THE UNIVERSITY OF CHICAGO

MPCS 51033 • SPRING 2017 • SESSION 4

BACKENDS FOR MOBILE APPLICATIONS

CLASS NEWS

CLASS NEWS

- Office hours tomorrow from 10-11:30 in Young 308

CLASS NEWS

- Talk about Firebase messaging with other messaging services later in the course
  - Has more to do with iOS APN and handling notifications than Firebase

DEEP DIVE INTO SHARDING

SHARDING

- Popular technique for specific uses, software and platform

Flickr Architecture

TUESDAY, NOVEMBER 13, 2007 AT 6:04PM

Update: Flickr hits 2 Billion photos served. That's a lot of hamburgers.

Flickr is both my favorite bird and the web's leading photo sharing site. Flickr has an amazing challenge, they must handle a vast sea of ever expanding new content, ever increasing legions of users, and a constant stream of new features, all while providing excellent performance. How do they do it?

Site: http://www.flickr.com

Information Sources

- Flickr and PHP (an early document)
- Capacity Planning for LAMP
- Federation at Flickr: Doing Billions of Queries a Day by Dathan Pattishall.
- Building Scalable Web Sites by Cal Henderson from Flickr.
- Database War Stories #3: Flickr by Tim O'Reilly
- Cal Henderson's Talks. A lot of useful PowerPoint presentations.

Platform

Cross-Group Transactions

Transactions and entity groups present a fundamental tension in designing data for the App Engine datastore. If all we cared about was data consistency, we'd put all our data in a single entity group, and every transaction would have a consistent and current view of the entire world—and would be battling with every other simultaneous transaction to commit a change. Conversely, if we wanted to avoid contention as much as possible, we'd keep every entity in its own group and never perform more than one operation in a transaction.

Entity groups are a middle ground. They allow us to define limited sets of data that demand strongly consistent transactional updates, and with some thought we can organize these boundaries so an increase in traffic does not result in an increase in

SHARDING

- Ancestor paths do a lot of the work for us

Keys, Paths, and Ancestors

To create an entity in a group with other entities, you associate it with the key of another entity from that group. One way to do this is to make the existing entity's key the *parent* of the new entity. The key of the parent becomes part of the key of the child. These parent–child relationships form a path of ancestors down to a *root* entity that does not have a parent. Every entity whose key begins with the same root is in the same group, including the root entity itself.

When you create an entity and do not specify a parent, the entity is created in a new group by itself. The new entity is the root of the new group.

We alluded to paths earlier when we discussed keys, so let's complete the picture. An entity's key consists of the path of ancestors in the entity's group, starting from the group's root. Each entity in the path is represented by the entity's kind followed by either the system-assigned numeric ID or the app-assigned string ID. The full path is a sequence of kind and ID pairs.

The following keys represent entities in the same group, because they all have the same root ancestor:

```
MessageBoard, "The_Archonville_Times"

MessageBoard, "The_Archonville_Times" / Message, "first!"

MessageBoard, "The_Archonville_Times" / Message, "pk_fest_aug_21"

MessageBoard, "The_Archonville_Times" / Message, "first!" / Message, "keep_clean"
```

SHARDING

Entity groups are a middle ground. They allow us to define limited sets of data that demand strongly consistent transactional updates, and with some thought we can organize these boundaries so an increase in traffic does not result in an increase in simultaneous updates to a single group. For example, if a user only ever updates an entity group dedicated to that user, then an increase in users will never increase the number of users writing to a given group simultaneously.

- Entity groups are the key

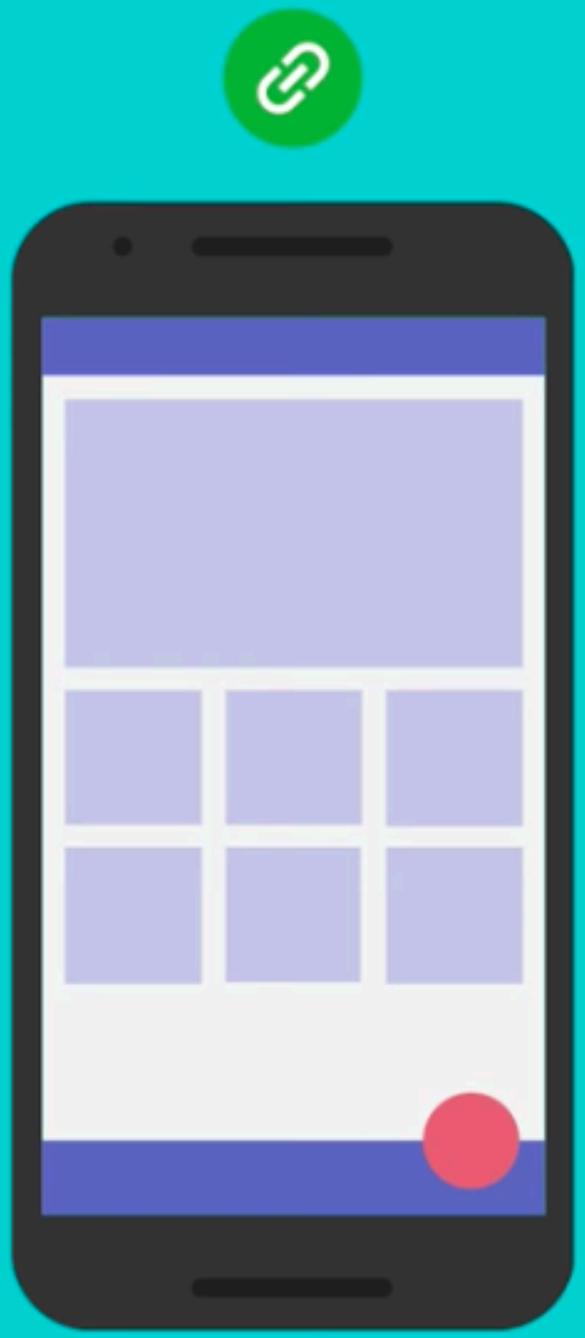
SHARDING

- Take away
  - Sharding is a well-used technique but the implementation specifics are based on problem, use, platform
  - Sharding on App Engine for counters is a great use
  - Take advantage of the design of Datastore

FIREBASE DYNAMIC LINKS

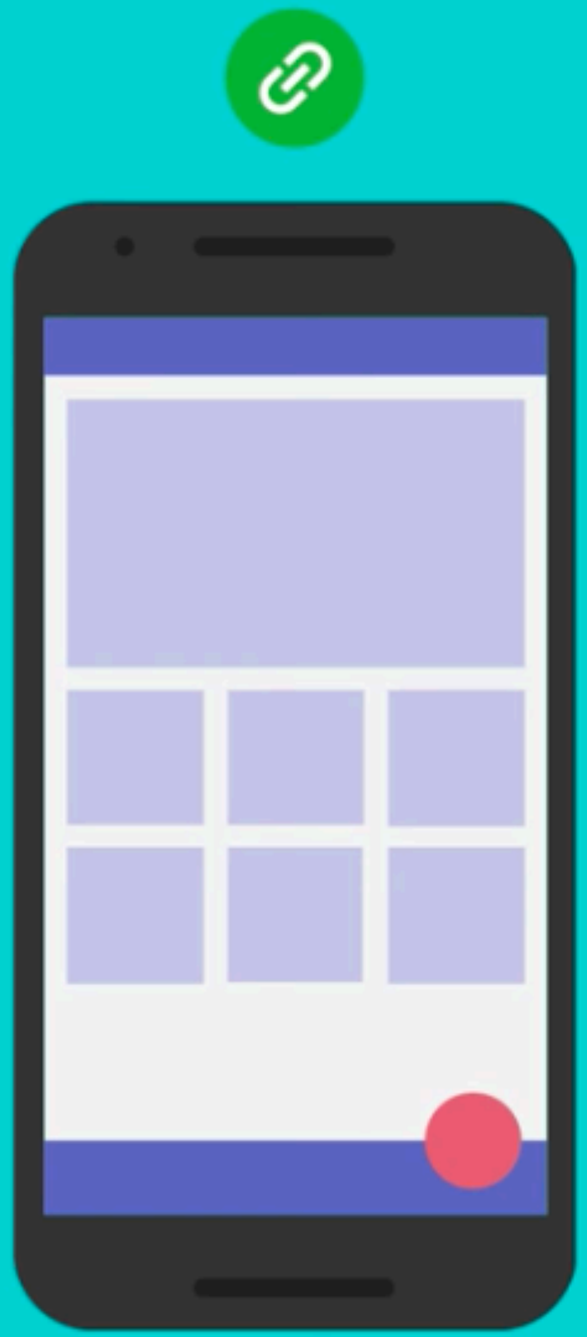
DYNAMIC LINKS

- Firebase dynamic links
  - https://youtu.be/LvY1JMcrPF8?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL



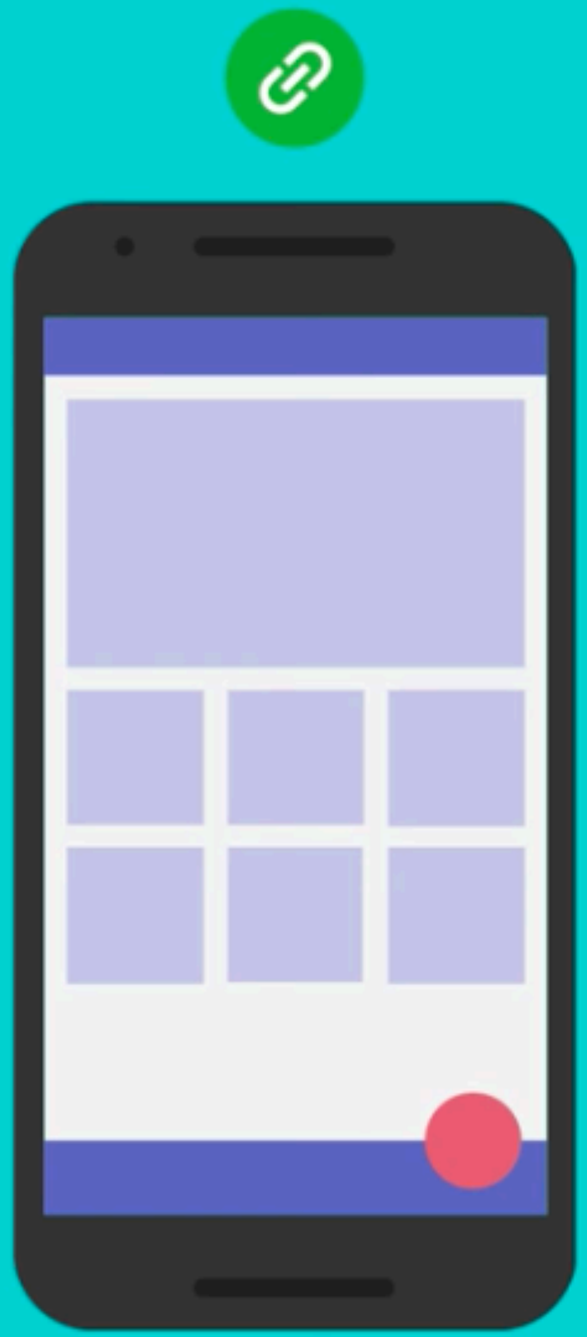
DYNAMIC LINKS

- Dynamic links can be taken directly to the linked content in your native app
- If a user opens the same Dynamic Link in a desktop browser, they can be taken to the equivalent content on your website



DYNAMIC LINKS

- Creating links
  - Firebase console
  - REST API
  - iOS Buider API
  - Programatically

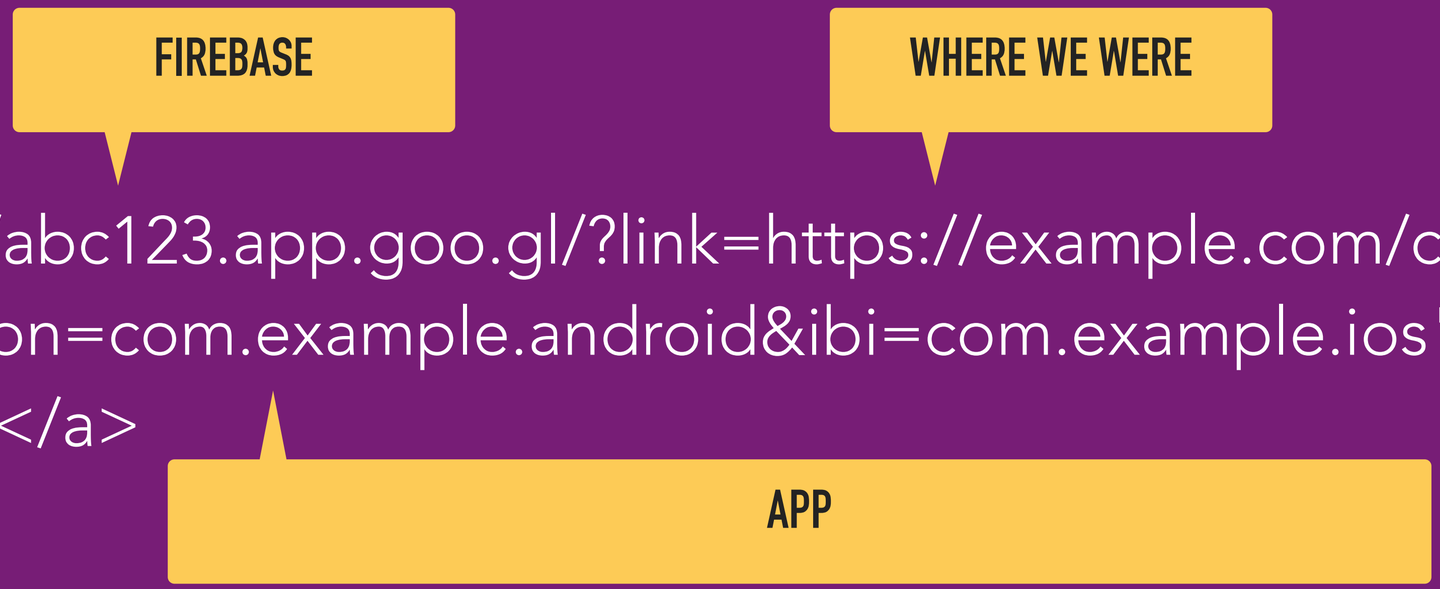


DYNAMIC LINKS

- Implementing deep links

1	Set up Firebase and the Dynamic Links SDK	Enable Firebase Dynamic Links for your Firebase project in the Firebase console. Then, include the Dynamic Links SDK in your app.
2	Create Dynamic Links	You can create Dynamic Links programmatically or by using the Firebase console.
3	Handle Dynamic Links in your app	When your app opens, use the Dynamic Links SDK to check if a Dynamic Link was passed to it. If so, get the link from the Dynamic Link data and handle the link as necessary.
4	View analytics data	Track the performance of your Dynamic Links in the Firebase console.

DYNAMIC LINKS



- Open this page in our app!

