twitch.tv/dancojocar
youtube.com/dancojocar

# Lecture #11
# Firebase Services

Mobile Applications
Fall 2023


2007 Chat App 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020

https://startupandrew.com/posting-experiences-in-the-haitian-earthquake/

# Firebase

Analytics

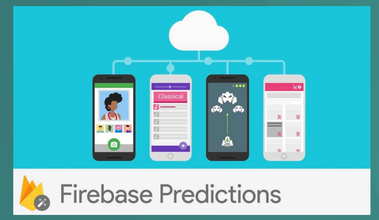
Cloud Function for Firebase

Firebase Authentication

Firebase Cloud Messaging

Cloud Firestore

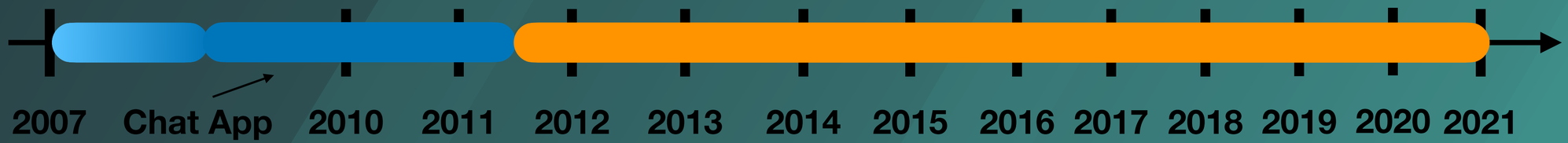
Cloud Storage for Firebase




Firebase Realtime Database

Firebase Hosting


ML Kit for Firebase

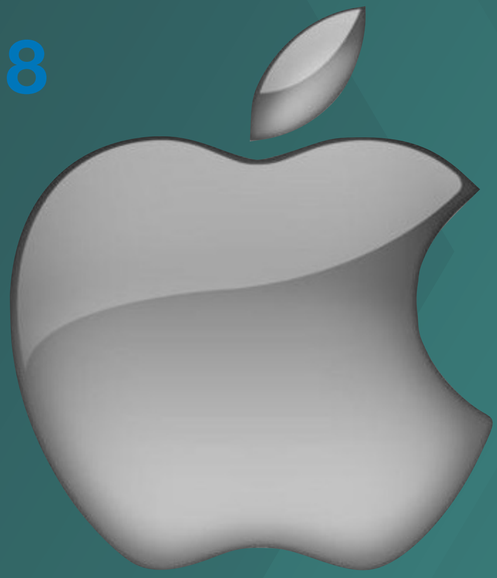




https://firebase.google.com/support/releases

# Firebase

**Over 2.5 million apps actively using Firebase every month!**

2018



# Firebase

**Over 3 million apps actively using Firebase every month!**


# Add Firebase to Your Project

- Using the assistant, in Android Studio:

  1.Tools -> Firebase.

  2. Select the service.

  3. Connect to Firebase.

- Manually:
  - Create a project in console.firebase.google.com
  - Download the config file.
  - Add the SDK.

# Add the SDK

Root-level build.gradle:

```
buildscript {
  // ...
  dependencies {
    // ...
    classpath ‘com.google.gms:google-services:4.3.10’ // google-services plugin
  }
}

allprojects {
  // ...
  repositories {
    // ...
    google() // Google's Maven repository
  }
}
```

# Add the SDK

Module-level build.gradle:

```
dependencies {
  // ...

  // Import the Firebase BoM
  implementation platform(‘com.google.firebase:firebase-bom:31.1.1')

  // When using the BoM, you don't specify versions in Firebase library dependencies

  // Declare the dependencies for the desired Firebase products
  // For example, declare the dependencies for Firebase Authentication and Cloud Firestore
  implementation 'com.google.firebase:firebase-auth-ktx'
  implementation 'com.google.firebase:firebase-firestore-ktx'
}
```

# Available Libraries

firebase.inappmessaging.display

## firebase.perf

com.google.firebase.perf

com.google.firebase.perf.metrics

## firebase.remoteconfig

com.google.firebase.remoteconfig

## firebase.storage

com.google.firebase.storage

## Inter-operational packages

com.google.firebase.auth.internal

com.google.firebase.ml.naturallanguage.languageid

com.google.firebase.ml.naturallanguage.smartreply

firebase.google.com/docs/reference/android/packages

# Realtime Database

- Realtime.
- Offline.
- Collaborate across devices with ease.
- Scale across multiple regions.

