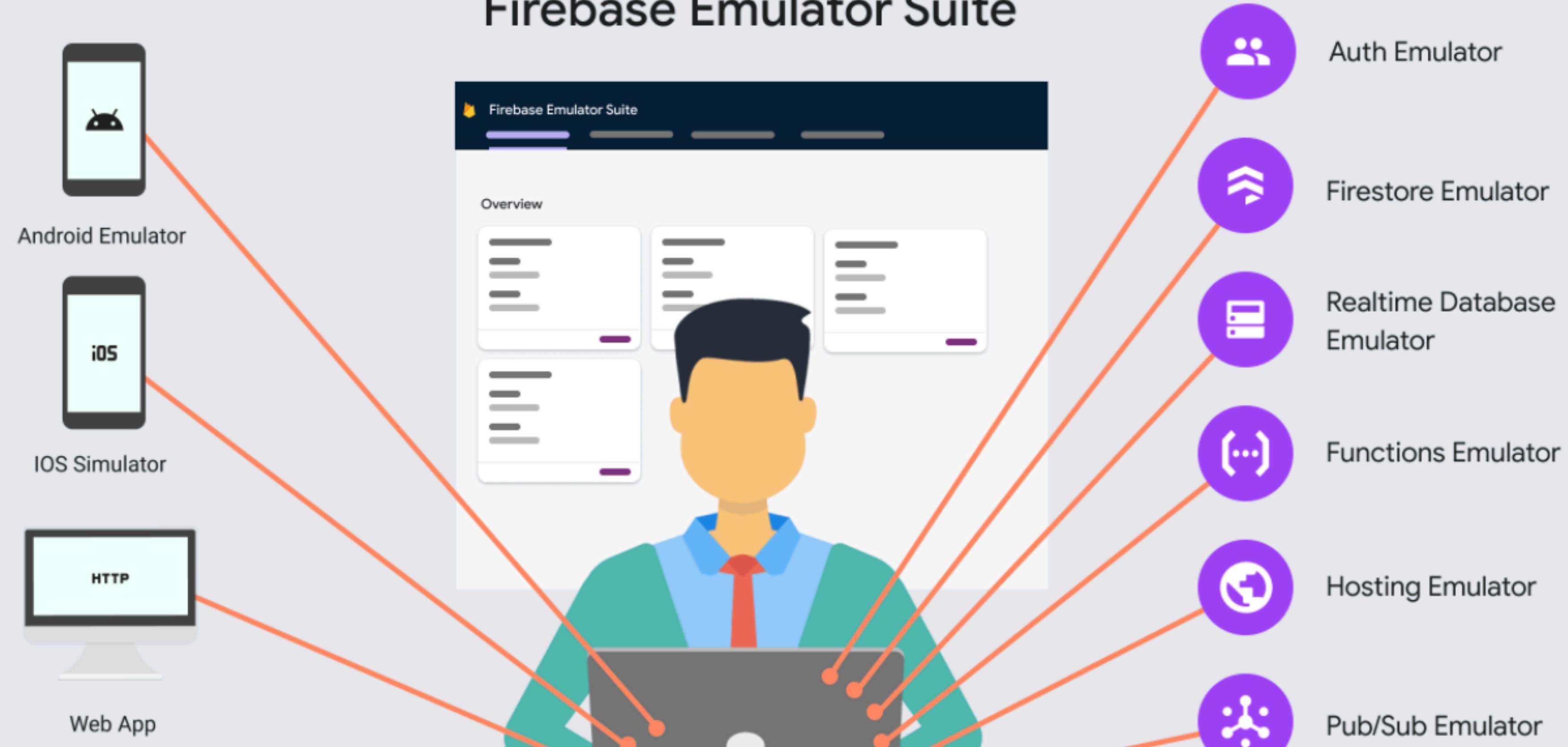


# Lecture #11

# Firebase Services

Mobile Applications  
Fall 2023

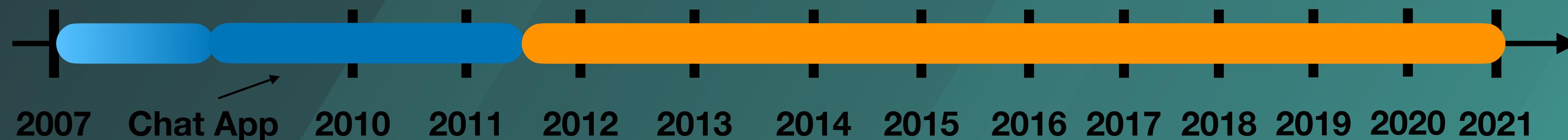
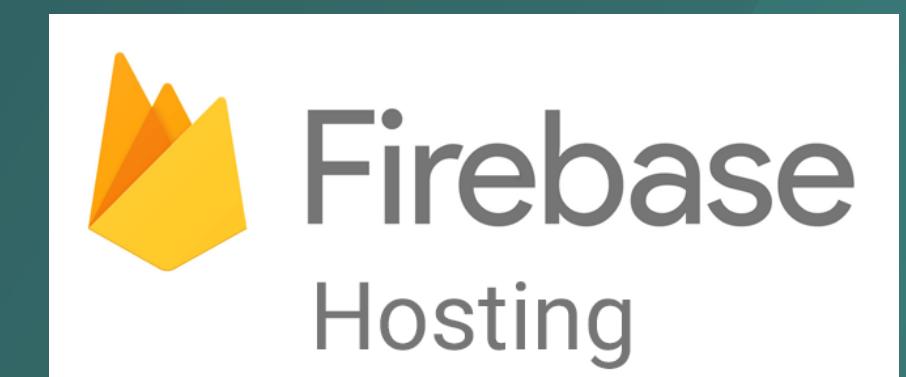
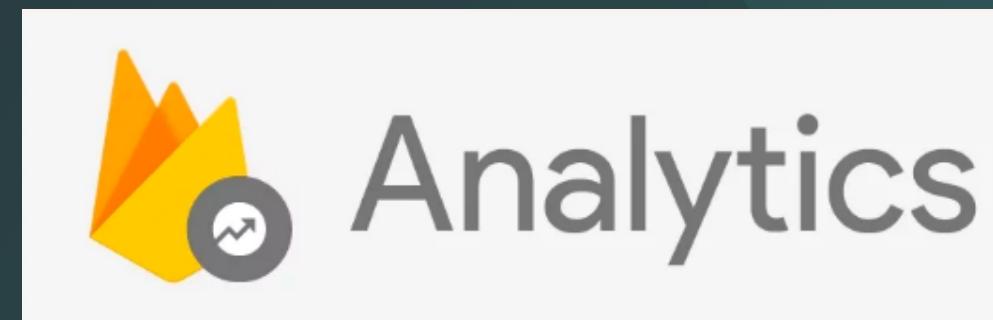
# Firebase Emulator Suite



<https://startupper.com/post/google-experience-support-releases-an-earthquake/>



# Firebase





# Firebase

Over 2.5 million apps  
actively using Firebase  
every month!

2018





# Firebase

Over 3 million apps  
actively using Firebase  
every month!



