

Lecture #9

Firebase Services

Mobile Applications 2020-2021

twitch.tv/dancojocar

youtube.com/dancojocar



James Tamplin



Andrew Lee

twitch.tv/dancojocar

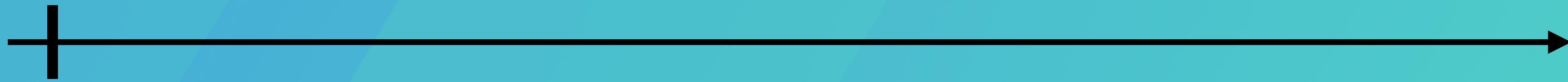
youtube.com/dancojocar



James Tamplin



Andrew Lee



2007



James Tamplin



Andrew Lee

**Various
projects**



2007



James Tamplin



Andrew Lee





James Tamplin



Andrew Lee

**Various
projects**





James Tamplin



Andrew Lee

**Various
projects**





James Tamplin



Andrew Lee

**Various
projects**





James Tamplin



Andrew Lee

Various
projects





James Tamplin



Andrew Lee

Various
projects



<https://startupandrew.com/posts/my-experiences-in-the-haitian-earthquake/>

twitch.tv/dancojocar

youtube.com/dancojocar



James Tamplin



Andrew Lee

**Various
projects**



twitch.tv/dancojocar

youtube.com/dancojocar



James Tamplin



Andrew Lee

Various
projects



twitch.tv/dancojocar

youtube.com/dancojocar



James Tamplin



Andrew Lee

Various
projects



twitch.tv/dancojocar

youtube.com/dancojocar



James Tamplin



Andrew Lee

Various
projects





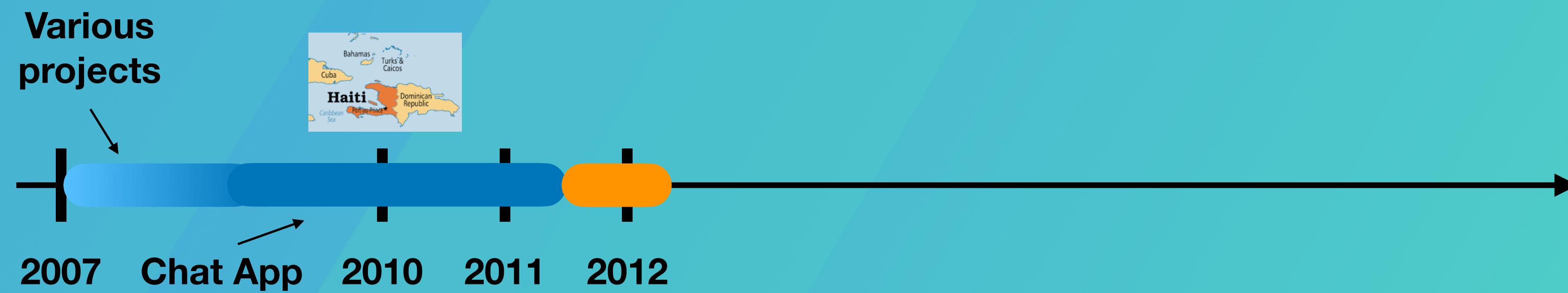
Firebase



James Tamplin



Andrew Lee





Firebase

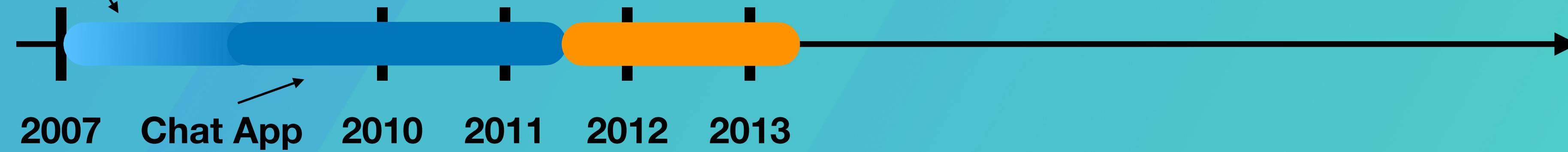


James Tamplin



Andrew Lee

Various
projects





Firebase

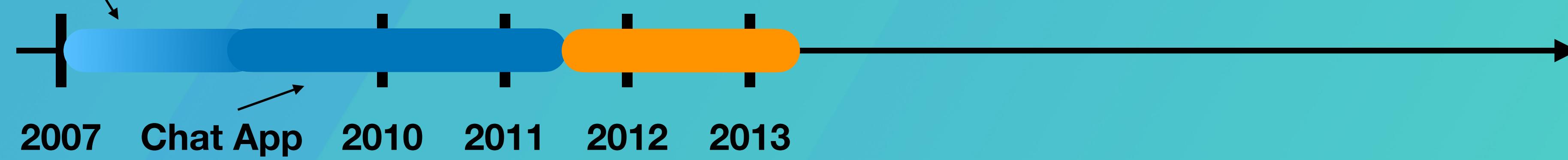


James Tamplin



Andrew Lee

Various
projects





Firebase

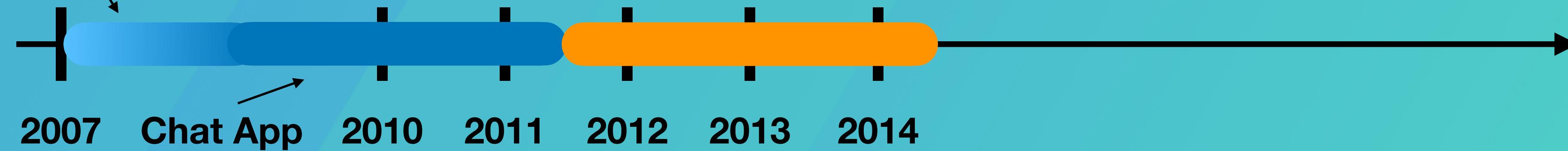


James Tamplin



Andrew Lee

Various
projects



twitch.tv/dancojocar

youtube.com/dancojocar



Firebase



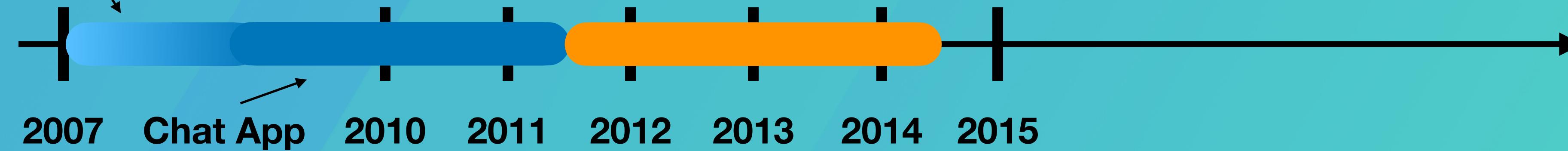
James Tamplin



Andrew Lee



Various
projects



twitch.tv/dancojocar

youtube.com/dancojocar



Firebase



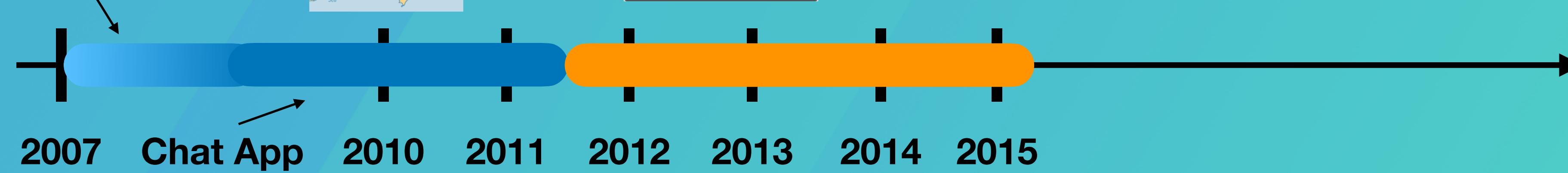
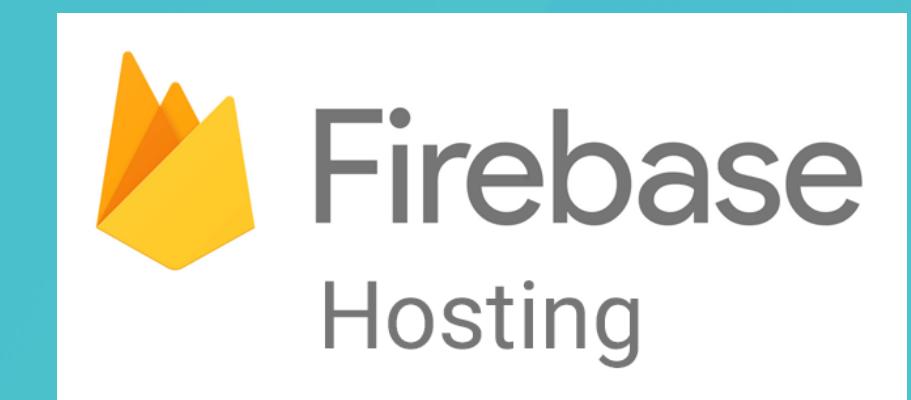
James Tamplin



Andrew Lee



Various
projects



twitch.tv/dancojocar

youtube.com/dancojocar

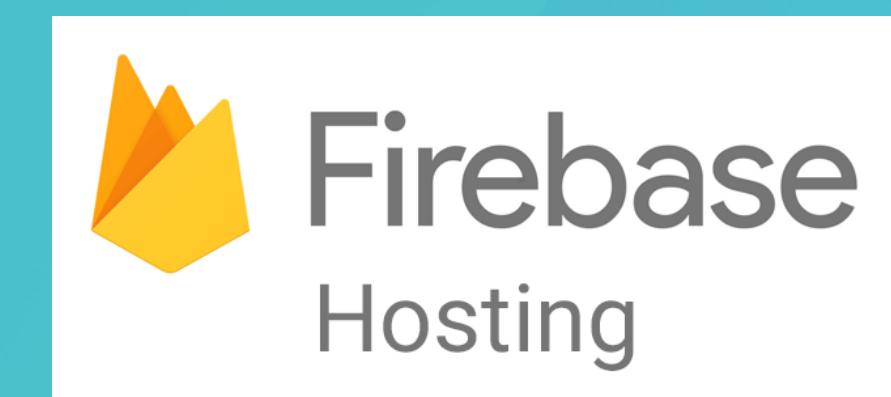
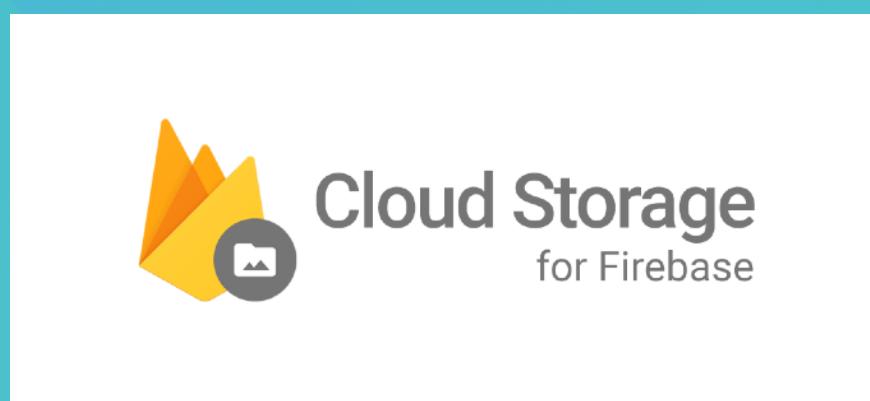
Firebase



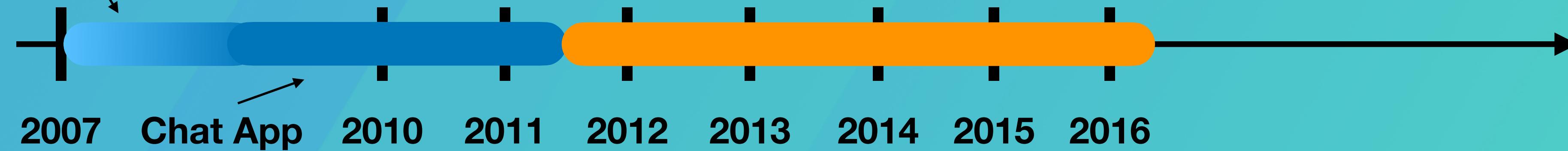
James Tamplin



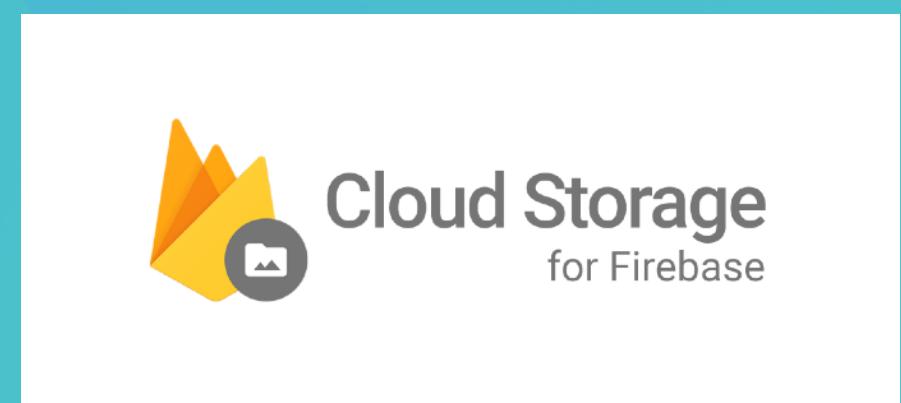
Andrew Lee



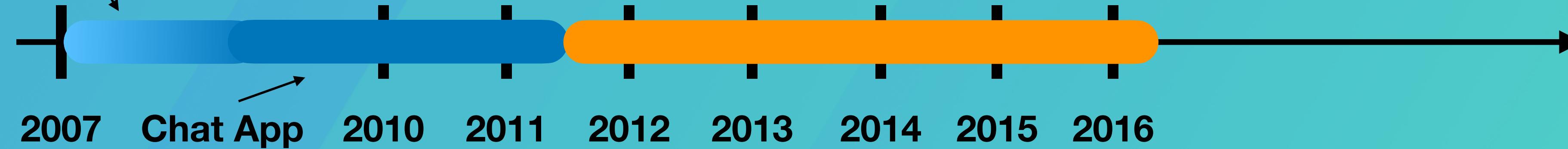
Various
projects



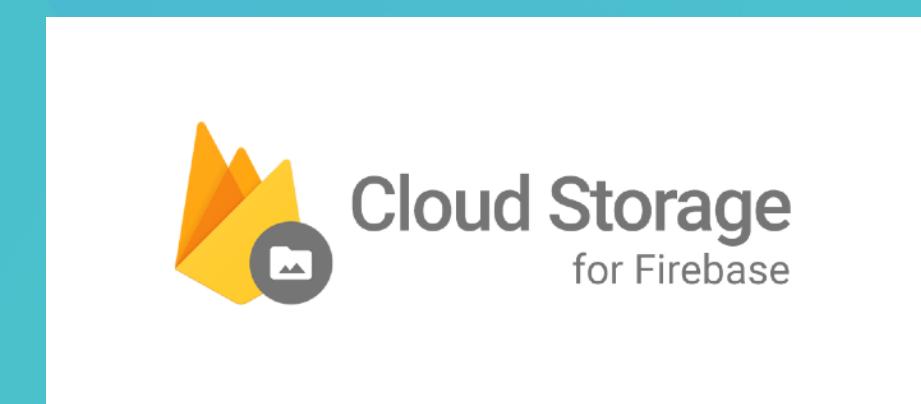
Firebase



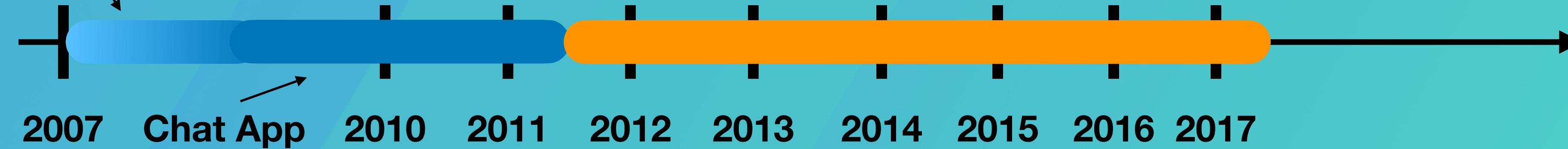
Various
projects



Firebase



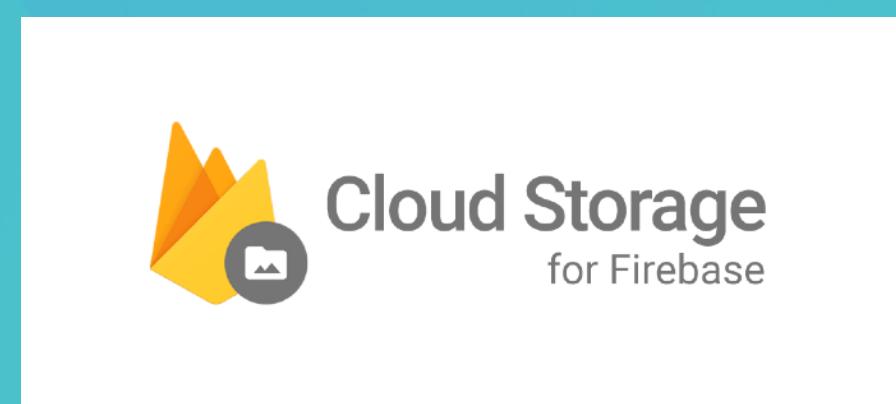
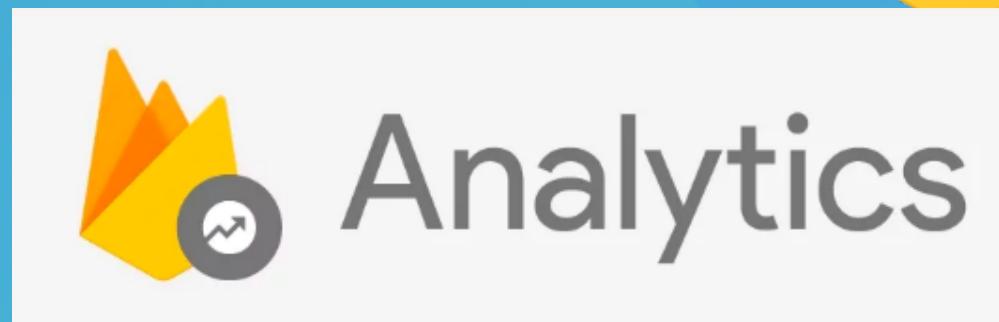
Various
projects



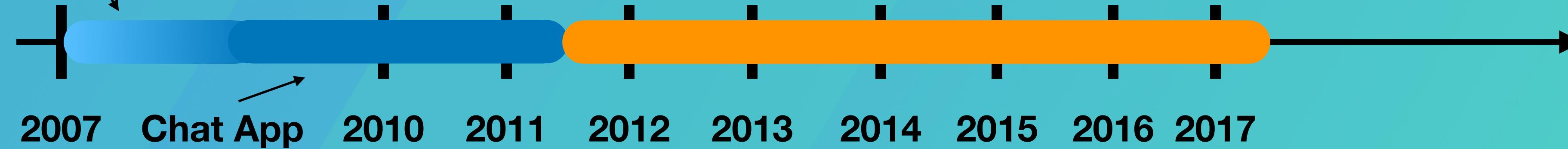
twitch.tv/dancojocar

youtube.com/dancojocar

Firebase



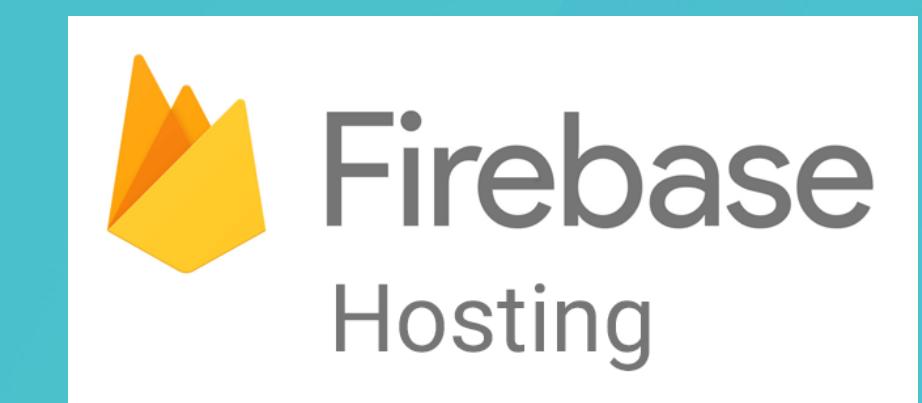
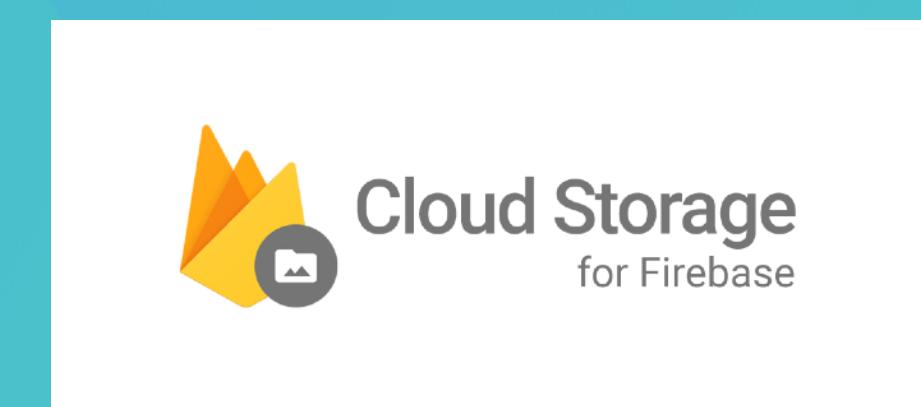
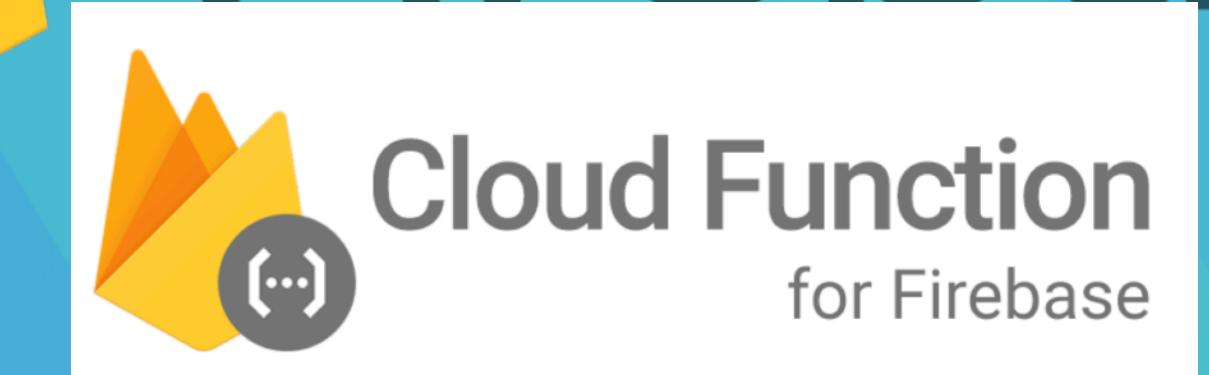
Various
projects