# Installation & Setup

Add the Realtime Database to your app

```
implementation ‘com.google.firebase:firebase-database-ktx’
```

Secure Your Data

```
{
{"rules": {
 "messages": {{
  "$message": {{
   "content": "Hello",
   "timestamp": 1405704370369
  }, ".read": "data.child('timestamp').val() >
      (now - 600000)",
  "message1": {{
   "content": "Goodbye",
   "timestamp": 1405704395231
  }, ".validate": “newData.hasChildren(['content',
      'timestamp'])
  ...
  }       && newData.child('content').isString()
}         && newData.child('timestamp').isNumber()"
    }
   }
  }
}
```

**https://firebase.google.com/docs/database/security/securing-data**

# Data Access

Write to your database

Read from your database

```
// Read from the database
myRef.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        // This method is called once with the initial value and again
        // whenever data at this location is updated.
        val value = dataSnapshot.getValue(String::class.java)
        Log.d(TAG, "Value is: $value")
    }

    override fun onCancelled(error: DatabaseError) {
        // Failed to read value
        Log.w(TAG, "Failed to read value.", error.toException())
    }
})
```

```
// Write a message to the database
val database = FirebaseDatabase.getInstance()
val myRef = database.getReference("message")

myRef.setValue("Hello, World!")
```

**https://firebase.google.com/docs/database/android/read-and-write**

# Update Data

```
private fun writeNewPost(
    userId: String,
    username: String,
    title: String,
    body: String
){
  // Create new post at /user-posts/$userid/$postid and at
  // /posts/$postid simultaneously
  val key = database.child("posts").push().key
  if (key == null) {
    Log.w(TAG, "Couldn't get push key for posts")
    return
  }
  val post = Post(userId, username, title, body)
  val postValues = post.toMap()
  val childUpdates = HashMap<String, Any>()
  childUpdates["/posts/$key"] = postValues
  childUpdates["/user-posts/$userId/$key"] = postValues
  database.updateChildren(childUpdates)
}
```

# Using Transactions

```
private fun onStarClicked(postRef: DatabaseReference) {
  postRef.runTransaction(object : Transaction.Handler {
    override fun doTransaction(mutableData: MutableData): Transaction.Result {
      val p = mutableData.getValue(Post::class.java)
          ?: return Transaction.success(mutableData)
      if (p.stars.containsKey(uid)) {
        // Unstar the post and remove self from stars
        p.starCount = p.starCount - 1
        p.stars.remove(uid)
      } else {
        // Star the post and add self to stars
        p.starCount = p.starCount + 1
        p.stars[uid] = true
      }
      // Set value and report transaction success
      mutableData.value = p
      return Transaction.success(mutableData)
    }
    override fun onComplete(
      databaseError: DatabaseError?,
      b: Boolean,
```

# Enabling Offline Capabilities

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

https://firebase.google.com/docs/database/android/offline-capabilities

# Enabling Offline Capabilities

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

Keeping Data Fresh

```
DatabaseReference scoresRef = FirebaseDatabase.getInstance()
    .getReference("scores");
scoresRef.keepSynced(true);
```

**https://firebase.google.com/docs/database/android/offline-capabilities**

# Enabling Offline Capabilities

```
FirebaseDatabase.getInstance().setPersistenceEnabled(true);
```

Keeping Data Fresh

```
DatabaseReference scoresRef = FirebaseDatabase.getInstance()
    .getReference("scores");
scoresRef.keepSynced(true);
scoresRef.keepSynced(false);
```

**https://firebase.google.com/docs/database/android/offline-capabilities**

# Enabling Offline Capabilities

Detecting Connection State

```
val connectedRef = FirebaseDatabase.getInstance().getReference(".info/connected")
connectedRef.addValueEventListener(object : ValueEventListener {
  override fun onDataChange(snapshot: DataSnapshot) {
    val connected = snapshot.getValue(Boolean::class.java) ?: false
    if (connected) {
      Log.d(TAG, "connected")
    } else {
      Log.d(TAG, "not connected")
    }
  }

  override fun onCancelled(error: DatabaseError) {
    Log.w(TAG, "Listener was cancelled")
  }
})
```

**https://firebase.google.com/docs/database/android/offline-capabilities**

# Enabling Offline Capabilities


Detecting Connection State

```
val connectedRef = FirebaseDatabase.getInstance().getReference(".info/connected")
connectedRef.addValueEventListener(object : ValueEventListener {
  override fun onDataChange(snapshot: DataSnapshot) {
    val connected = snapshot.getValue(Boolean::class.java) ?: false
    if (connected) {
      Log.d(TAG, "connected")
    } else {
      Log.d(TAG, "not connected")
    }
  }

  override fun onCancelled(error: DatabaseError) {
    Log.w(TAG, "Listener was cancelled")
  }
})
```

**https://firebase.google.com/docs/database/android/offline-capabilities**

# Authentication

Key capabilities:

- Email and password based authentication.
- Federated identity provider integration:
  - Google, Facebook, Twitter, Github.
- Phone number authentication.
- Custom auth system integration.
- Anonymous auth.