DYNAMIC LINKS

DYNAMIC LINKS

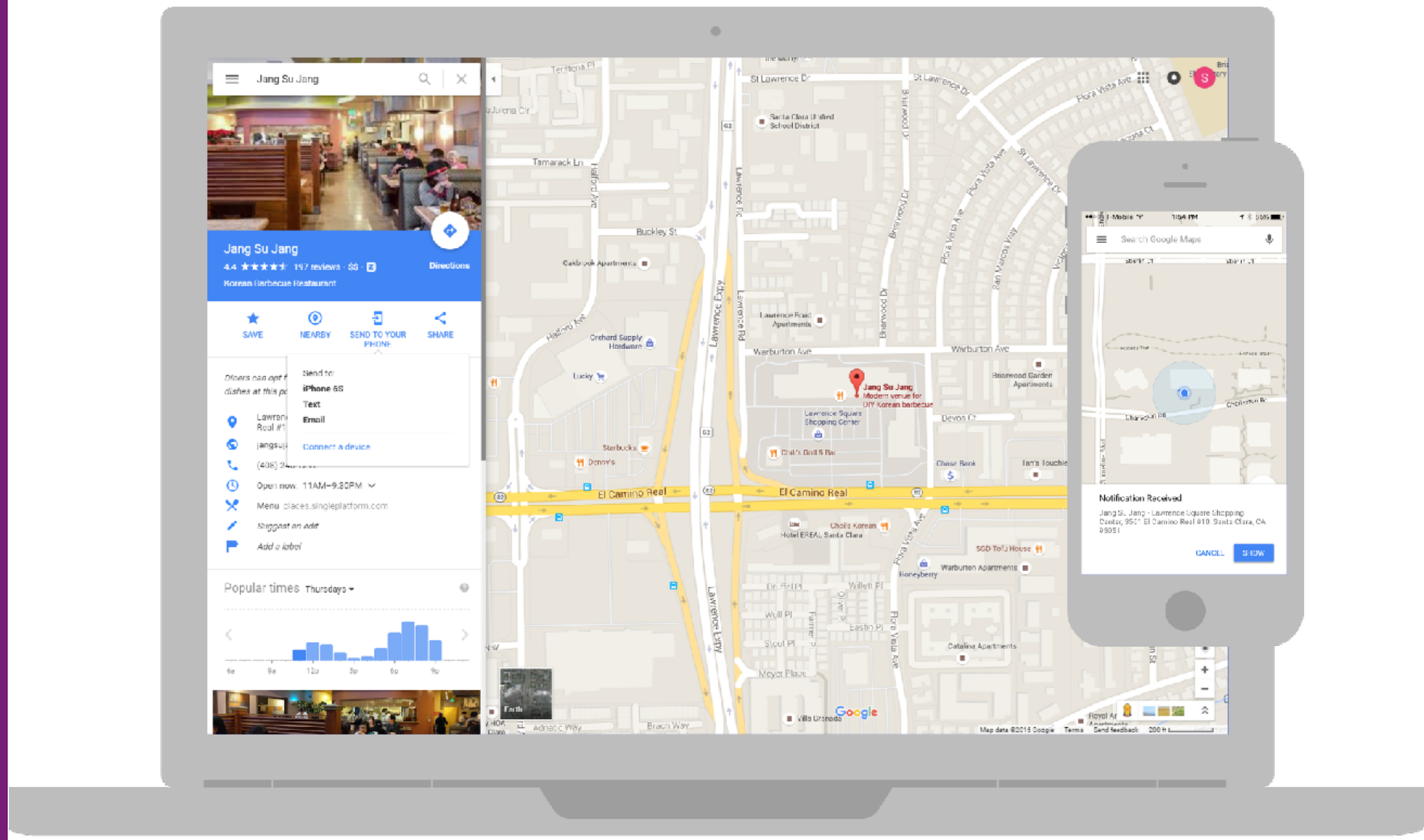
URL SHORTENER

- https://abc123.app.goo.gl/WXYZ

DYNAMIC LINKS

USE CASES

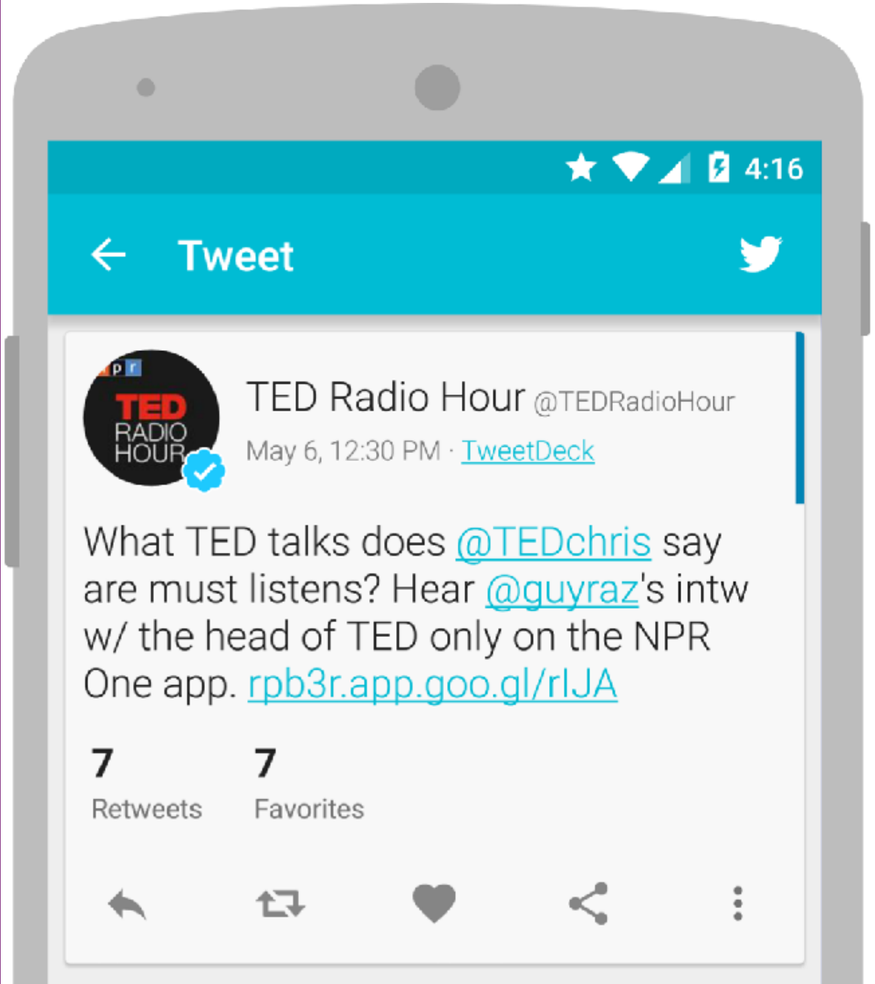
- Converting web (or desktop) users to app users
    - Don't loose place



DYNAMIC LINKS

USE CASES

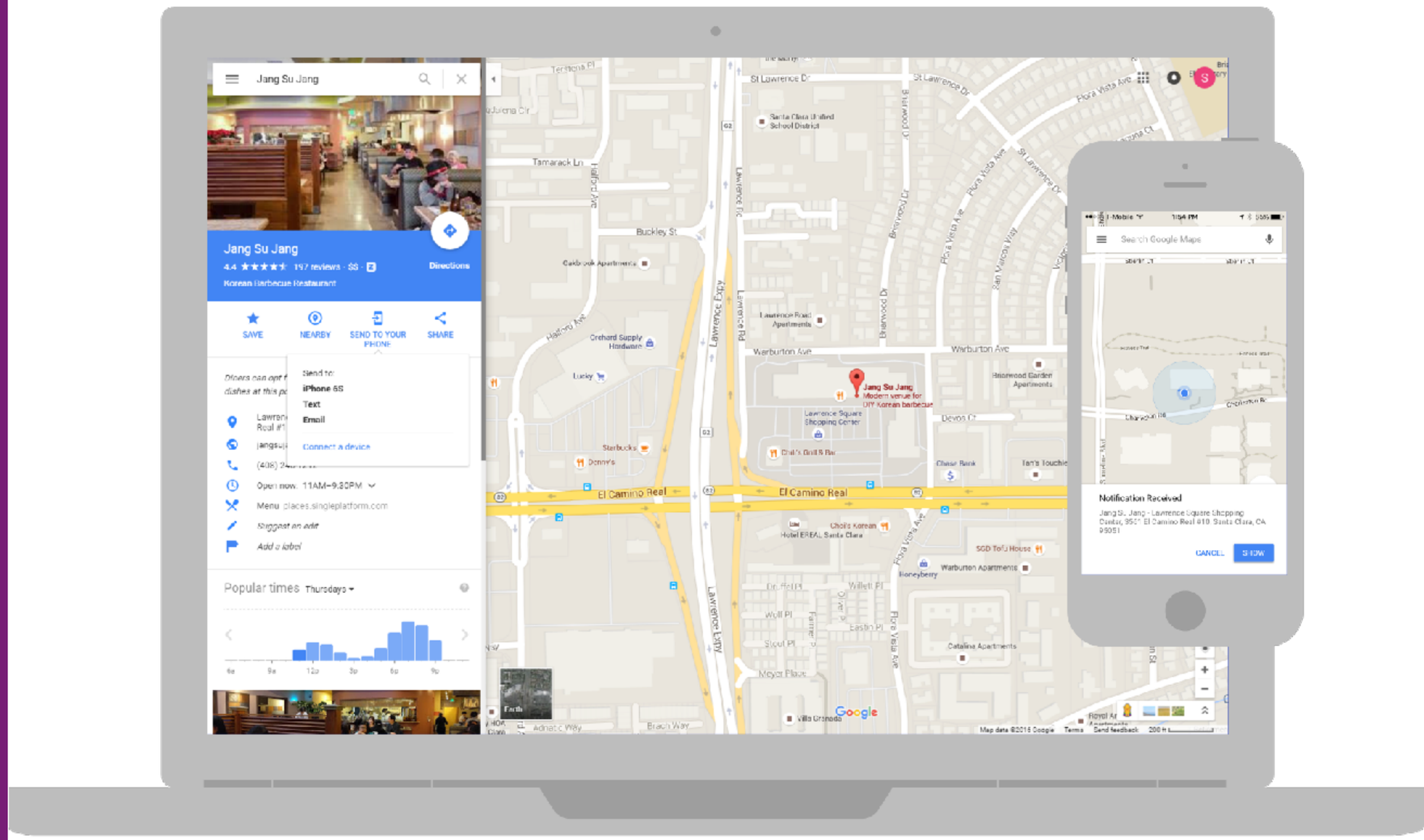
- Send links through social media
  - Direct content links
  - Track engagement



DYNAMIC LINKS

USE CASES

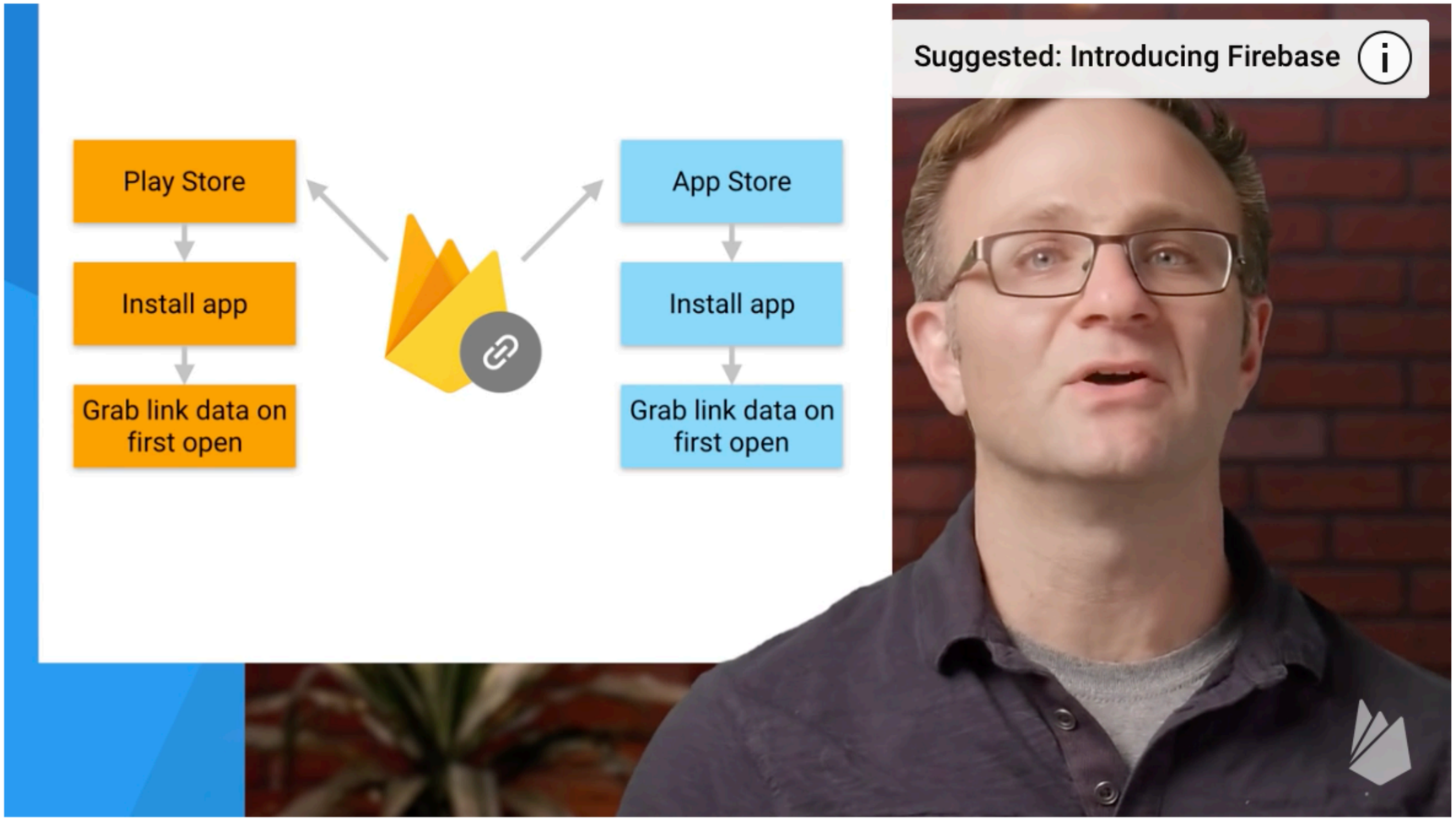
- Send links through social media
  - Direct content links
  - Track engagement