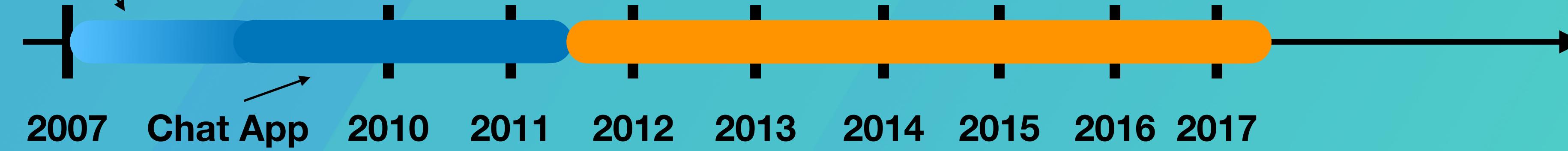
twitch.tv/dancojocar

youtube.com/dancojocar

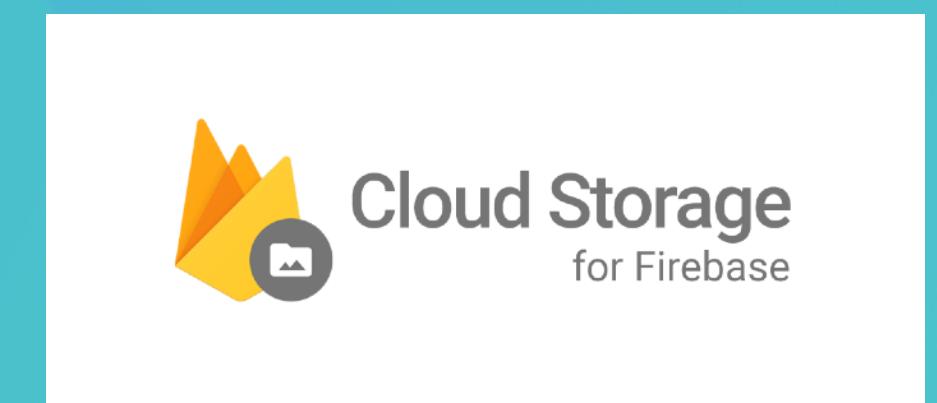
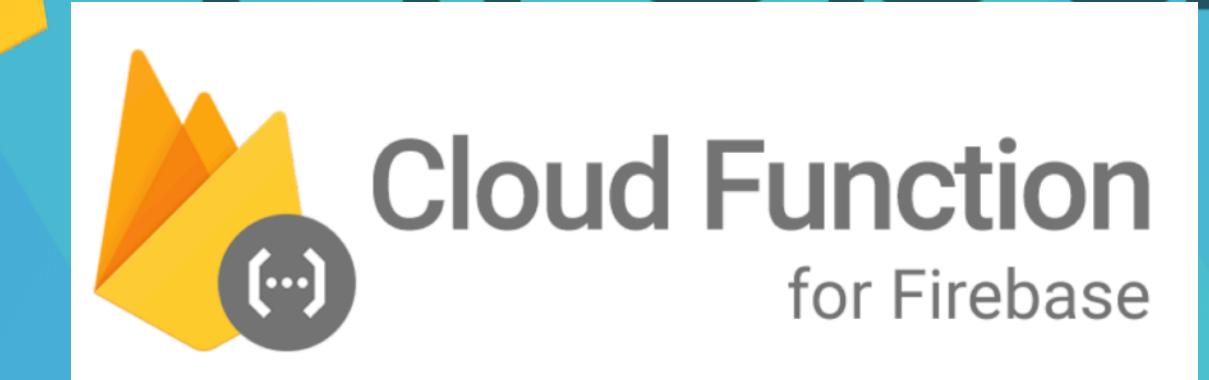
Firebase



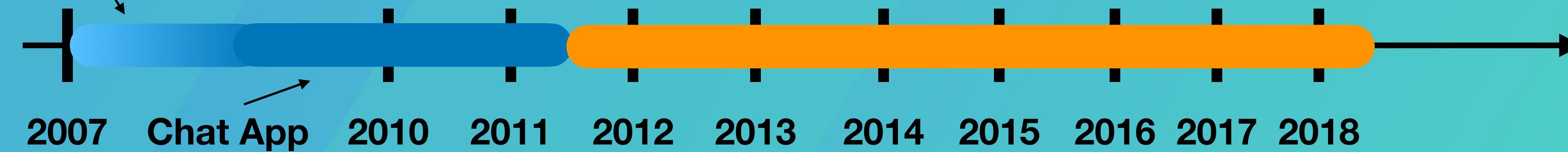
Various
projects



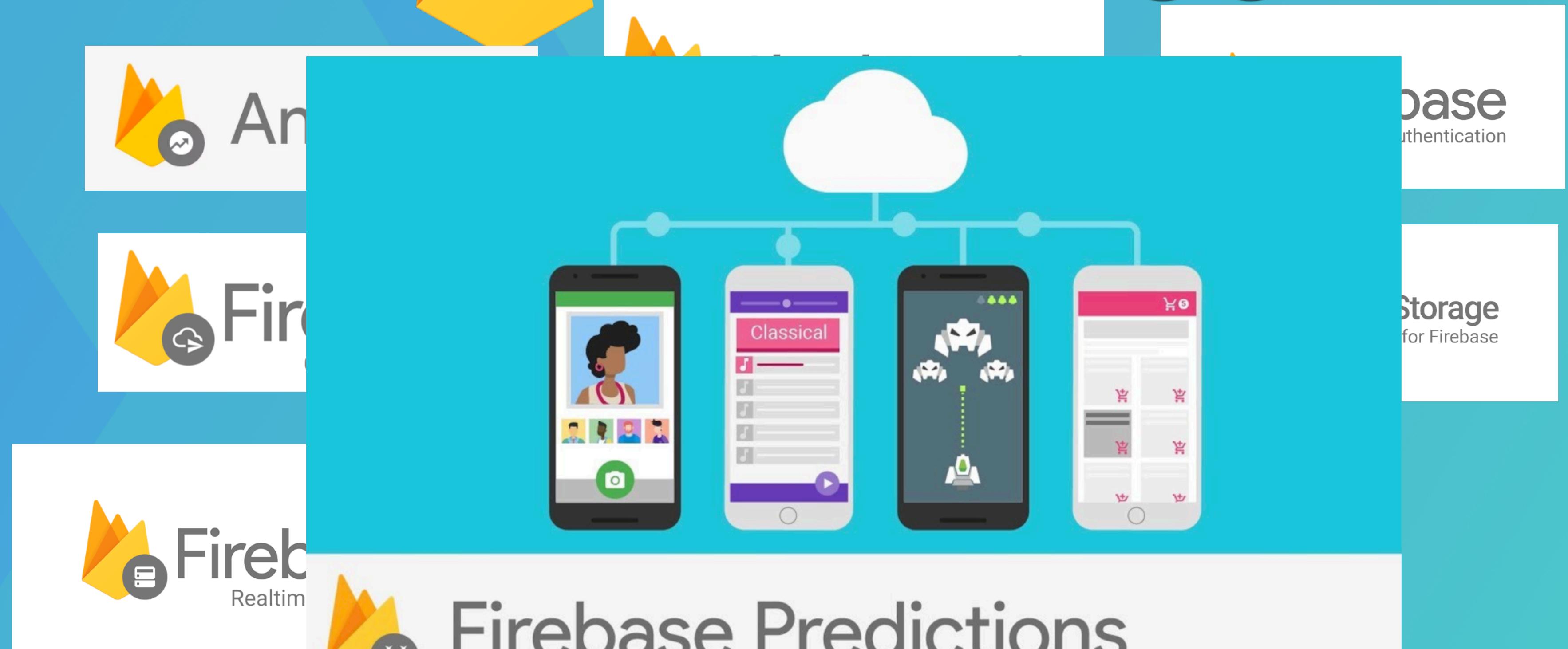
Firebase



Various
projects



Firebase



Various
projects

2007

Chat App

2010

2011

2012

2013

2014

2015

2016

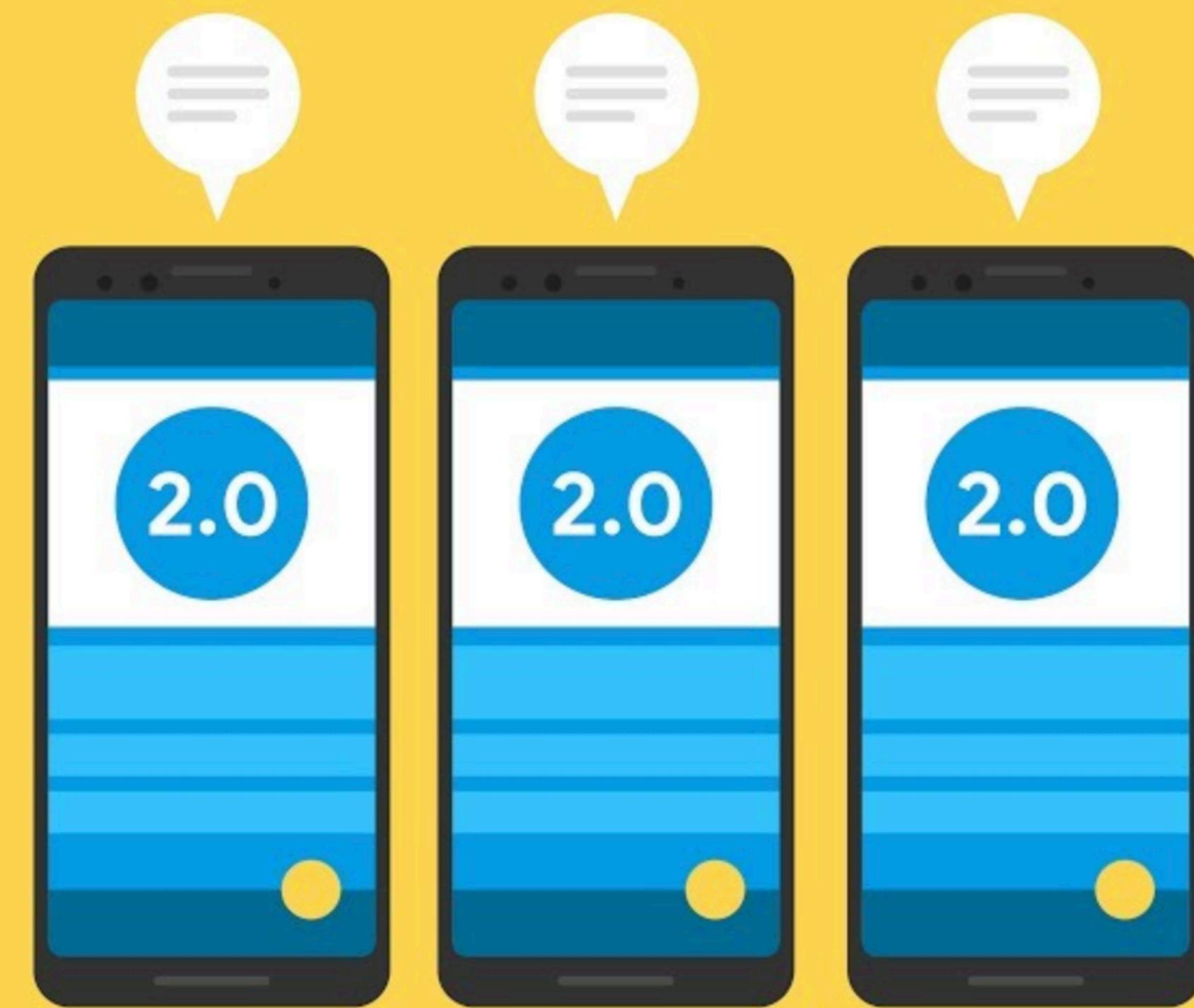
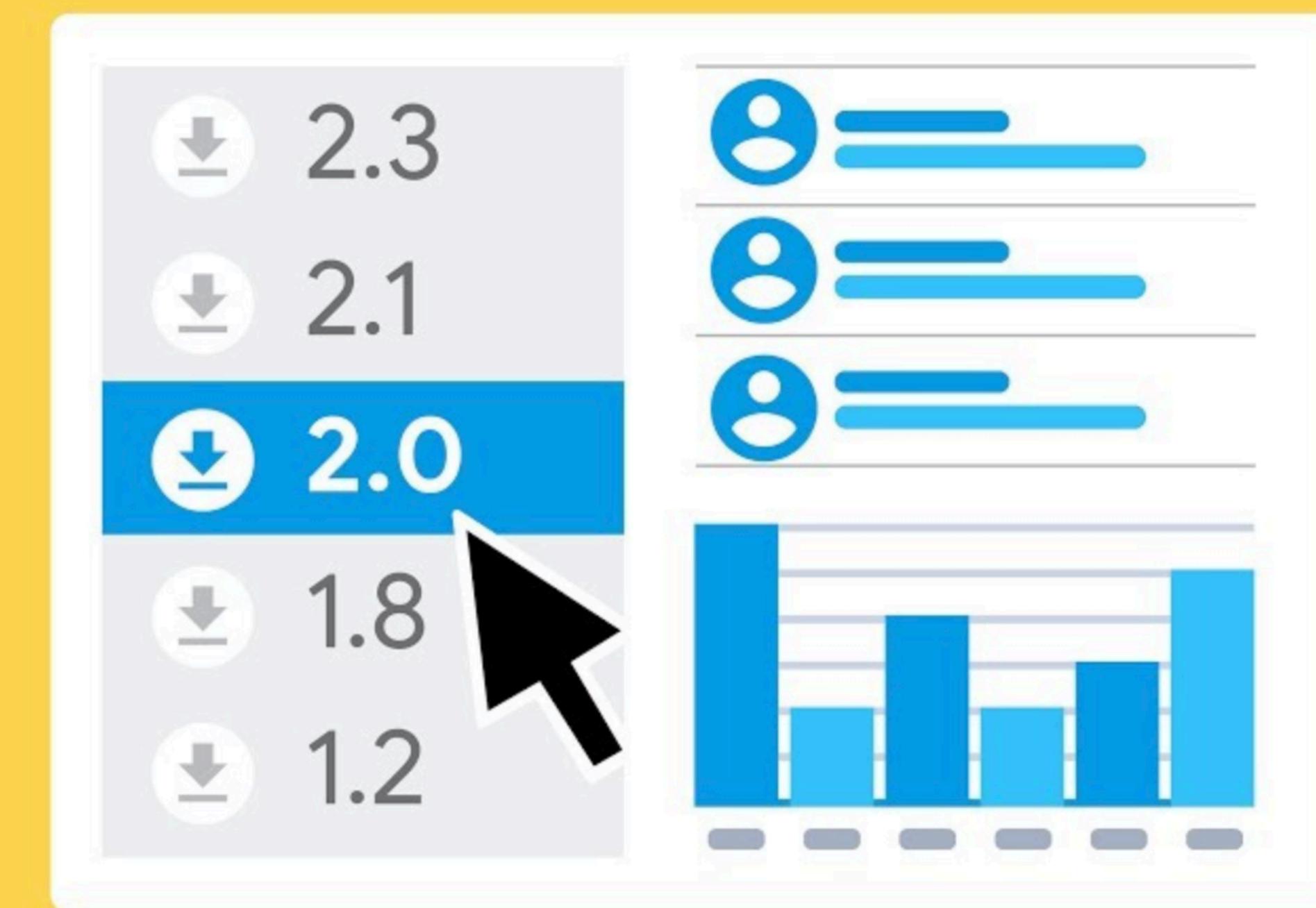
2017

2018

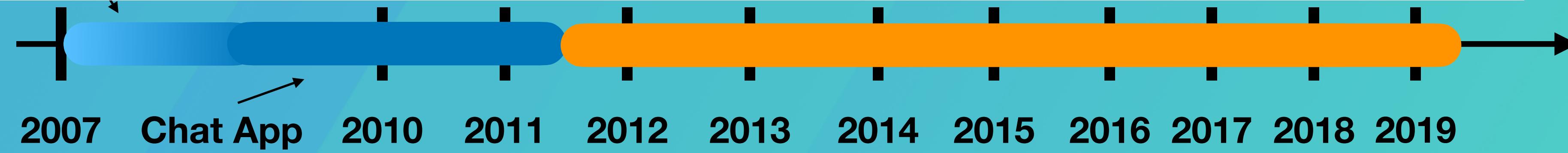
2019

Built with
Firebase

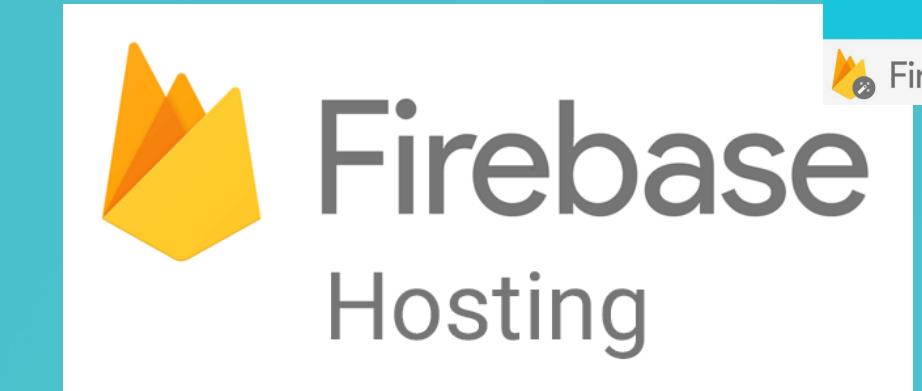
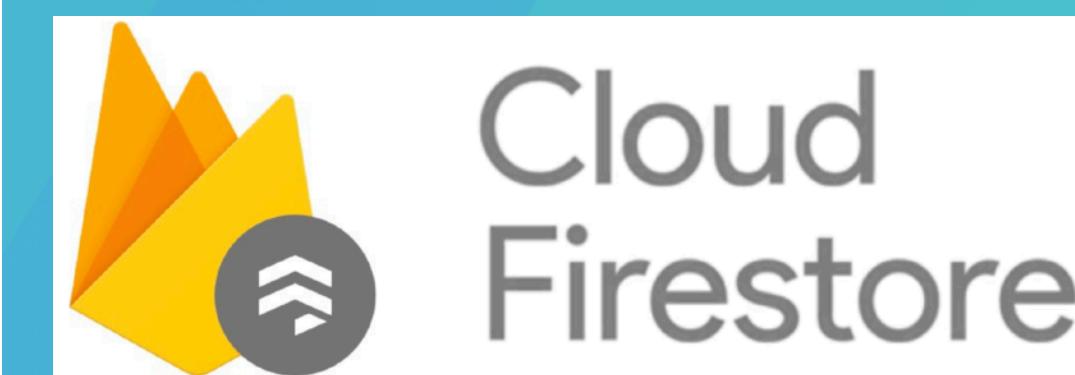
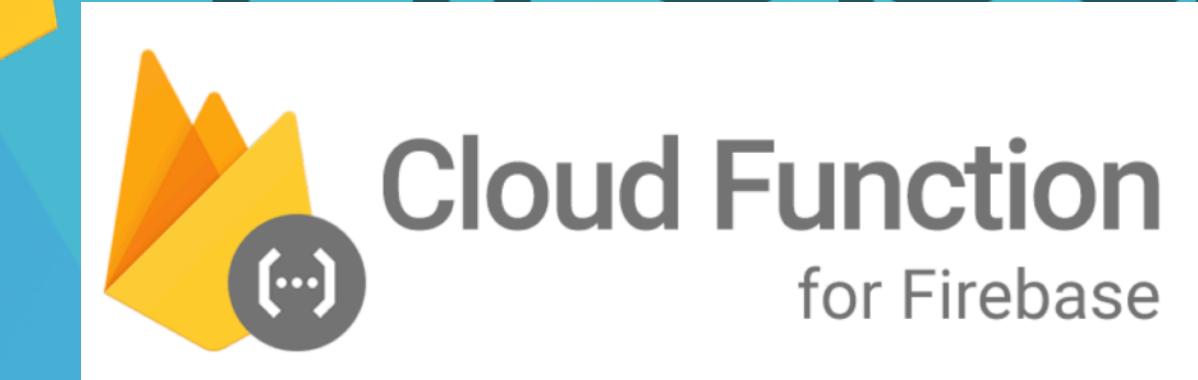
ML Kit
for Firebase