# Authentication

Key capabilities:

- Email and password based authentication.
- Federated identity provider integration:
  - Google, Facebook, Twitter, Github, Apple.
- Phone number authentication.
- Custom auth system integration.
- Anonymous auth.

# Authenticate Using Google Sign-In

- Dependencies
- Integrate Google Sign-In
- Use shared auth
- Use the token

```
implementation 'com.google.firebase:firebase-auth-ktx'
implementation 'com.google.android.gms:play-services-auth'
```

```
// Configure Google Sign In
val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(getString(R.string.default_web_client_id))
        .requestEmail()
        .build()
mGoogleSignInClient = GoogleSignIn.getClient(this, gso)
```

```
// Initialize Firebase Auth
auth = Firebase.auth
```

```
override fun onStart() {
    super.onStart()
    // Check if user is signed in (non-null) and update UI accordingly.
    val currentUser = auth.currentUser
    updateUI(currentUser)
}
```

```
private fun signIn() {
    val signInIntent = mGoogleSignInClient.signInIntent
    startActivityForResult(signInIntent, RC_SIGN_IN)
}
```

```
public override fun onActivityResult(requestCode: Int, resultCode: Int,
    data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    // Result returned from launching the Intent from
    // GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        val task = GoogleSignIn.getSignedInAccountFromIntent(data)
        try {
            // Google Sign In was successful, authenticate with Firebase
            val account = task.getResult(ApiException::class.java)
            firebaseAuthWithGoogle(account!!)
        } catch (e: ApiException) {
            // Google Sign In failed, update UI appropriately
            Log.w(TAG, "Google sign in failed", e)
            // ...
        }
    }
}
```

```
private fun firebaseAuthWithGoogle(acct: GoogleSignInAccount) {
    Log.d(TAG, "firebaseAuthWithGoogle:" + acct.id!!)

    val credential = GoogleAuthProvider.getCredential(acct.idToken, null)
    auth.signInWithCredential(credential)
        .addOnCompleteListener(this) { task ->
            if (task.isSuccessful) {
                // Sign in success, update UI with the signed-in user's information
                Log.d(TAG, "signInWithCredential:success")
                val user = auth.currentUser
                updateUI(user)
            } else {
                // If sign in fails, display a message to the user.
                Log.w(TAG, "signInWithCredential:failure", task.exception)
                updateUI(null)
            }
        }
}
```