ANDROID



iOS

# 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.firebaseio.google.com](https://console.firebaseio.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.firebaseio: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.firebaseio:firebase-auth-ktx'  
    implementation 'com.google.firebaseio:firebase-firebase-ktx'  
}
```

# Available Libraries

firebase.inappmessaging.display

---

firebase.perf

---

[com.google.firebaseio.perf](#)

---

[com.google.firebaseio.perf.metrics](#)

---

firebase.remoteconfig

---

[com.google.firebaseio.remoteconfig](#)

---

firebase.storage

---

[com.google.firebaseio.storage](#)

---

Inter-operational packages

---

[com.google.firebaseio.auth.internal](#)

---

[com.google.firebaseio.ml.naturallanguage.languageid](#)

---

[firebase.google.com/docs/reference/android/packages](https://firebase.google.com/docs/reference/android/packages)

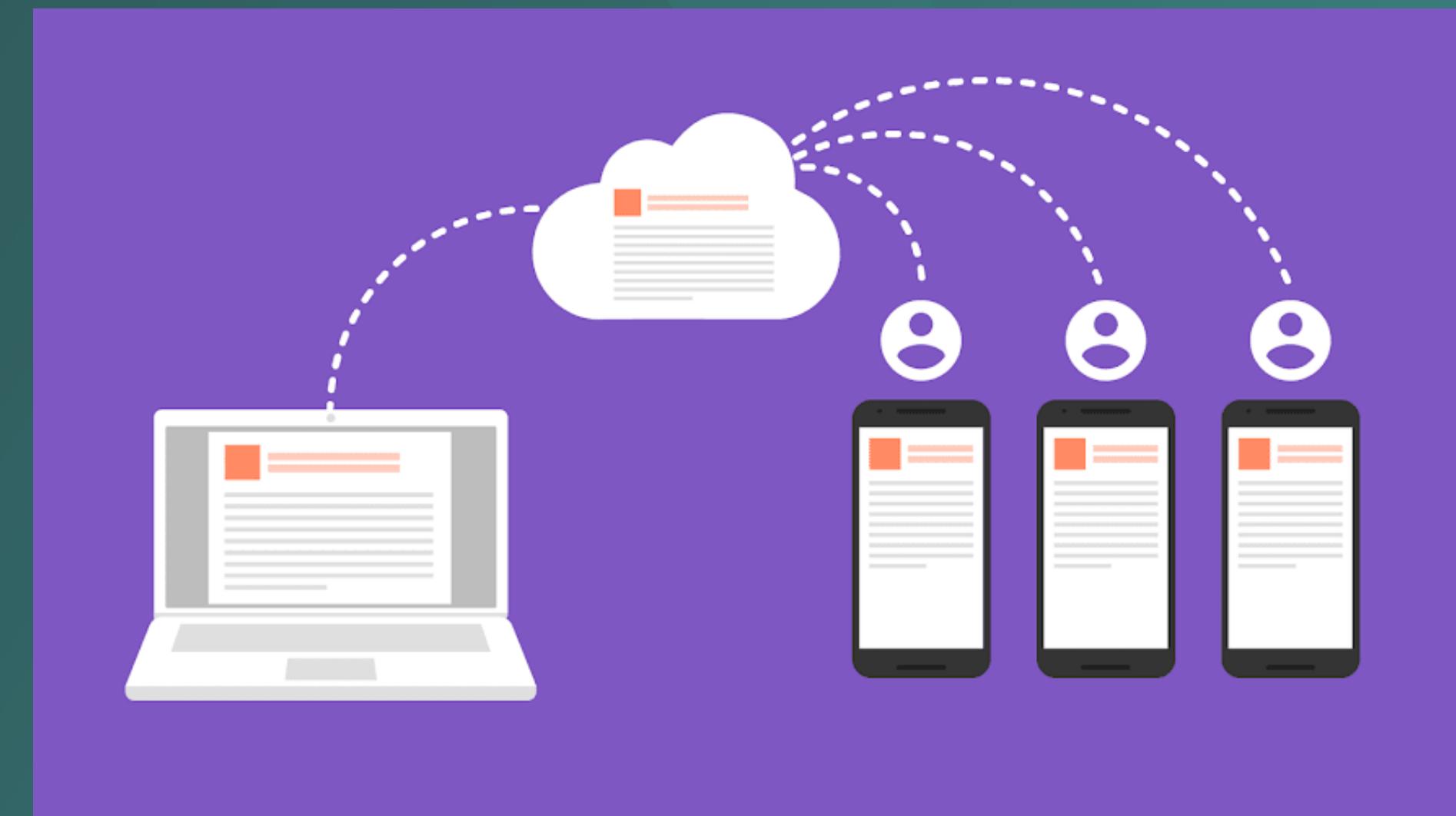
---

[com.google.firebaseio.ml.naturallanguage.smartreply](#)

---

# Realtime Database

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



# Installation & Setup

## Secure Your Data

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

implementation 'com.google.firebase:firebase-database-ktx'

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

# Data Access

```
// Read from the database
myRef.addValueEventListener(object : ValueEventListener {
    override fun onDataChange(dataSnapshot: DataSnapshot) {
        val myRef = dataSnapshot.getReference("message")
        // This method is called for every change in the database.
        // Whenever data at this location is updated.
        myRef.addValueListener(object : ValueEventListener {
            override fun onDataChange(dataSnapshot: DataSnapshot) {
                Log.d(TAG, "Value is: ${dataSnapshot.value}")
            }
            override fun onCancelled(error: DatabaseError) {
                // Failed to read value
                Log.w(TAG, "Failed to read value.", error.toException())
            }
        })
    }
})
```

<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);
```

# Enabling Offline Capabilities

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

Keeping Data Fresh

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

# Enabling Offline Capabilities

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

Keeping Data Fresh

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

# 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")
    }
})
```

# Enabling Offline Capabilities

DEMO

## 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")
    }
})
```