SETTING UP DYNAMICS LINKS

SETTING UP DYNAMICS LINKS

- https://youtu.be/sFPo296OQqk

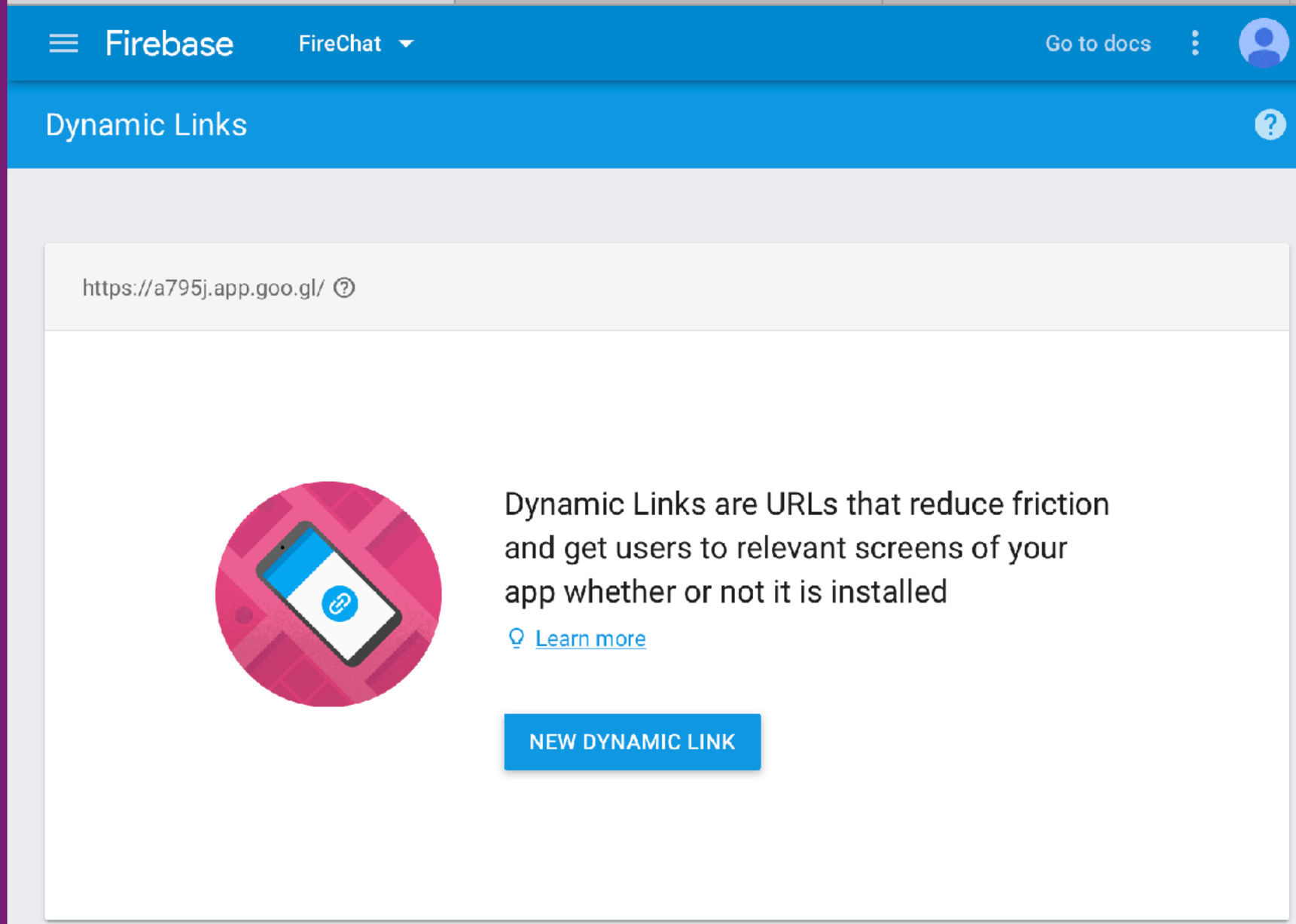


SETTING UP DYNAMICS LINKS

- Custom URL for your application

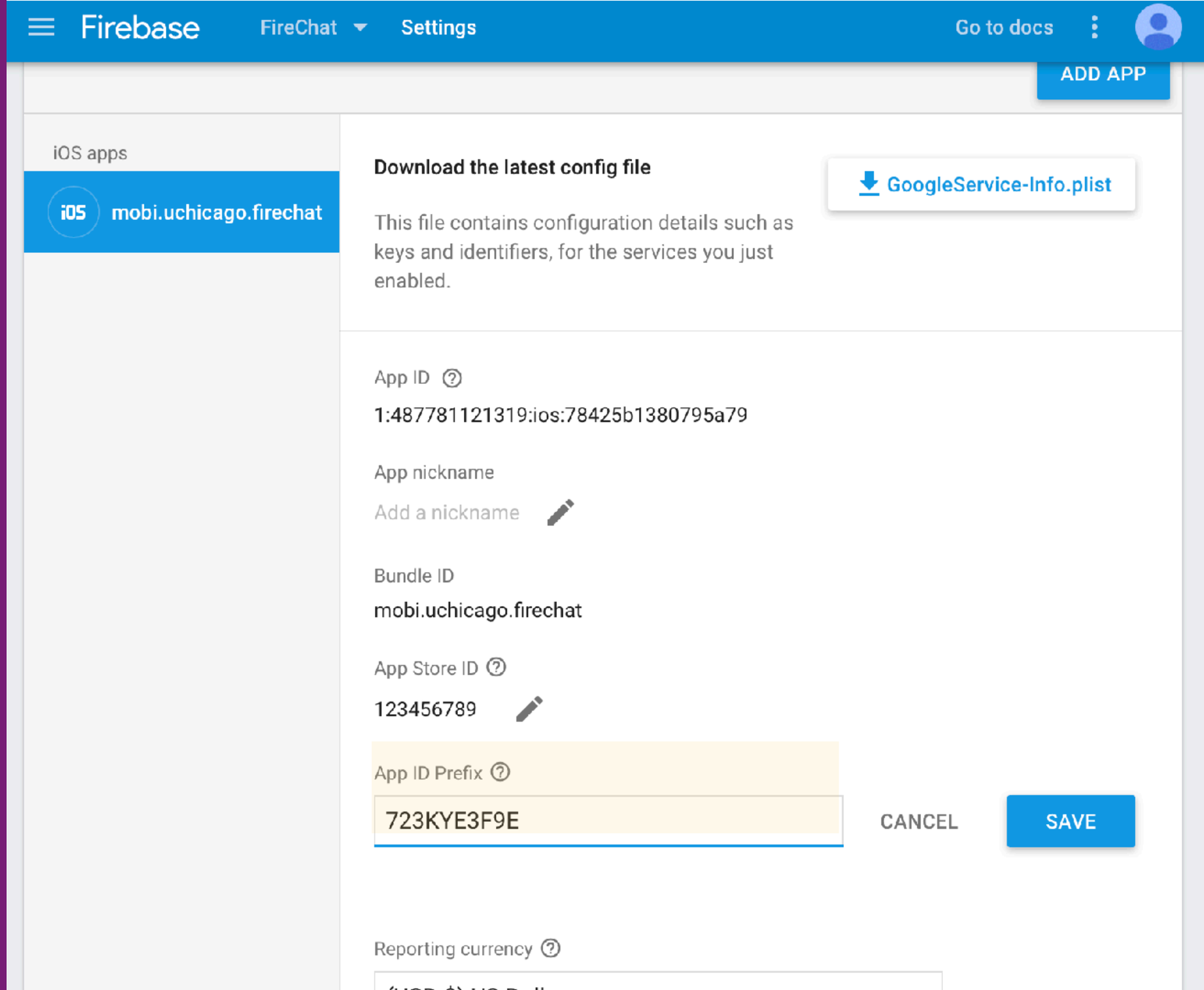
SETTING UP DYNAMICS LINKS

- Set up in the console



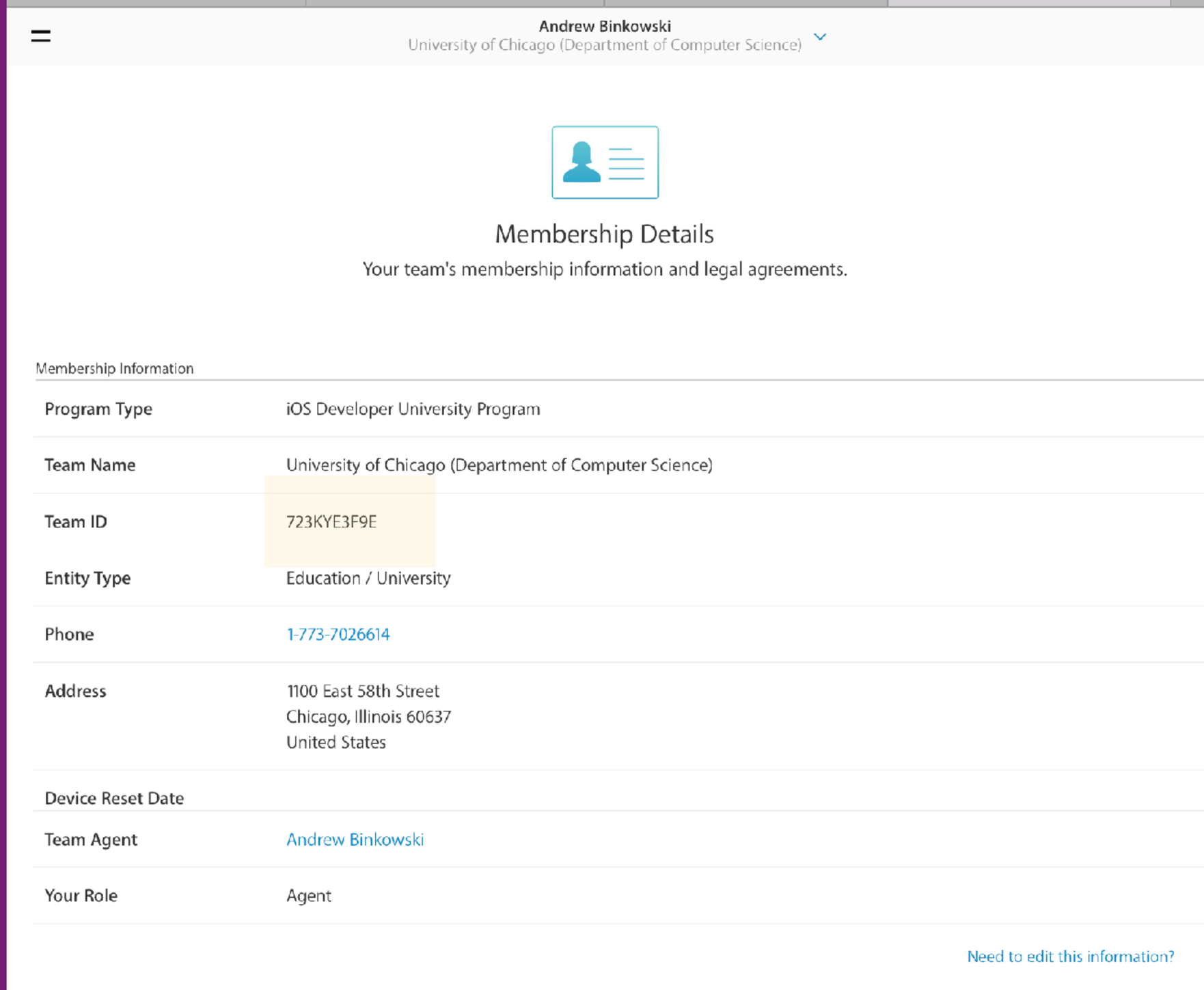
SETTING UP DYNAMICS LINKS

- Project settings