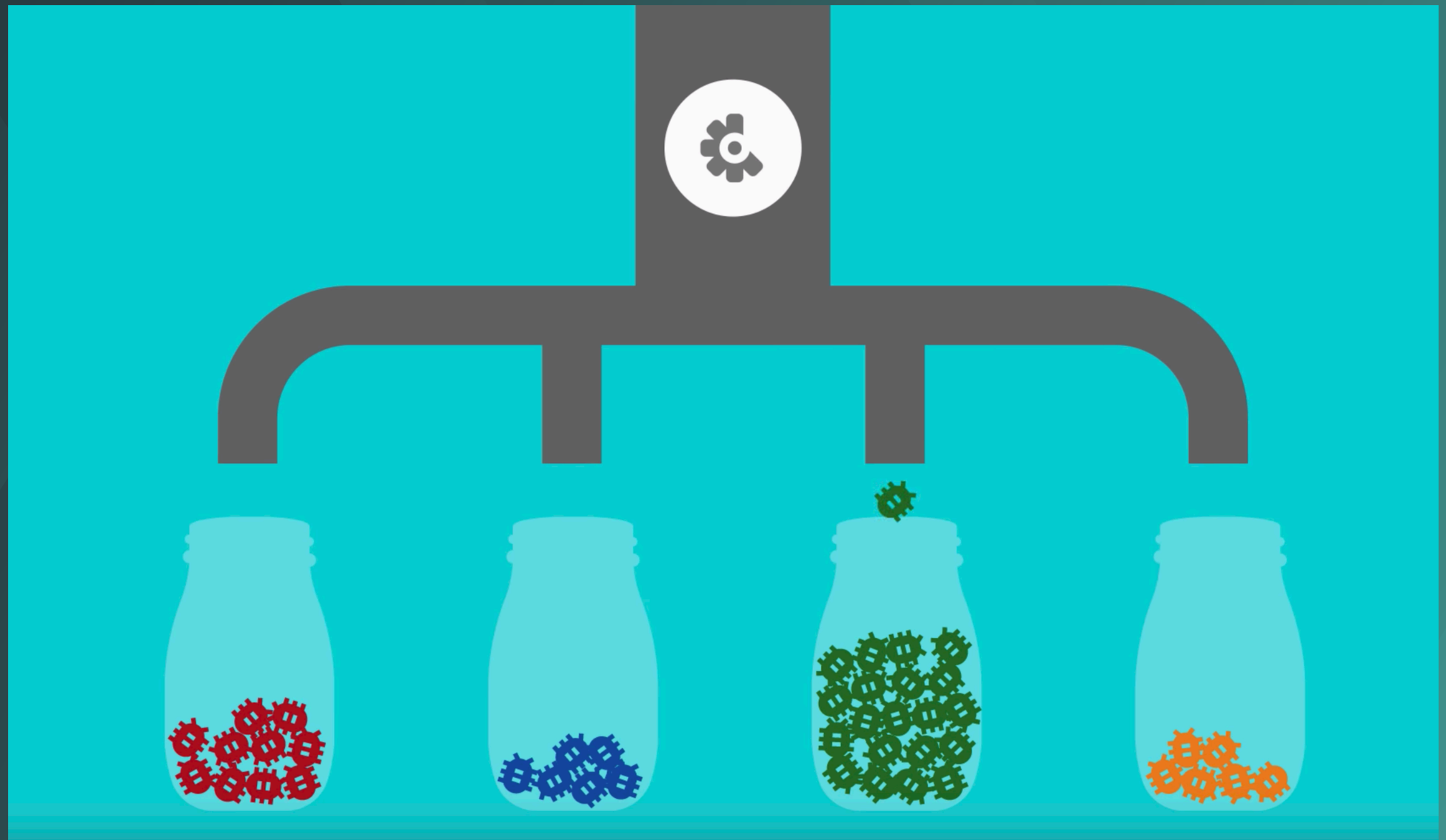
# Sign out a User

DEMO

```
FirebaseAuth.getInstance().signOut()
```


Error
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum est ac ultrices porttitor. Suspendisse pellentesque risus vehicula, vehicula erat a, tempor nisi. Donec vel suscipit sapien, vitae sollicitudin dui. Fusce vitae nulla ornare, sollicitudin nibh et, malesuada eros. In nec nisi consequat, laoreet massa et, molestie nisi. Proin eget dui lacus. Phasellus auctor faucibus facilisis.
Sed nec arcu nibh. Maecenas nulla velit, hendrerit at arcu ut, viverra ullamcorper eros. Proin efficitur libero in urna gravida, dictum tempor nibh bibendum. Ut eu libero non leo elementum molestie. Nullam convallis convallis tellus, vitae ultrices nisi bibendum non. Suspendisse placerat lorem augue, nec bibendum nisi mollis non. Mauris viverra posuere diam in laoreet. Sed non erat rutrum, bibendum risus a, molestie arcu. Praesent sit amet arcu ut libero faucibus lacinia.

# Crashlytics



# Enable the SDK

```
buildscript {
    repositories {
      // Make sure that you have the following two repositories
      google()  // Google's Maven repository
      mavenCentral()  // Maven Central repository
    }

    dependencies {
        ...
        classpath 'com.android.tools.build:gradle:7.2.0'

        // Make sure that you have the Google services Gradle plugin dependency
        classpath 'com.google.gms:google-services:4.3.14'

        // Add the dependency for the Crashlytics Gradle plugin
        classpath 'com.google.firebase:firebase-crashlytics-gradle:2.9.2'
    }
}
```

# Enable the SDK

```
plugins {
    id 'com.android.application'

    // Make sure that you have the Google services Gradle plugin
    id 'com.google.gms.google-services'

    // Add the Crashlytics Gradle plugin
    id 'com.google.firebase.crashlytics'
    ...
}

dependencies {
    // Import the BoM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:31.1.1')

    // Add the dependencies for the Crashlytics and Analytics libraries
    // When using the BoM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebase:firebase-crashlytics-ktx'
    implementation 'com.google.firebase:firebase-analytics-ktx'
}
```

# Test Implementation

```
val crashButton = Button(this)
crashButton.text = "Test Crash"
crashButton.setOnClickListener {
  throw RuntimeException("Test Crash") // Force a crash
}
```

# Customize

**AndroidManifest.xml**

```
<meta-data
    android:name="firebase_crashlytics_collection_enabled"
    android:value="false" />
```

# Customize

**AndroidManifest.xml**

```
<meta-data
    android:name="firebase_crashlytics_collection_enabled"
    android:value="false" />
```

Enable collection for selected users:

```
val crashlytics = FirebaseCrashlytics.getInstance()
Crashlytics.setUserIdentifier("myAppUserId")
```

# Customize

**AndroidManifest.xml**

```
<meta-data
    android:name="firebase_crashlytics_collection_enabled"
    android:value="false" />
```

Enable collection for selected users:

```
val crashlytics = FirebaseCrashlytics.getInstance()
Crashlytics.setUserIdentifier("myAppUserId")
```

Crash report and `Log.println`:

```
Crashlytics.log(Log.DEBUG, "tag", "message")
```

# Customize

**`AndroidManifest.xml`**

```
<meta-data
    android:name="firebase_crashlytics_collection_enabled"
    android:value="false" />
```