Firebase App Distribution



Firebase



Various
projects



2007

Chat App

2010

2011

2012

2013

2014

2015

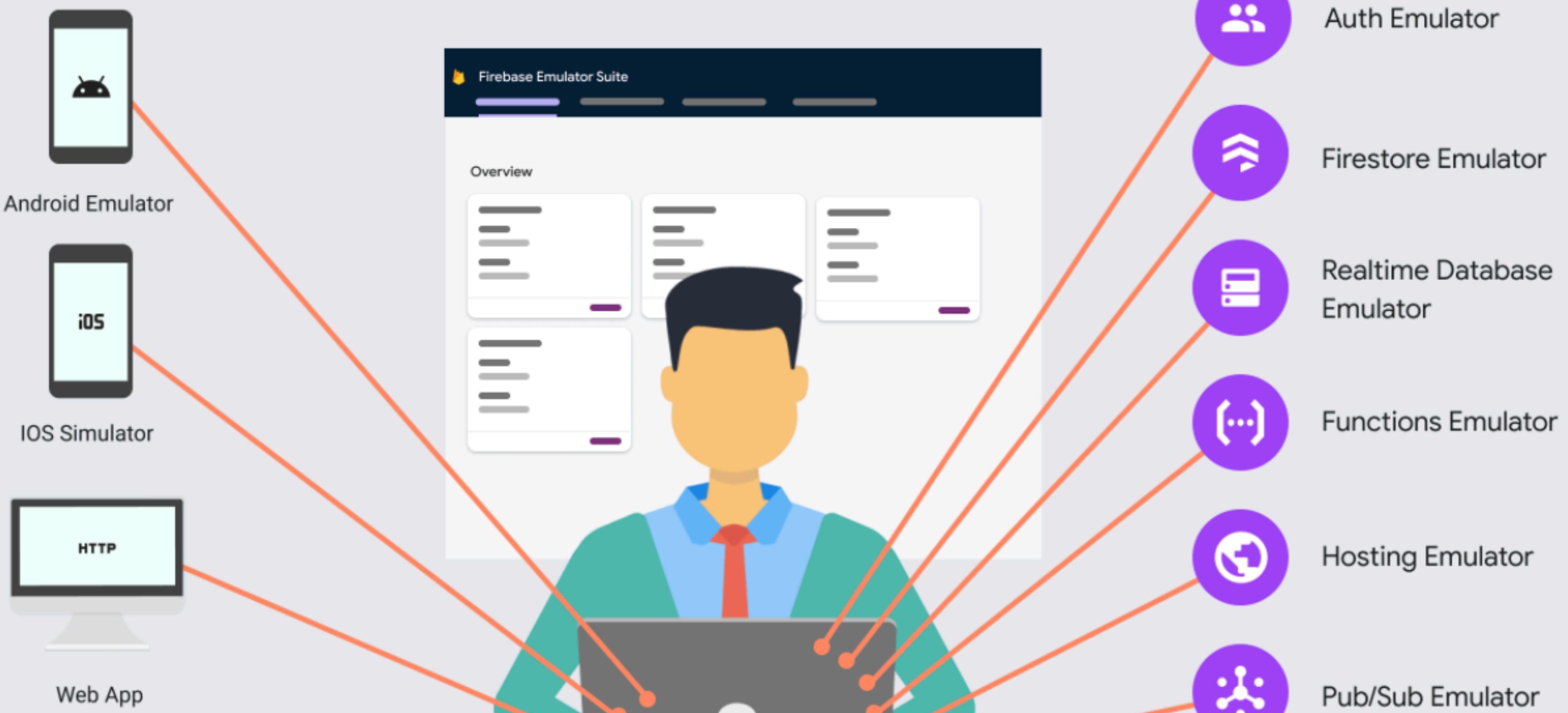
2016

2017

2018

2019

Firebase Emulator Suite



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

twitch.tv/dancojocar

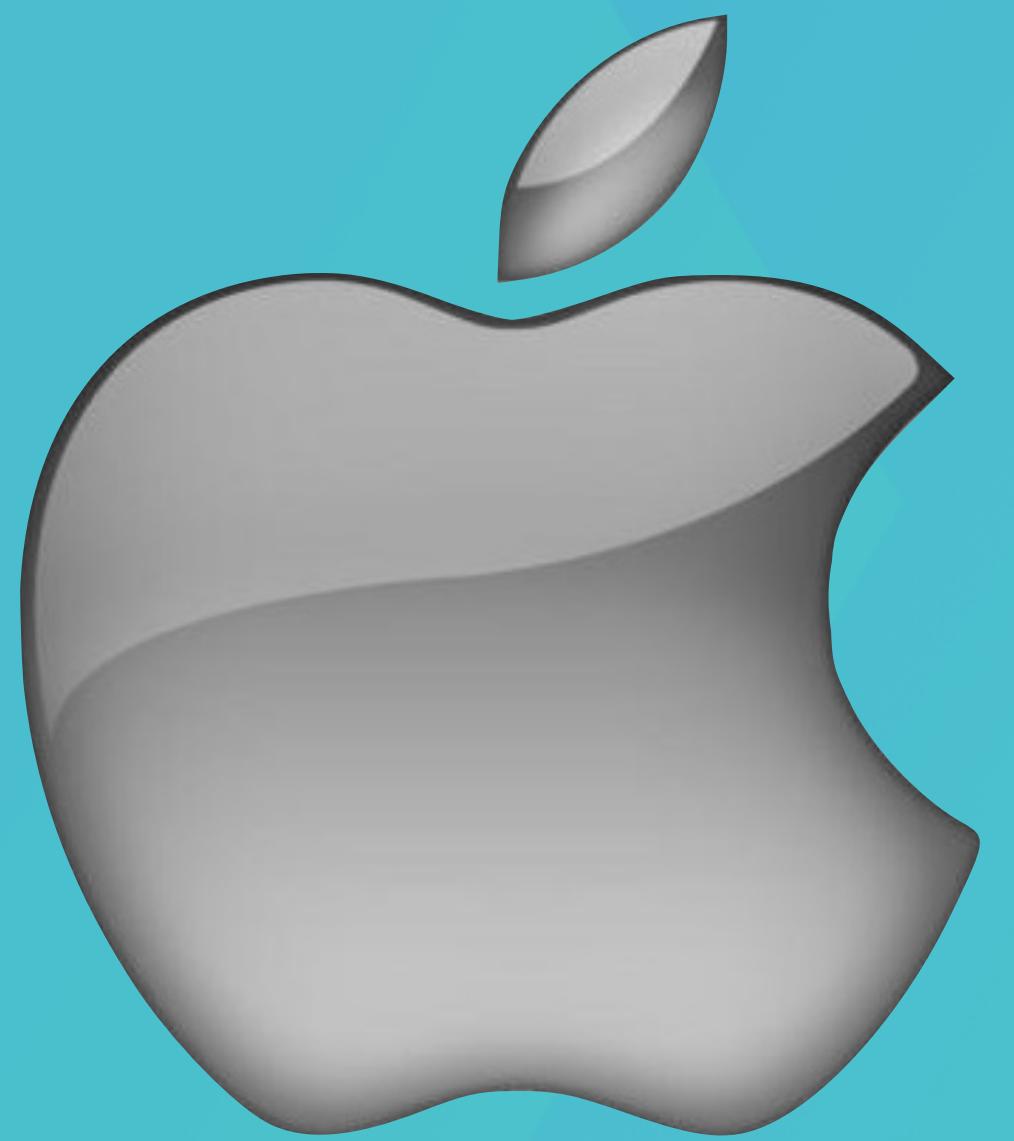
youtube.com/dancojocar



Firebase



ANDROID



iOS

twitch.tv/dancojocar

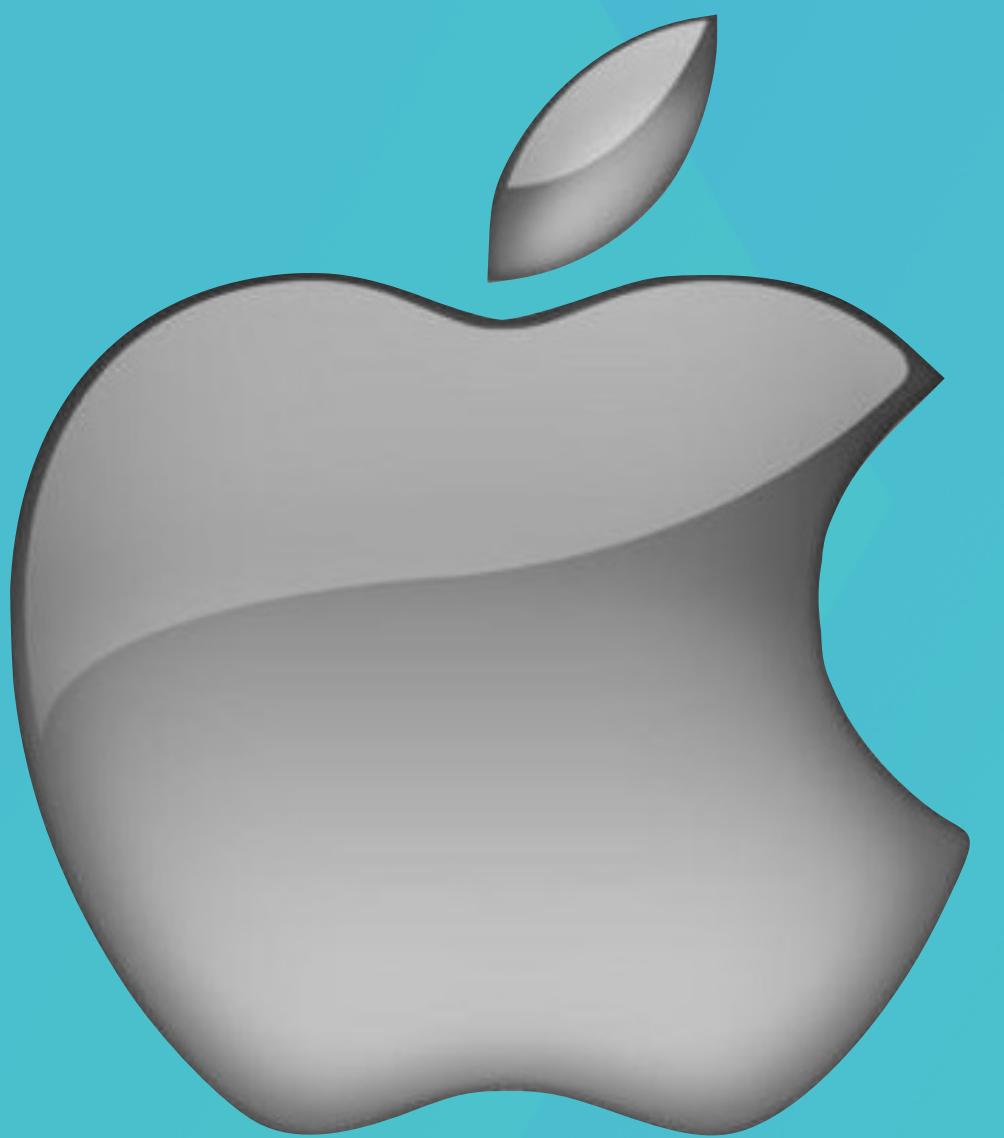
youtube.com/dancojocar



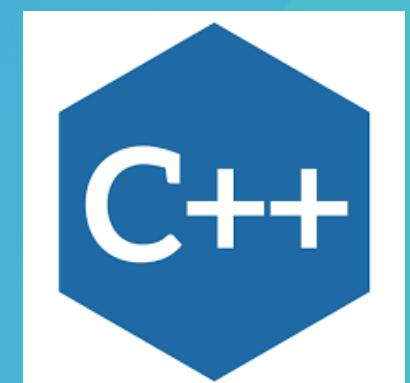
Firebase



ANDROID



iOS



twitch.tv/dancojocar

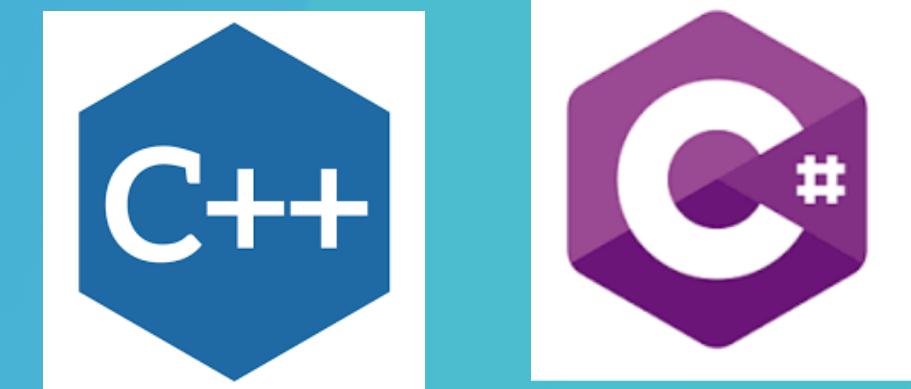
youtube.com/dancojocar



Firebase



ANDROID



iOS



twitch.tv/dancojocar

youtube.com/dancojocar

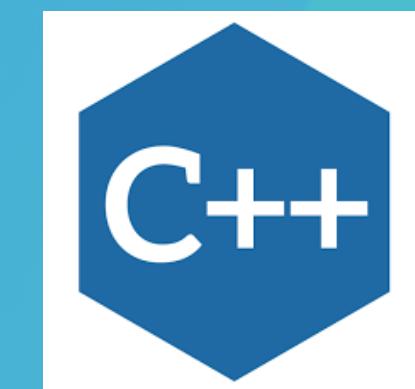


Firebase

Over 1.5 million apps
actively using Firebase
every month!



ANDROID



twitch.tv/dancojocar

youtube.com/dancojocar



twitch.tv/dancojocar

youtube.com/dancojocar



Firebase

Over 2.5 million apps
actively using Firebase
every month!



ANDROID



iOS

Add Firebase to Your Project

twitch.tv/dancojocar

youtube.com/dancojocar



Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

- Using the assistant, in Android Studio:

- 1.Tools -> Firebase.
2. Select the service.
3. Connect to Firebase.

▼  **Analytics**

Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)

►  [Get Started with Firebase Analytics](#)

►  **Cloud Messaging**

Deliver and receive messages and notifications reliably across cloud and device. [More info](#)

►  **Authentication**

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)

►  **Realtime Database**

Store and sync data in realtime across all connected clients. [More info](#)



Firebase gives you the tools and infrastructure from Google to help you develop, grow and earn money from your app. [Learn more](#)

- 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

- Download the config file.

- Add the SDK.

▼  **Analytics**

Measure user activity and engagement with free, easy, and unlimited analytics. [More info](#)

►  [Get Started with Firebase Analytics](#)

►  **Cloud Messaging**

Deliver and receive messages and notifications reliably across cloud and device. [More info](#)

►  **Authentication**

Sign in and manage users with ease, accepting emails, Google Sign-In, Facebook and other login providers. [More info](#)

►  **Realtime Database**

Store and sync data in realtime across all connected clients. [More info](#)

Add the SDK

Root-level build.gradle:

```
buildscript {  
    // ...  
    dependencies {  
        // ...  
        classpath 'com.google.gms:google-services:4.3.4' // 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:26.1.1')  
  
    // 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

ads

com.google.android.gms.ads	Contains classes for Google Mobile Ads.
com.google.android.gms.ads.appopen	
com.google.android.gms.ads.doubleclick	Contains classes for DoubleClick for Publishers.
com.google.android.gms.ads.formats	Contains classes for native ads functionality within Google Mobile Ads.
com.google.android.gms.ads.initialization	Contains classes related to SDK initialization.
com.google.android.gms.ads.instream	Contains classes related to the instream ad format.
com.google.android.gms.ads.mediation	Contains classes for Google Mobile Ads mediation adapters.
com.google.android.gms.ads.mediation.admob	Contains classes for the Google AdMob mediation adapter.
com.google.android.gms.ads.mediation.customevent	Contains classes for Google Mobile Ads mediation custom events.
com.google.android.gms.ads.mediation.rtb	Contains classes for Google Mobile Ads RTB mediation adapters.
com.google.android.gms.ads.reward	Contains classes for Rewarded Video Ads.
com.google.android.gms.ads.reward.mediation	Contains classes for Rewarded Video Ads mediation adapters.
com.google.android.gms.ads.rewarded	Contains classes for Rewarded Ads.
com.google.android.gms.ads.search	Contains classes for Search Ads for Apps.

Available Libraries

appindexing

[com.google.android.gms.appindexing](#)

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

The Firebase App Indexing API lets apps index personal content and log user actions with Google.

[com.google.firebaseio.appindexing.builders](#)

This package contains builders and convenience methods to help construct common types of indexable objects and actions.

appinvite

[com.google.android.gms.appinvite](#)

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

measurement

[com.google.android.gms.measurement](#)

measurement.impl

[com.google.android.gms.measurement](#) Contains classes that configure Firebase Analytics core services.

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

Contains public API classes for Firebase Analytics.

Available Libraries

firebase

[com.google.firebaseio](#)

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

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

firebase.auth

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

crashlytics

[crashlytics](#)

[crashlytics.core](#)

[fabric](#)

firebase.database

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

Available Libraries

`firebase.dynamiclinks`

`com.google.firebaseio.dynamiclinks`

`firebase.firestore`

`com.google.firebaseio`

`com.google.firebaseio.firebaseio`

`firebase.functions`

`com.google.firebaseio.functions`

`firebase.iid`

`com.google.firebaseio.iid`

`firebase.inappmessaging`

`com.google.firebaseio.inappmessaging`

Available Libraries

`firebase.inappmessaging.display`

[com.google.firebaseio.inappmessaging.display](#)

`firebase.messaging`

[com.google.firebaseio.messaging](#) Contains public API classes for Firebase Cloud Messaging.

`firebase.ml.common`

[com.google.firebaseio.ml.common](#)

[com.google.firebaseio.ml.common.modeldownload](#)

`firebase.ml.interpreter`

[com.google.firebaseio.ml.custom](#)

`firebase.ml.naturallanguage`

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

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

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

Available Libraries

firebase.perf

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

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

firebase.remoteconfig

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

firebase.storage

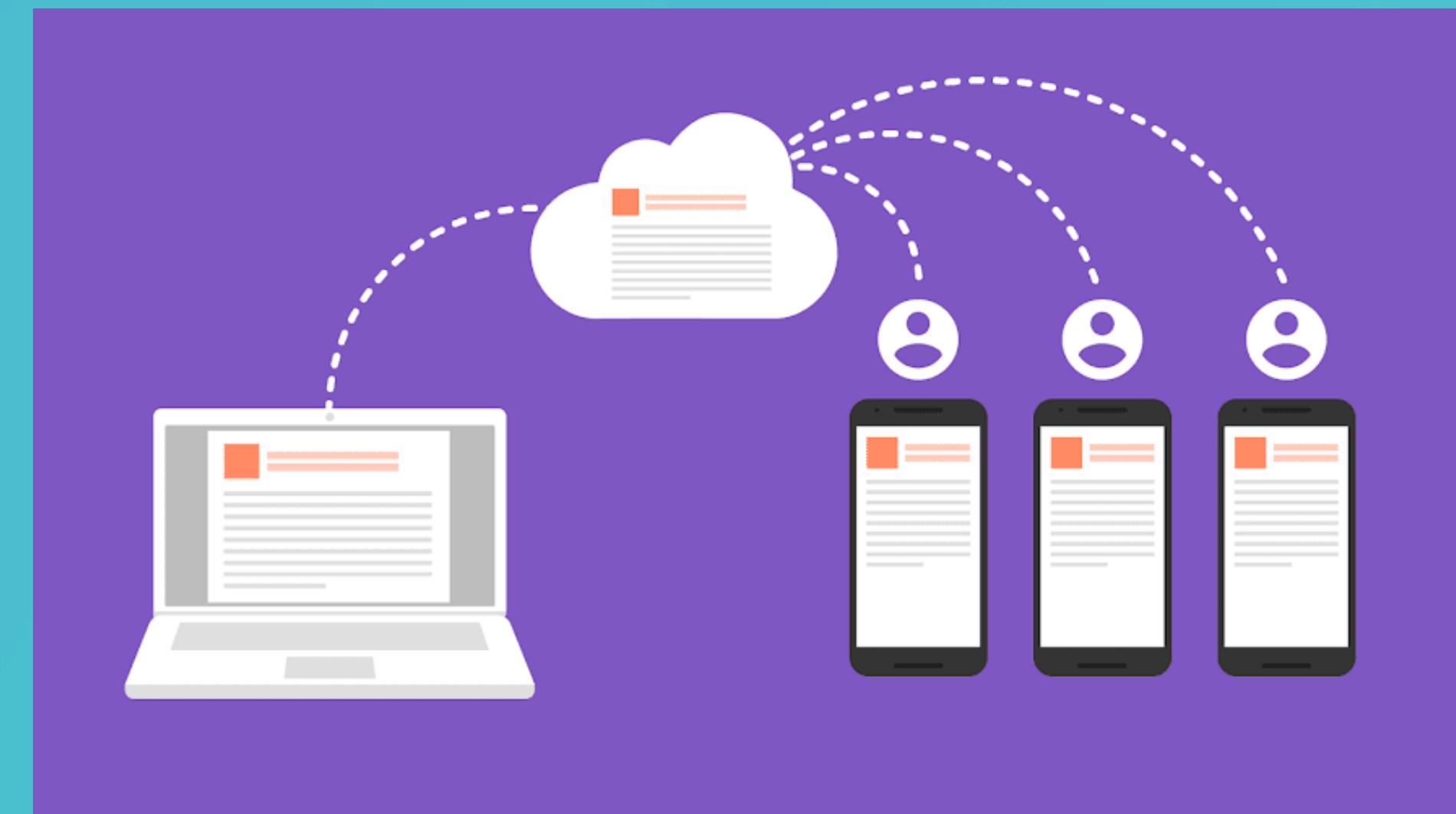
[com.google.firebaseio.storage](#)

Inter-operational packages

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

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.firebaseio:firebase-database-ktx:19.5.1'
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}

{
  "rules": {
    "messages": {
      "$message": {
        // only messages from the last ten minutes can be read
        ".read": "data.child('timestamp').val() > (now - 600000)",
        // new messages must have a string content and a number timestamp
        ".validate": "newData.hasChildren(['content', 'timestamp'])
          && newData.child('content').isString()
          && newData.child('timestamp').isNumber()"
      }
    }
  }
}
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}
```

```
{
  "rules": {
    "messages": {
      "$message": {
        // only messages from the last ten minutes can be read
        ".read": "data.child('timestamp').val() > (now - 600000)",
        // new messages must have a string content and a number timestamp
        ".validate": "newData.hasChildren(['content', 'timestamp'])
          && newData.child('content').isString()
          && newData.child('timestamp').isNumber()"
      }
    }
  }
}
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}

{
  "rules": {
    "messages": {
      "$message": {
        // only messages from the last ten minutes can be read
        ".read": "data.child('timestamp').val() > (now - 600000)",
        // new messages must have a string content and a number timestamp
        ".validate": "newData.hasChildren(['content', 'timestamp'])
          && newData.child('content').isString()
          && newData.child('timestamp').isNumber()"
      }
    }
  }
}
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}

{
  "rules": {
    "messages": {
      "$message": {
        // only messages from the last ten minutes can be read
        ".read": "data.child('timestamp').val() > (now - 600000)",
        // new messages must have a string content and a number timestamp
        ".validate": "newData.hasChildren(['content', 'timestamp'])
          && newData.child('content').isString()
          && newData.child('timestamp').isNumber()"
      }
    }
  }
}
```

Installation & Setup

Add the Realtime Database to your app

```
implementation 'com.google.firebaseio:firebase-database-ktx:19.5.1'
```

Secure Your Data

```
{
  "messages": {
    "message0": {
      "content": "Hello",
      "timestamp": 1405704370369
    },
    "message1": {
      "content": "Goodbye",
      "timestamp": 1405704395231
    },
    ...
  }
}

{
  "rules": {
    "messages": {
      "$message": {
        // only messages from the last ten minutes can be read
        ".read": "data.child('timestamp').val() > (now - 600000)",
        // new messages must have a string content and a number timestamp
        ".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

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

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

Data Access

Write to your database

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

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

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())
    }
})
```

Data Access

Write to your database

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

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

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())
    }
})
```

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,
            dataSnapshot: DataSnapshot?
        ) {
            // Transaction completed
            Log.d(TAG, "postTransaction:onComplete:" + databaseError!!)
        }
    })
}
```

```
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,
            dataSnapshot: DataSnapshot?
        ) {
            // Transaction completed
            Log.d(TAG, "postTransaction:onComplete:" + databaseError!!)
        }
    })
}
```

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);
```

DEMO

Enabling Offline Capabilities

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

Keeping Data Fresh

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

scoresRef.keepSynced(false);
```

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:

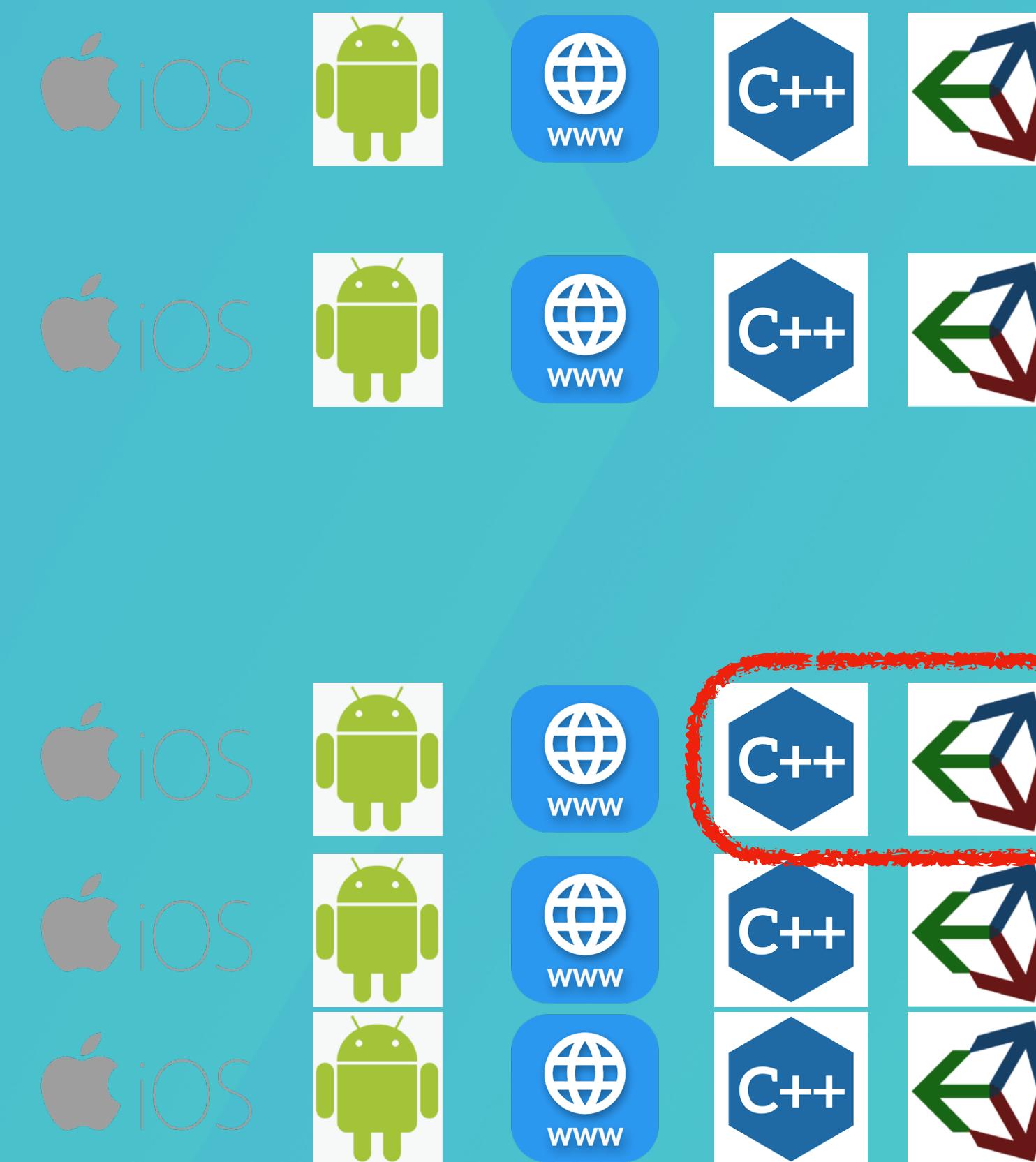
- Email and password based authentication.
- Federated identity provider integration:
 - Google, Facebook, Twitter, Github.
 - Phone number authentication.
 - Custom auth system integration.
 - Anonymous auth.



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

twitch.tv/dancojocar
youtube.com/dancojocar

Authenticate Using Google Sign-In

- Dependencies

```
implementation 'com.google.firebaseio:firebase-auth:20.0.1'  
implementation 'com.google.android.gms:play-services-auth:19.0.0'
```

Authenticate Using Google Sign-In

- Dependencies
- Integrate Google Sign-In

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

Authenticate Using Google Sign-In

twitch.tv/dancojocar

youtube.com/dancojocar

- Dependencies
- Integrate Google Sign-In

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

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

Authenticate Using Google Sign-In

twitch.tv/dancojocar

youtube.com/dancojocar

- Dependencies
- Integrate Google 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)
            // ...
        }
    }
}
```

Authenticate Using Google Sign-In

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

```
private lateinit var auth: FirebaseAuth
// ...
// Initialize Firebase Auth
auth = FirebaseAuth.getInstance()

// ...