# 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

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

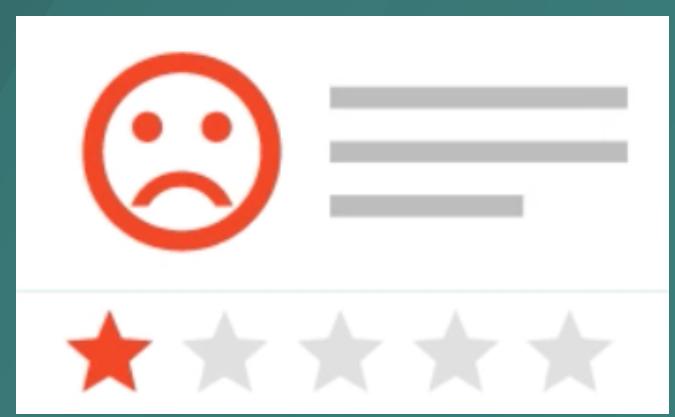
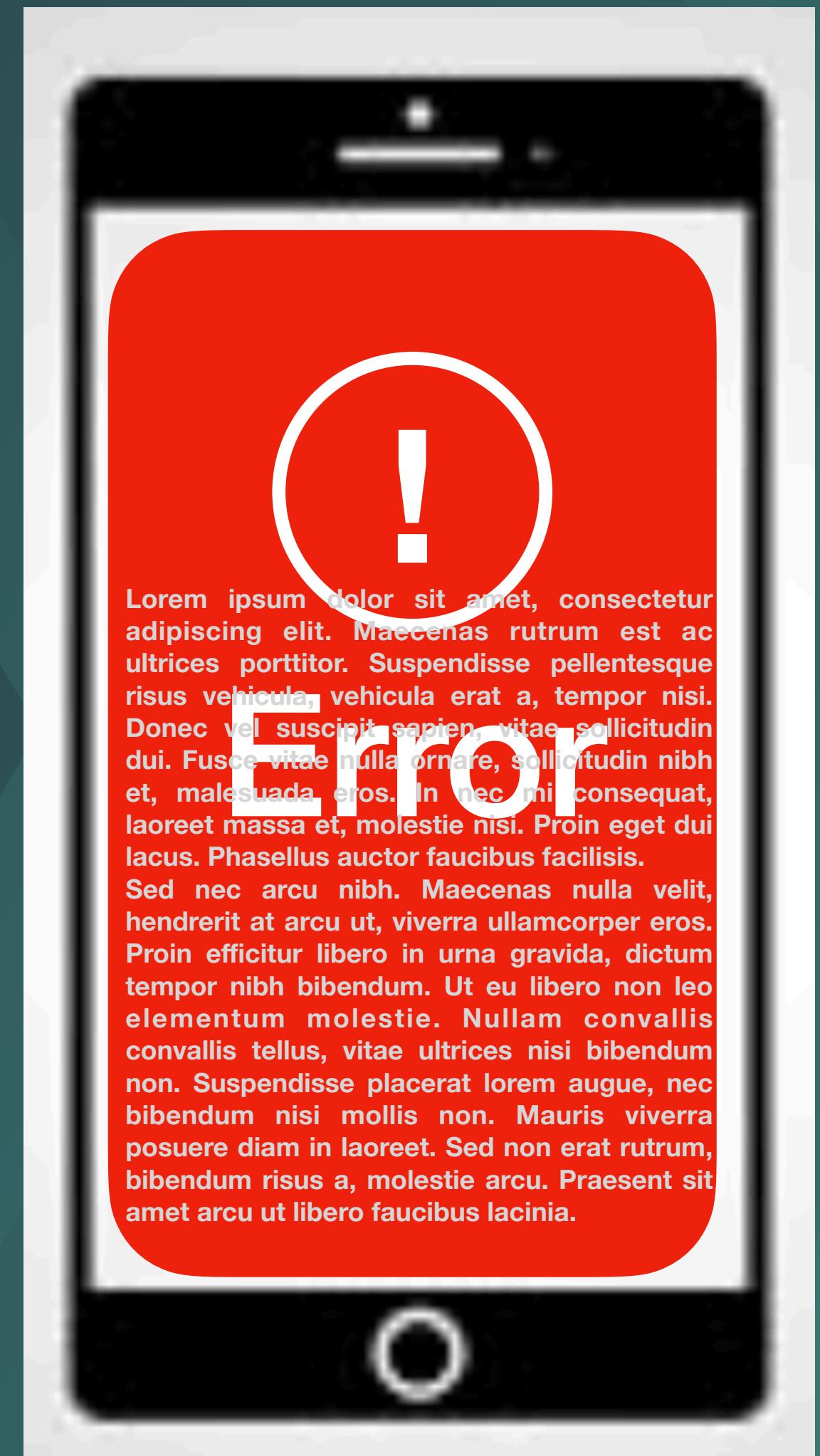
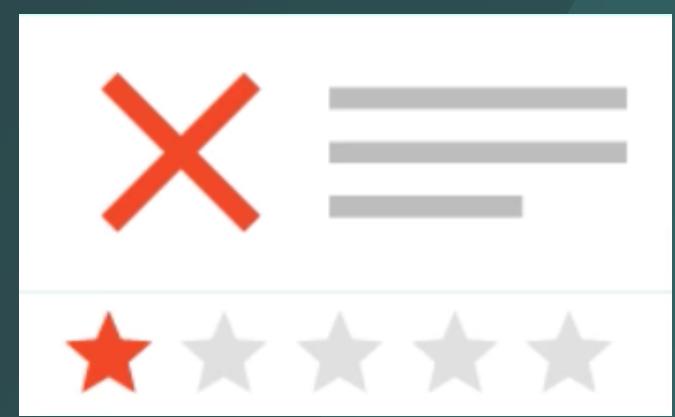
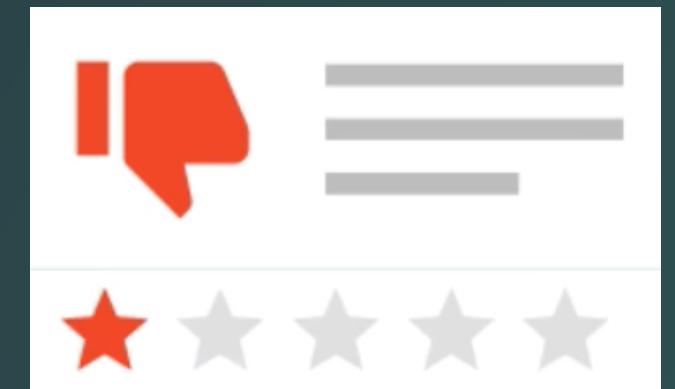
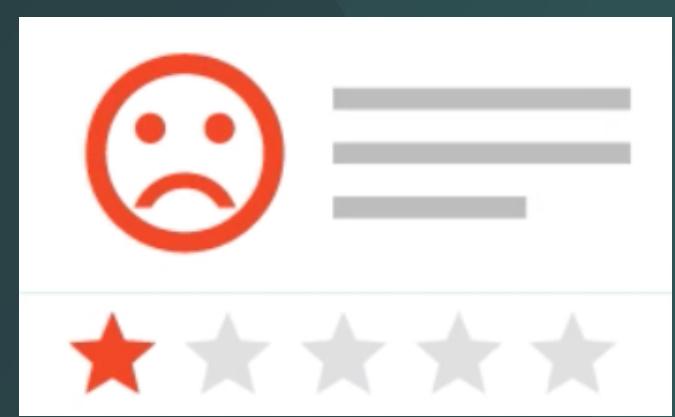
```
// Result returned from launching the Intent from  
  
private fun firebaseAuthWithGoogle(acct: GoogleSignInAccount) {  
    Log.d(TAG, "firebaseAuthWithGoogle:" + acct.idToken)  
    // Implementation com.google.firebase.auth.FirebaseAuth.kt  
    .requestIdToken(getString(R.string.default_web_client_id))  
    // Implementation com.google.android.gms.play-services-auth  
    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)
```

# Sign out a User

DEMO

`FirebaseAuth.getInstance().signOut()`

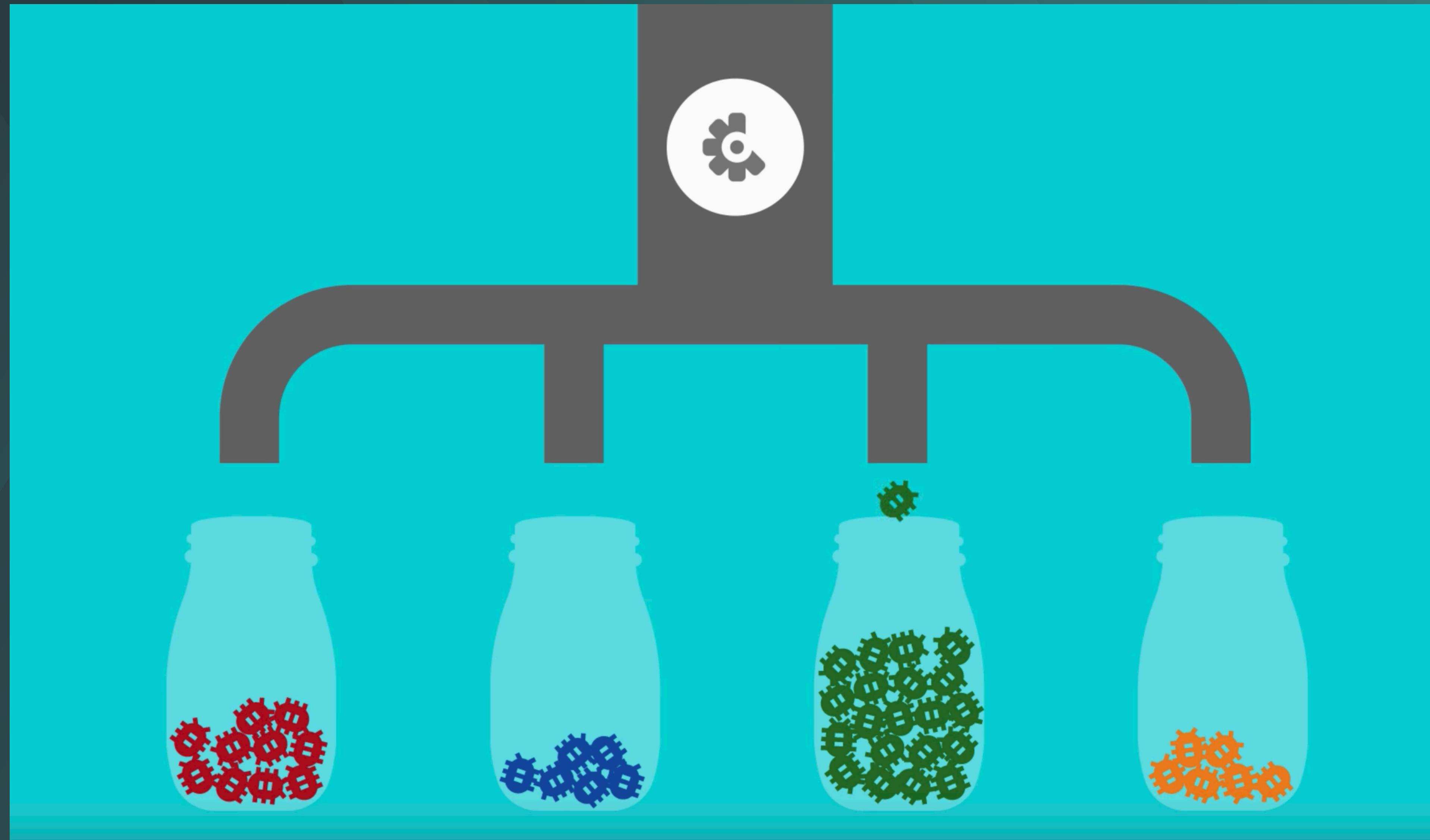








# Crashlytics



# 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.firebaseio: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.firebaseio.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