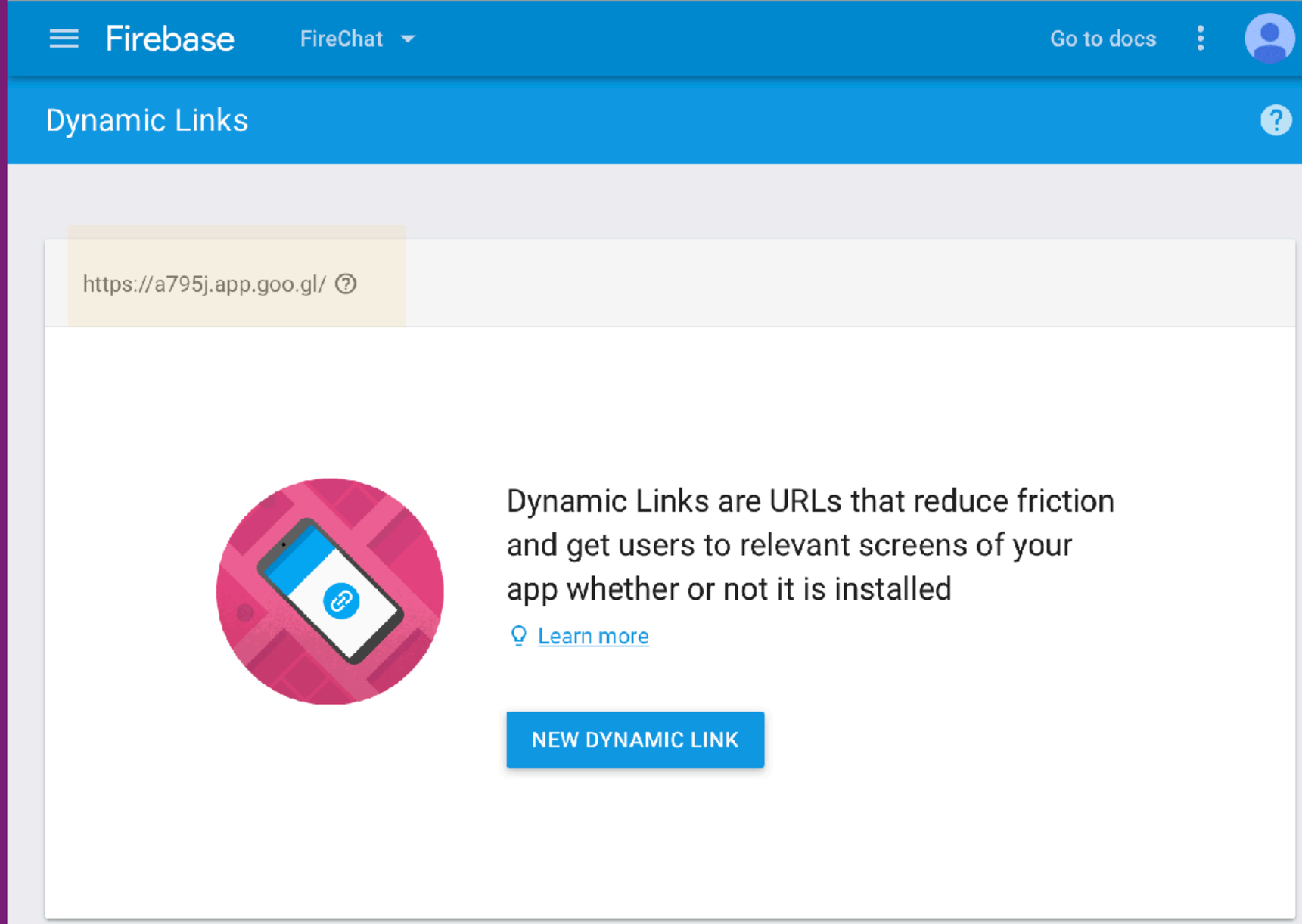
SETTING UP DYNAMICS LINKS

- Apple developer > Membership

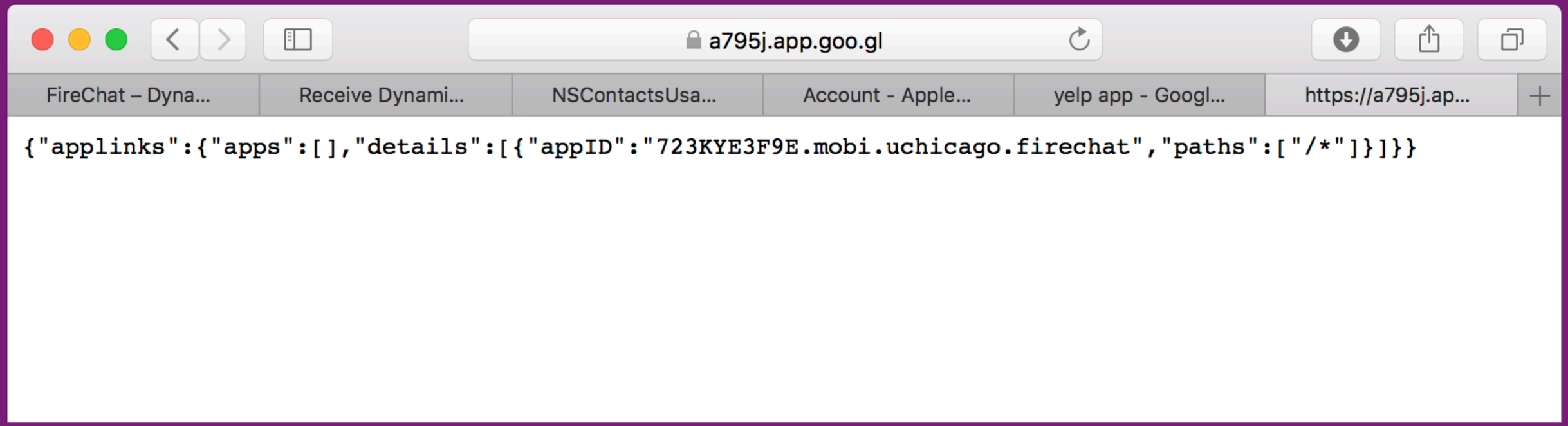


SETTING UP DYNAMICS LINKS

- Dynamic links domain



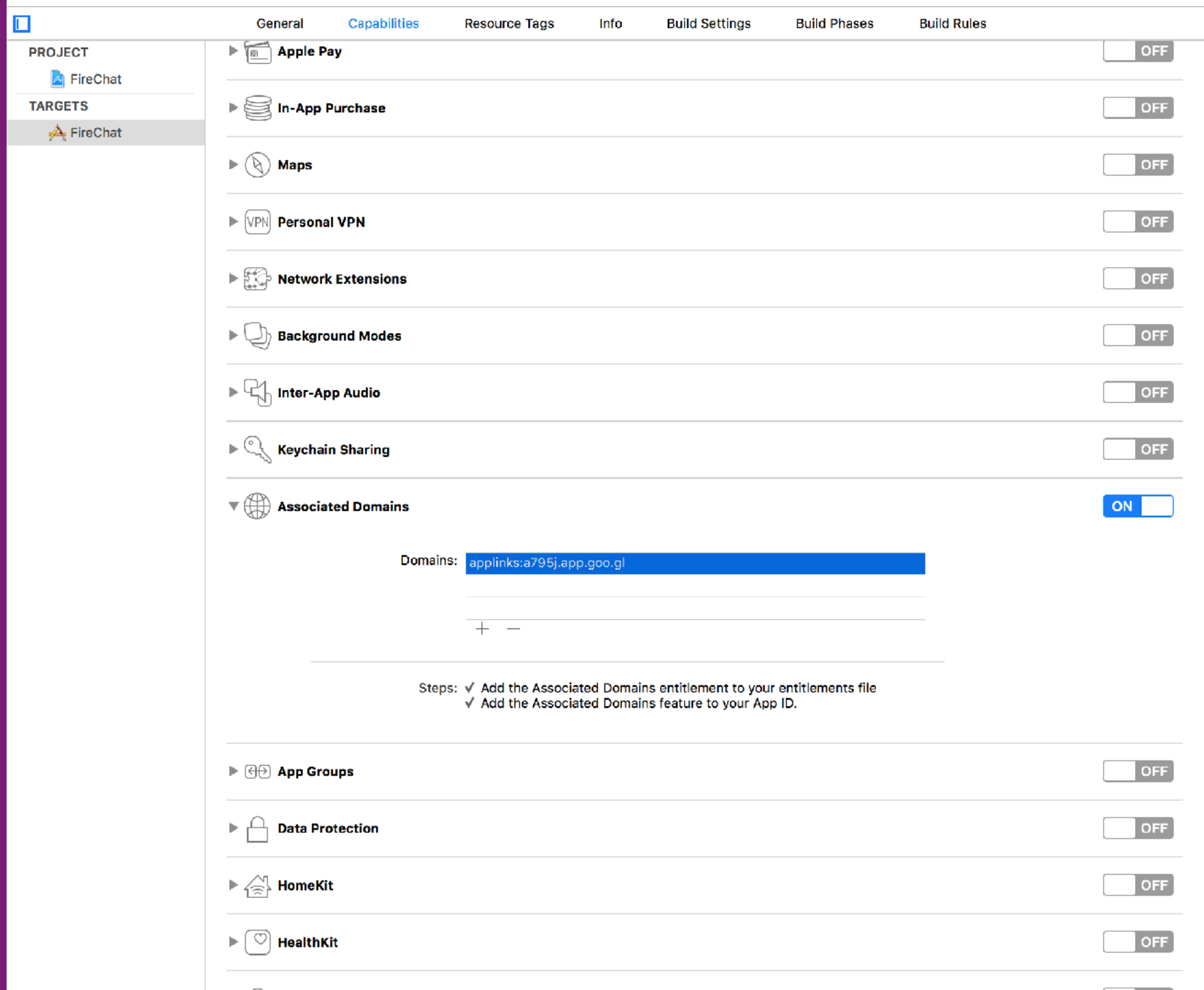
SETTING UP DYNAMICS LINKS



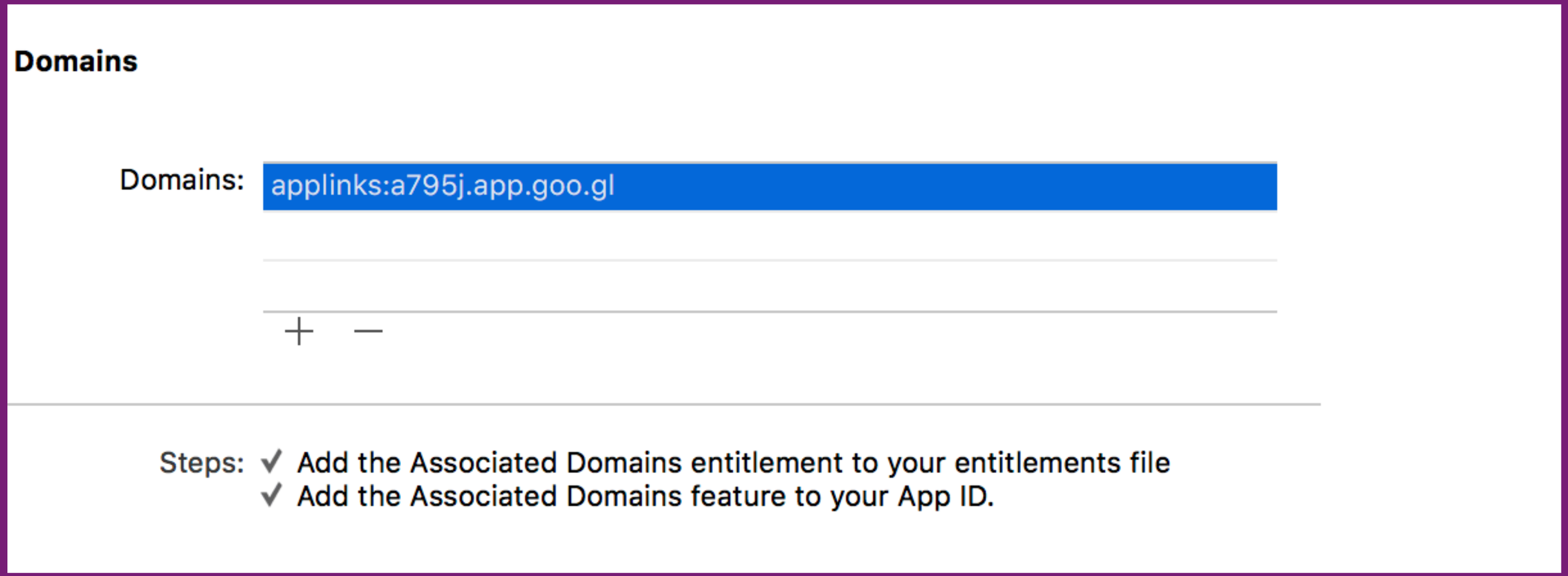
- https://a795j.app.goo.gl/apple-app-site-association
- This URL will redirect to my application

