signed in user.

+ Now, open ChatActivity.java file and add the following codes:

_Declare few variables right after the class declaration:

```
///////////////////////////////////////////////////////////////////////////////////
//Post a chat message - 1: Declare variables
private FloatingActionButton sendBtn;
private EditText input;
```

_Inside onCreate() method, add the below codes at the end of method:

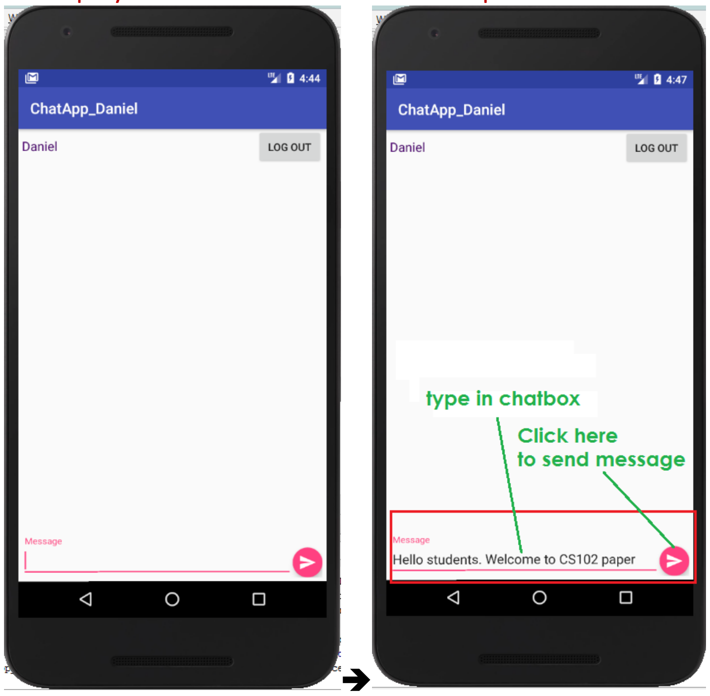
```
///////////////////////////////////////////////////////////////////////////////
//Post a chat message - 3: Set click listener for the sendBtn
sendBtn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        //Check if the input field is not empty, send the message & delete it after sending out
        if (!input.getText().toString().isEmpty()) {
            //Read the input field and push a new instance of ChatMessage to the Firebase database
            //We also create a node "GroupChatMessages" to store those chat messages
            FirebaseDatabase.getInstance().getReference().child("GroupChatMessages")
                    .push()
                    .setValue(new ChatMessage(input.getText().toString(),
                            FirebaseAuth.getInstance()
                                    .getCurrentUser()
                                    .getDisplayName()));
            //Clear the input
            input.setText("");
        }
    }
});
```

Note that: Data in the **Firebase real-time database** is always stored as key-value pairs. However, if you observe the code above, you'll see that we're calling **setValue()** without specifying any key. That's allowed only because the call to the **setValue() method** is preceded by a call to the **push() method**, which automatically generates a new key.

+ Build and run your app on an AVD (Nexus 5X):
On the app side:

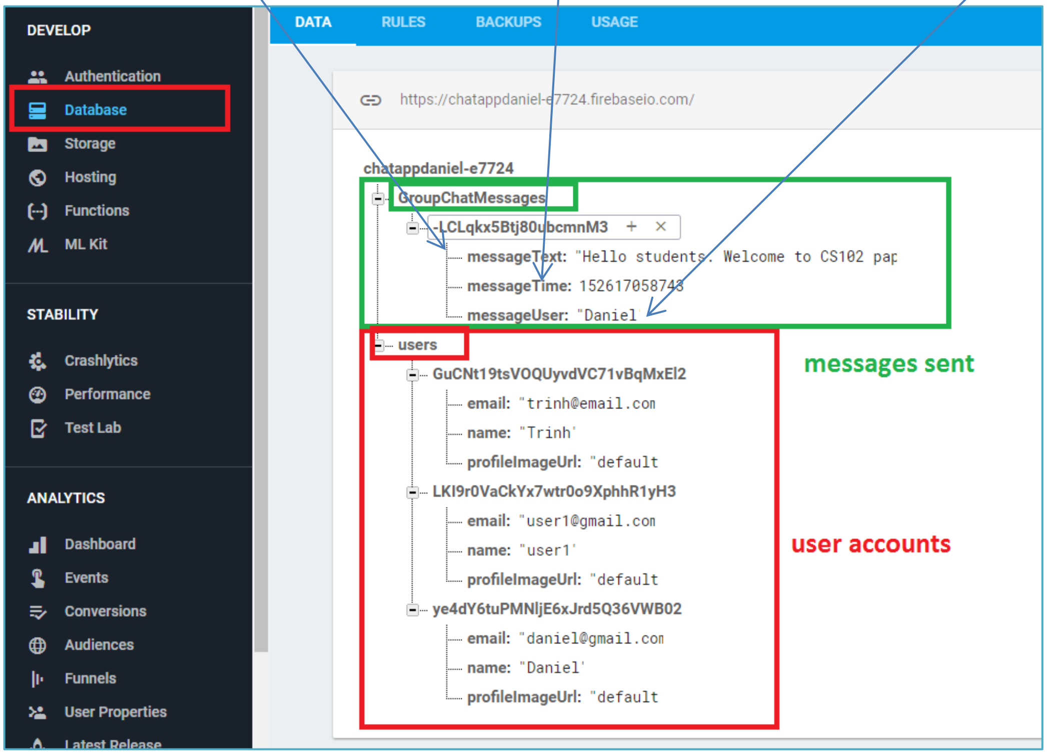
- First of all, **login** the app with your user account;
- Then type a message, such as "Hello students. Welcome to CS102 paper", in the chat box;
- Click "**send**" button to send the message to group chat room;