DEMO

Log non-fatal exceptions

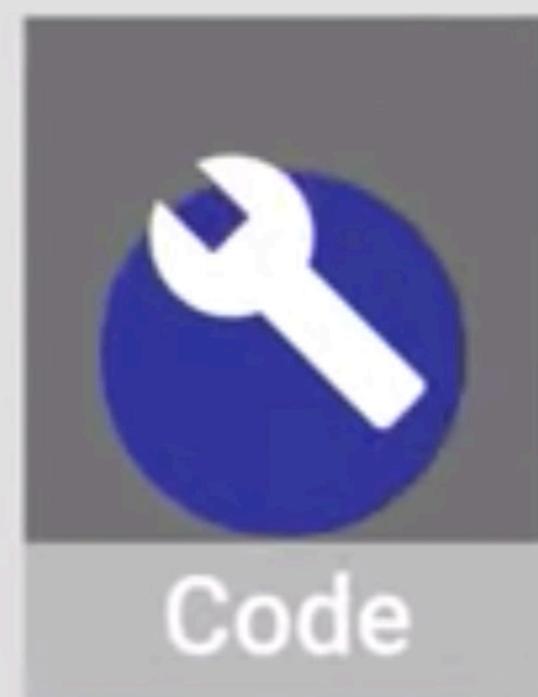
```
Crashlytics.logException(e)
```

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 */)
```



**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.**



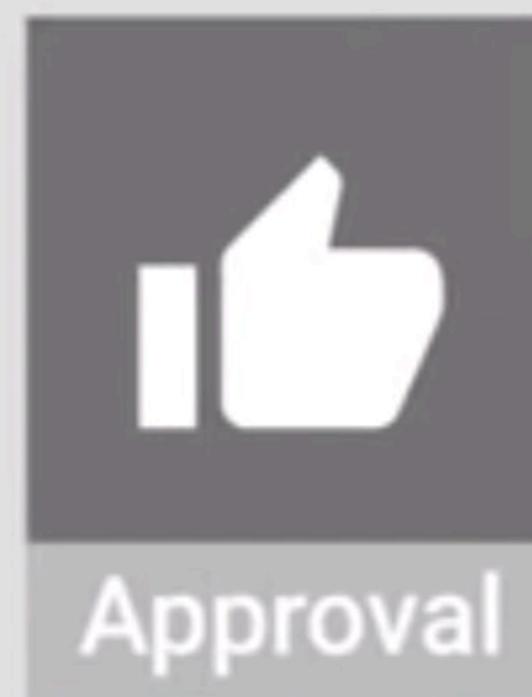