Enable collection for selected users:

```
val crashlytics = FirebaseCrashlytics.getInstance()
Crashlytics.setUserIdentifier("myAppUserId")
```

Crash report and `Log.println`:

```
Crashlytics.log(Log.DEBUG, "tag", "message")
```

Crash report only:

```
Crashlytics.log("message")
```

# Customize

Log non-fatal exceptions

```
Crashlytics.logException(e)
```

Add custom keys:

```
Crashlytics.setString(key, "foo" /* string value */)
Crashlytics.setBool(key, true /* boolean value */)
Crashlytics.setDouble(key, 1.0 /* double value */)
Crashlytics.setFloat(key, 1.0f /* float value */)
Crashlytics.setInt(key, 1 /* int value */)
```

# Customize


Log non-fatal exceptions

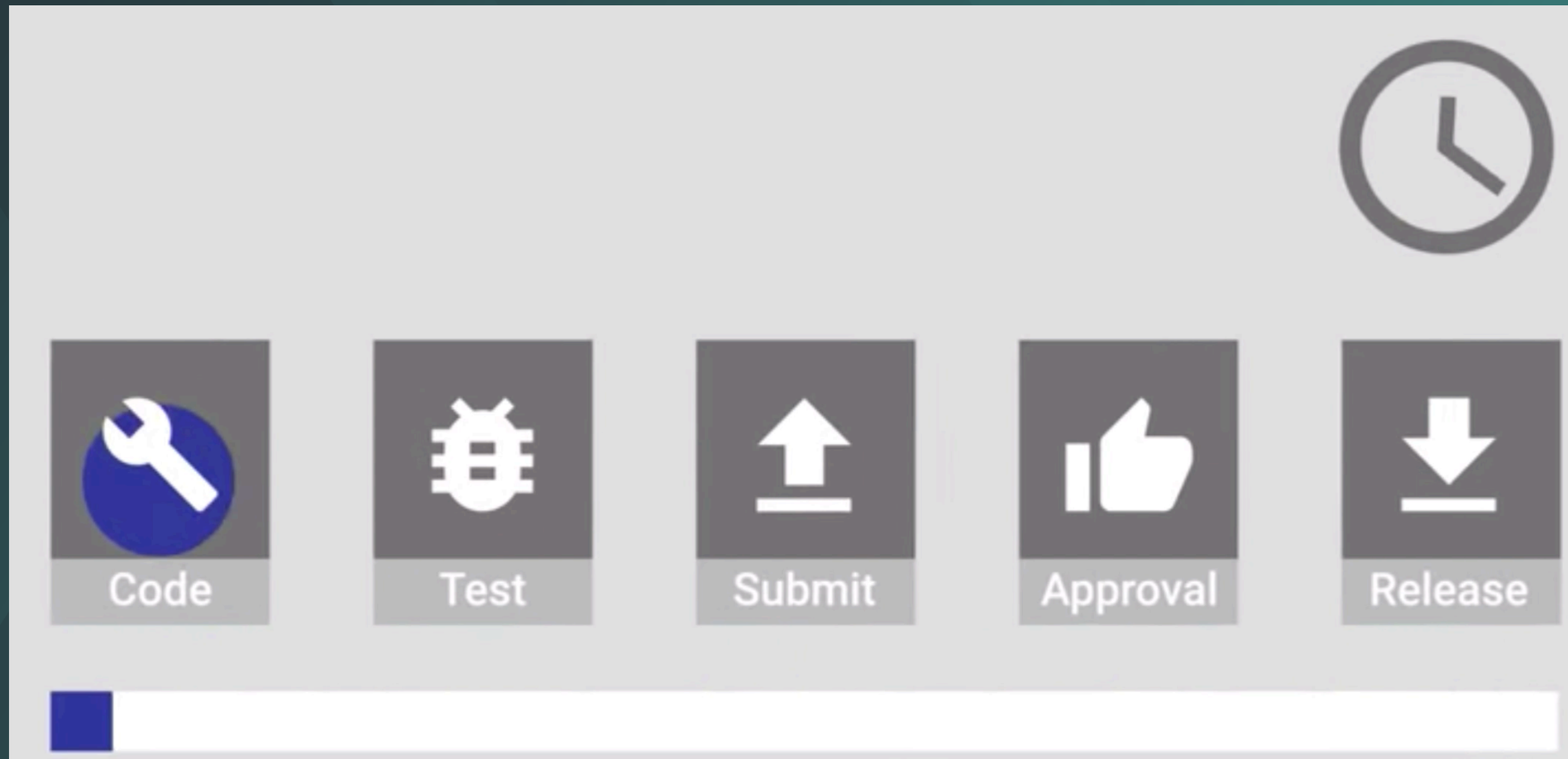
```
Crashlytics.logException(e)
```

Add custom keys:

```
Crashlytics.setString(key, "foo" /* string value */)
Crashlytics.setBool(key, true /* boolean value */)
Crashlytics.setDouble(key, 1.0 /* double value */)
Crashlytics.setFloat(key, 1.0f /* float value */)
Crashlytics.setInt(key, 1 /* int value */)
```