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

Authenticate Using Google Sign-In

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

```
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.  
            }  
        }  
}
```

Authenticate Using Google Sign-In

twitch.tv/dancojocar

youtube.com/dancojocar

```
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)
                Snackbar.make(main_layout, "Authentication Failed.", Snackbar.LENGTH_SHORT).show()
                updateUI(null)
            }
        }
    // ...
}
```

DEMO

Sign out a User

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



twitch.tv/dancojocar

youtube.com/dancojocar



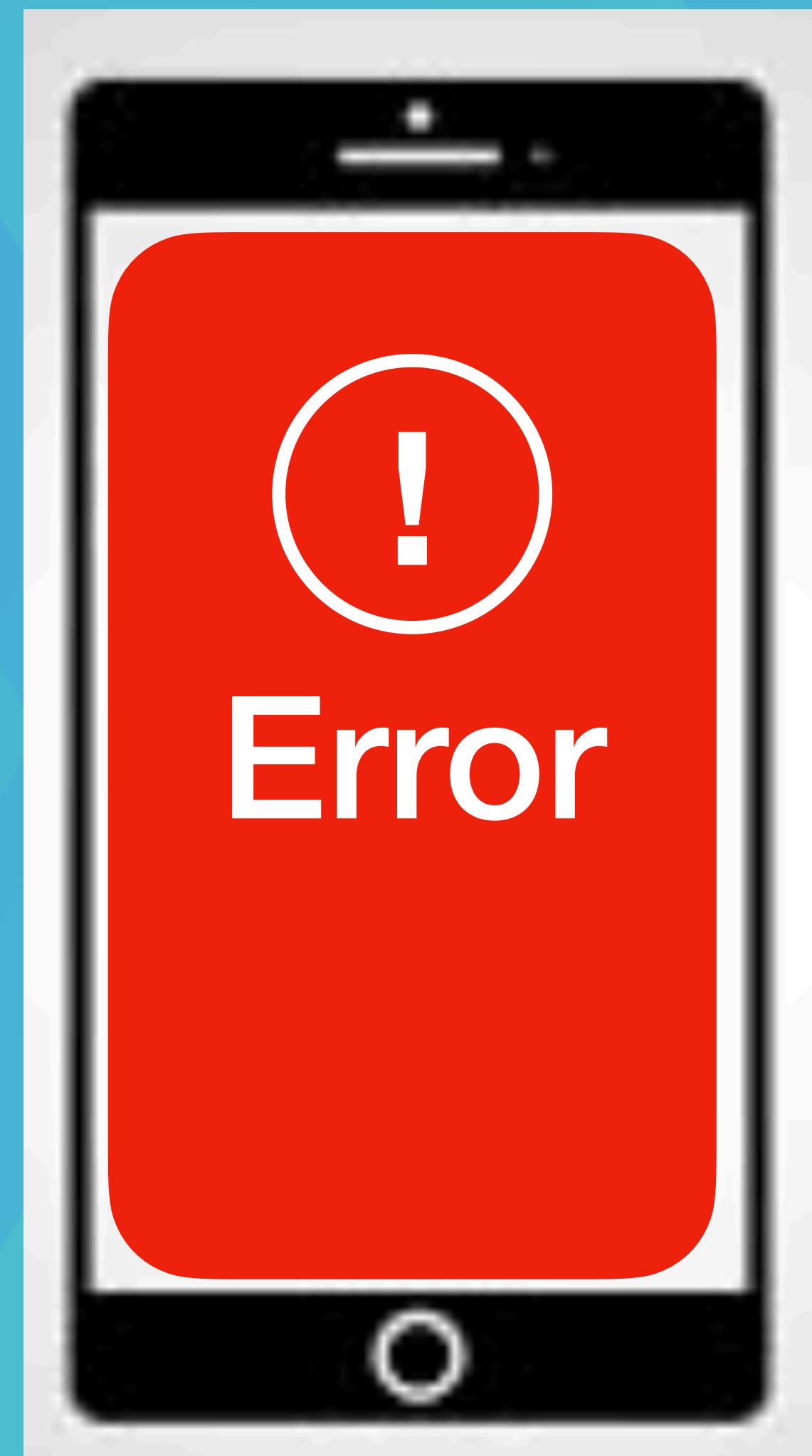
twitch.tv/dancojocar

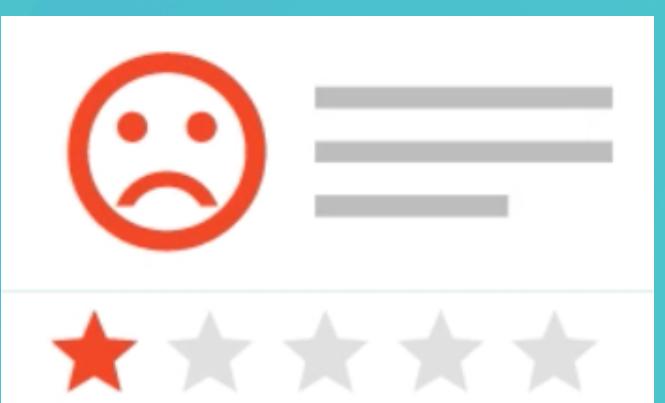
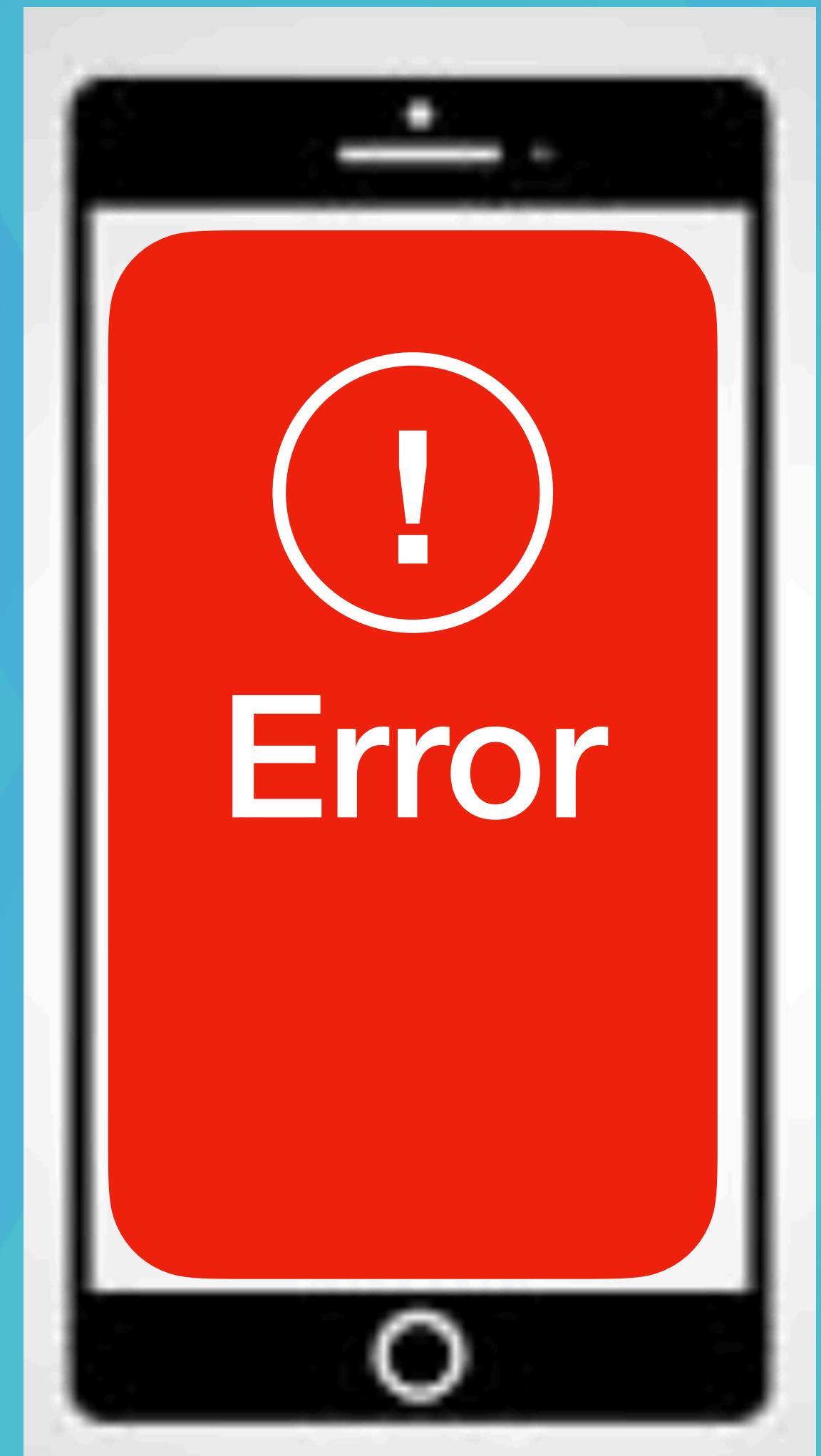
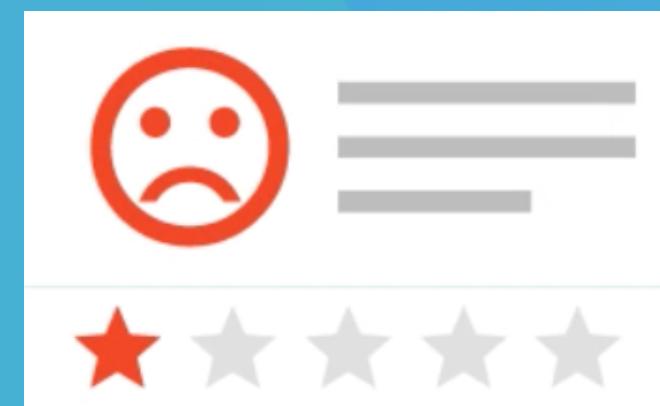
youtube.com/dancojocar

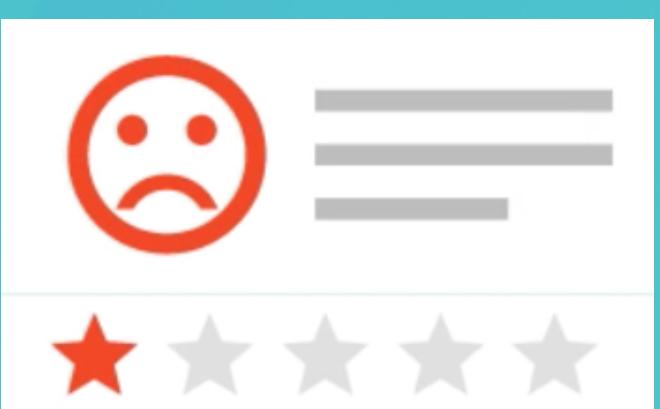
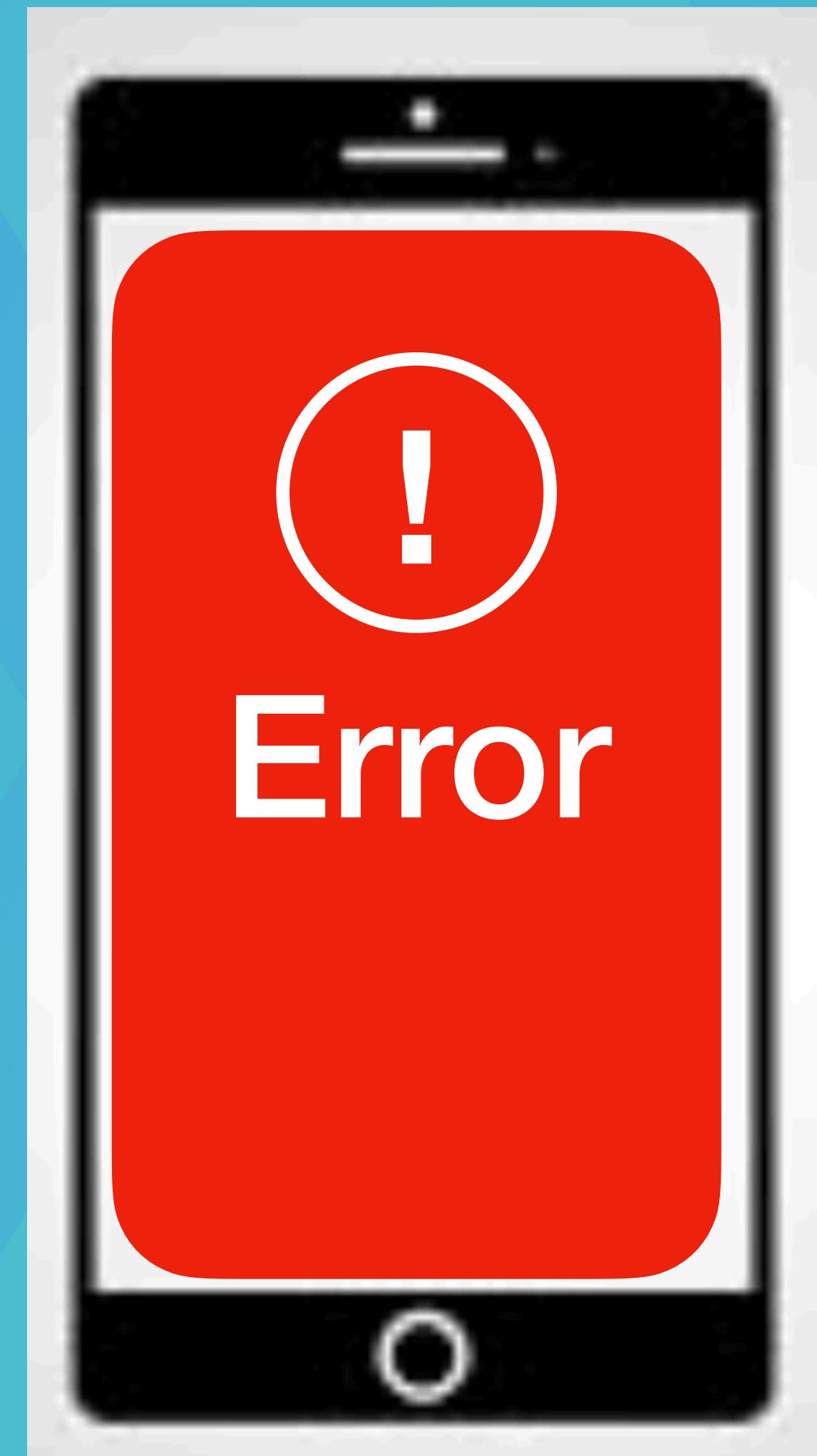
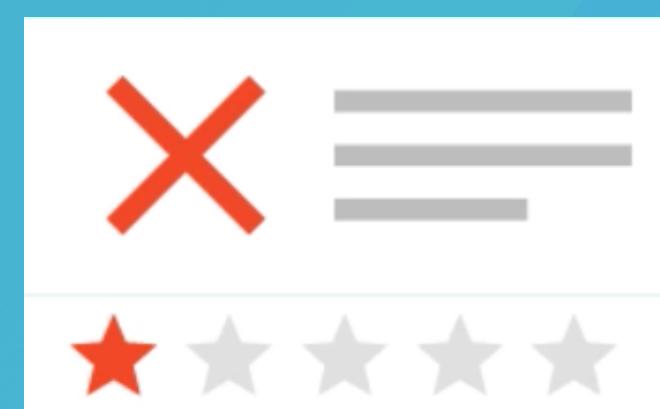
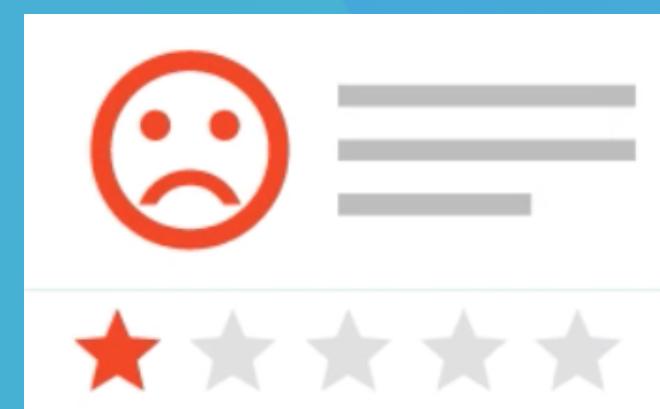


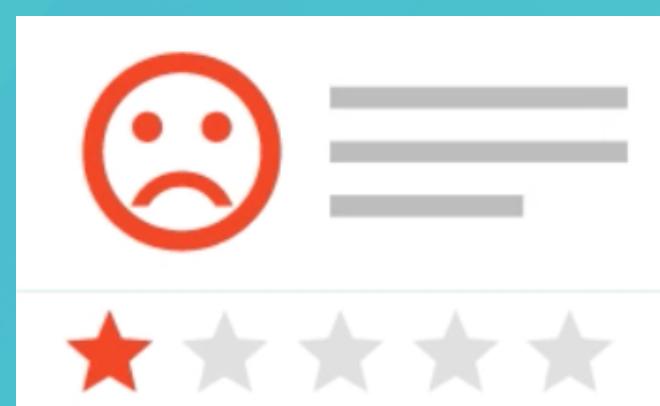
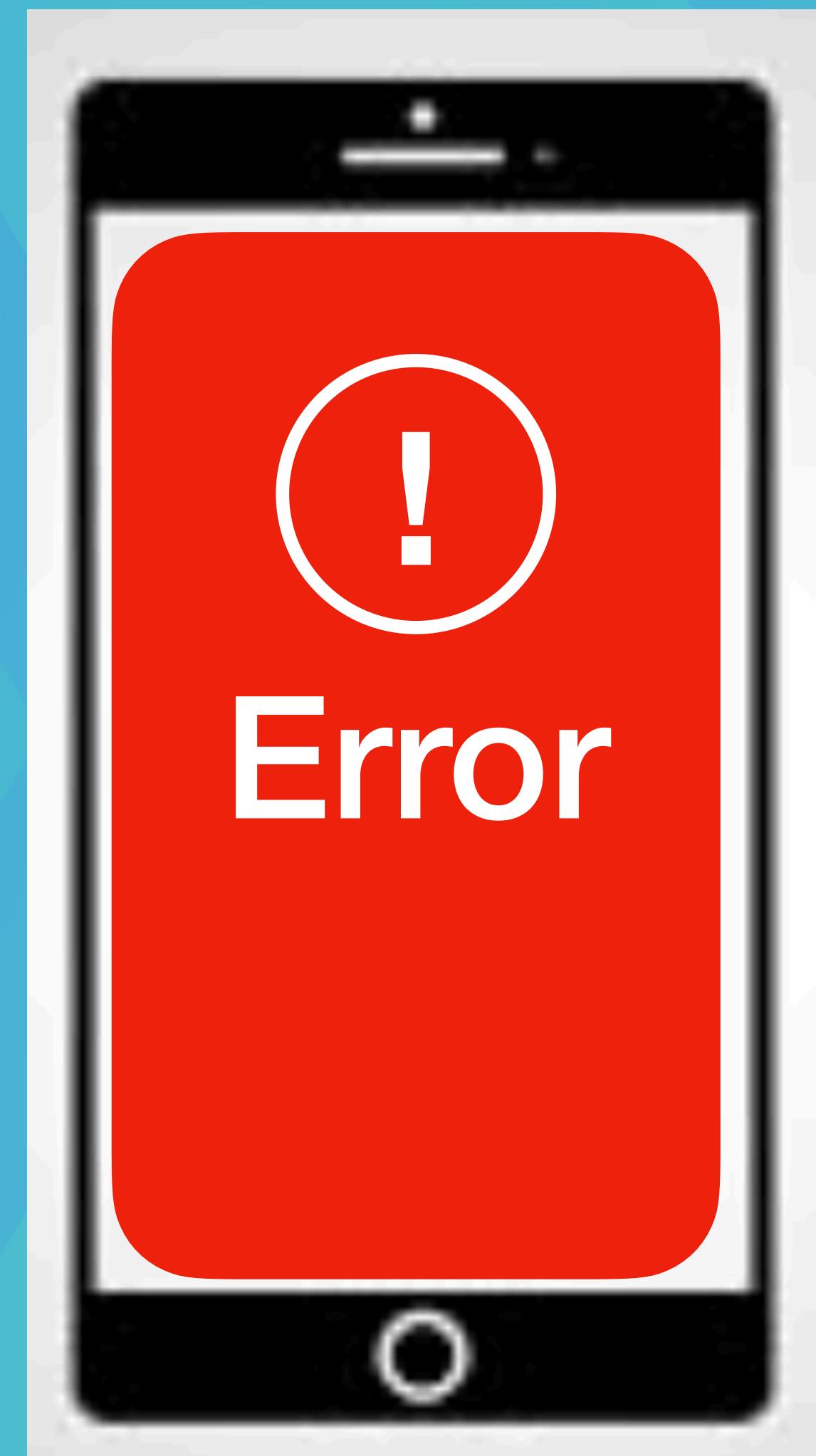
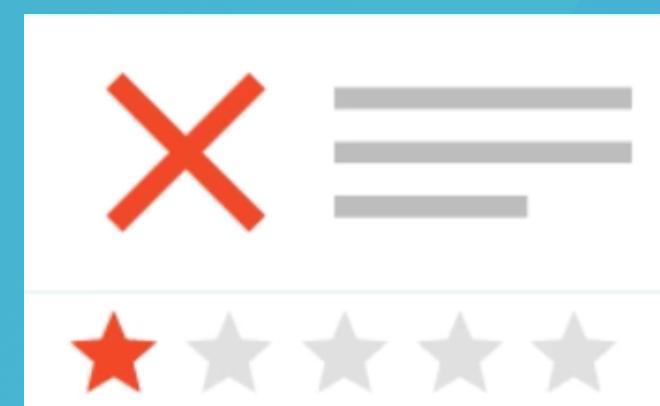
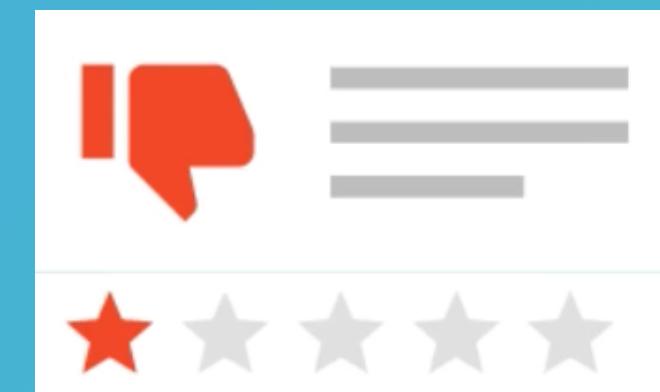
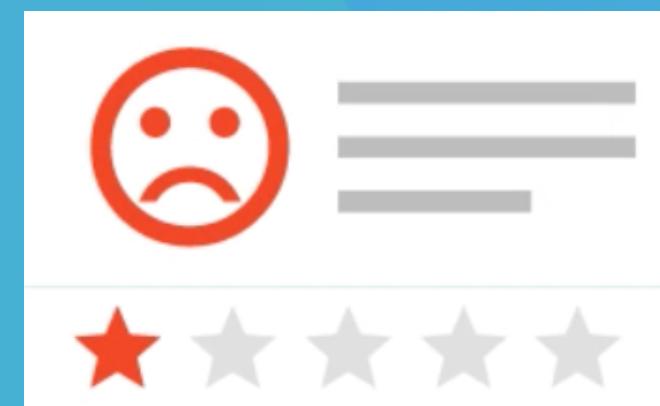
twitch.tv/dancojocar