Code



Test



Submit

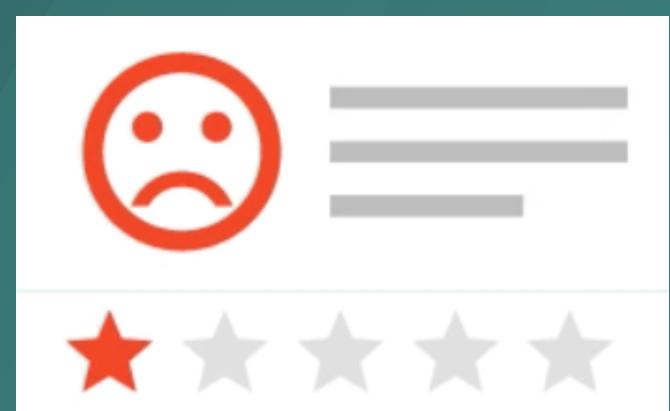
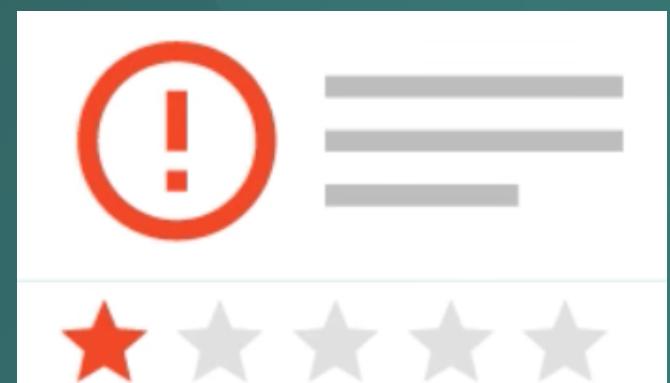
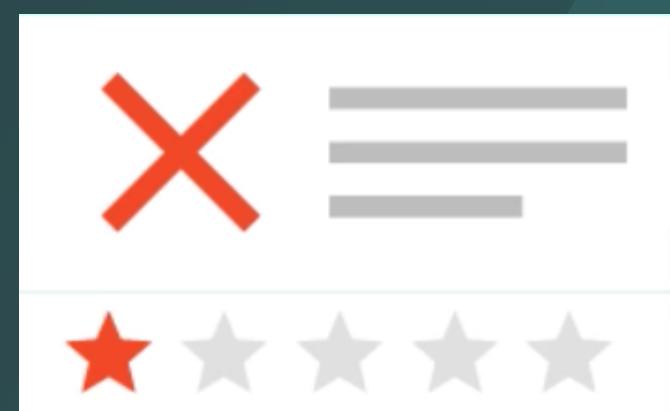
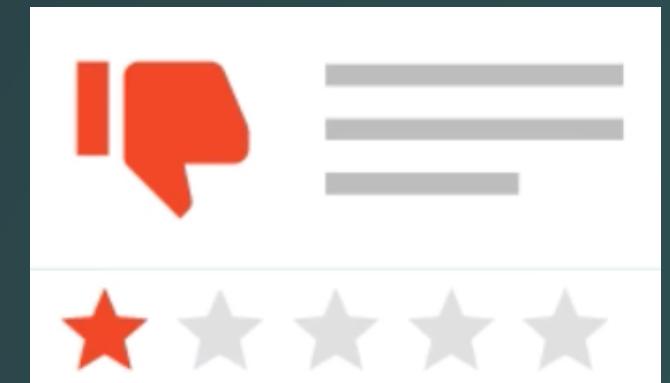
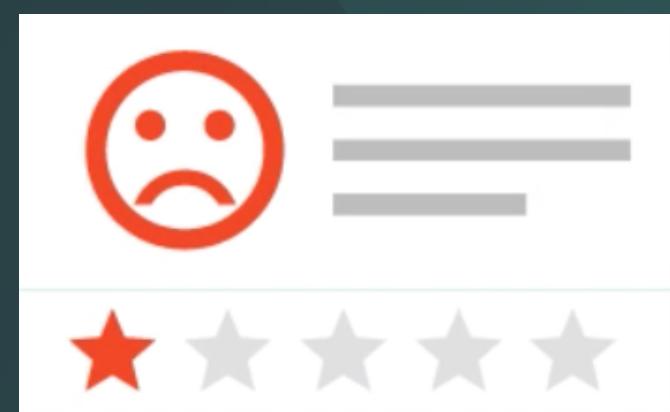


Approval

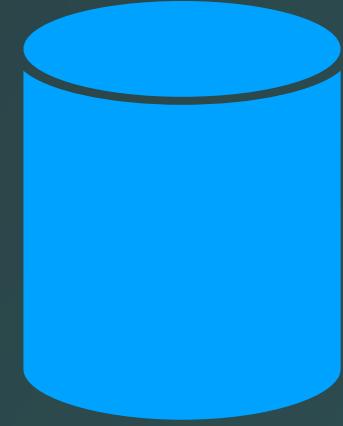