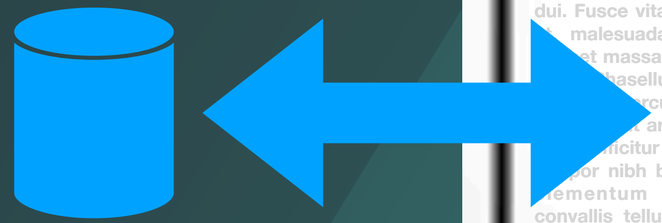
**https://firebase.google.com/docs/crashlytics/customize-crash-reports**


**Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum est ac ultrices porttitor. Suspendisse pellentesque risus vehicula, vehicula erat a, tempor nisi.**
Donec vel suscipit sapien, vitae sollicitudin dui. Fusce vitae nulla ornare, sollicitudin nibh et, malesuada eros. In nec mi consequat, laoreet massa et, molestie nisi. Proin eget dui lacus. Phasellus auctor faucibus facilisis.
Sed nec arcu nibh. Maecenas nulla velit, hendrerit at arcu ut, viverra ullamcorper eros. Proin efficitur libero in urna gravida, dictum tempor nibh bibendum. Ut eu libero non leo elementum molestie. Nullam convallis convallis tellus, vitae ultrices nisi bibendum non. Suspendisse placerat lorem augue, nec bibendum nisi mollis non. Mauris viverra posuere diam in laoreet. Sed non erat rutrum, bibendum risus a, molestie arcu. Praesent sit amet arcu ut libero faucibus lacinia.