SETTING UP DYNAMICS LINKS

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain

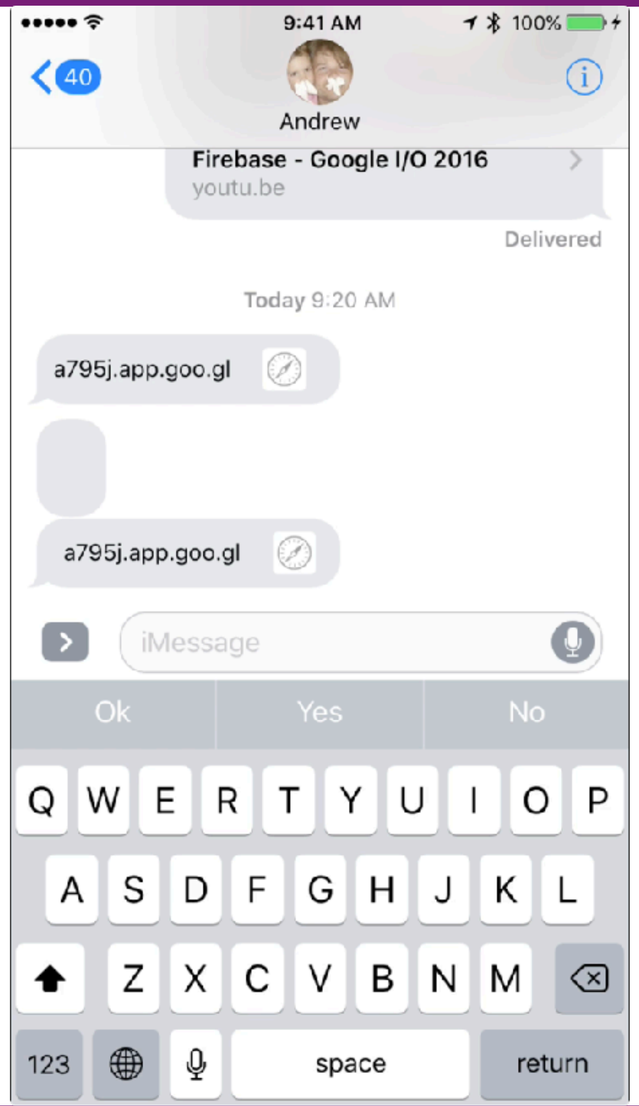


SETTING UP DYNAMICS LINKS

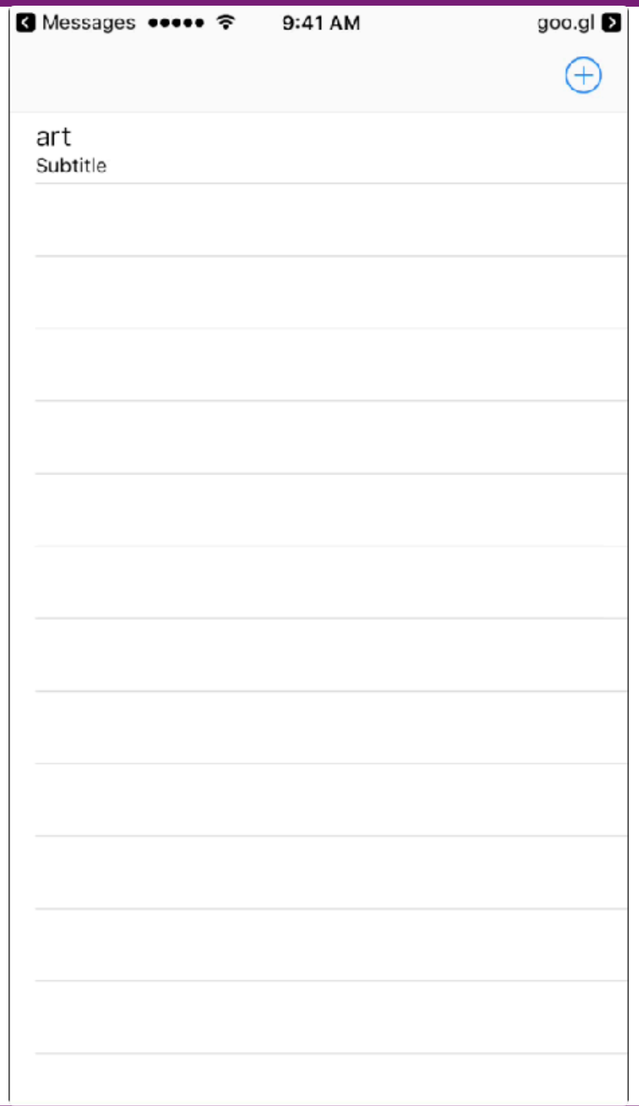


- https://a795j.app.goo.gl

SETTING UP DYNAMICS LINKS



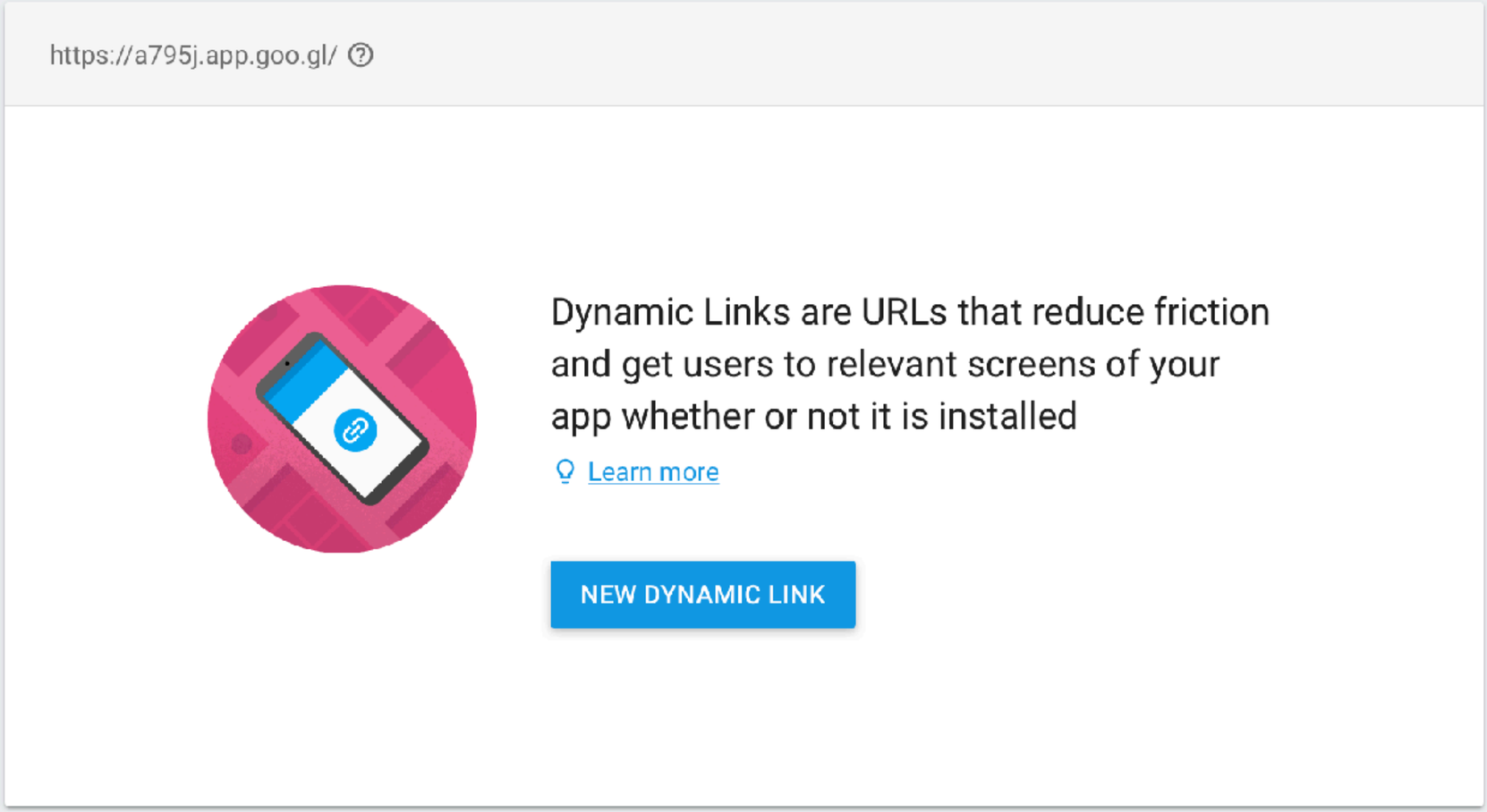
TEST ON DEVICE



CREATING A DYNAMIC LINK

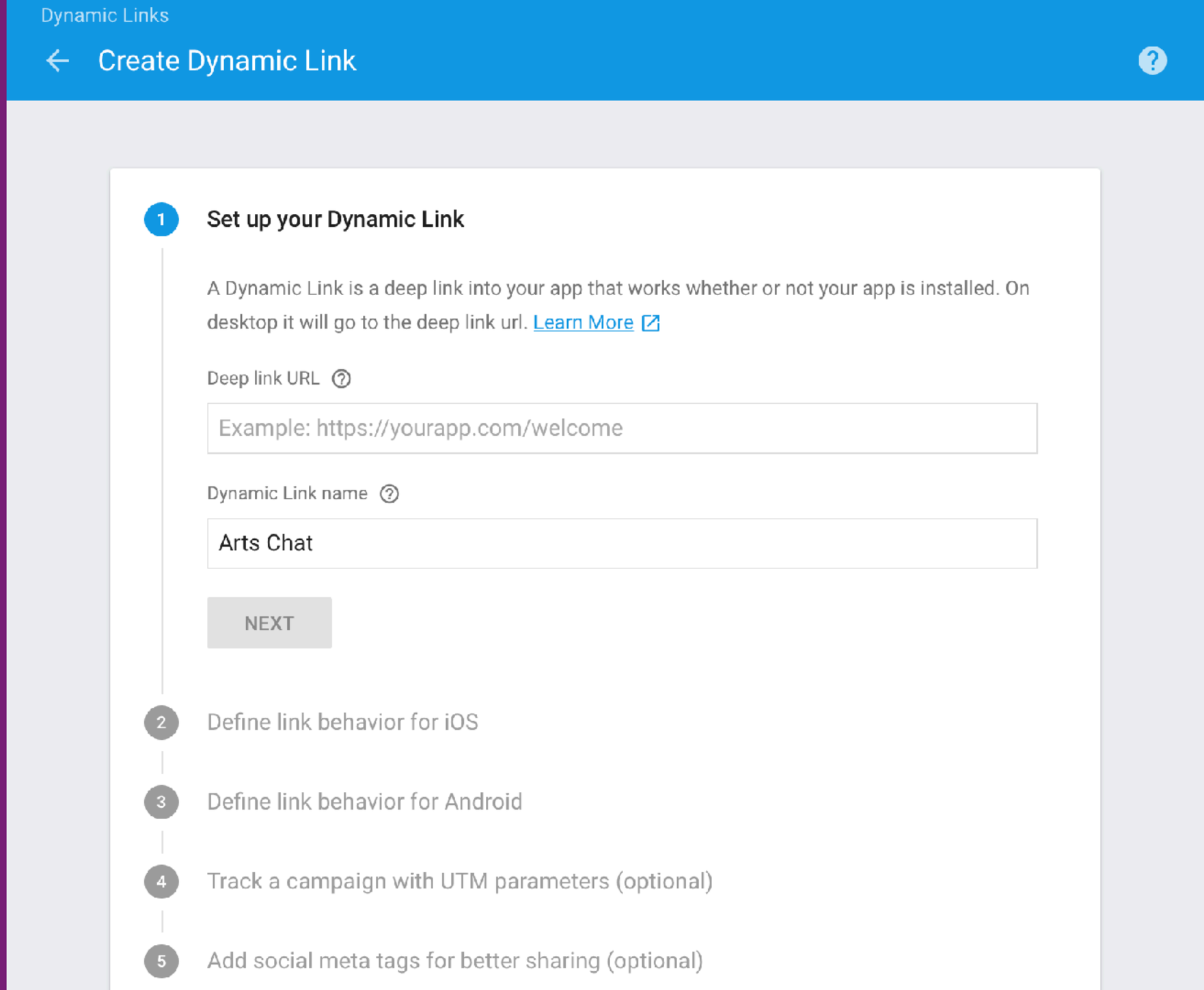
CREATING A DYNAMIC LINK

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain



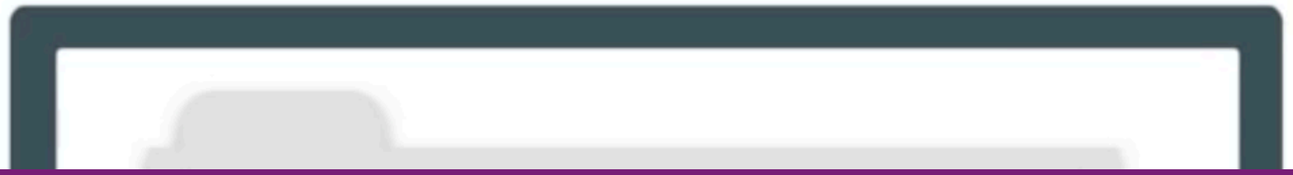
CREATING A DYNAMIC LINK

- Set up iOS app to handled Universal Links
- Firebase is hosting our associated domain



CREATING A DYNAMIC LINK

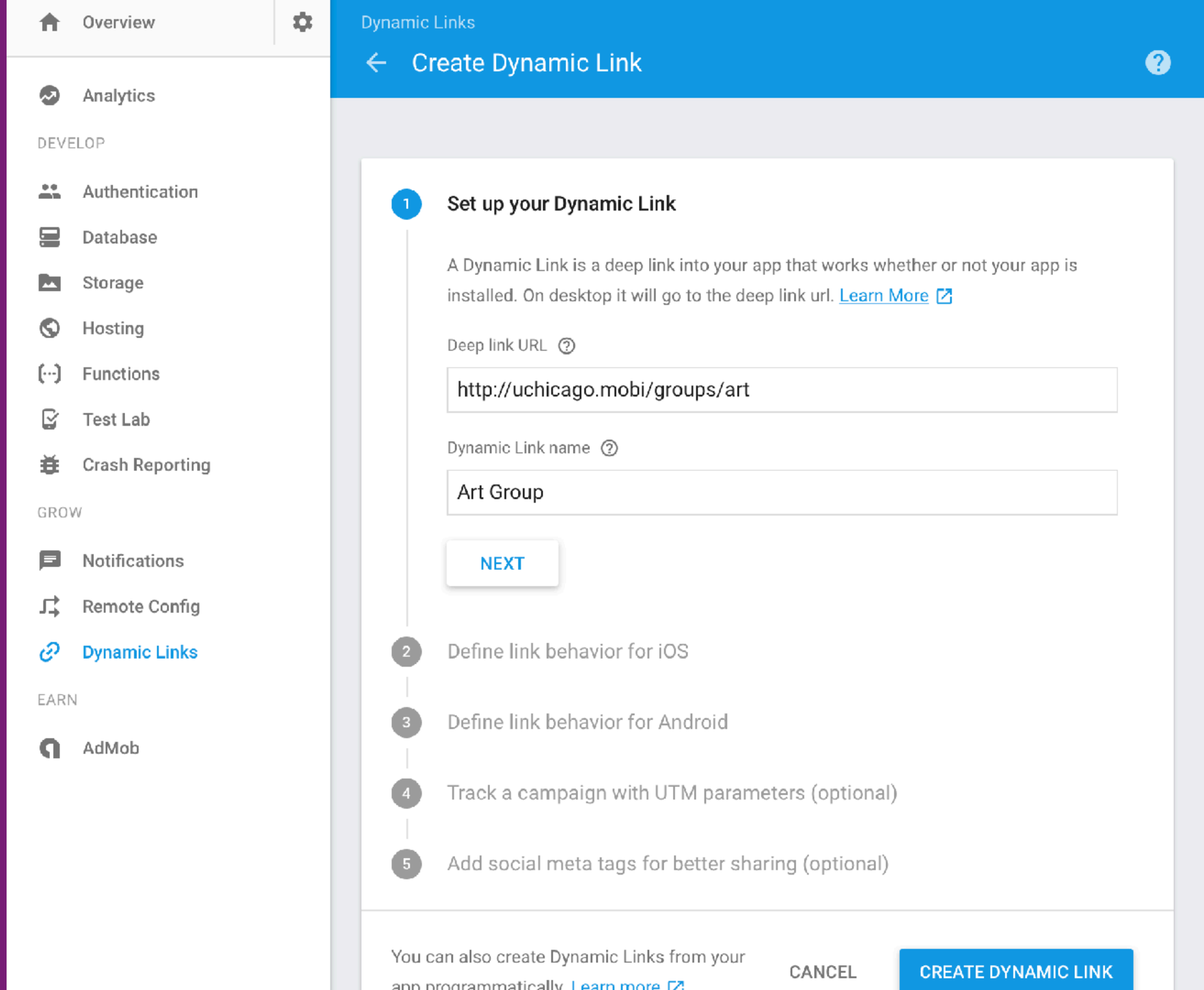
```
https://abcde.app.goo.gl/?
link=http://moviereviews.example.com/reviewID%eD42663
&isi=48151632&ibi=com.example.moviereviews
```



- All data is passed in through URL parameters
- Allows the links to be "universal"