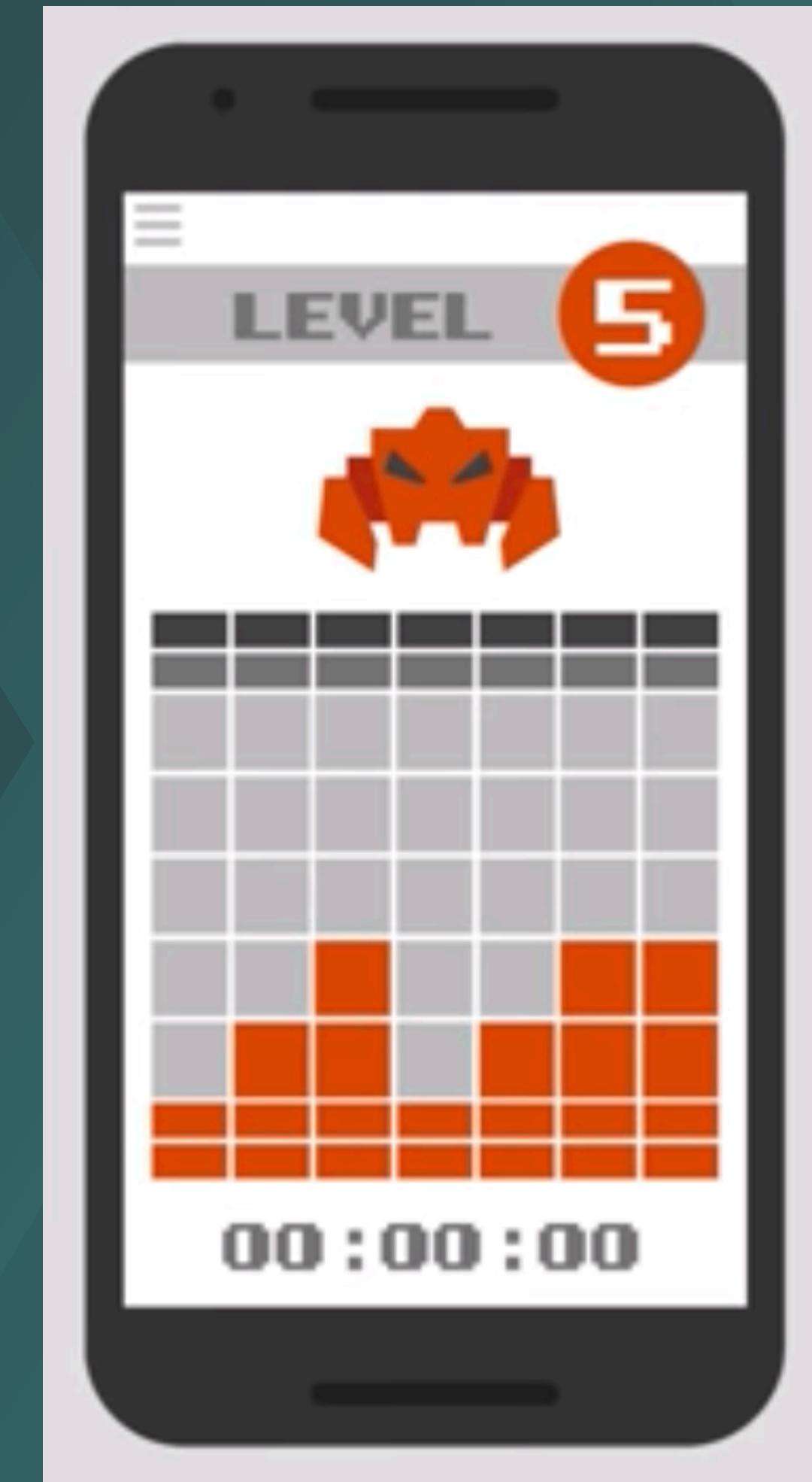
Release

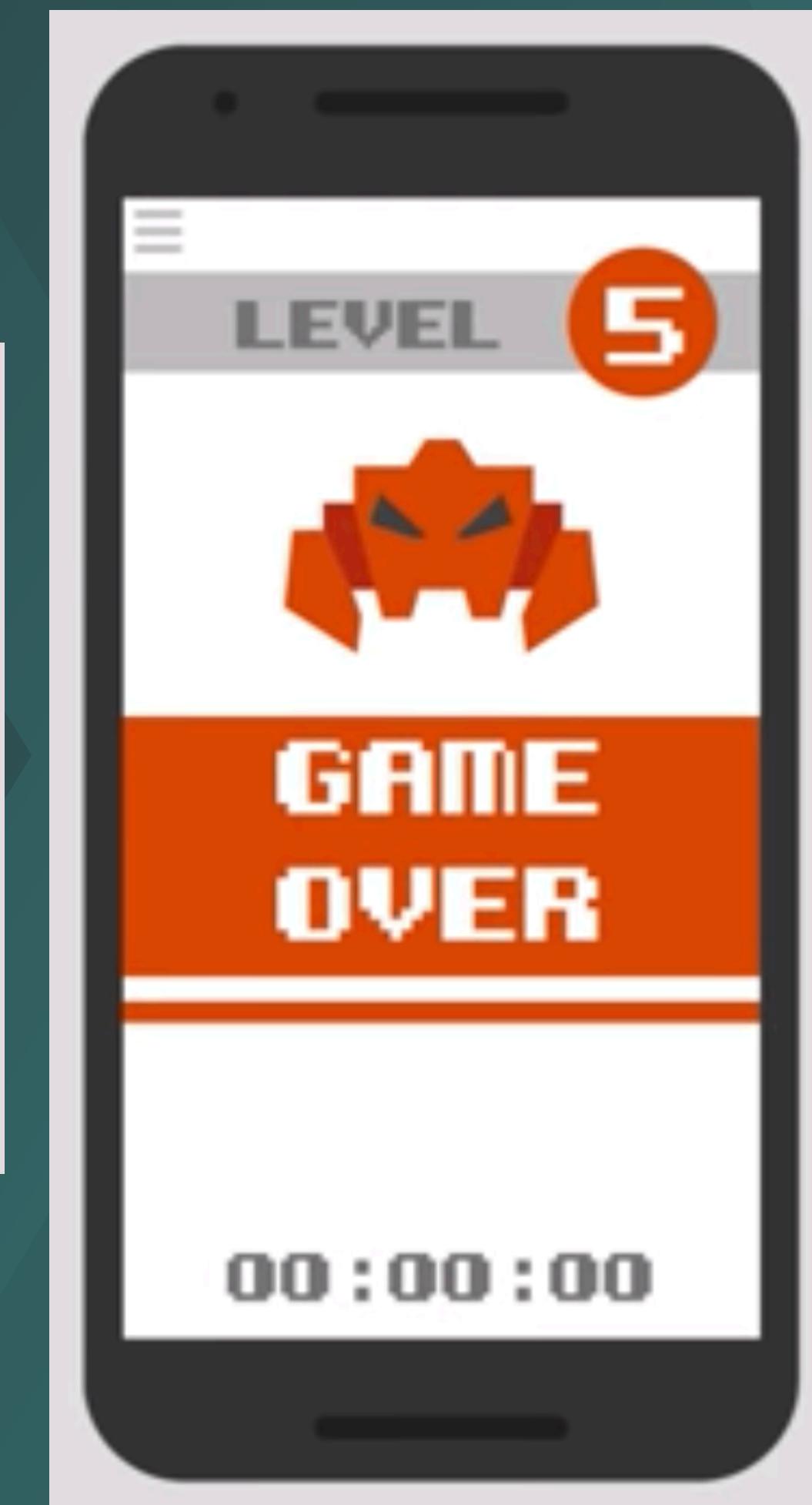
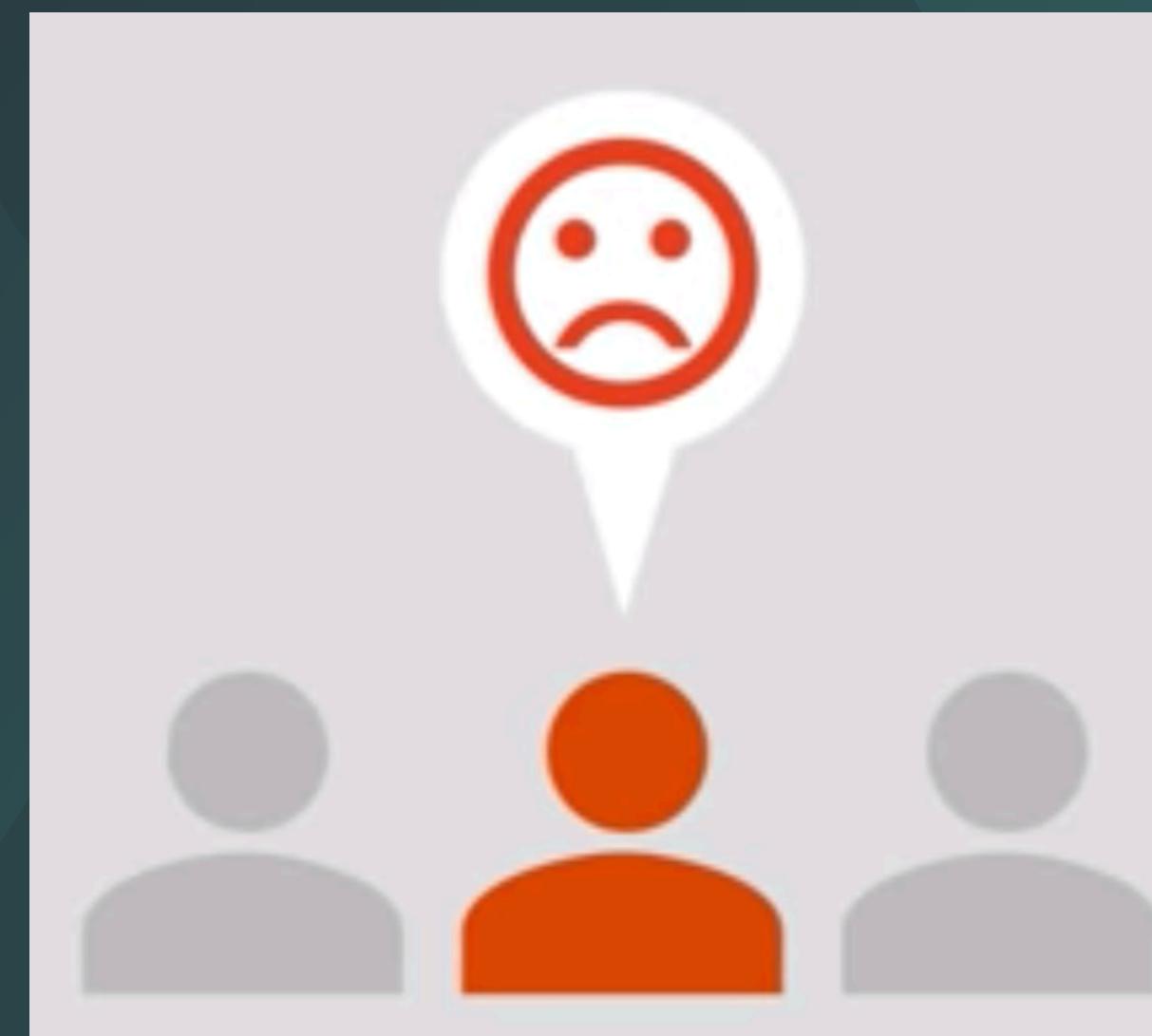




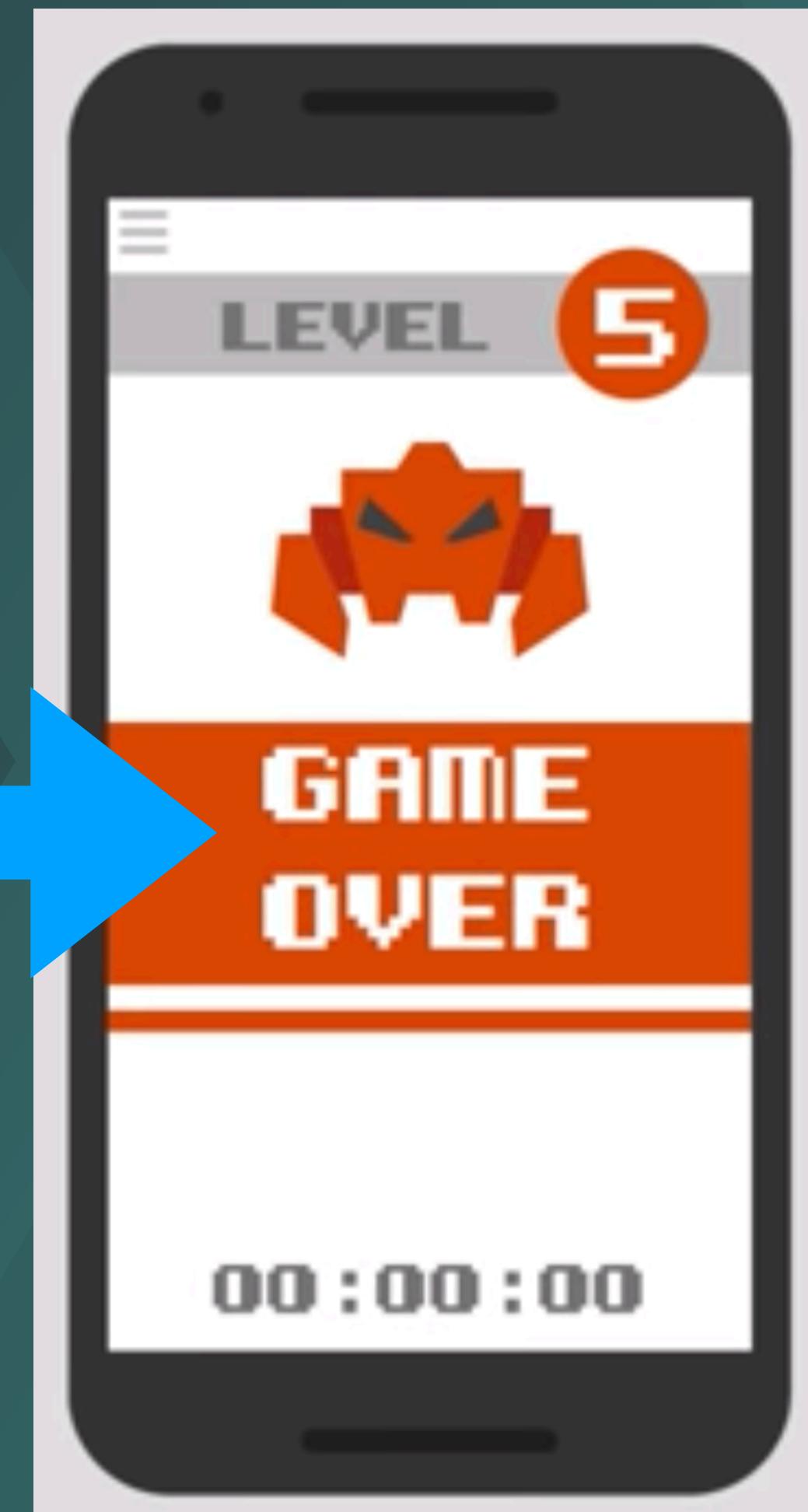
**Realtime**



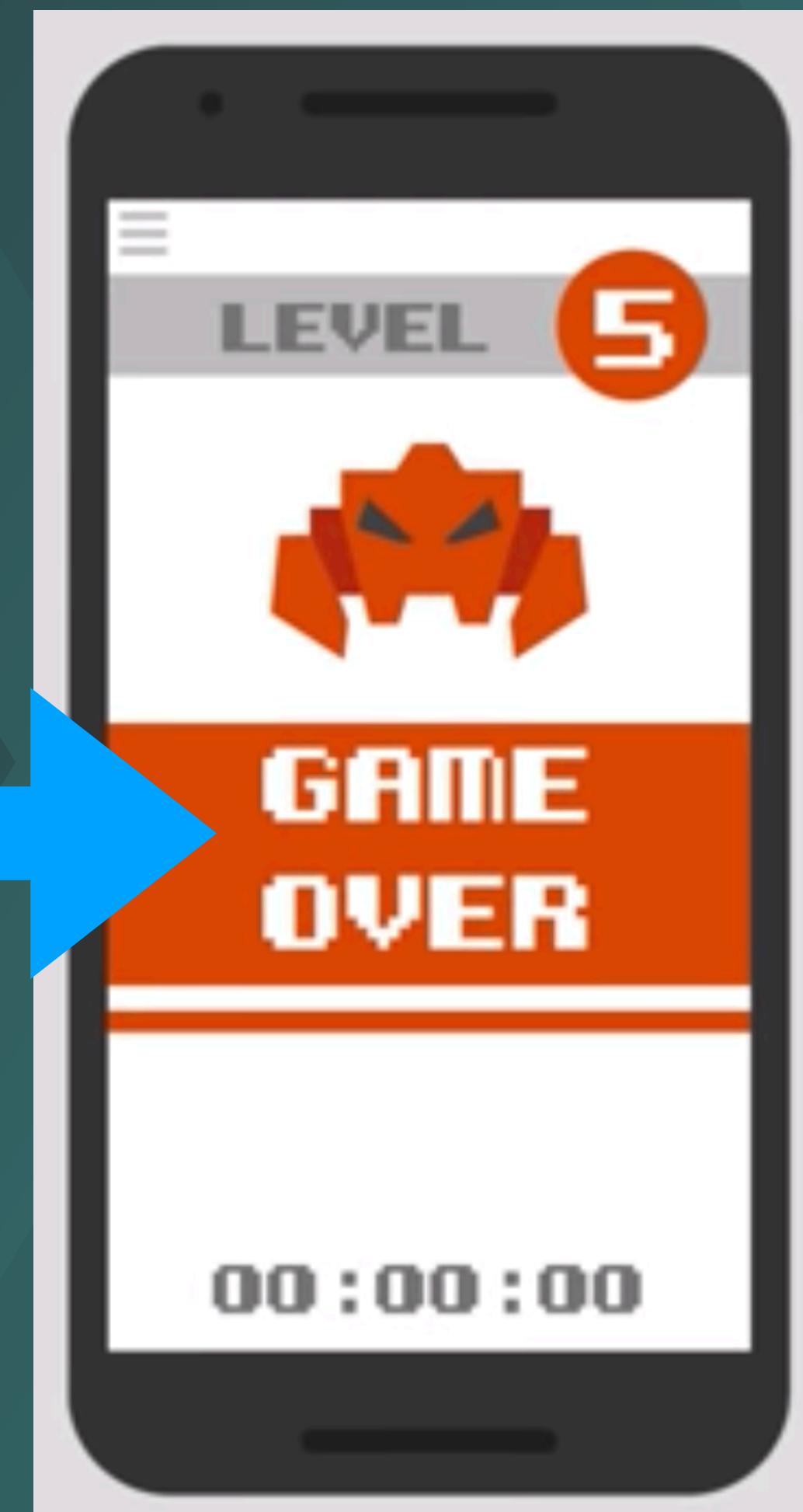
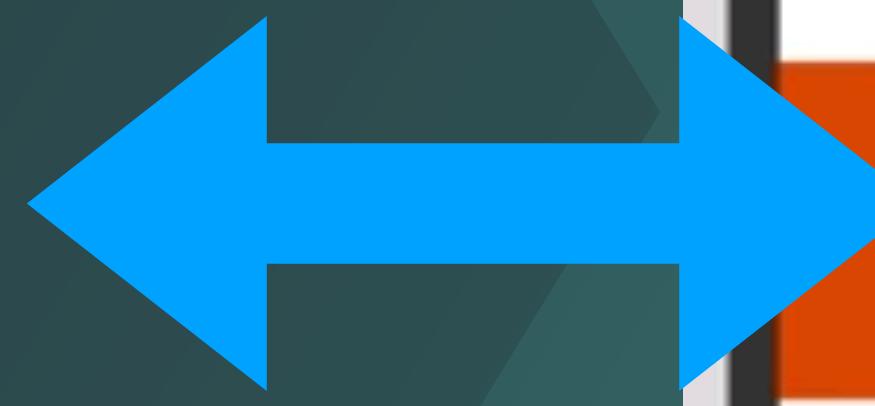