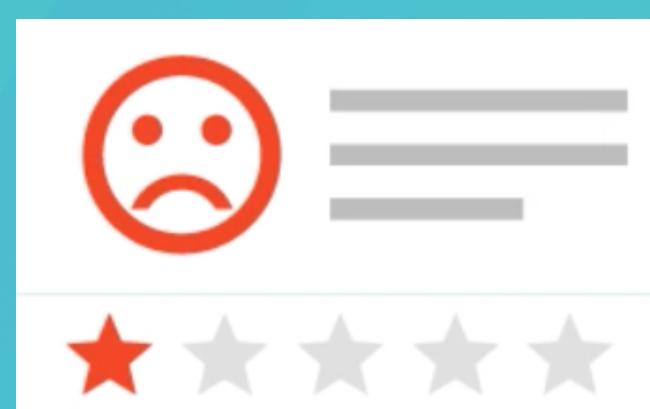
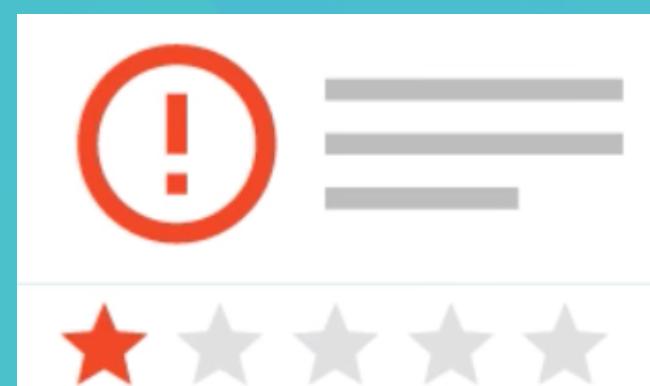
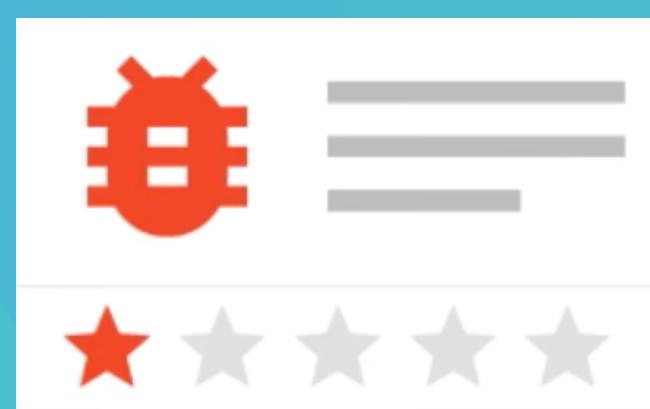
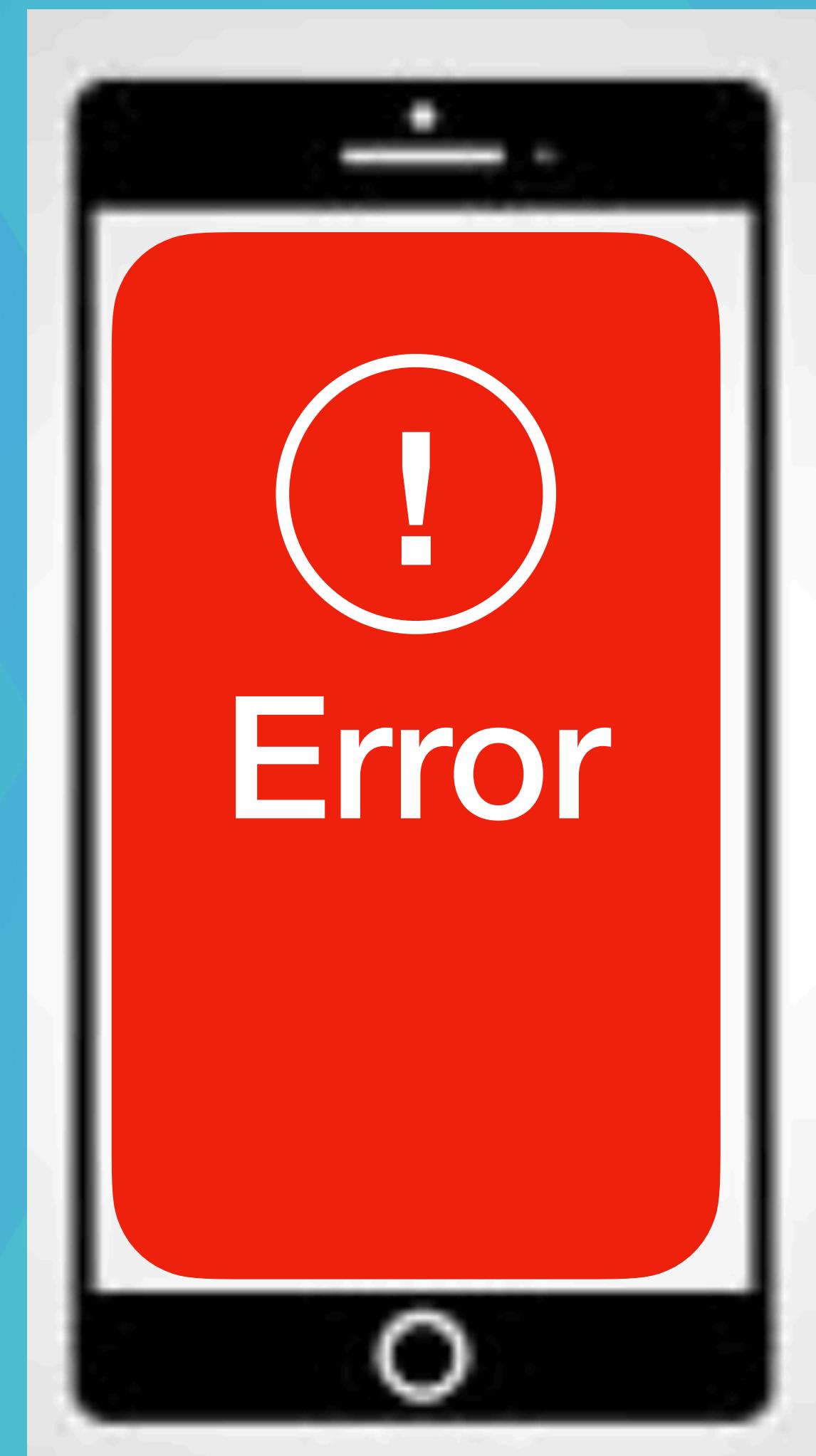
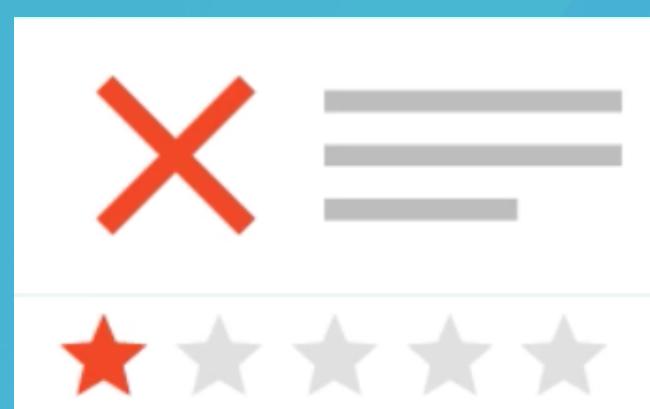
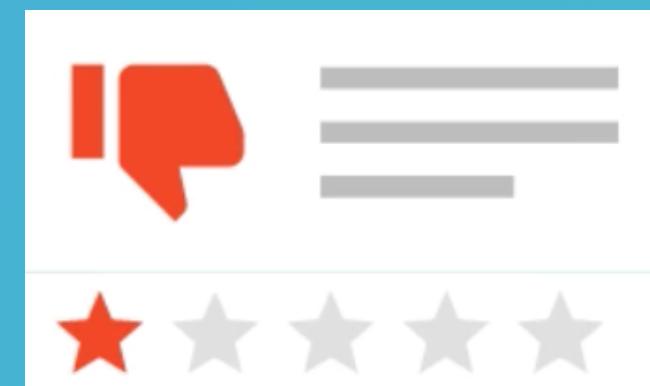
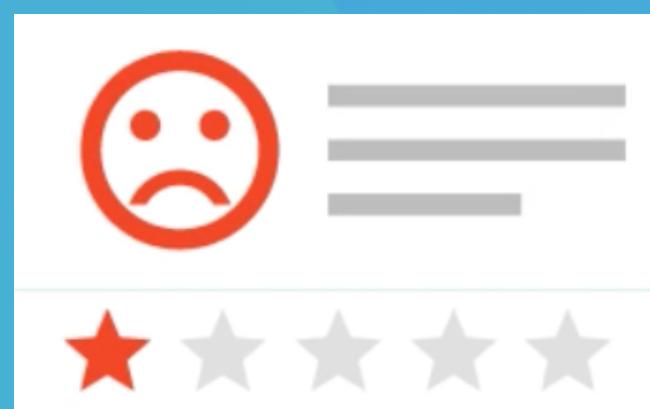
youtube.com/dancojocar









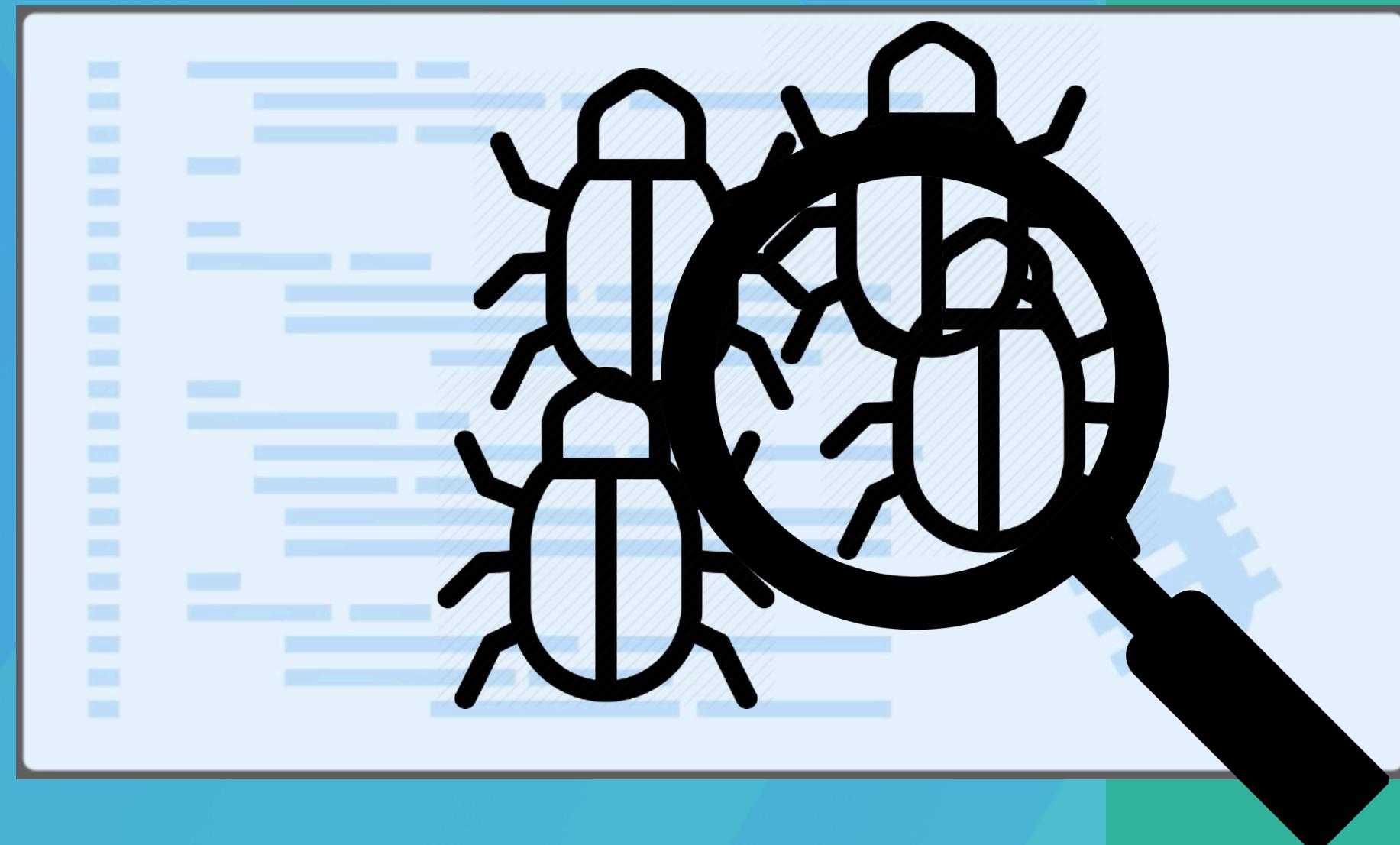




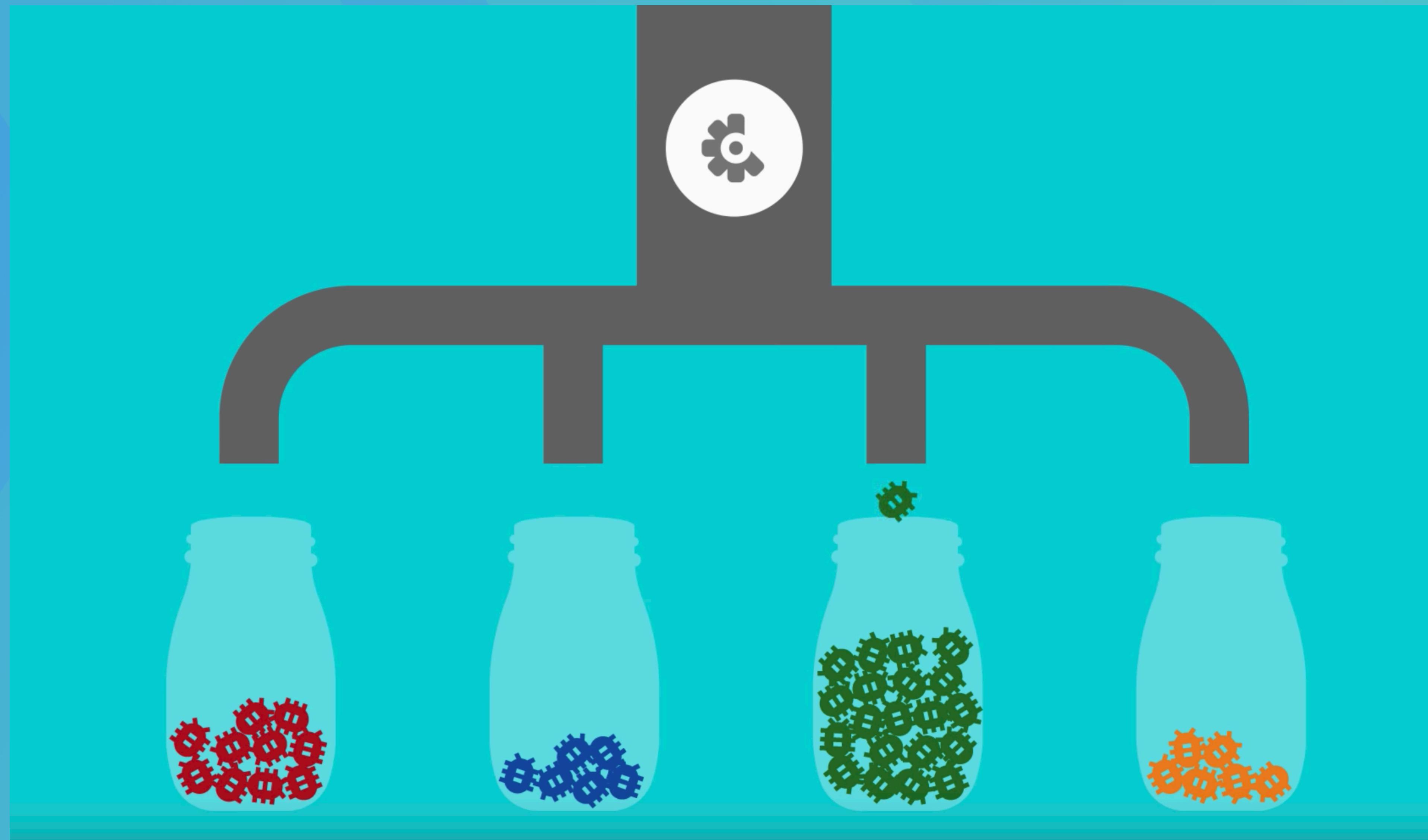


twitch.tv/dancojocar

youtube.com/dancojocar



Crashlytics



Enable the SDK

twitch.tv/dancojocar

youtube.com/dancojocar

```
buildscript {  
    repositories {  
        // Add the following repositories:  
        google() // Google's Maven repository  
    }  
    dependencies {  
        // ...  
        classpath 'com.google.gms:google-services:4.3.4' // Google Services plugin  
        // Add dependency  
        classpath 'com.google.firebaseio:firebase-crashlytics-gradle:2.4.1'  
        // Crashlytics plugin  
    }  
}  
allprojects {  
    repositories {  
        // Check that you have the following line (if not, add it):  
        google() // Google's Maven repository  
    }  
}
```

Enable the SDK

twitch.tv/dancojocar

youtube.com/dancojocar

```
apply plugin: 'com.android.application'  
// Google Services Gradle plugin  
apply plugin: 'com.google.gms.google-services'  
  
// Apply the Crashlytics Gradle plugin  
apply plugin: 'com.google.firebaseio.crashlytics'  
  
dependencies {  
    // ...  
    // Import the BoM for the Firebase platform  
    implementation platform('com.google.firebaseio:firebase-bom:26.1.1')  
  
    // Declare the dependencies for the Crashlytics and Analytics libraries  
    // When using the BoM, you don't specify versions in Firebase library dependencies  
    implementation 'com.google.firebaseio:firebase-crashlytics-ktx'  
    implementation 'com.google.firebaseio:firebase-analytics-ktx'  
}
```

Test Implementation

```
val crashButton = Button(this)
crashButton.text = "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

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")
```

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

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")
```

Add custom keys:

```
Crashlytics.setInt("current_level", 3)  
Crashlytics.setString("last_UI_action", "teamed the dragon!")
```

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")
```

Add custom keys:

```
Crashlytics.setInt("current_level", 3)  
Crashlytics.setString("last_UI_action", "teamed the dragon!")
```

Set user IDs

```
Crashlytics.setUserIdentifier("user123456789")
```

DEMO

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")
```

Add custom keys:

```
Crashlytics.setInt("current_level", 3)  
Crashlytics.setString("last_UI_action", "teamed the dragon!")
```

Set user IDs