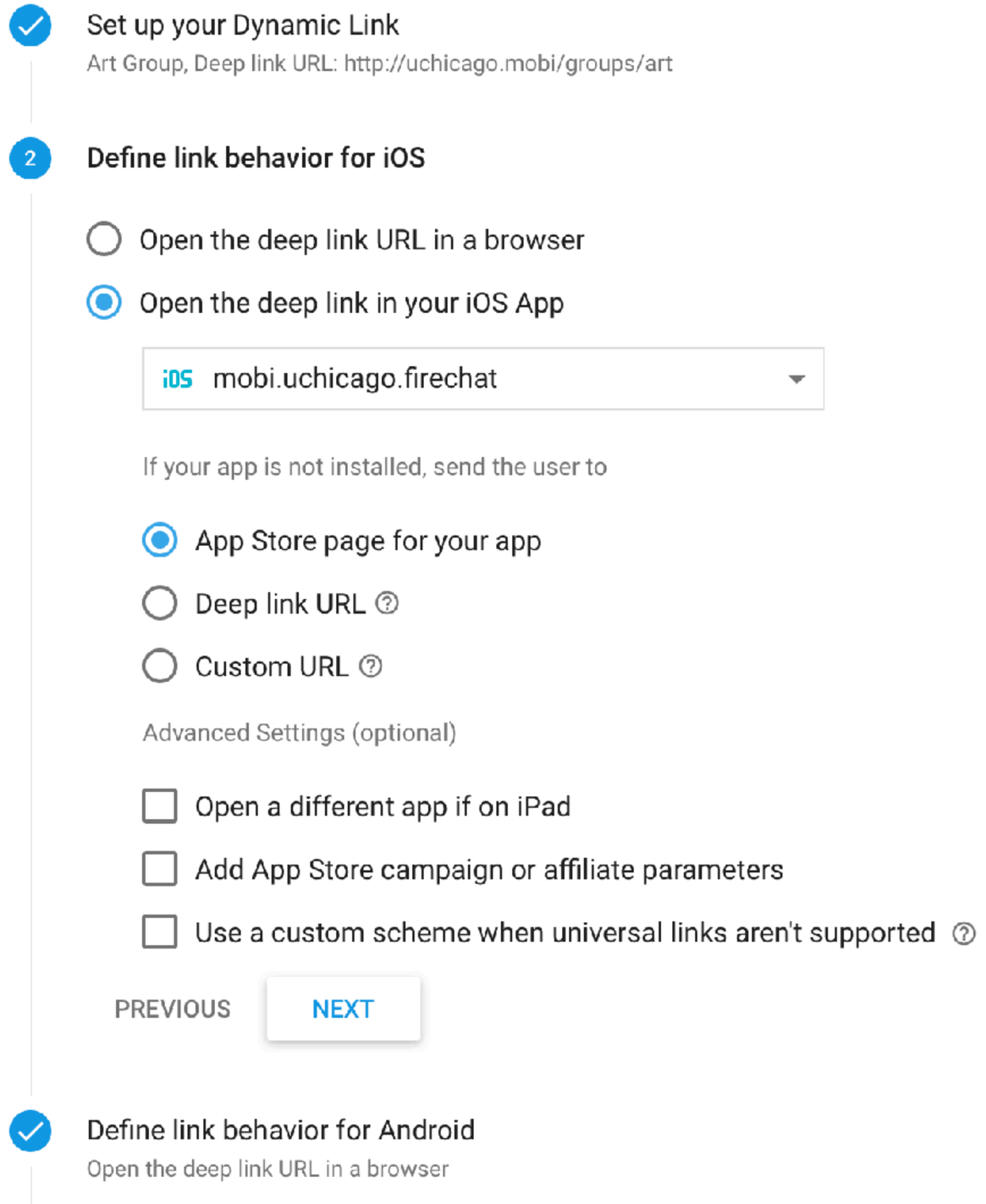
CREATING A DYNAMIC LINK

- URL contains information that will be read in the iOS app



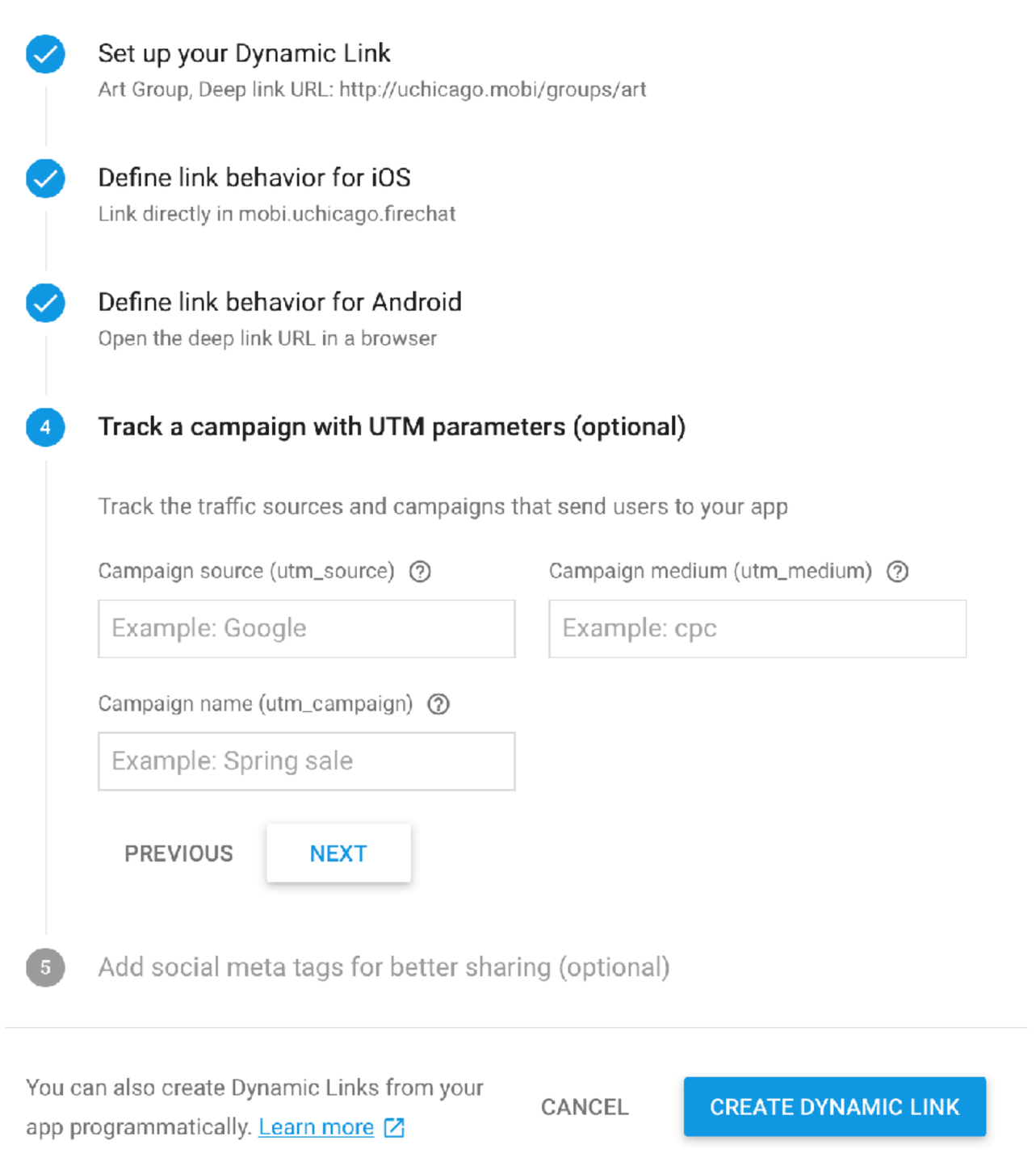
CREATING A DYNAMIC LINK

- Define the behavior of the link in the selected application



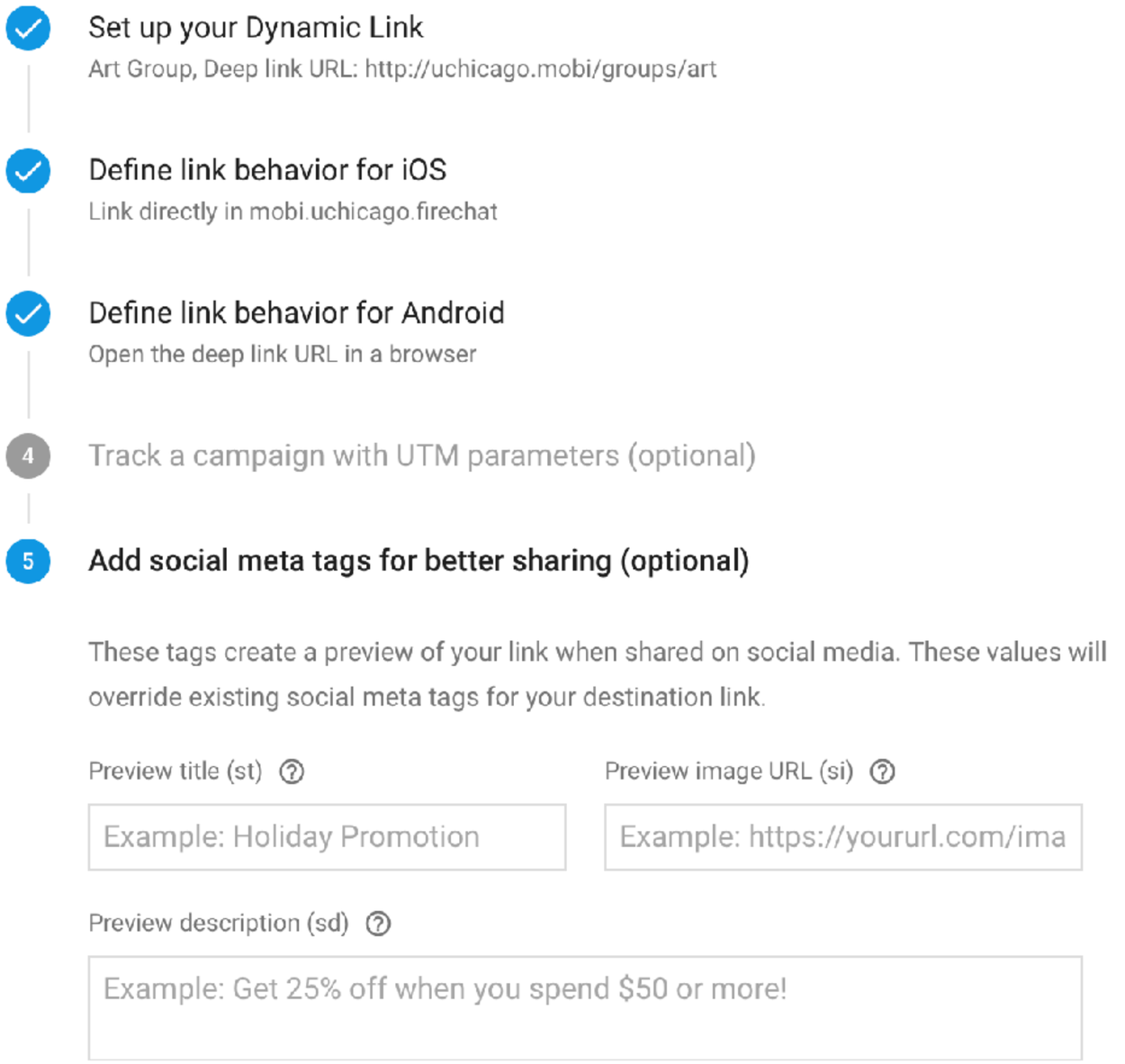
CREATING A DYNAMIC LINK

- Analytics tags

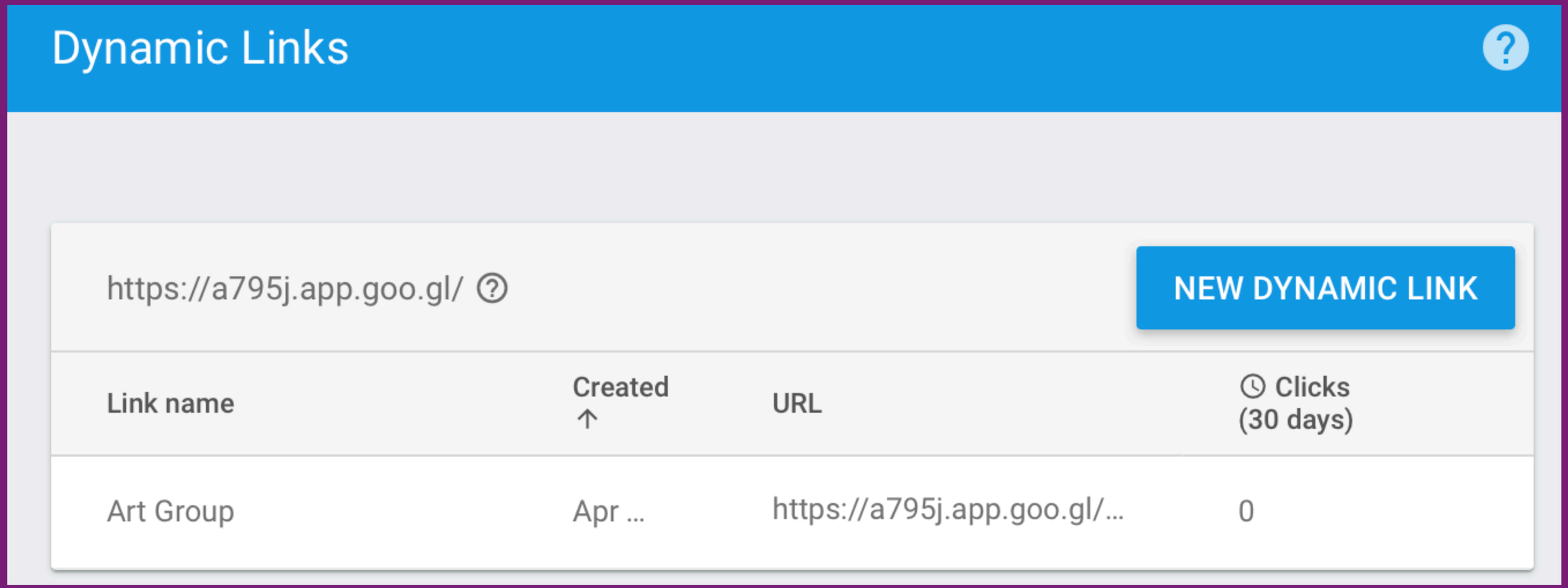


CREATING A DYNAMIC LINK

- Social media tags and data



CREATING A DYNAMIC LINK



- Your link 🎉

CREATING A DYNAMIC LINK

Link name

Art Group

Deep link

http://uchicago.mobi/groups/art

FALL BACK

iOS app

iOS mobi.uchicago.firechat

FULL LINK

Long Dynamic Link

https://a795j.app.goo.gl/?link=http://uchicago.mobi/groups/art&isi=284910350&ibi=mobi.uchicago.firechat

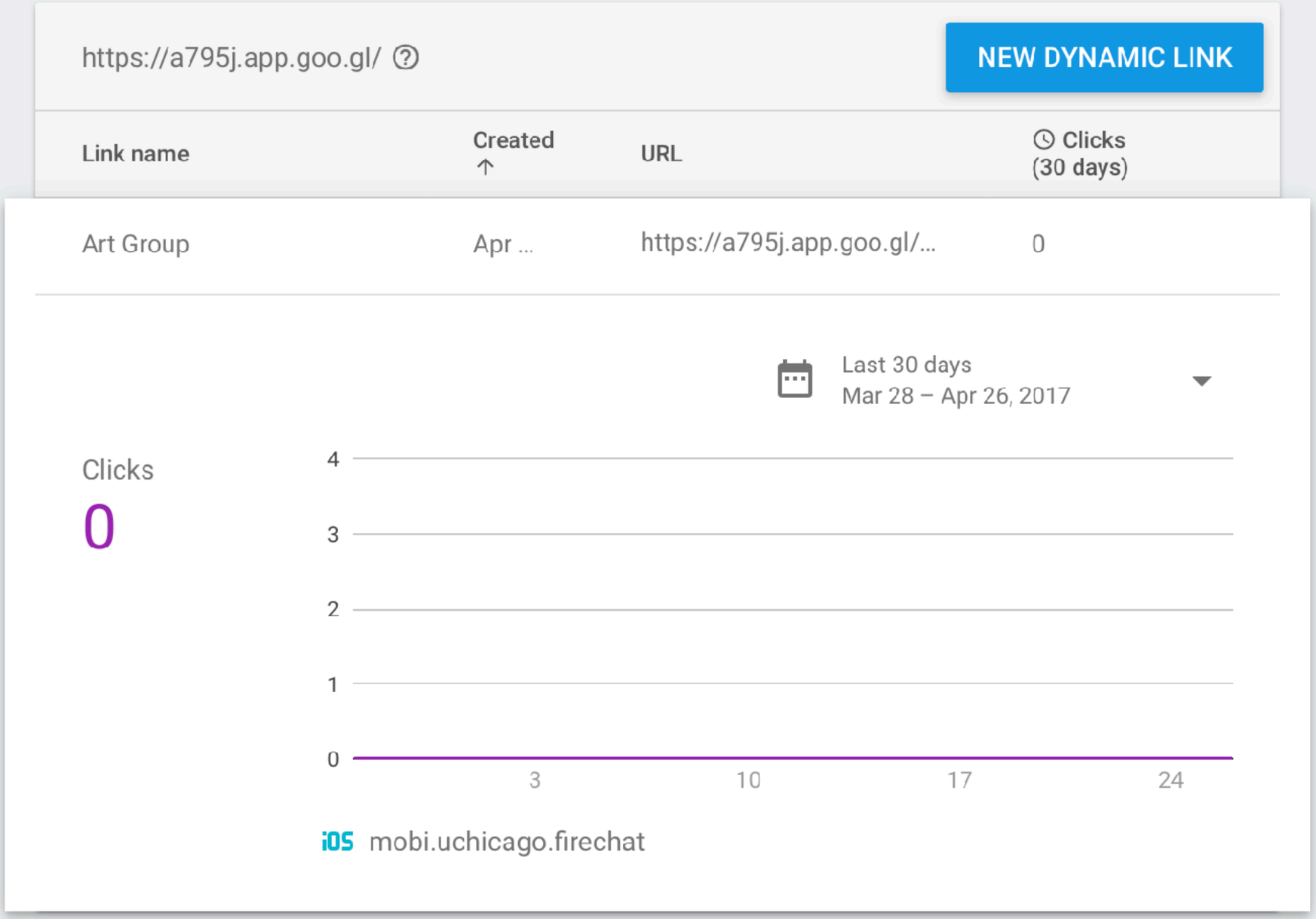
Short Dynamic Link

https://a795j.app.goo.gl/6SuK

SHORTENED LINK

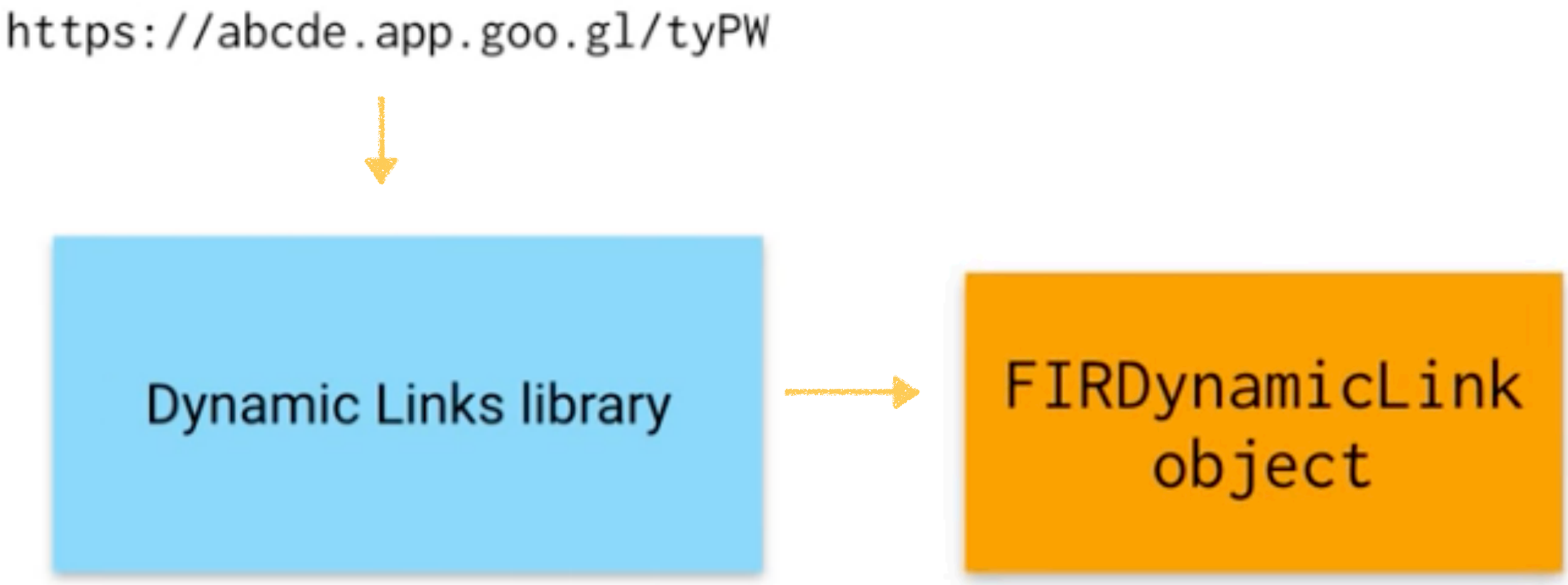
CREATING A DYNAMIC LINK

- Analytics for specific link



HANDLING DYNAMIC LINKS

HANDLING DYNAMIC LINKS



HANDLING DYNAMIC LINKS

```
# Uncomment the next line to define a global platform for your project
 platform :ios, '9.0'

target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'

  pod 'Firebase/DynamicLinks'

  pod 'SDWebImage'

end
```