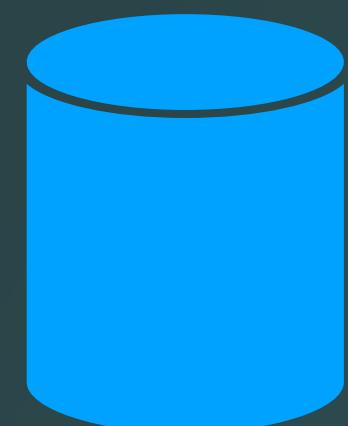


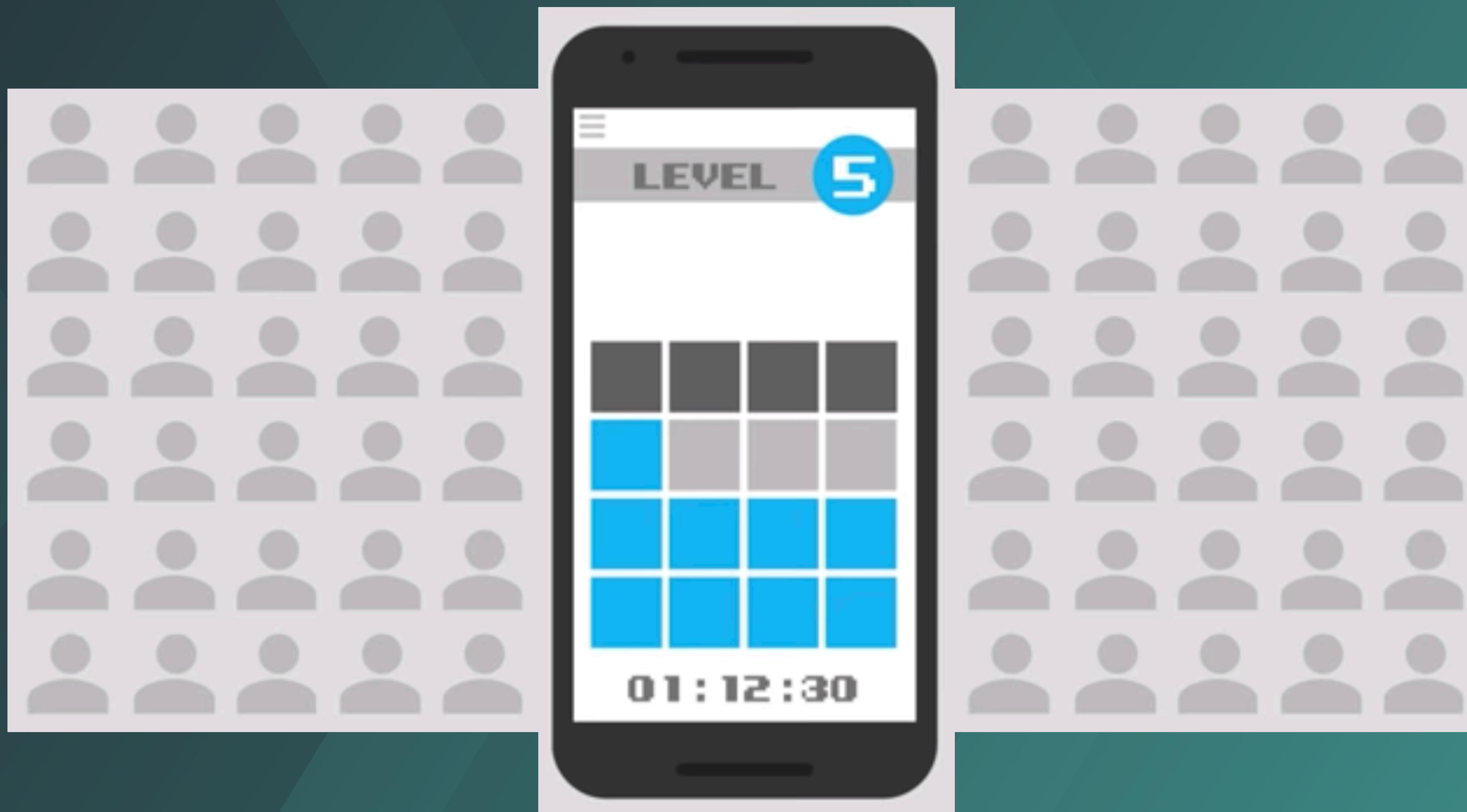


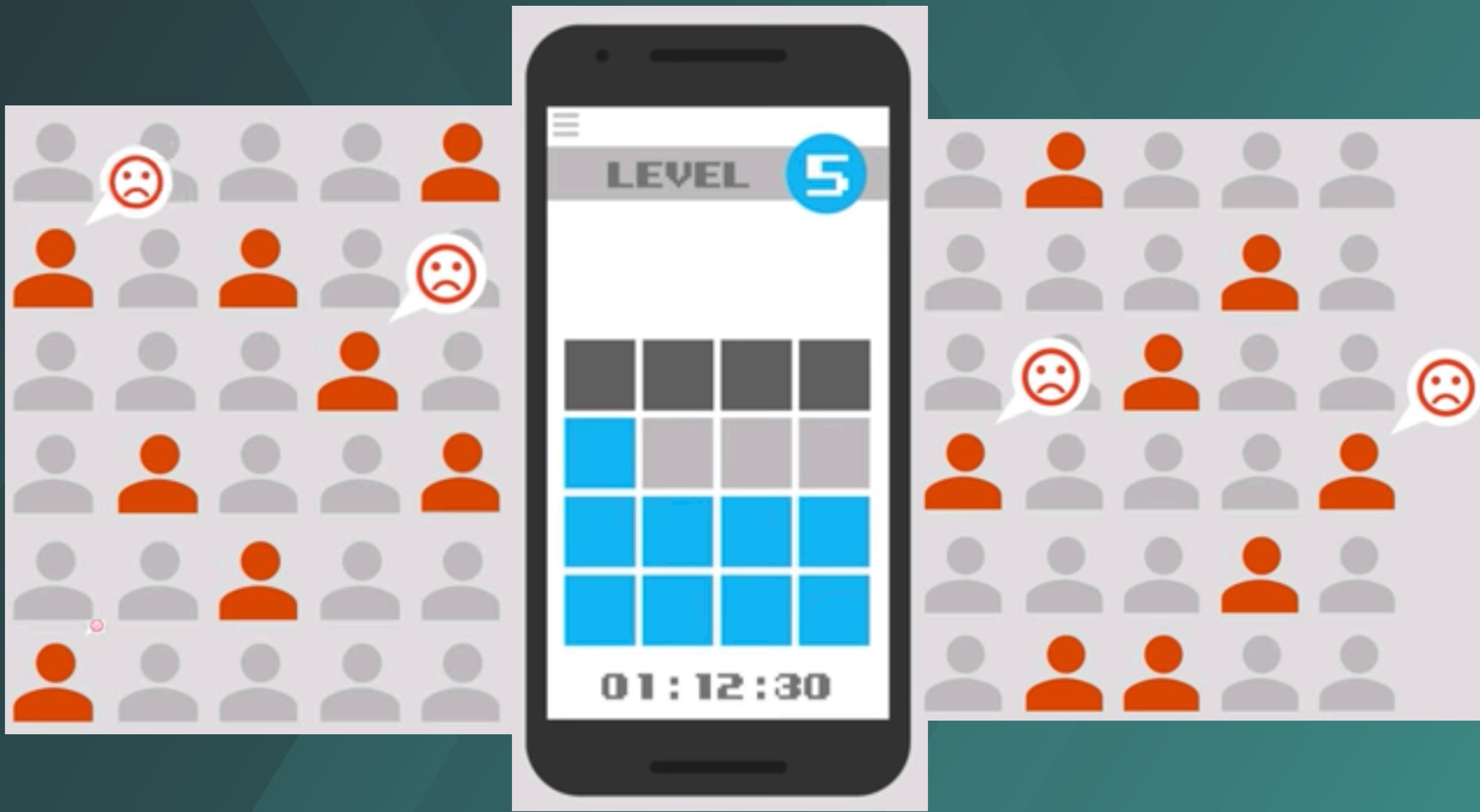
Realtime



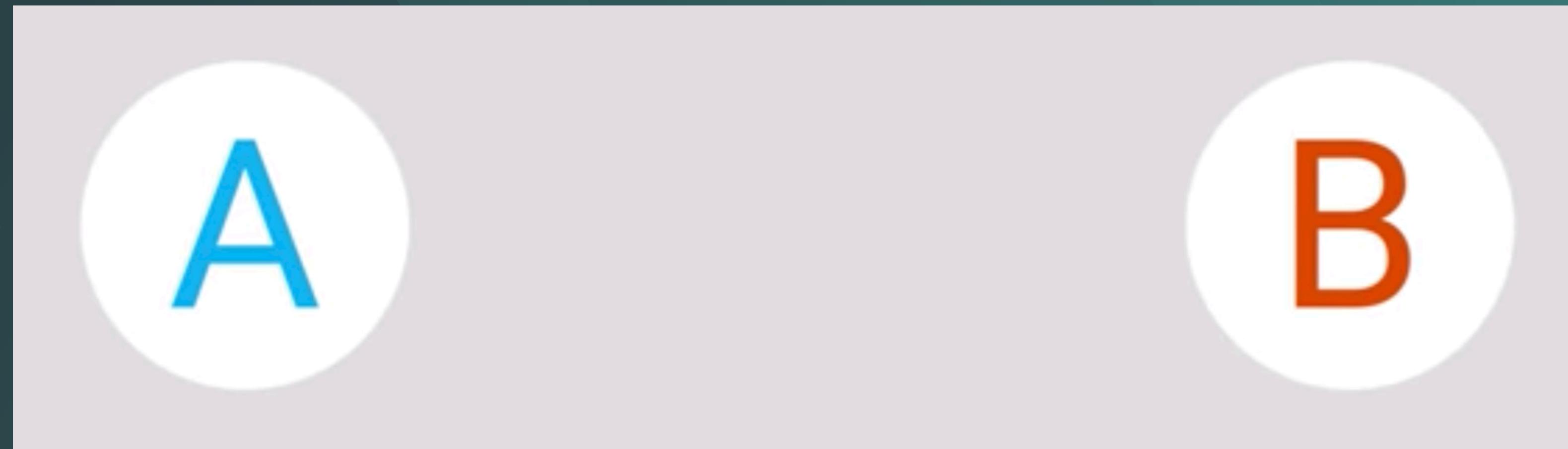
Realtime







# Remote Config



# Using Remote Config

DEMO

1

Add Firebase to your app