```
Crashlytics.setUserIdentifier("user123456789")
```

Log non-fatal exceptions

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

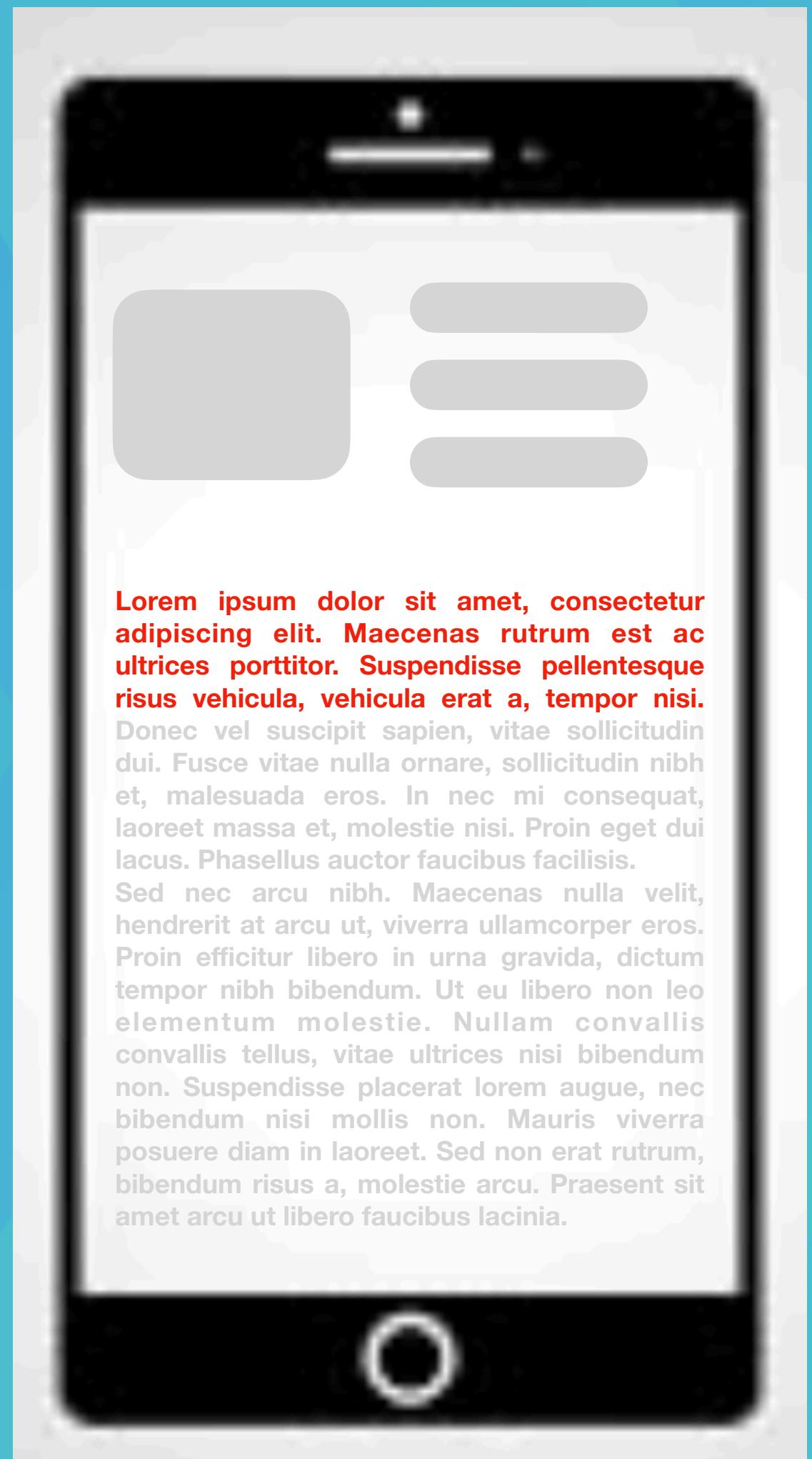
twitch.tv/dancojocar

youtube.com/dancojocar

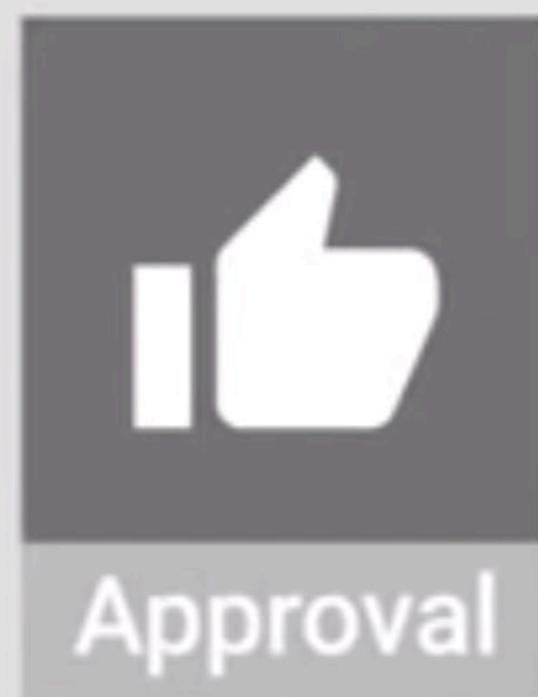
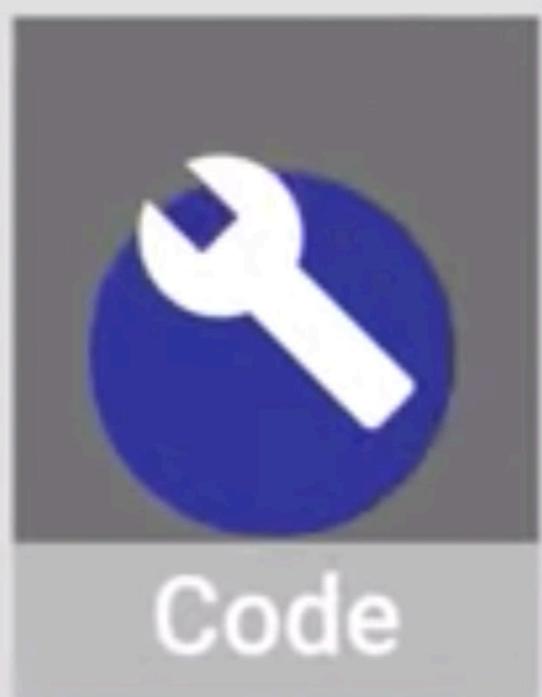


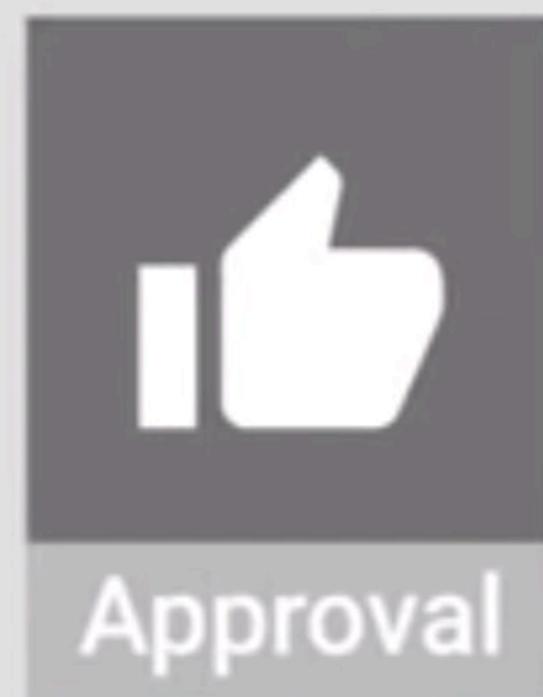
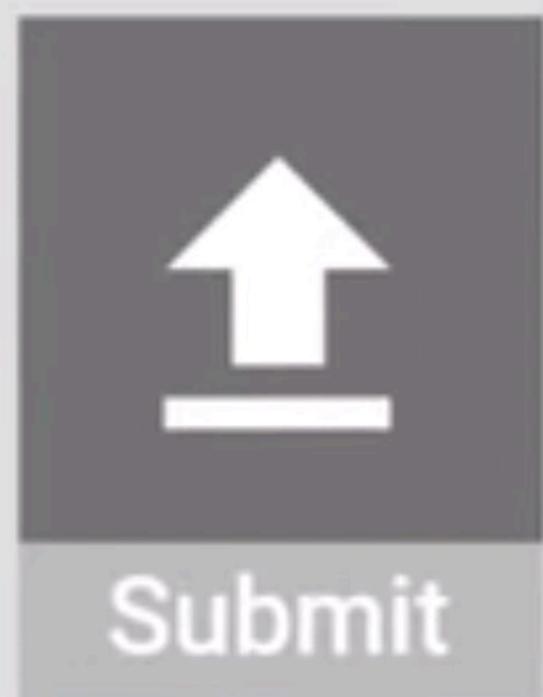
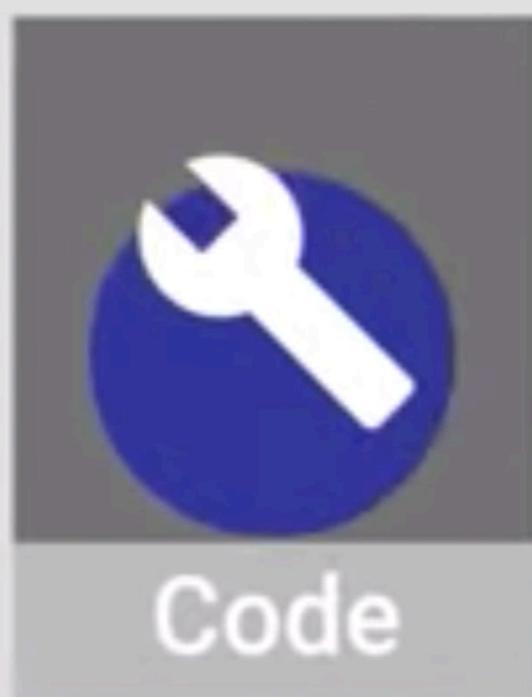
twitch.tv/dancojocar

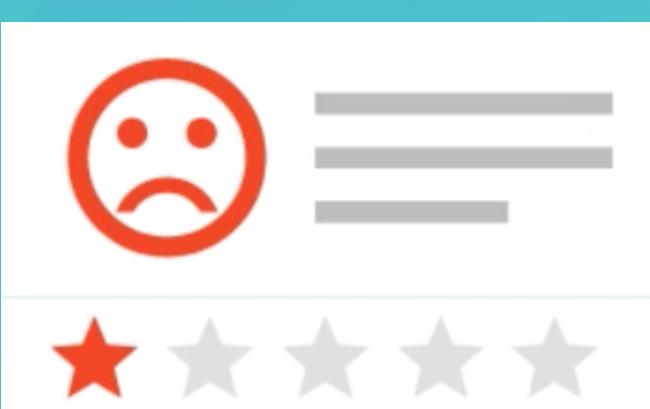
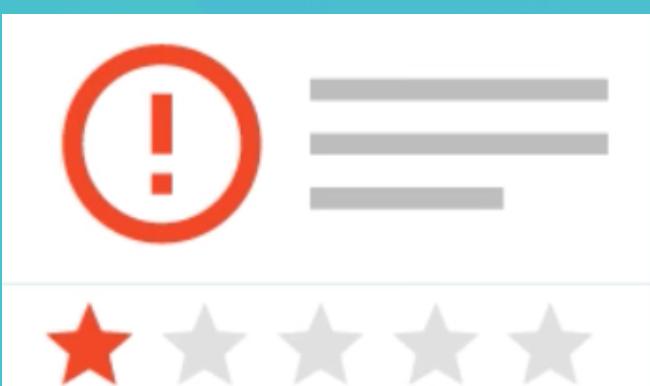
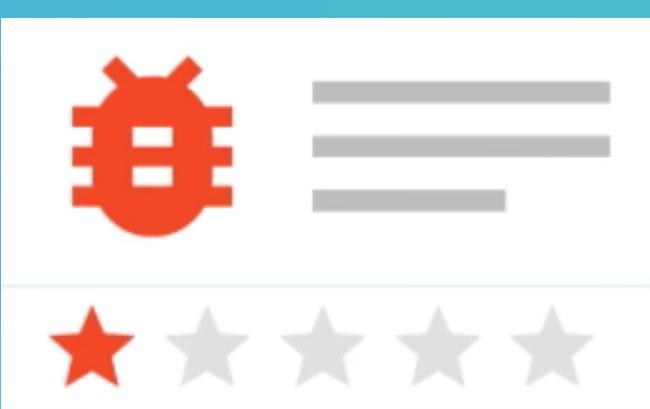
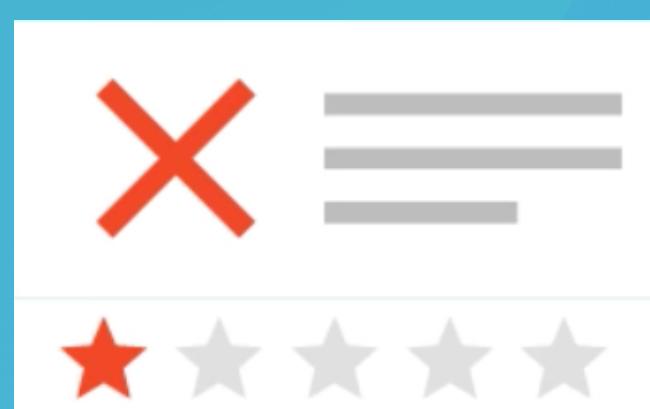
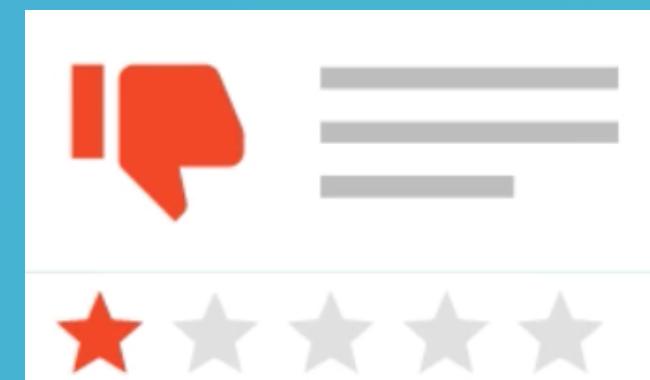
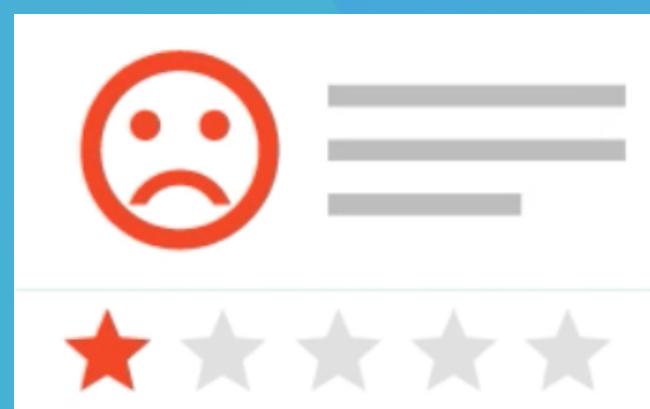
youtube.com/dancojocar



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 lacinia. 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.

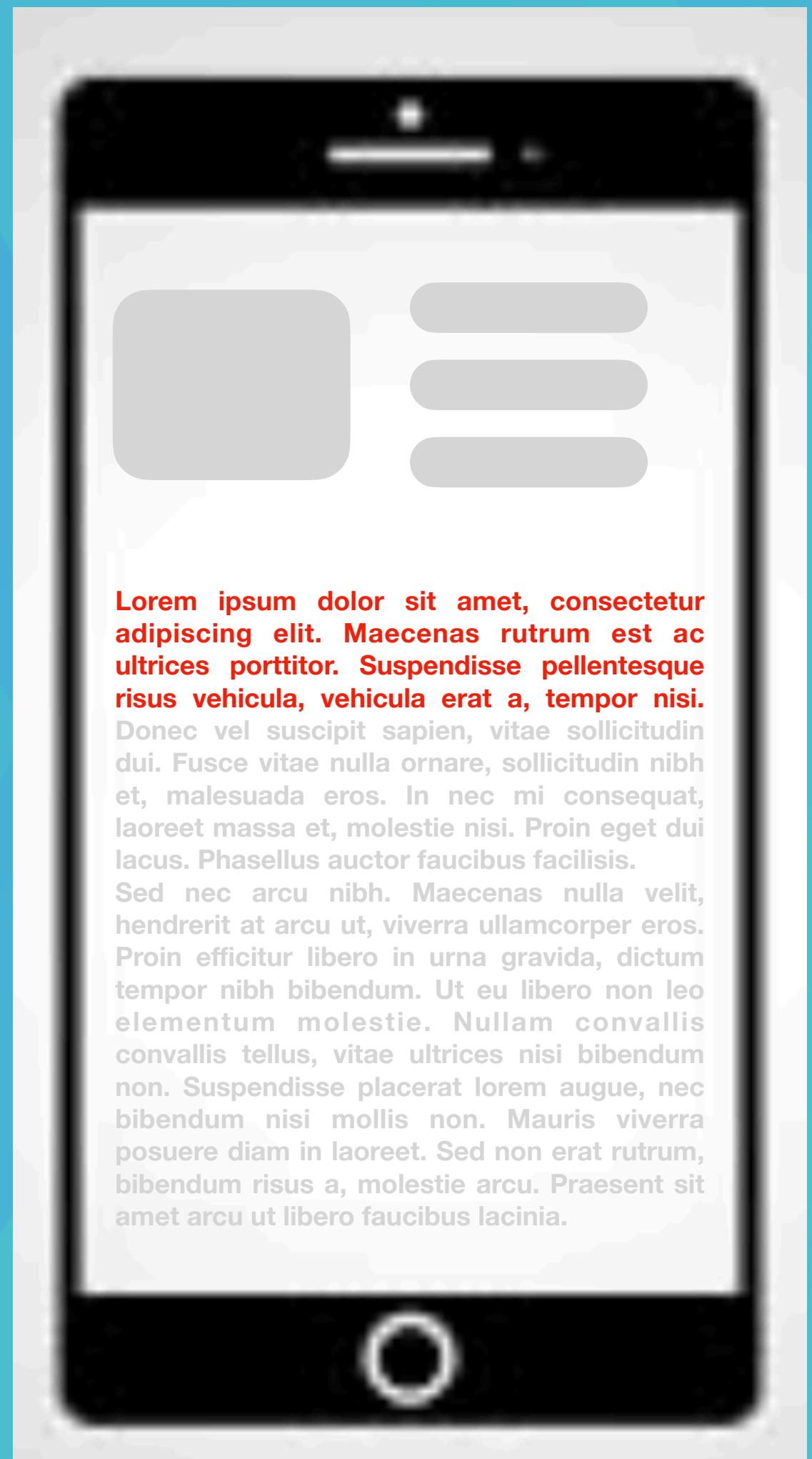






twitch.tv/dancojocar

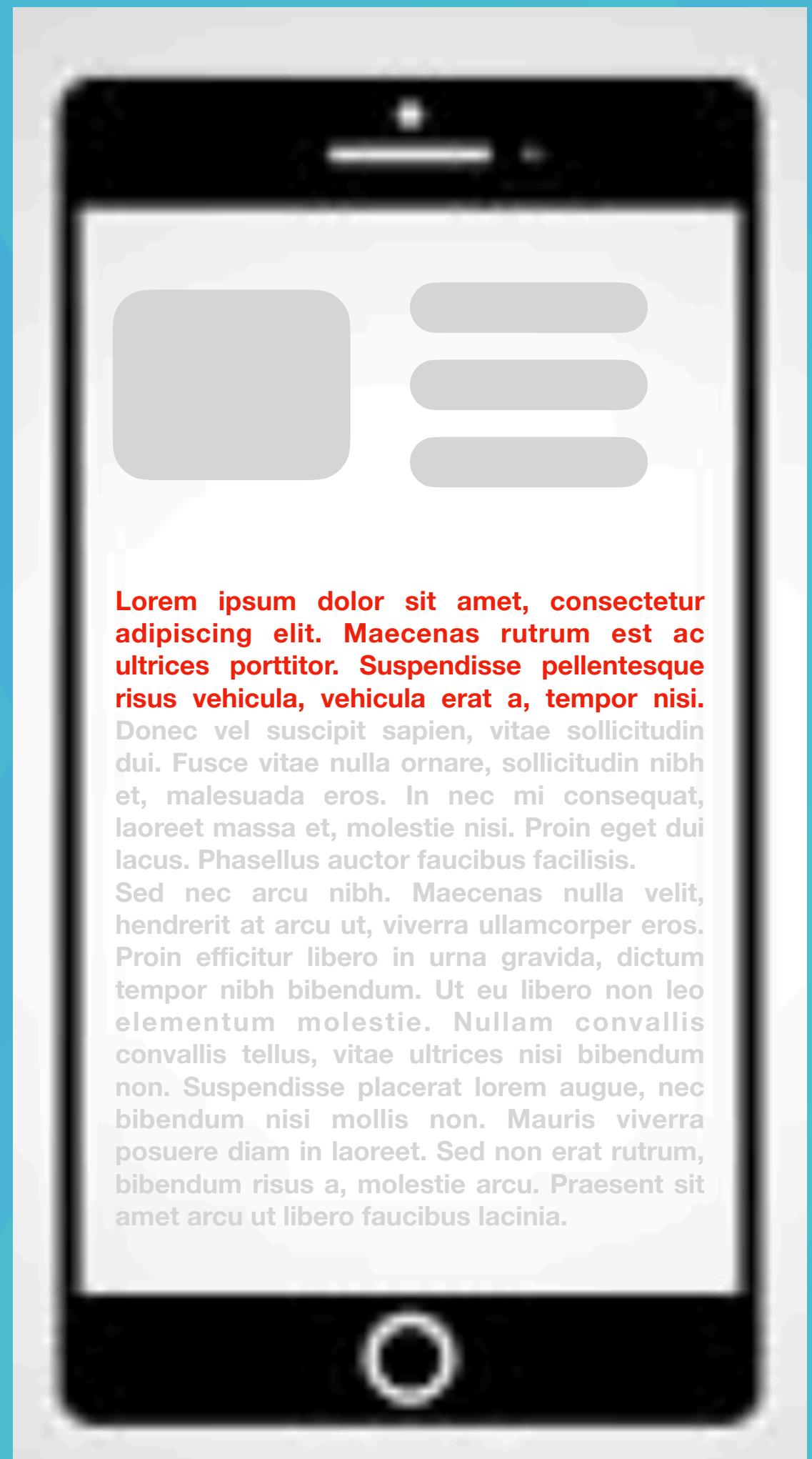
youtube.com/dancojocar

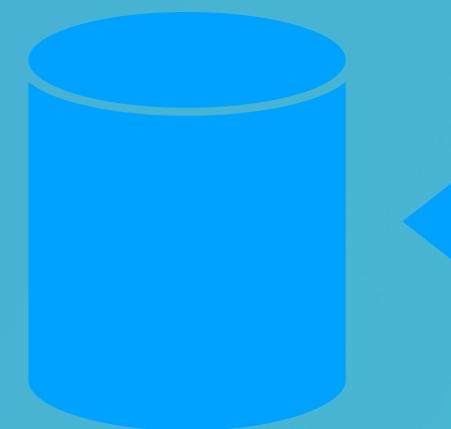
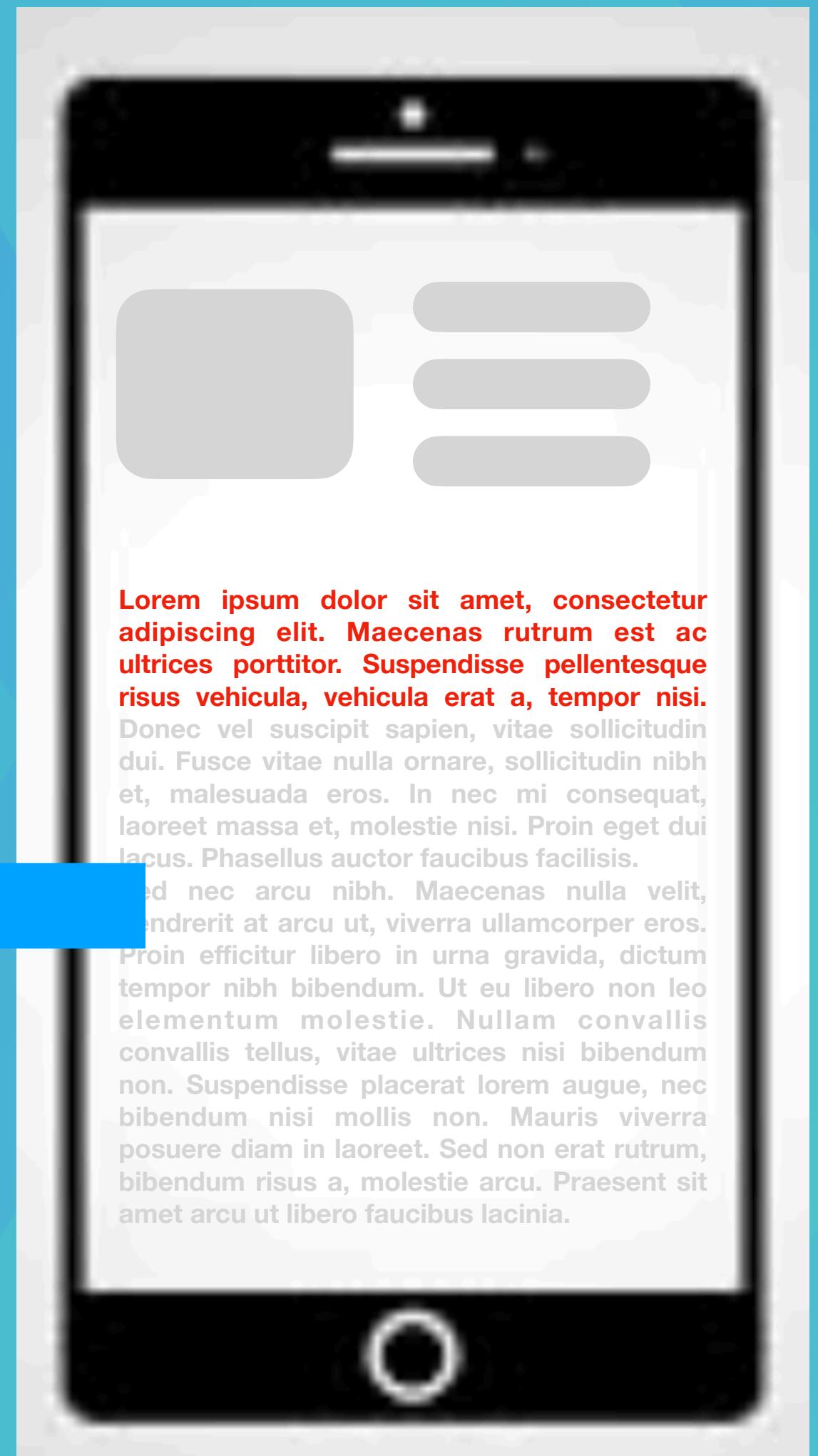


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas rutrum est ac ultrices porttitor. Suspendisse pellentesque risus vehicula, vehicula erat a, tempor nisi.
Donec vel suscipit sapien, vitae sollicitudin dui. Fusce vitae nulla ornare, sollicitudin nibh et, malesuada eros. In nec mi consequat, laoreet massa et, molestie nisi. Proin eget dui lacinia. 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.

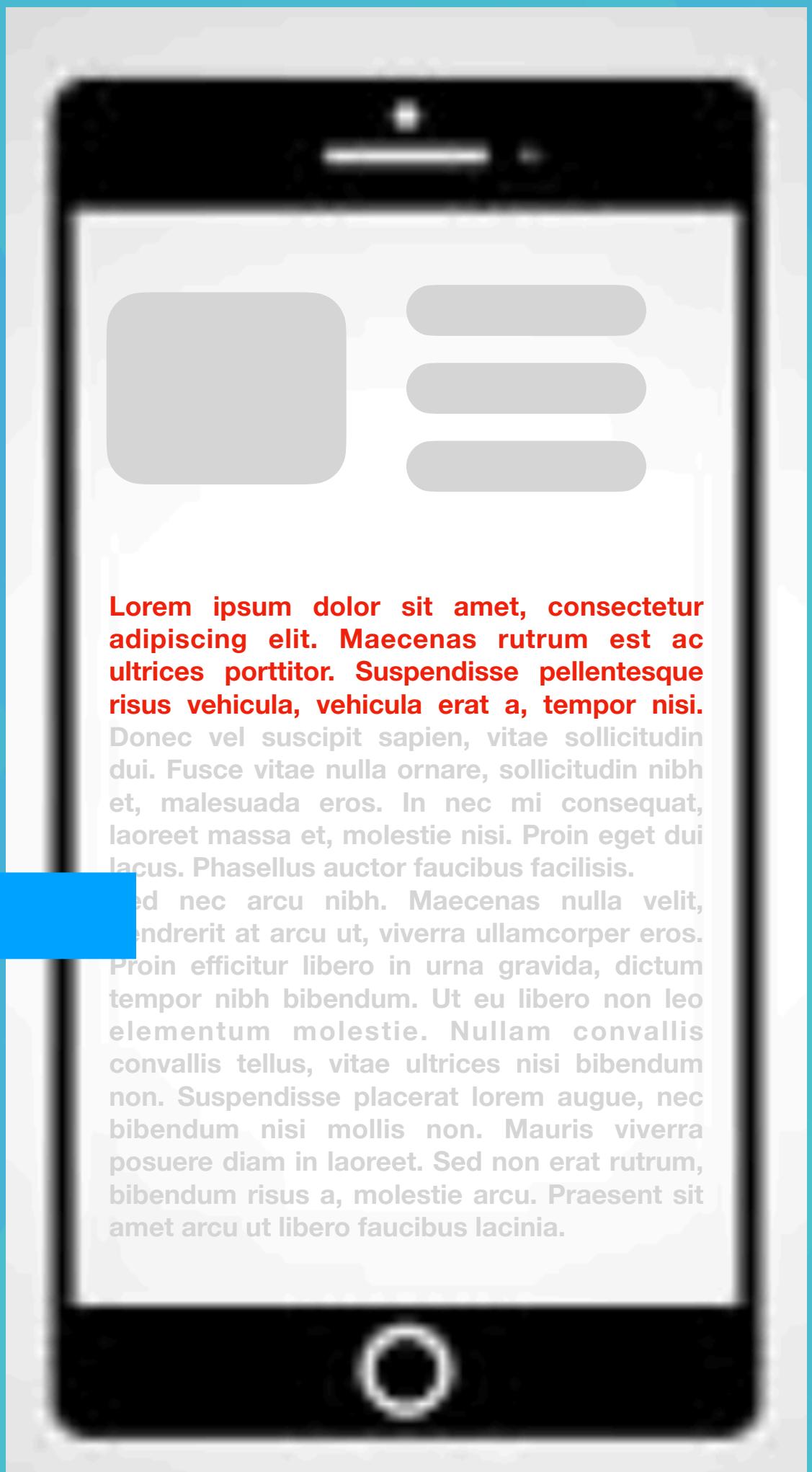
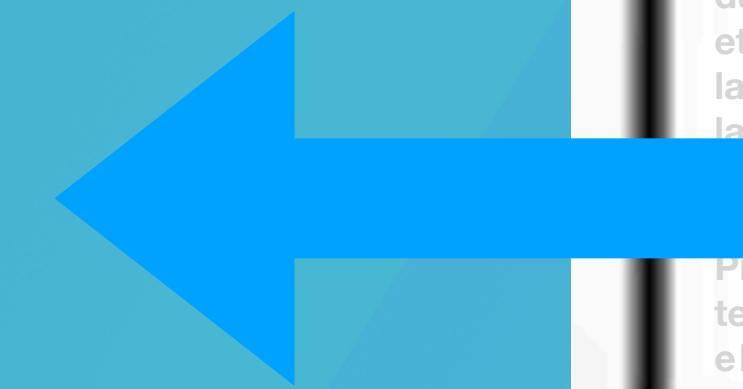
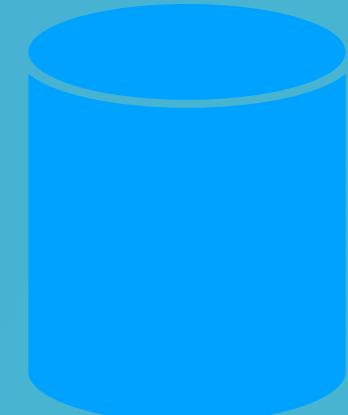
twitch.tv/dancojocar

youtube.com/dancojocar

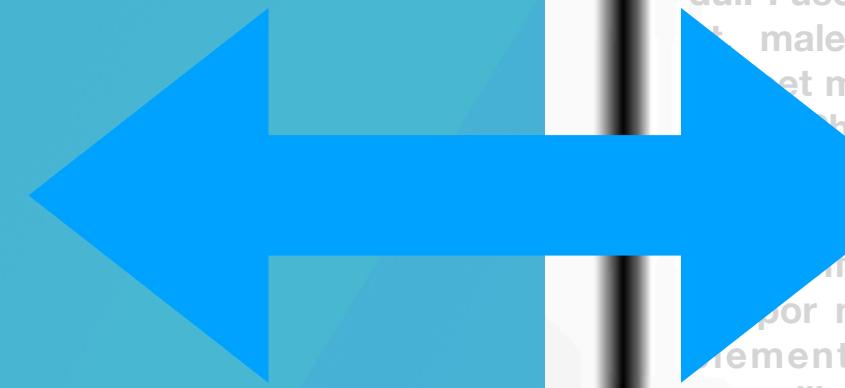




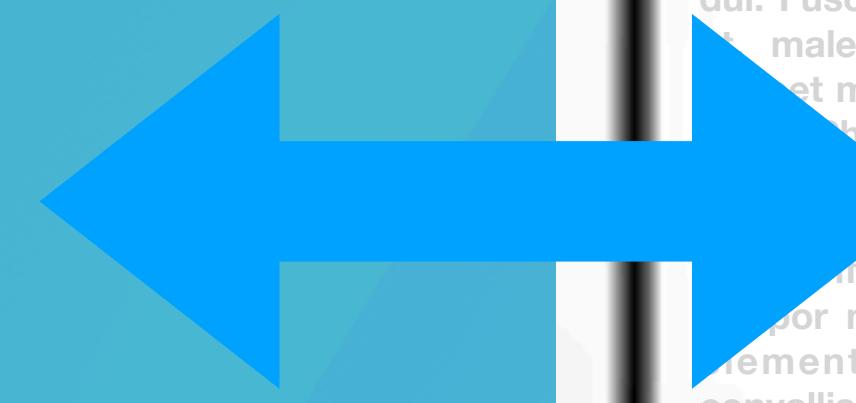
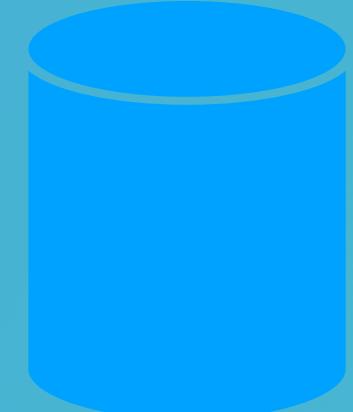
Realtime



Realtime

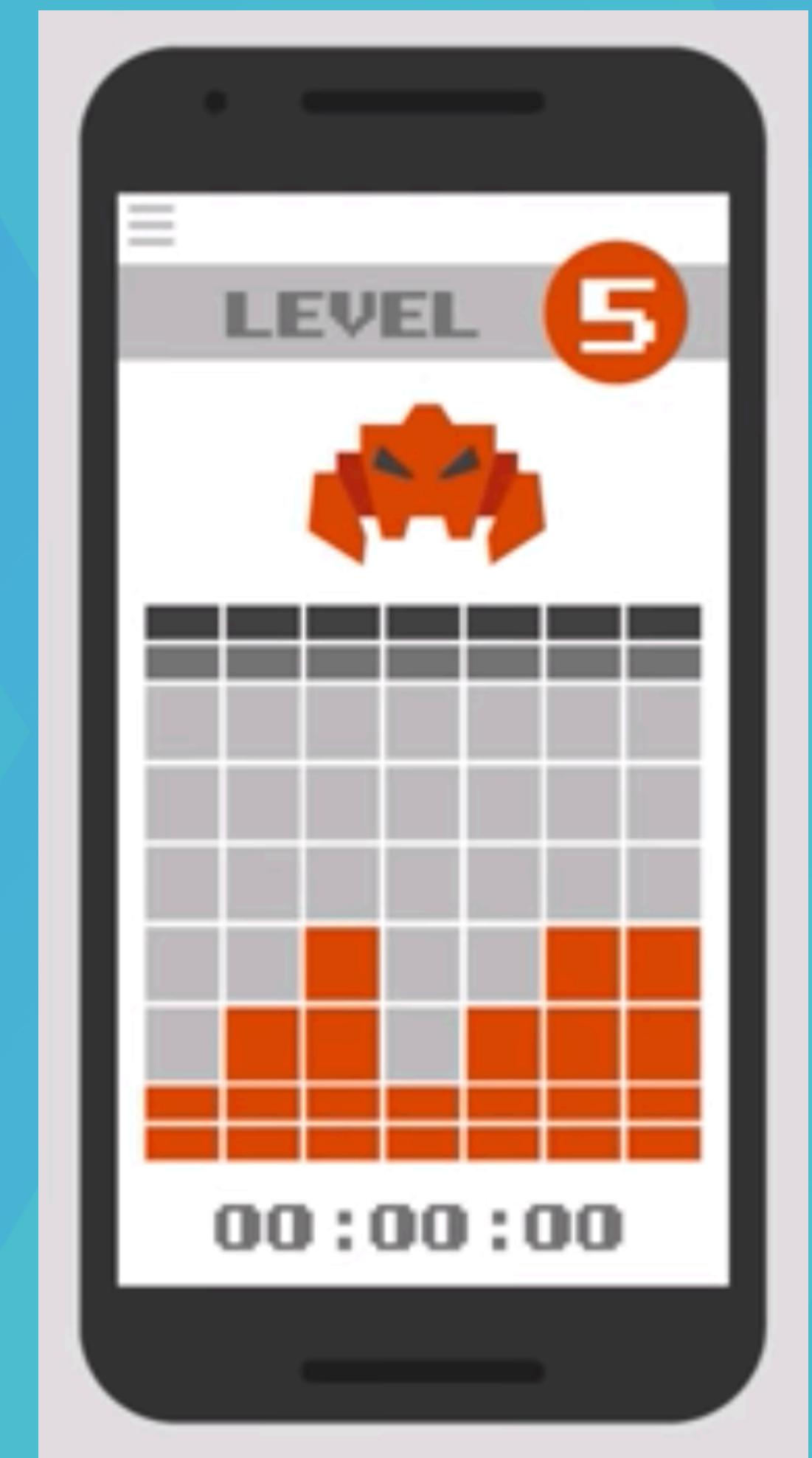


Realtime



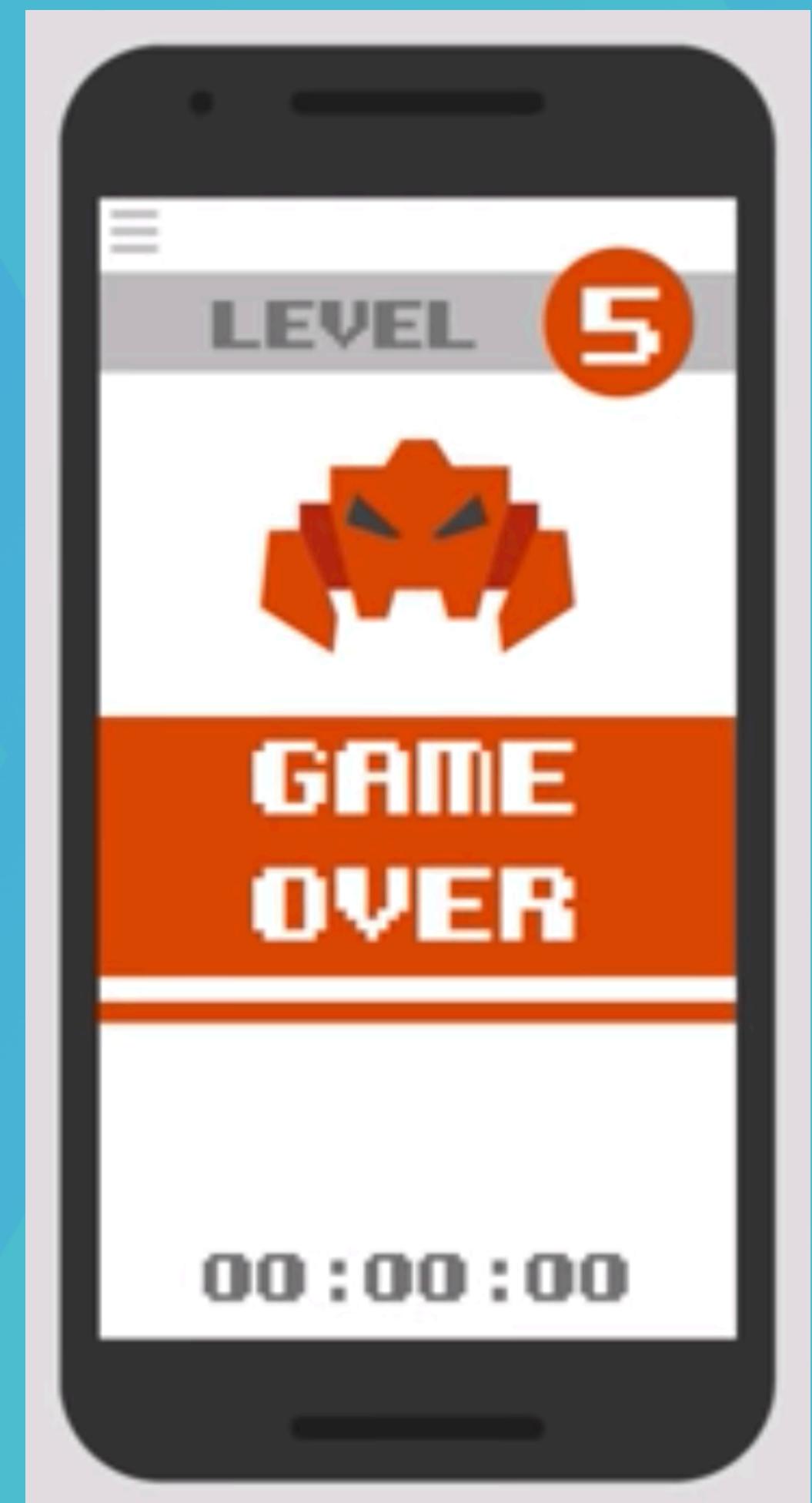
twitch.tv/dancojocar

youtube.com/dancojocar



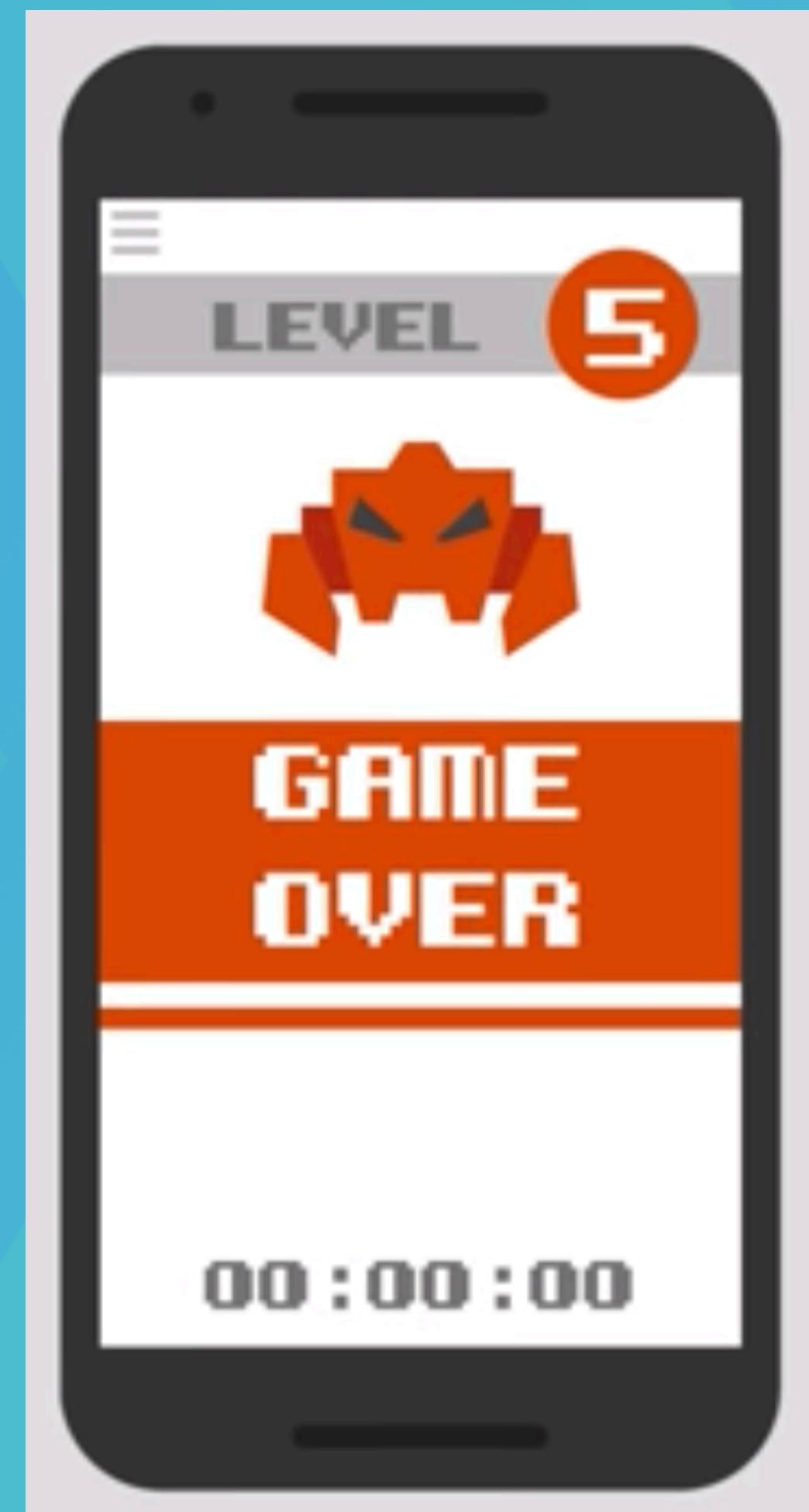
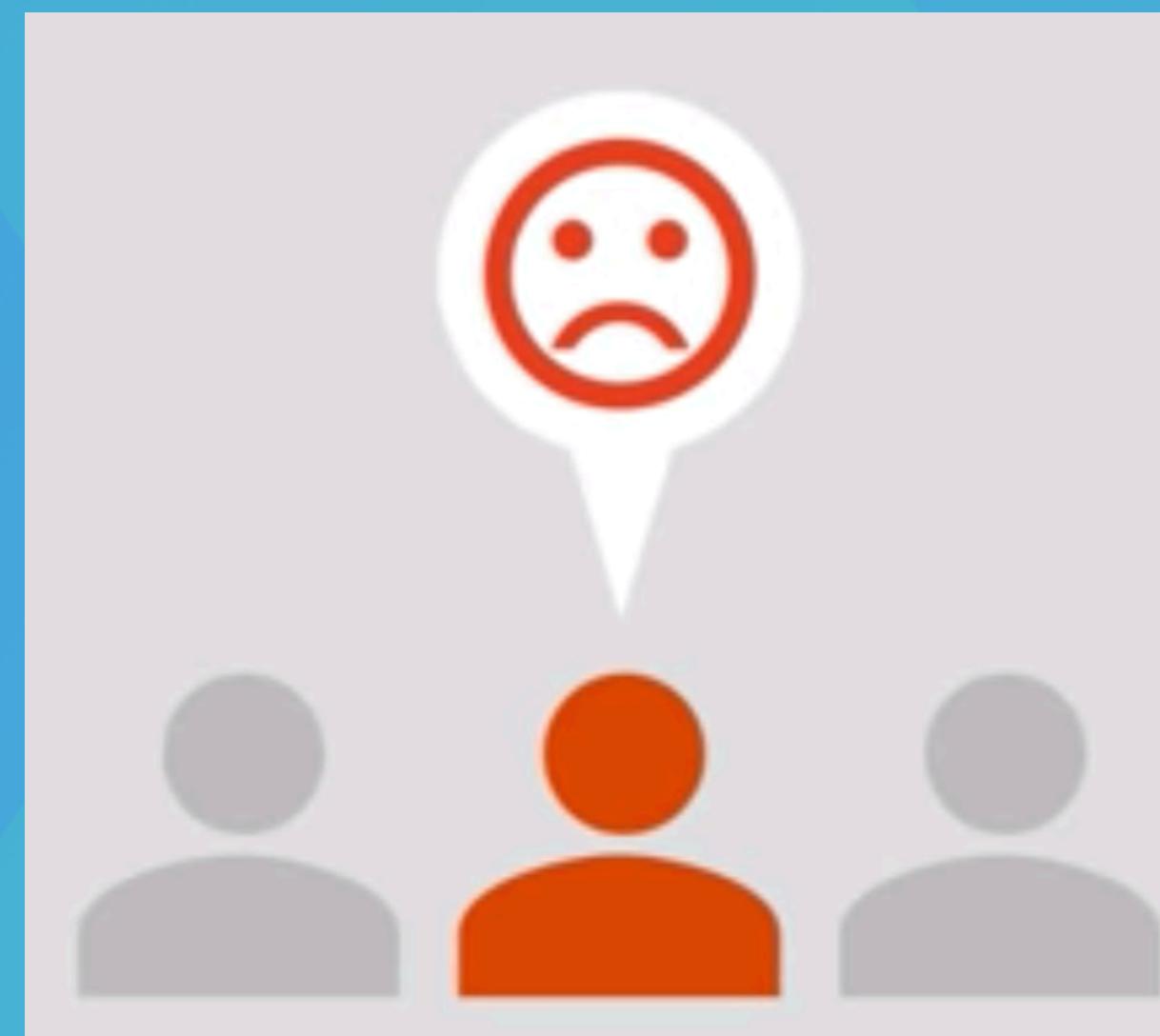
twitch.tv/dancojocar

youtube.com/dancojocar



twitch.tv/dancojocar

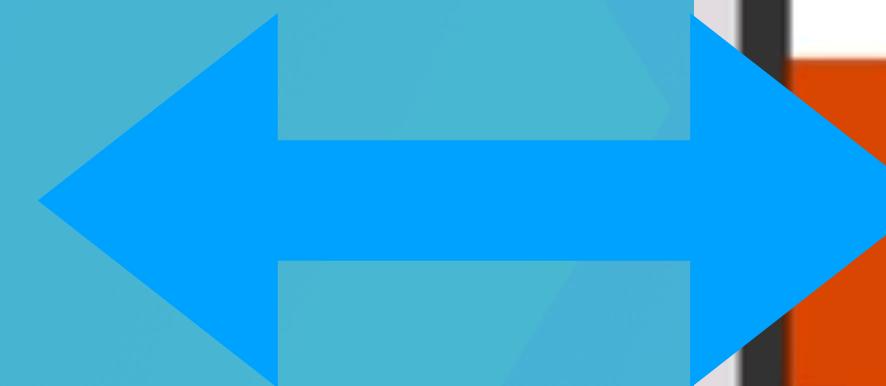
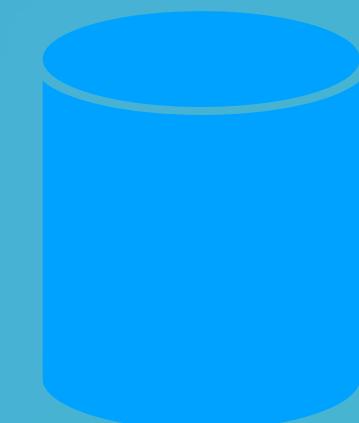
youtube.com/dancojocar