HANDLING DYNAMIC LINKS

- Two different flows
  - Custom URL Scheme
    - First install
    - Will direct to the app store id that you specified
  - Deep link
    - iOS functions
    - Will launch the app and pass the link

HANDLING DYNAMIC LINKS

```
// Handle dynamic links
@available(iOS 8.0, *)
func application(_ application: UIApplication,
                 continue userActivity: NSUserActivity,
                 restorationHandler: @escaping ([Any]?) -> Void) -> Bool {
  guard let dynamicLinks = FIRDynamicLinks.dynamicLinks() else {
    return false
  }
  let handled = dynamicLinks.handleUniversalLink(userActivity.webpageURL!) { (dynamiclink, error) in
    if let dynamiclink = dynamiclink, let _ = dynamiclink.url {
      self.handleIcomingLink(dynamiclink)
    } else {
      print("Error: \(String(describing: error?.localizedDescription))")
    }
  }
  return handled
}
```

- In App Delegate detect open from dynamic links

HANDLING DYNAMIC LINKS

```
// Open the app from URL callback during Google authentication
// or from an intial install on a deep link
@available(iOS 9.0, *)
func application(_ application: UIApplication,
                 open url: URL,
                 options: [UIApplicationOpenURLOptionsKey : Any]) -> Bool {



  let dynamicLink = FIRDynamicLinks.dynamicLinks()?.dynamicLink(fromCustomSchemeURL: url)
  if dynamicLink != nil {
    // Handle the deep link. For example, show the deep-linked content or
    // apply a promotional offer to the user's account.
    // ...
    self.handleIcomingLink(dynamicLink!)
    print("Deep link of first launch or old device")
    return true
  }
  return GIDSignIn.sharedInstance().handle(url,
                                           sourceApplication:options[UIApplicationOpenURLOptionsKey.sourceApplication] as? String,
                                           annotation: [:])
}
```

- In App Delegate, handle a first launch from deep link

HANDLING DYNAMIC LINKS

```
func handleIcomingLink(_ dynamicLink: FIRDynamicLink) {
  print("DynamicLink: \(dynamicLink)")
  guard let pathComponents = dynamicLink.url?.pathComponents  else { return }
  for component in pathComponents {
    print("\t>> component: \(component)")
  }
}
```

```
DynamicLink: <FIRDynamicLink: 0x17024a4a0, url [http://uchicago.mobi/
groups/art], confidence: weak>
    >> component: /
    >> component: groups
    >> component: art
```

DO SOMETHING USEFUL HERE
ADD USER TO GROUP

HANDLING DYNAMIC LINKS

- Link confidence
- Depends on the content

`matchConfidence = weak`

"20% off coupon"

Probably okay

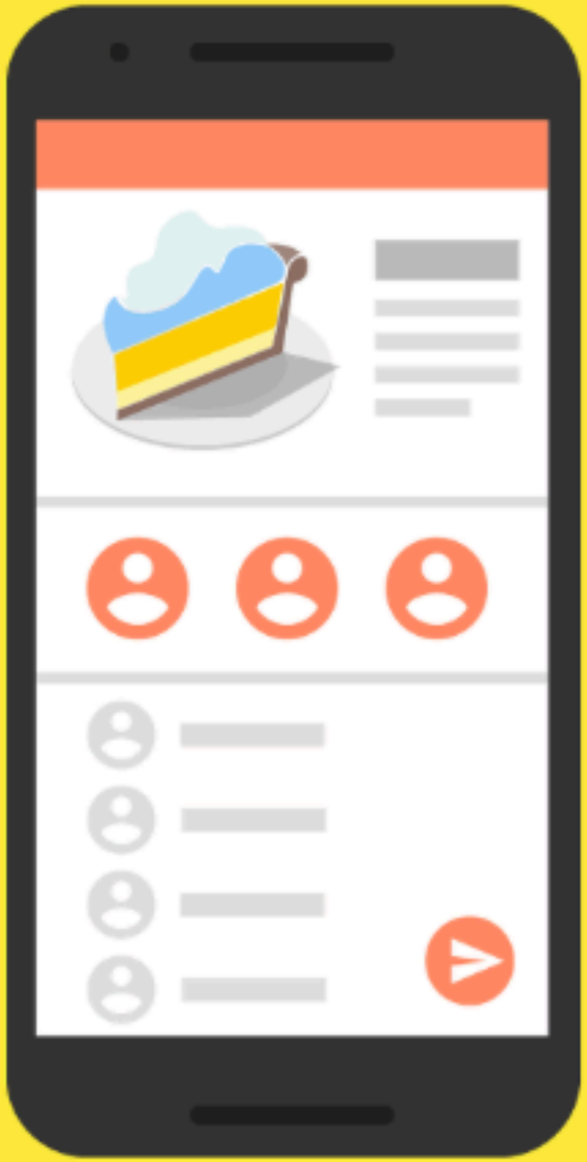
`matchConfidence = weak`

"Shared by Joan Smith!"

FIREBASE INVITES

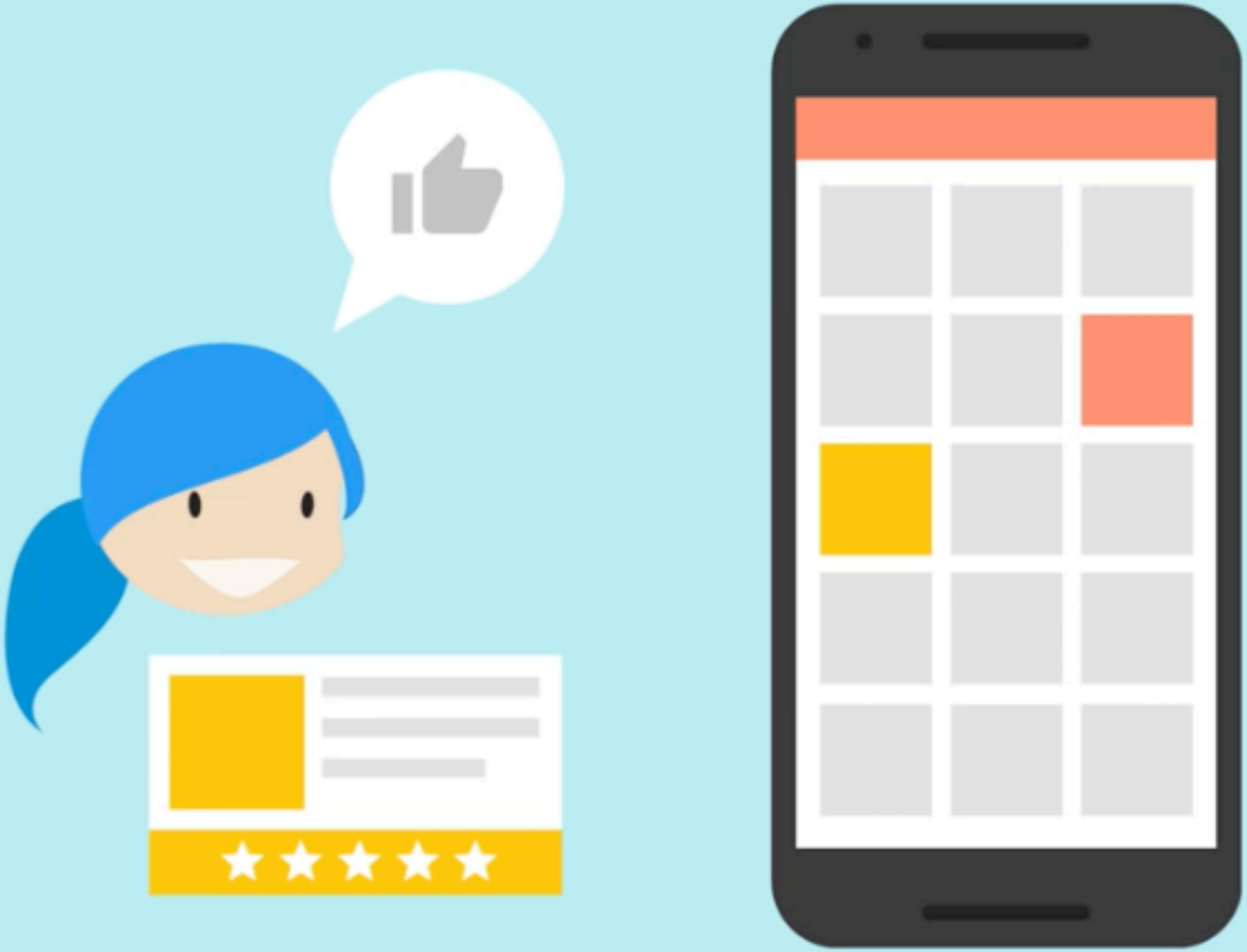
INVITES

- https://youtu.be/LkaIJCZ_HyM?list=PLl-K7zZEsYLmOF_07IayrTntevxtbUxDL



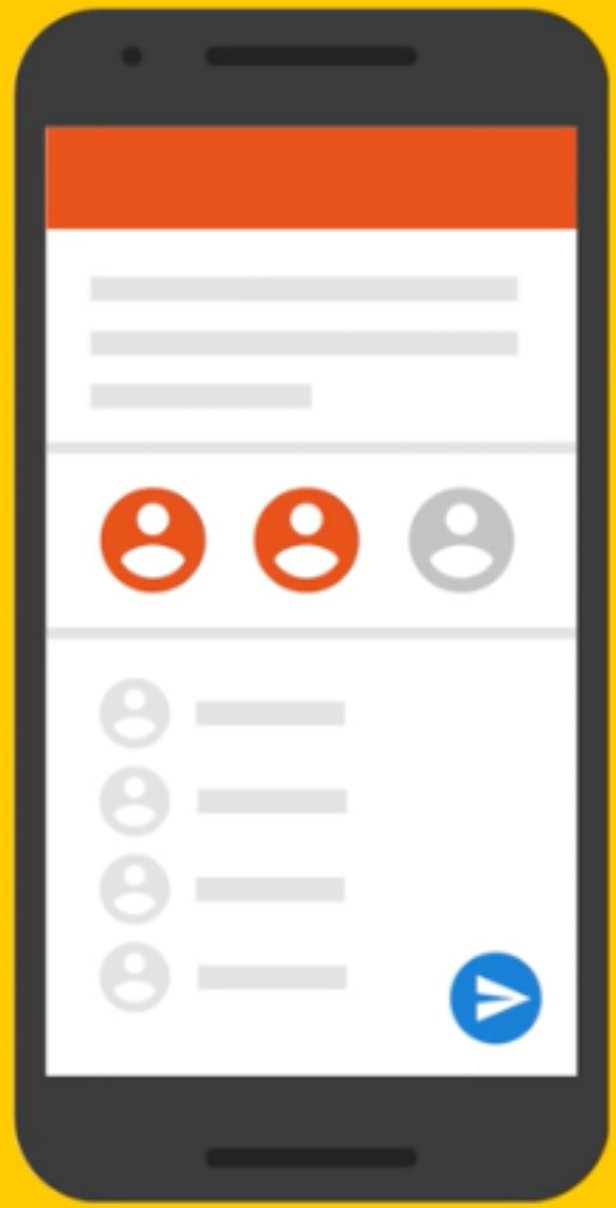
INVITES

- Firebase invites is mechanism for delivering dynamic links
- Built-in UI (material design style)
- Delivered from your authenticated account



INVITES

- Simpler than cut/pasting links in app
- No need to build additional controllers and UI to support a common task
- Tracking/analytics that are more consistent



INVITES

1	Handle links in your app	Enable linking directly to the content you want to share. You handle links on iOS by using custom URL schemes or Universal Links and on Android by using intent filters.
2	Add Share button to your app	When users click a Share button, use the Firebase Invites SDK to set up and open the sharing screen.
3	Handle Dynamic Links in your app	To enable your app to receive invitations, when your app opens, use the Dynamic Links SDK to check if a Dynamic Link was passed to it. If so, get the link from the Dynamic Link data and handle the link as necessary.

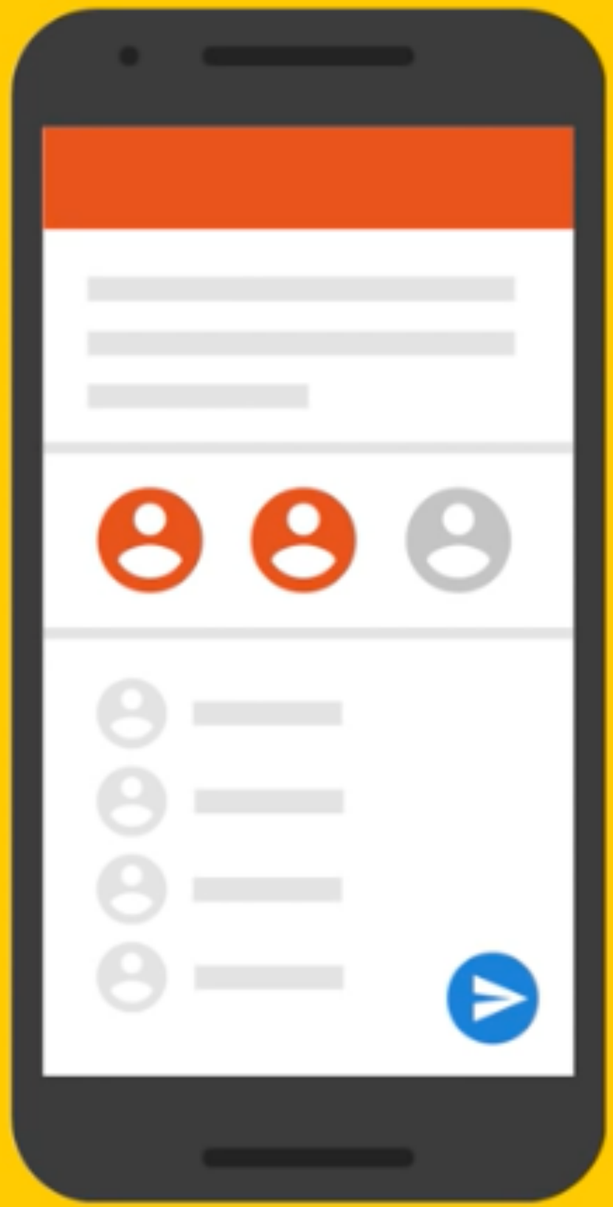
INVITES

```
# Uncomment the next line to define a global platform for your project
 platform :ios, '9.0'

target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'

  pod 'Firebase/DynamicLinks'

  pod 'SDWebImage'

end
```

NEED TO POD INSTALL AGAIN

INVITES

- Users must be authenticated using "Google Sign-in" to use Invites



INVITES

```
// For iOS8 and older
func application(_ application: UIApplication,
                 open url: URL, sourceApplication: String?, annotation: Any) -> Bool {
  if let invite = FIRInvites.handle(url, sourceApplication:sourceApplication, annotation:annotation) as? FIRReceivedInvite {
    let matchType =
      (invite.matchType == .weak) ? "Weak" : "Strong"
    print("Invite received from: \(sourceApplication ?? "") Deeplink: \(invite.deepLink)," +
      "Id: \(invite.inviteId), Type: \(matchType)")
    return true
  }

  return GIDSignIn.sharedInstance().handle(url, sourceApplication: sourceApplication, annotation: annotation)
}
```