```
implementation 'com.google.firebaseio: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)

# Make \$

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



# AdMob

Implement AdMob in AndroidManifest.xml

```
<manifest>
    implementation 'com.google.android.gms:play-services-ads'
    <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]"/>           Initialize the SDK
    </application>
</manifest>
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    // Sample AdMob app ID: ca-app-pub-3940256099942544~3347511713
    MobileAds.initialize(this)
}
```

# Ad Formats

- Banner
- Interstitial

```
class MainActivity : Activity() {  
    • Native <com.google.android.gms.ads.AdView  
val adLoader = AdLoader.Builder("http://schemas.android.com/apk/res/m  
    • InterstitialAd mInterstitialAd: InterstitialAd ->  
    import com.google.android.gms.ads.AdRequestContent"  
    override fun onCreate(savedInstanceState: Bundle?) {  
    import com.google.android.gms.ads.MobileAds  
    super.onCreate(savedInstanceState) mAppContent  
import com.google.android.gms.ads.reward.RewardedVideoAd  
    .setContentView(R.layout.activity_main)  
    override android.widget.LinearLayoutParentBottom: Int  
    class MainActivity : AppCompatActivity() RewardVideoAdListener {  
        MobileAds.initialize(this, logging, altering the UI, and so on.  
    } "ca-app-pub-394025609942544/6300978111"  
private lateinit var mRewardedVideoAd: RewardedVideoAd  
    .withNativeAdOptions(NativeAdOptions.Builder()  
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
        mInterstitialAd.adUnitId = "ca-app-pub-394025609942544/1033173712"  
        super.onViewCreated(view, savedInstanceState) TANNER  
        mInterstitialAd.loadAd(AdRequest.Builder().build())  
    } setContenViewAndLayout.activity_main  
} .build()  
MobileAds.initialize(this, "ca-app-pub-394025609942544~3347511713")  
https://play.google.com/store/apps/details?id=com.google.samples.gwt.admob&hl=en
```



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.



# 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.