You will see that the message has been sent and the chat box is set to empty. But you don't see the sent message display on the chat window. We'll implement it later



+ Open Firebase console and open Database to see:

You will see that there is a new message added under the child node "**GroupChatMessages**" in database. The message has 3 data: messageText (message body), messageTime (When it was posted) and messageUser (who posted it)



```
https://chatappdaniel-e7724.firebaseio.com/
chatappdaniel-e7724
  GroupChatMessages
    -LCLqkx5Btj80ubcmnM3
      messageText: "Hello students. Welcome to CS102 pap
      messageTime: 1526170587438
      messageUser: "Daniel"
  users
    GuCNt19tsVOQUyvdVC71vBqMxEl2
      email: "trinh@email.com
      name: "Trinh"
      profileImageUrl: "default
    LKI9r0VaCkYx7wtr0o9XphhR1yH3
      email: "user1@gmail.com
      name: "user1"
      profileImageUrl: "default
    ye4dY6tuPMNljE6xJrd5Q36VWB02
      email: "daniel@gmail.com
      name: "Daniel"
      profileImageUrl: "default
```

Step 6: Display the Chat Messages on Chat window

Add the third party library: Firebase-UI to your Android project:

We'll be using **one external library** in this project: **Firebase-UI**:

- Read more about **FirebaseUI** at: https://firebase.googleblog.com/2015/08/build-better-mobile-apps-with-firebaseui_78.html & check **FirebaseUI version** at https://github.com/firebase/FirebaseUI-Android

+ Open the **build.gradle** file of the app module and add the following implementation dependencies to it:

```
//To integrate Firebase-ui library
implementation 'com.firebaseui:firebase-ui:2.2.0'//FirebaseUI
```

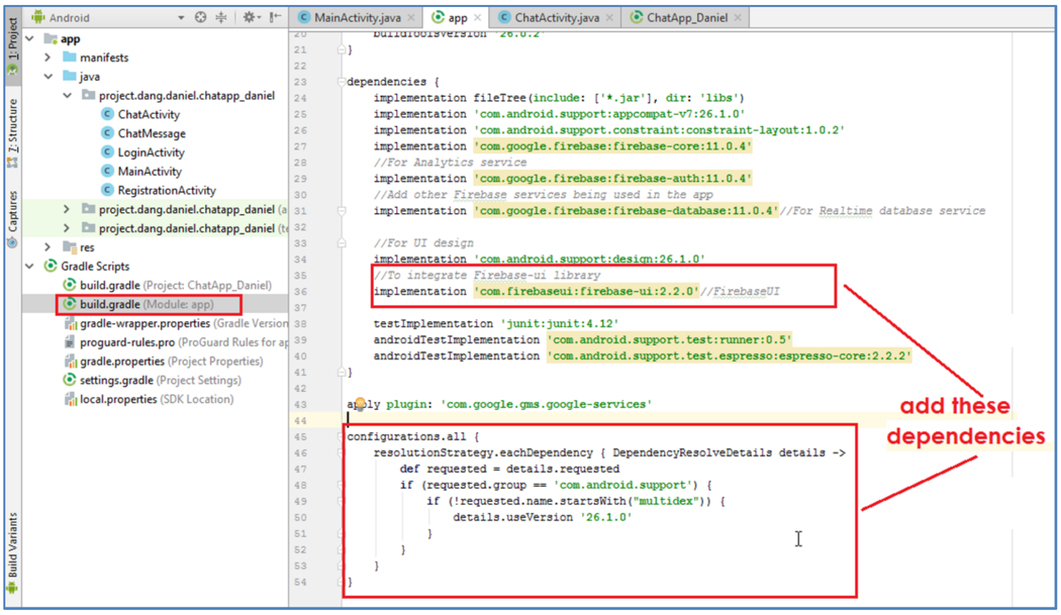
+ Press the "**Sync Now"** button to update the project.