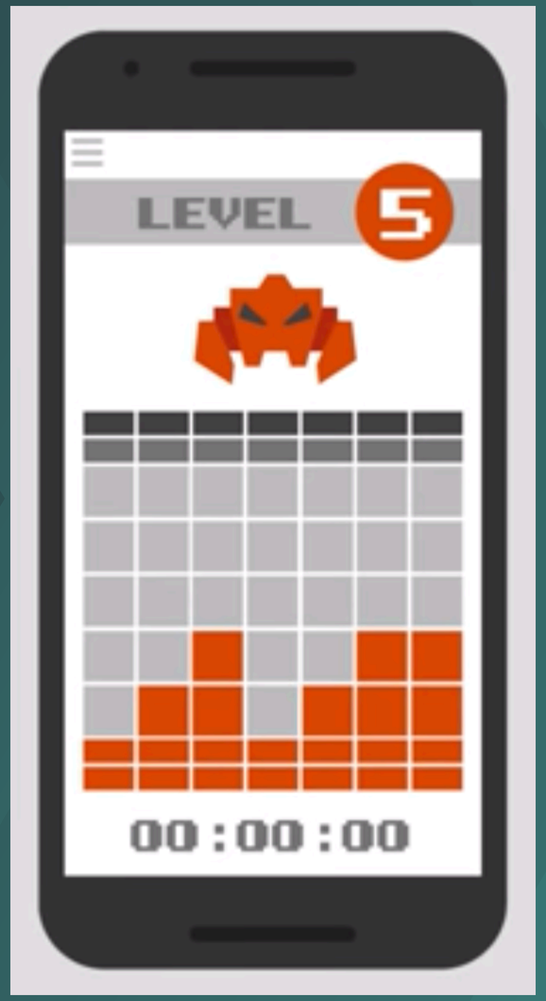

Code
Test
Submit
Approval
Release

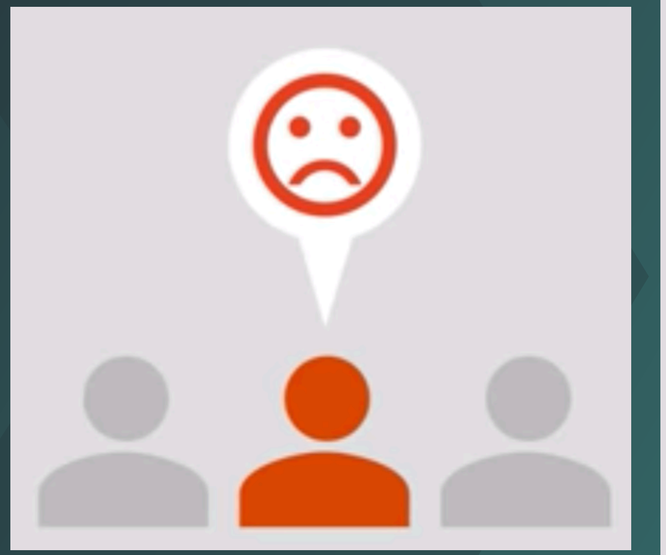

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum est ac ultrices porttitor. Suspendisse pellentesque risus vehicula, vehicula erat a, tempor nisi. Donec vel suscipit sapien, vitae sollicitudin dui. Fusce vitae nulla ornare, sollicitudin nibh et, malesuada eros. In nec mi consequat, laoreet massa et, molestie nisi. Proin eget dui lacus. Phasellus auctor faucibus facilisis.
Sed nec arcu nibh. Maecenas nulla velit, hendrerit at arcu ut, viverra ullamcorper eros. Proin efficitur libero in urna gravida, dictum tempor nibh bibendum. Ut eu libero non leo elementum molestie. Nullam convallis convallis tellus, vitae ultrices nisi bibendum non. Suspendisse placerat lorem augue, nec bibendum nisi mollis non. Mauris viverra posuere diam in laoreet. Sed non erat rutrum, bibendum risus a, molestie arcu. Praesent sit amet arcu ut libero faucibus lacinia.

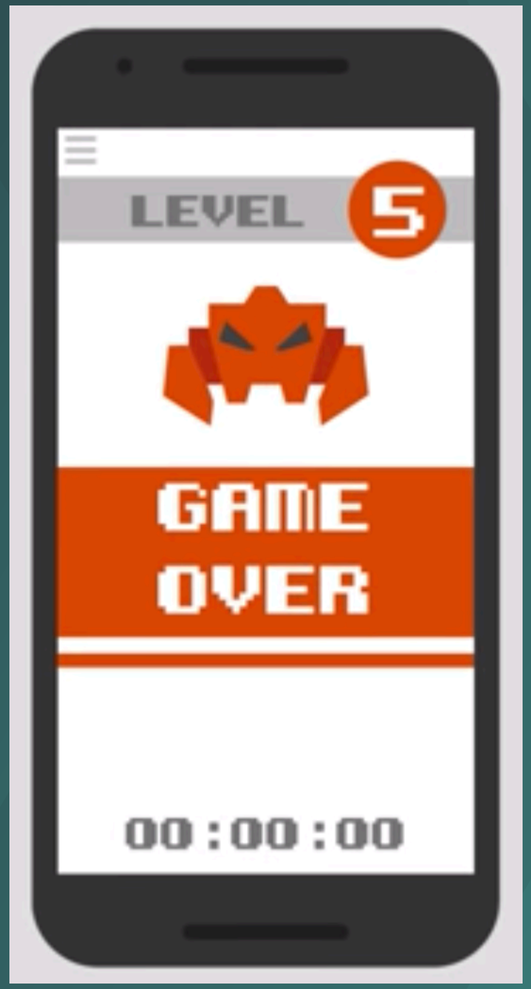
Realtime



Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum est ac ultrices porttitor. Suspendisse pellentesque risus vehicula, vehicula erat a, tempor nisi. Donec vel suscipit sapien, vitae sollicitudin dui. Fusce vitae nulla ornare, sollicitudin nibh malesuada eros. In nec mi consequat, massa et, molestie nisi. Proin eget dui hasellus auctor faucibus facilisis. rcu nibh. Maecenas nulla velit, arcu ut, viverra ullamcorper eros. icitur libero in urna gravida, dictum or nibh bibendum. Ut eu libero non leo ementum molestie. Nullam convallis convallis tellus, vitae ultrices nisi bibendum non. Suspendisse placerat lorem augue, nec bibendum nisi mollis non. Mauris viverra posuere diam in laoreet. Sed non erat rutrum, bibendum risus a, molestie arcu. Praesent sit amet arcu ut libero faucibus lacinia.