twitch.tv/dancojocar

youtube.com/dancojocar

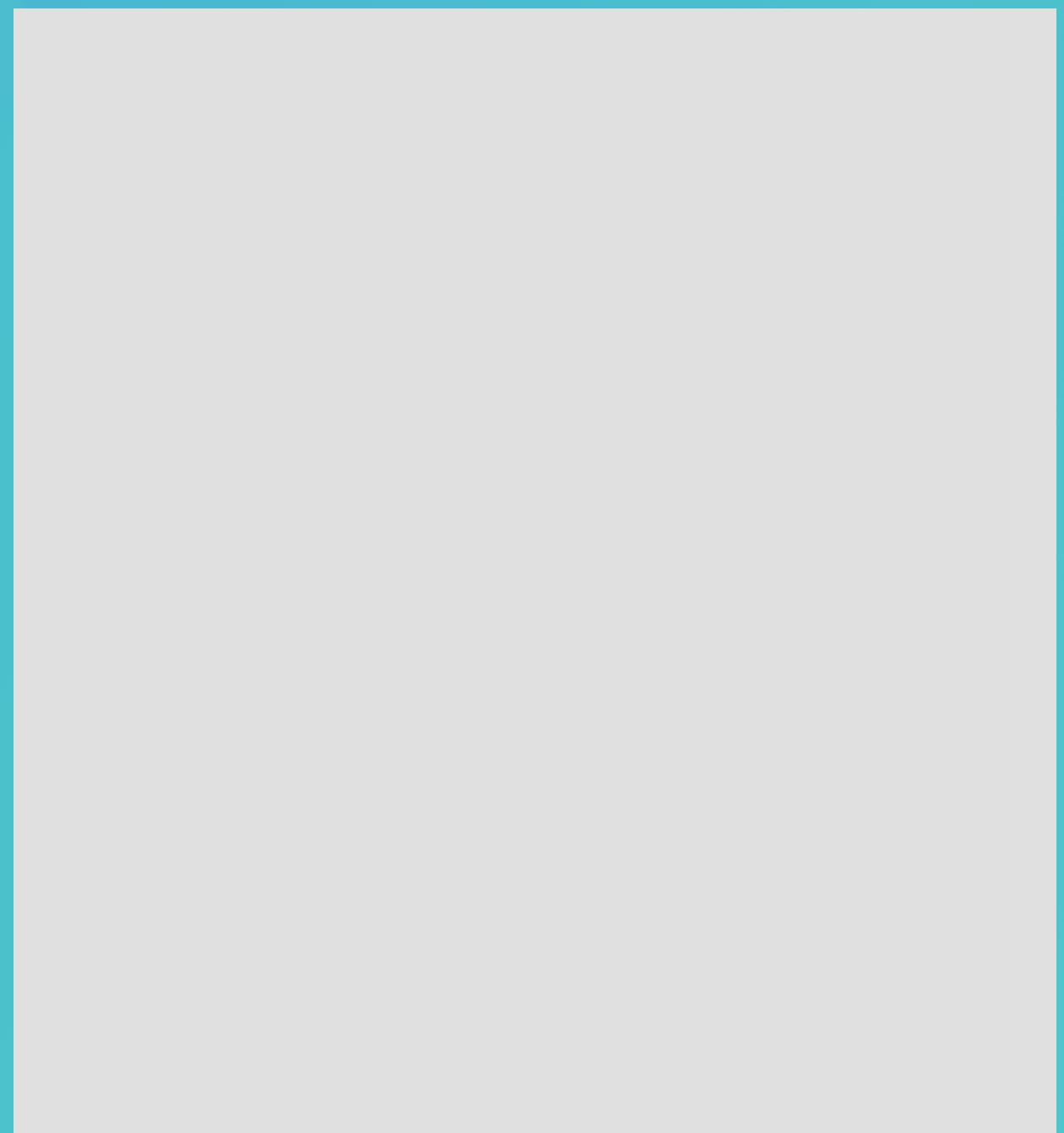
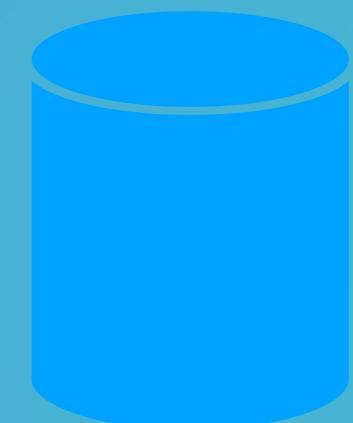
Realtime



twitch.tv/dancojocar

youtube.com/dancojocar

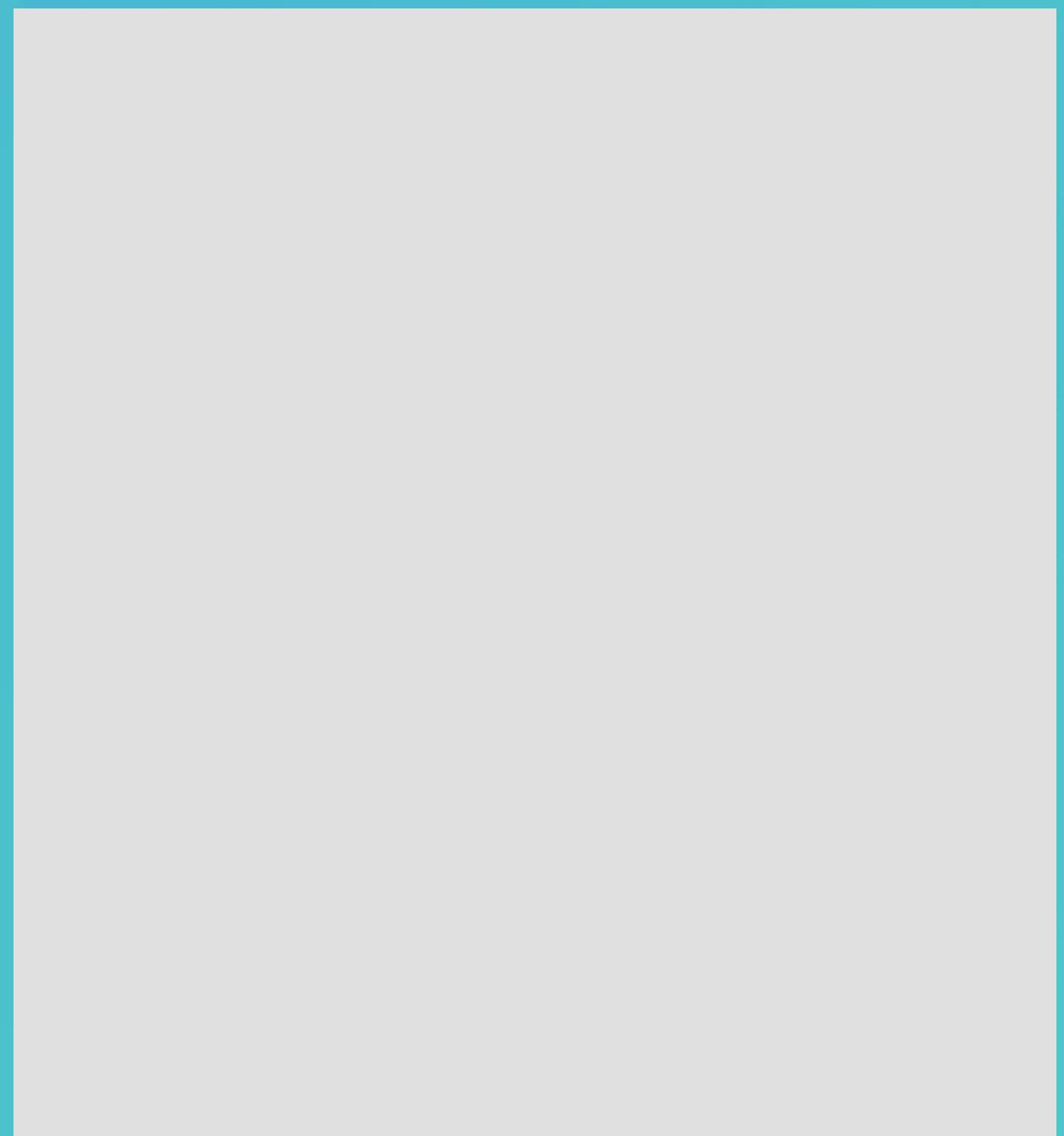
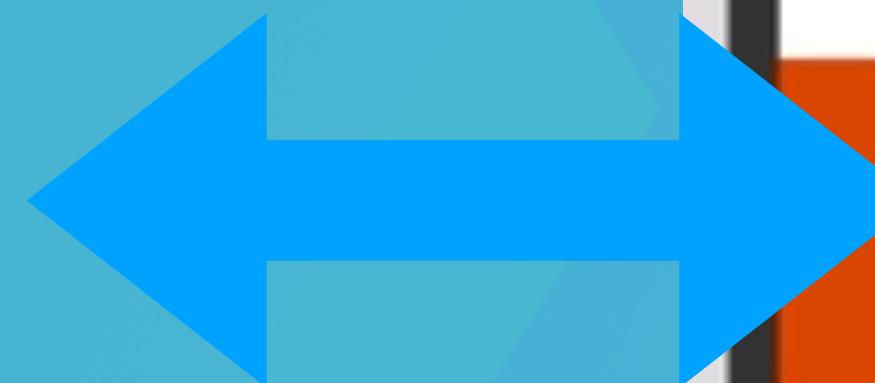
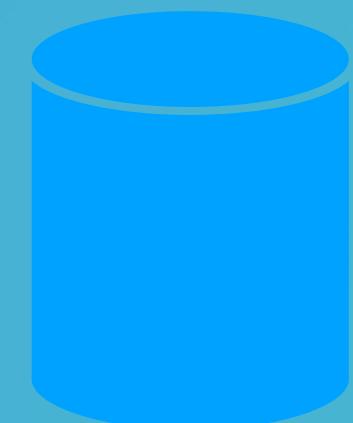
Realtime



twitch.tv/dancojocar

youtube.com/dancojocar

Realtime



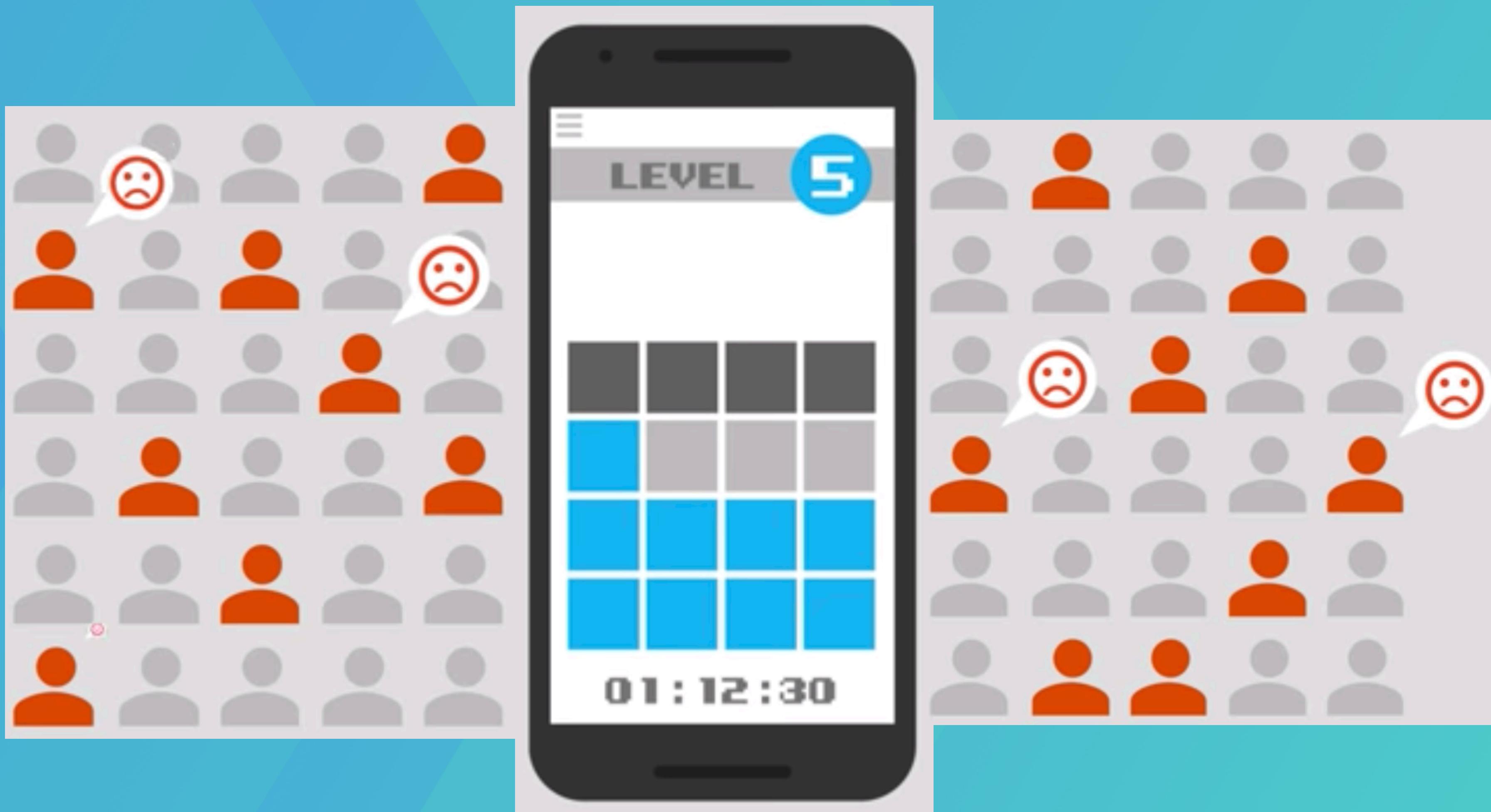
twitch.tv/dancojocar

youtube.com/dancojocar



twitch.tv/dancojocar

youtube.com/dancojocar



twitch.tv/dancojocar

youtube.com/dancojocar

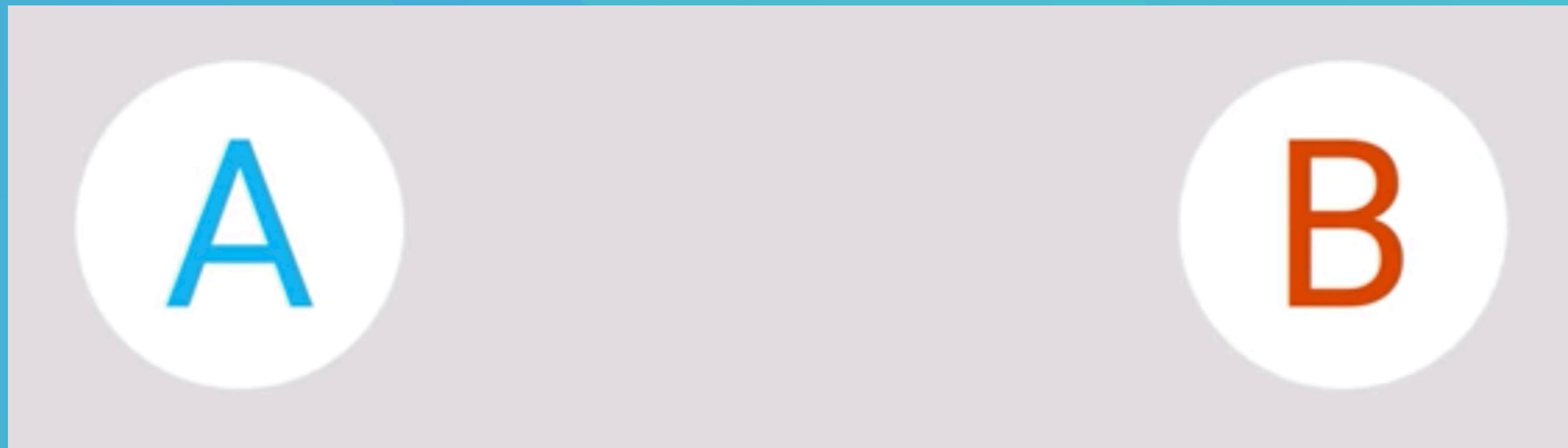


A



B

Remote Config



Using Remote Config



Add Firebase to your app

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

Using Remote Config

1

Add Firebase to your app

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

2

Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

Using Remote Config

1

Add Firebase to your app

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

2

Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

3

Set in-app default parameter values

Using Remote Config

1

Add Firebase to your app

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

2

Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

3

Set in-app default parameter values

4

Get parameter values to use in your app

Using Remote Config

- 1 Add Firebase to your app

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

- 2 Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

- 3 Set in-app default parameter values

- 4 Get parameter values to use in your app

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

Using Remote Config

- 1 Add Firebase to your app

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

- 2 Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

- 3 Set in-app default parameter values

- 4 Get parameter values to use in your app

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

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

DEMO

Using Remote Config

- 1 Add Firebase to your app

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

- 2 Get the Remote Config singleton object