+ If you face up error: "Attribute meta-data#android.support.VERSION@value value=(26.1.0) from [com.android.support:design:26.1.0] AndroidManifest.xml:28:13-35 is also present at [com.android.support:customtabs:25.4.0] AndroidManifest.xml:25:13-35 value=(25.4.0)."

You add the below code snippet at the end of gradle file:

```
configurations.all {
    resolutionStrategy.eachDependency { DependencyResolveDetails details ->
        def requested = details.requested
        if (requested.group == 'com.android.support') {
            if (!requested.name.startsWith("multidex")) {
                details.useVersion '26.1.0'
            }
        }
    }
}
```

+ The **gradle file** looks like that:


Android
MainActivity.java
app
ChatActivity.java
ChatApp_Daniel
1: Project
7: Structure
Captures
Build Variants
app
manifests
java
project.dang.daniel.chatapp_daniel
ChatActivity
ChatMessage
LoginActivity
MainActivity
RegistrationActivity
project.dang.daniel.chatapp_daniel (a
project.dang.daniel.chatapp_daniel (t
res
Gradle Scripts
build.gradle (Project: ChatApp_Daniel)
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Version
proguard-rules.pro (ProGuard Rules for ap
gradle.properties (Project Properties)
settings.gradle (Project Settings)
local.properties (SDK Location)
add these dependencies

```
    buildToolsVersion '26.0.2'
}

dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:26.1.0'
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'
    implementation 'com.google.firebase:firebase-core:11.0.4'
    //For Analytics service
    implementation 'com.google.firebase:firebase-auth:11.0.4'
    //Add other Firebase services being used in the app
    implementation 'com.google.firebase:firebase-database:11.0.4'//For Realtime database service

    //For UI design
    implementation 'com.android.support:design:26.1.0'
    //To integrate Firebase-ui library
    implementation 'com.firebaseui:firebase-ui:2.2.0'//FirebaseUI

    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:0.5'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:2.2.2'
}

apply plugin: 'com.google.gms.google-services'

configurations.all {
    resolutionStrategy.eachDependency { DependencyResolveDetails details ->
        def requested = details.requested
        if (requested.group == 'com.android.support') {
            if (!requested.name.startsWith("multidex")) {
                details.useVersion '26.1.0'
            }
        }
    }
}
```

Firebase-UI library has a very handy class called **FirebaseListAdapter**, which dramatically reduces the effort required to populate a **ListView** using data present in the **Firebase real-time database**. We'll be using it now to fetch and display all the **ChatMessage** objects that are present in the database.

- **FirebaseListAdapter** is an abstract class and has an abstract **populateView() method**, which must be overridden. As its name suggests, **populateView()** is used to populate the views of each list item.

+ Open ChatActivity.java file and add below codes to it:

_Declare few variables right after the class declaration:

```
/////////////////////////////////////////////////////////////////////////////////
//Display a chat message - 1: Declare variables
private FirebaseListAdapter<ChatMessage> adapter;
private ListView listOfMessages;
```

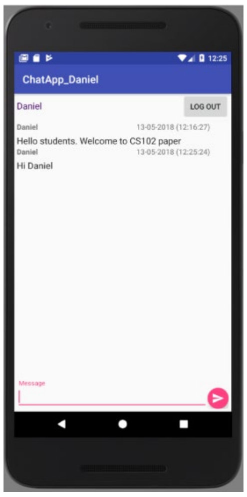
_Inside onCreate() method, add the below codes at the end of method:

```
/////////////////////////////////////////////////////////////////////////////
//Display a chat message - 2: displayChatMessages() method
//Find reference for the listOfMessage listView
listOfMessages = (ListView) findViewById(R.id.list_of_messages);
//Collect all the posted messages stored inside the node "GroupChatMessages" of Firebase Database
adapter = new FirebaseListAdapter<ChatMessage>(this, ChatMessage.class, R.layout.message,
        FirebaseDatabase.getInstance().getReference().child("GroupChatMessages")) {
    @Override
    protected void populateView(View view, ChatMessage model, int position) {
        //1: Get references to the views of message.xml
        TextView messageText = (TextView) view.findViewById(R.id.message_text);
        TextView messageUser = (TextView) view.findViewById(R.id.message_user);
        TextView messageTime = (TextView) view.findViewById(R.id.message_time);
        //2: Set the chat text
        messageText.setText(model.getMessageText());
        messageUser.setText(model.getMessageUser());
        //3: Format the date before showing it on chat room
        messageTime.setText(android.text.format.DateFormat.format("dd-MM-yyyy (HH:mm:ss)",
                model.getMessageTime()));

    }
};

//Populate listview with chat messages
listOfMessages.setAdapter(adapter);
```