LEVEL
5
00 : 00 : 00

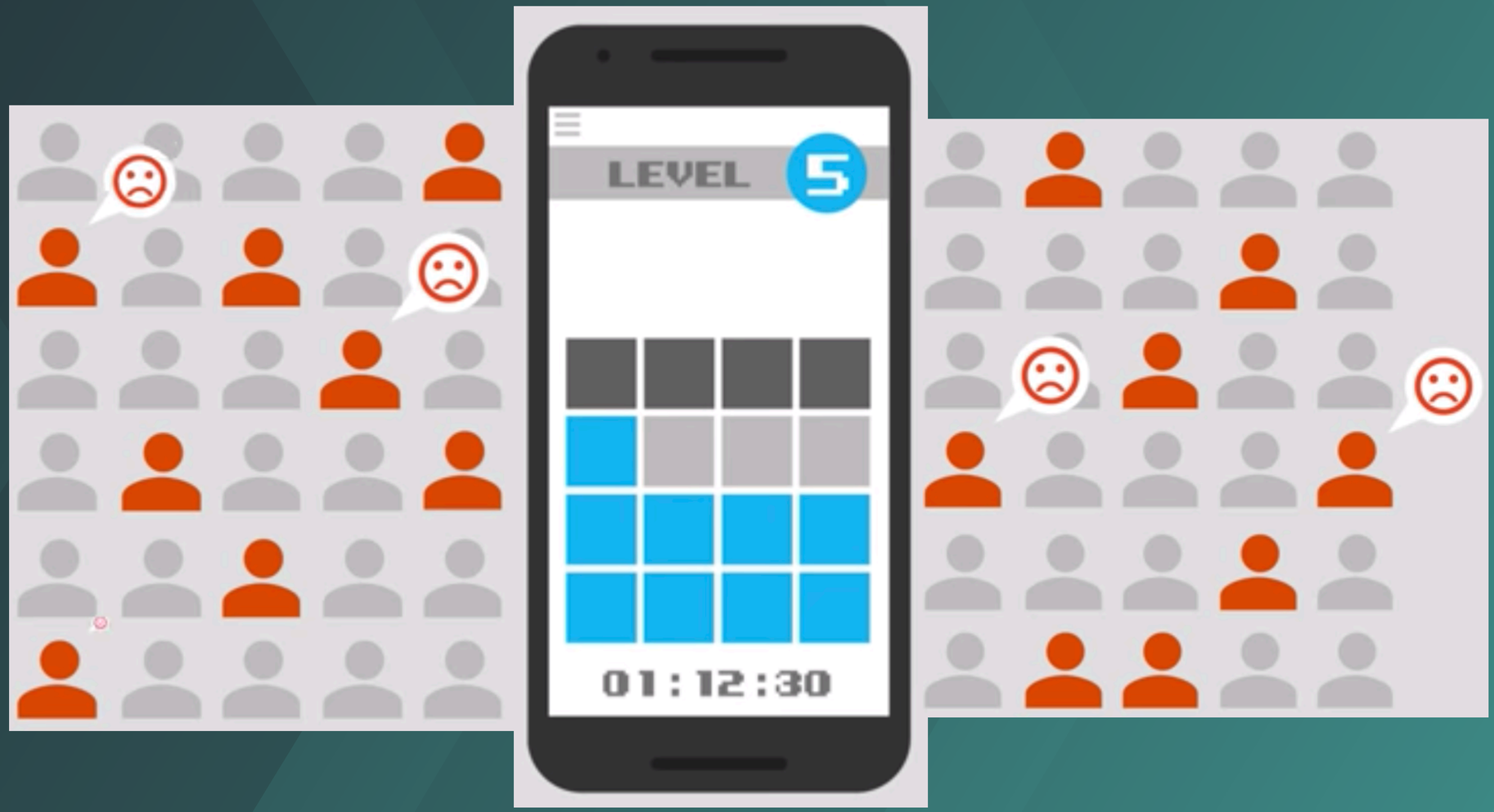



LEVEL
5
GAME
OVER
00 : 00 : 00


Realtime
LEVEL
5
GAME
OVER
00 : 00 : 00


Realtime
LEVEL
5
GAME
OVER
00 : 00 : 00


LEVEL
5
01:12:30


LEVEL
5
01:12:30

# Remote Config

# Using Remote Config


1. Add Firebase to your app

```
implementation 'com.google.firebase:firebase-config-ktx'
```

2. Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

5. Set parameter values in the service (as needed)

6. Fetch and activate values from the service (as needed)

https://firebase.google.com/docs/remote-config/use-config-android

# Make $

- In-app purchases.
- Subscriptions.
- Advertising.
- Paid apps.
- E-commerce.


**https://developer.android.com/distribute/best-practices/earn/monetization-options**

# AdMob

Import the Mobile Ads SDK

```
implementation ‘com.google.android.gms:play-services-ads’
```

Update your AndroidManifest.xml

```
<manifest>
  <application>
    <!-- Sample AdMob App ID: ca-app-pub-3940256099942544~3347511713 -->
    <meta-data
      android:name="com.google.android.gms.ads.APPLICATION_ID"
      android:value="[ADMOB_APP_ID]"/>
  </application>
</manifest>
```

Initialize the SDK

```
override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)
  // Sample AdMob app ID: ca-app-pub-3940256099942544~3347511713
  MobileAds.initialize(this)
}
```

# Ad Formats

- Banner
- Interstitial
- Native
- RewardedVideo

DEMO

# Google Play Billing Overview

DEMO

- Types of in-app products:
  - One-time products.
  - Subscriptions.
- In-app product configuration options:
  - Title.
  - Description.
  - Product ID.
  - Price / Default Price.


**https://developer.android.com/google/play/billing/billing_overview**

# Lecture outcomes

- Use Firebase Realtime Database.
- How to use Remove Config with A-B testing.
- Authenticate your users using Firebase, with GoogleSignIn and EmailPassword methods.
- How to use Firebase Realtime Database.
- Using AdMob to display ads.