```
FirebaseRemoteConfig.getInstance()
```

- 3 Set in-app default parameter values

- 4 Get parameter values to use in your app

- 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.



Make \$

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



twitch.tv/dancojocar

youtube.com/dancojocar

AdMob

Import the Mobile Ads SDK

```
implementation 'com.google.firebaseio:firebase-ads:19.6.0'
```

AdMob

Import the Mobile Ads SDK

```
implementation 'com.google.firebaseio:firebase-ads:19.6.0'
```

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>
```

AdMob

Import the Mobile Ads SDK

```
implementation 'com.google.firebaseio:firebase-ads:19.6.0'
```

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, "YOUR_ADMOB_APP_ID")
}
```

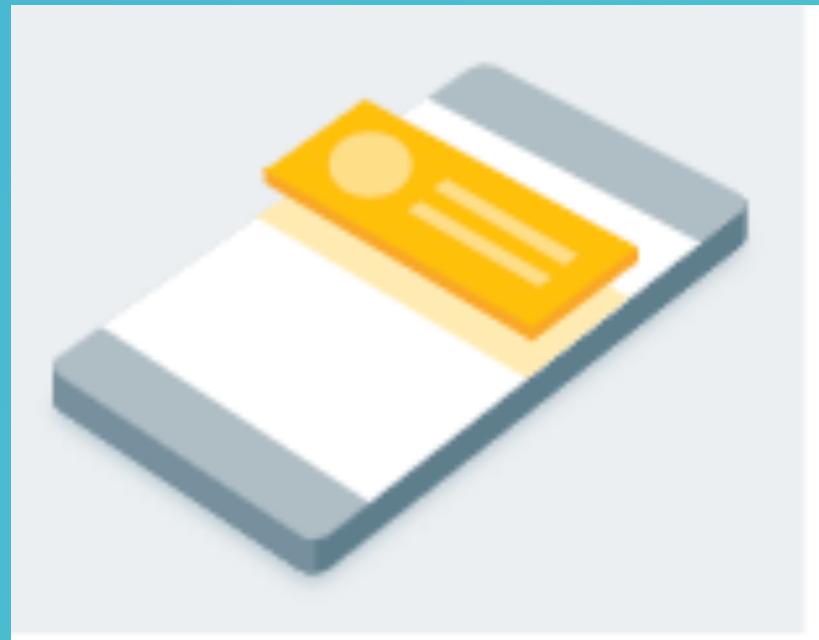
twitch.tv/dancojocar

youtube.com/dancojocar

Ad Formats

Ad Formats

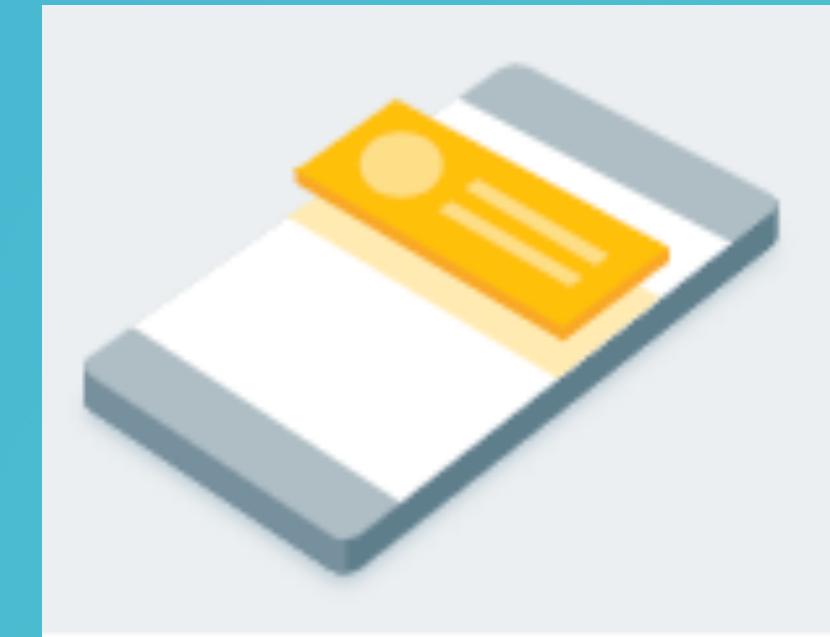
- Banner



```
<com.google.android.gms.ads.AdView  
    xmlns:ads="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/adView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerHorizontal="true"  
    android:layout_alignParentBottom="true"  
    ads:adSize="BANNER"  
    ads:adUnitId="ca-app-pub-3940256099942544/6300978111">  
</com.google.android.gms.ads.AdView>
```

Ad Formats

- Banner



```
<com.google.android.gms.ads.AdView  
    xmlns:ads="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/adView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_centerHorizontal="true"  
    android:layout_alignParentBottom="true"  
    ads:adSize="BANNER"  
    ads:adUnitId="ca-app-pub-3940256099942544/6300978111">  
</com.google.android.gms.ads.AdView>
```

```
val adView = AdView(this)  
adView.adSize = AdSize.BANNER  
adView.adUnitId = "ca-app-pub-3940256099942544/6300978111"  
// TODO: Add adView to your view hierarchy.
```

Ad Formats

- Banner
- Interstitial



```
class MainActivity : Activity() {

    private lateinit var mInterstitialAd: InterstitialAd

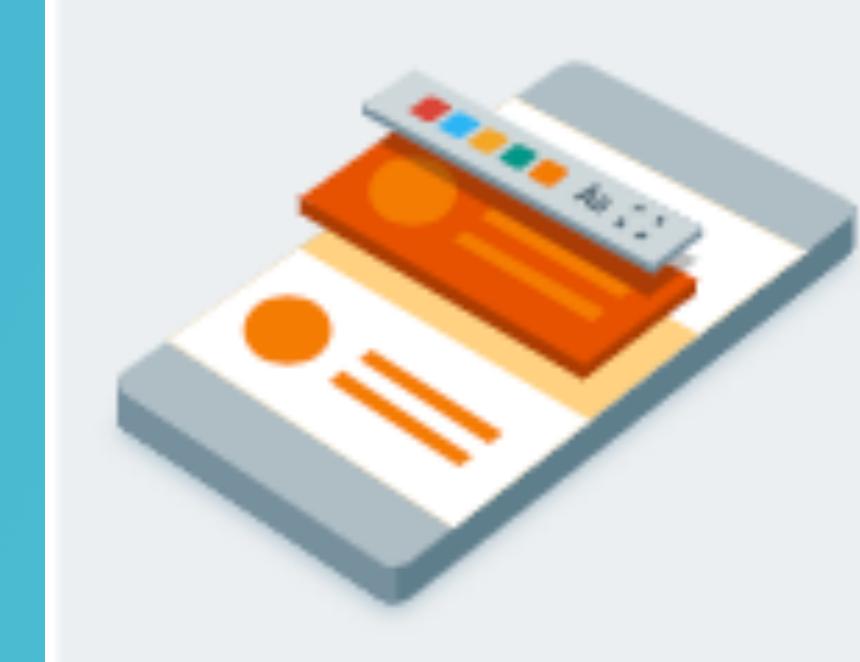
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        MobileAds.initialize(this,
            "ca-app-pub-3940256099942544~3347511713" )

        mInterstitialAd = InterstitialAd(this)
        mInterstitialAd.adUnitId = "ca-app-pub-3940256099942544/1033173712"
        mInterstitialAd.loadAd(AdRequest.Builder().build())
    }
}
```

Ad Formats

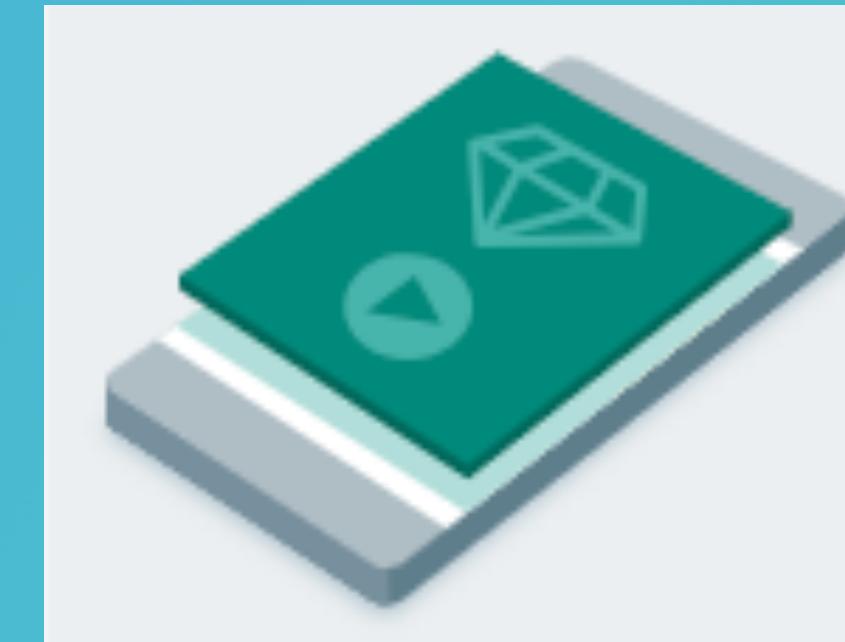
- Banner
- Interstitial
- Native



```
val adLoader = AdLoader.Builder(this, "ca-app-pub-3940256099942544/2247696110")
    .forUnifiedNativeAd { ad : UnifiedNativeAd ->
        // Show the ad.
    }
    .withAdListener(object : AdListener() {
        override fun onAdFailedToLoad(errorCode: Int) {
            // Handle the failure by logging, altering the UI, and so on.
        }
    })
    .withNativeAdOptions(NativeAdOptions.Builder()
        // Methods in the NativeAdOptions.Builder class can be
        // used here to specify individual options settings.
        .build())
    .build()
```

Ad Formats

- Banner
- Interstitial
- Native
- Rewarded Video



```
import com.google.android.gms.ads.AdRequest
import com.google.android.gms.ads.MobileAds
import com.google.android.gms.ads.reward.RewardedVideoAd

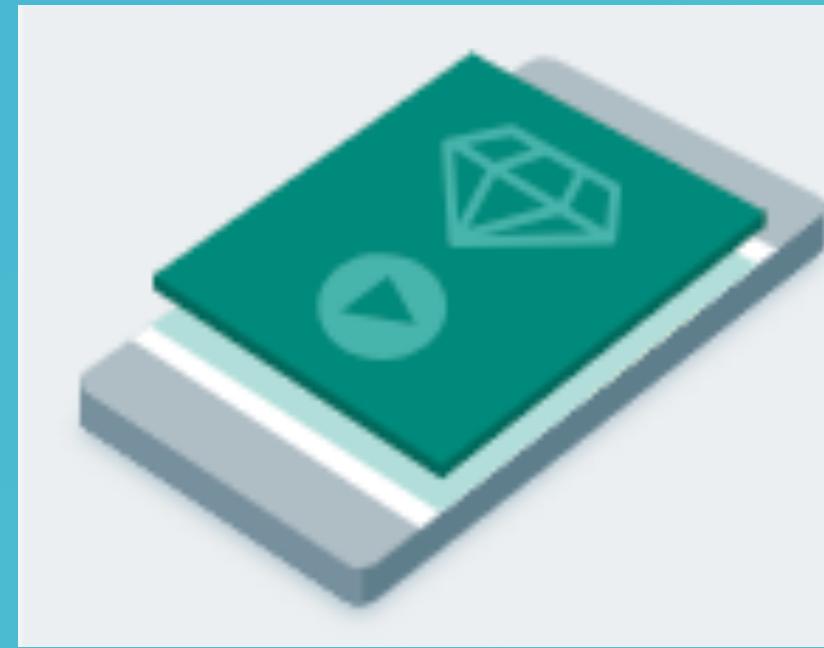
class MainActivity : AppCompatActivity(), RewardedVideoAdListener {

    private lateinit var mRewardedVideoAd: RewardedVideoAd

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        MobileAds.initialize(this, "ca-app-pub-3940256099942544~3347511713")

        // Use an activity context to get the rewarded video instance.
    }
}
```

Ad Formats



DEMO

```
import com.google.android.gms.ads.AdRequest
import com.google.android.gms.ads.MobileAds
import com.google.android.gms.ads.reward.RewardedVideoAd

class MainActivity : AppCompatActivity(), RewardedVideoAdListener {

    private lateinit var mRewardedVideoAd: RewardedVideoAd

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        MobileAds.initialize(this, "ca-app-pub-3940256099942544~3347511713")

        // Use an activity context to get the rewarded video instance.
        mRewardedVideoAd = MobileAds.getRewardedVideoAdInstance(this)
        mRewardedVideoAd.rewardedVideoAdListener = this
    }

    ...
}
```

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.