+ Build and run your app for the first time on AVD (Nexus 5X) to see the main screen:

The group chat app is now ready. Run it and a post new message to see them pop up immediately in the ListView.

- **Share the app with your friends and you should be able to see their messages too as soon as they post them.**



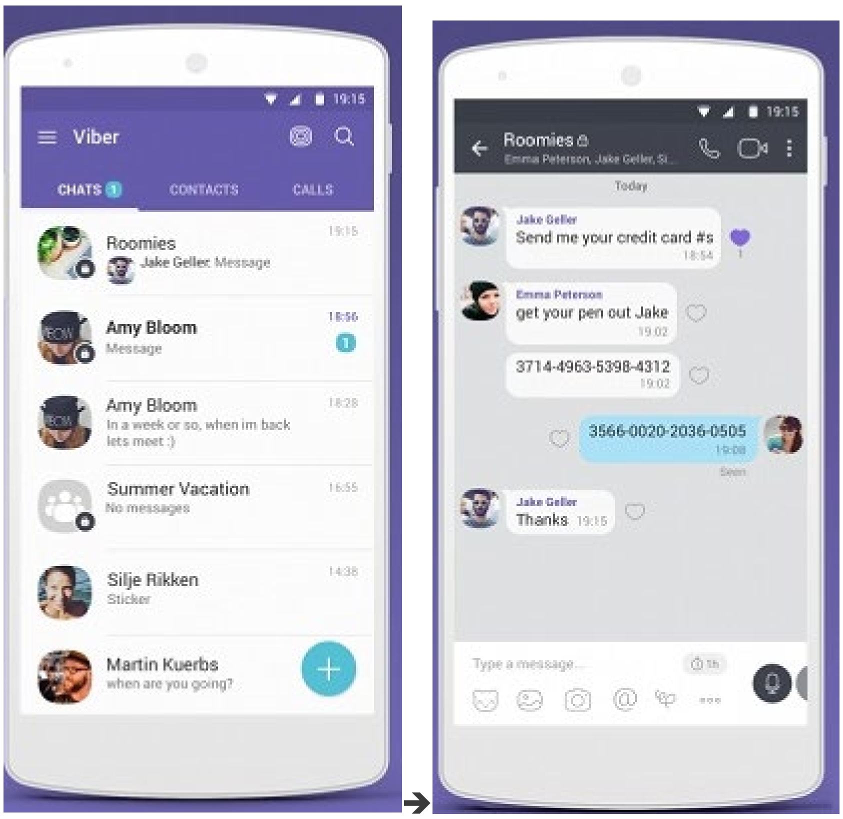
Daniel 13-05-2018 (12:16:27)
Hello students. Welcome to CS102 paper
Daniel 13-05-2018 (12:25:24)
Hi Daniel

In this tutorial, you learned how to **use Firebase and FirebaseUI to create a very simple group chat application**. You also saw how easy it is to work with the classes available in FirebaseUI to quickly create new screens and implement complex functionality.

PART 5: Handle the individual chat (Extra work)

For any well-known chat app such as Viber, Skype, etc., we can start an individual chat with one of our friend.

Now, it's your turn to implement the individual chat: "A user can click a button "View all users" to display a list view of all users on database. Then we can pick up one specific user and start sending message to that user".



Roomies 19:15
Jake Geller: Message

Amy Bloom 18:56
Message

Amy Bloom 18:28
In a week or so, when im back lets meet :)

Summer Vacation 16:55
No messages

Silje Rikken 14:38
Sticker

Martin Kuerbs
when are you going?

Jake Geller
Send me your credit card #s 18:54

Emma Peterson
get your pen out Jake 19:02

3714-4963-5398-4312 19:02

3566-0020-2036-0505 19:08
Seen

Jake Geller
Thanks 19:15