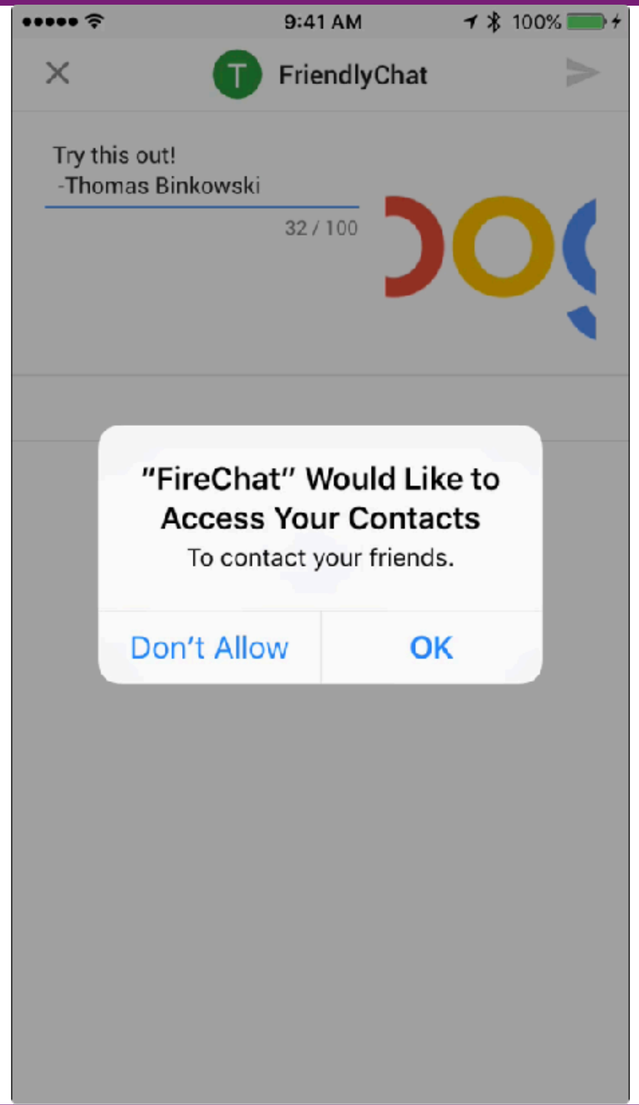
- Handle open from invite similar to dynamic links
- Parse to action

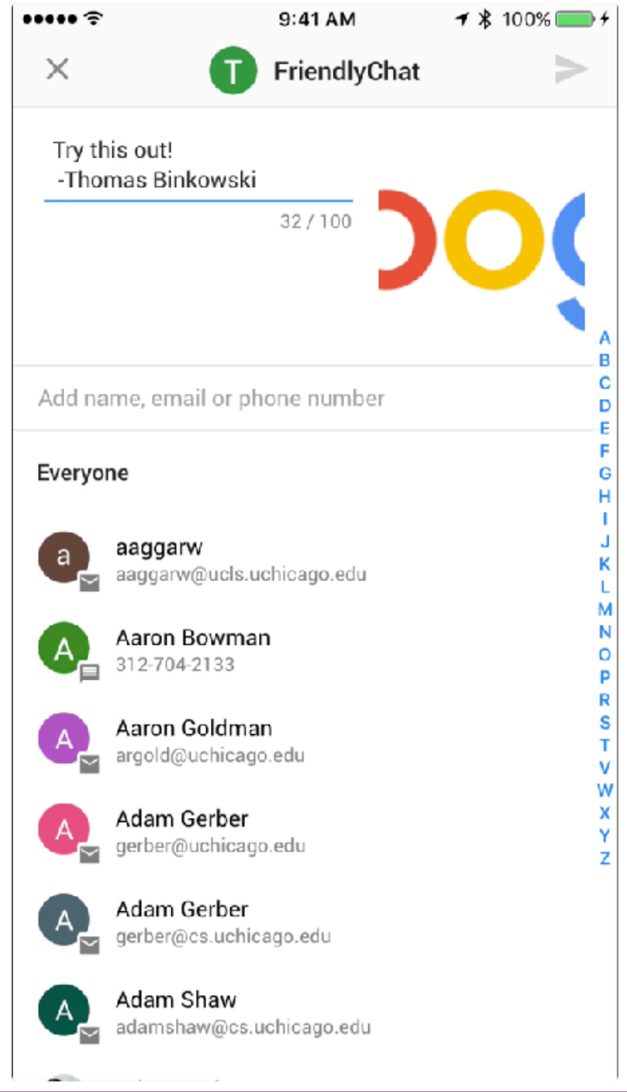
FIREBASE INVITES

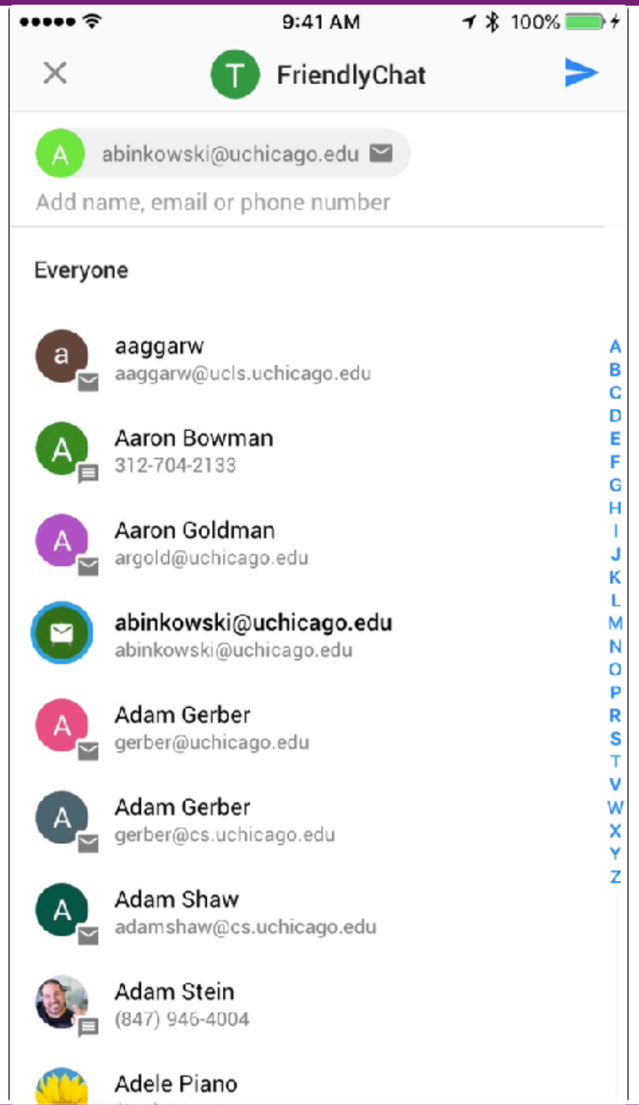
SEND AN INVITE

```
if let invite = FIRInvites.inviteDialog() {
  invite.setInviteDelegate(self as FIRInviteDelegate)

  // NOTE: You must have the App Store ID set in your developer console project
  // in order for invitations to successfully be sent.

  // A message hint for the dialog. Note this manifests differently depending on the
  // received invitation type. For example, in an email invite this appears as the subject.
  invite.setMessage("Try this out!\n -\(FIRAuth.auth()?.currentUser?.displayName ?? "")")
  // Title for the dialog, this is what the user sees before sending the invites.
  invite.setTitle("FriendlyChat")
  invite.setDeepLink("app_url")
  invite.setCallToActionText("Install!")
  invite.setCustomImage("https://www.google.com/images/branding/googlelogo/2x/googlelogo_color_272x92dp.png")
  invite.open()
}
```

FIREBASE INVITES

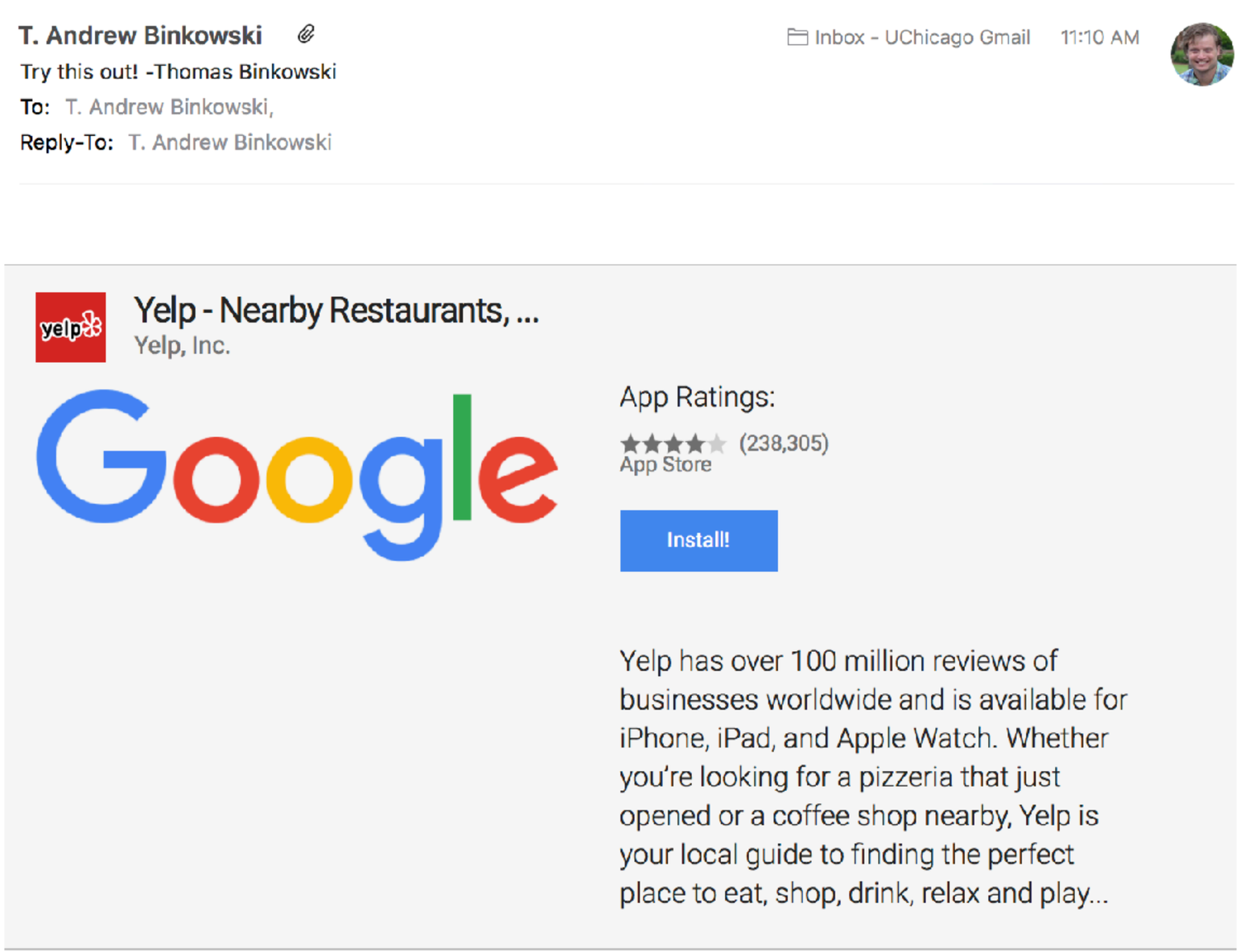






FIREBASE INVITES

- It uses your App Store id (hence yelp)
- The install link is customized by you (dynamic link) on mobile device only



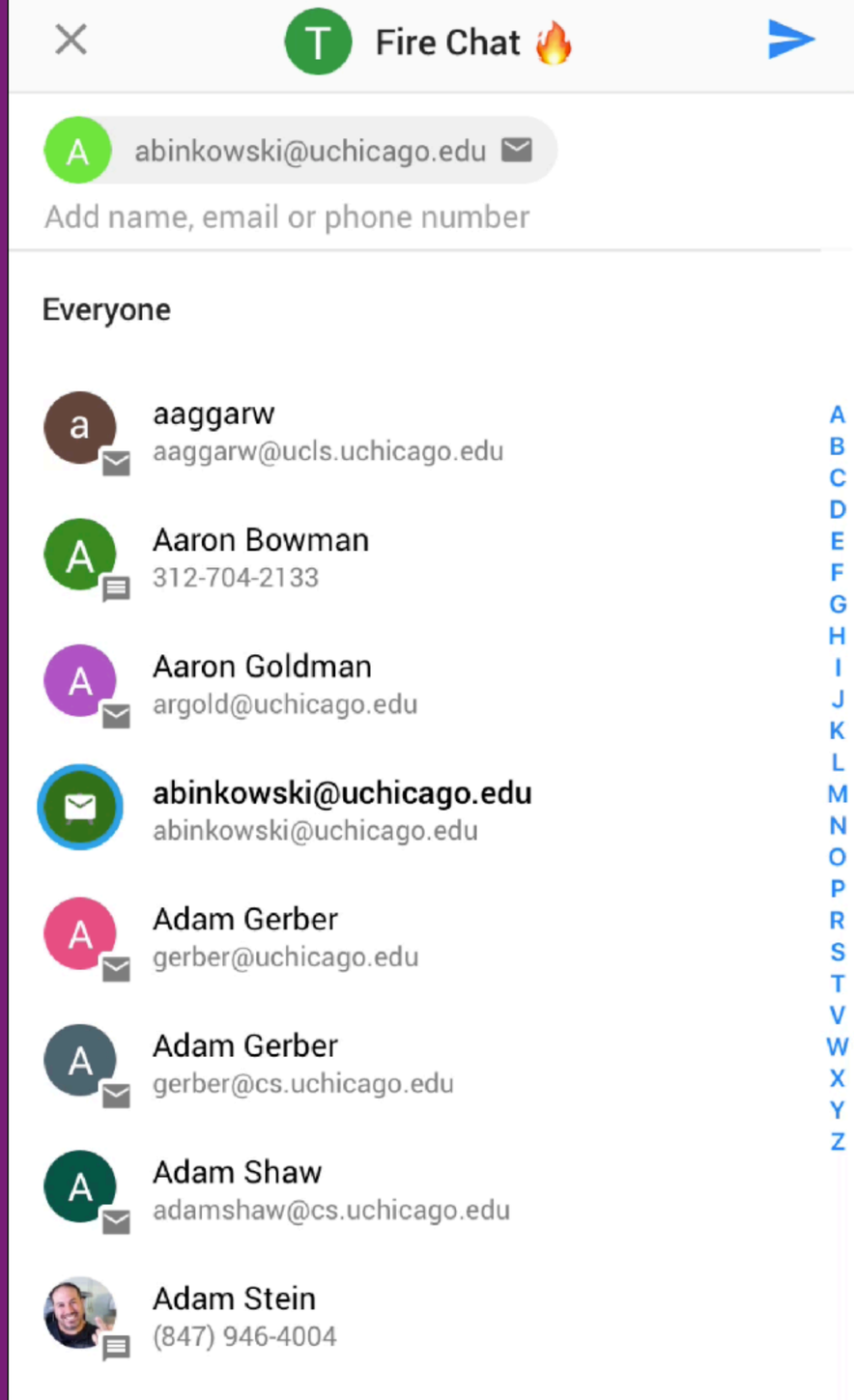
FIREBASE INVITES

```
if let invite = FIRInvites.inviteDialog() {
  invite.setInviteDelegate(self as FIRInviteDelegate)

  // NOTE: You must have the                    eloper console project
  // in order for invitation

  // A message hint for the                     ifferently depending on the
  // received invitation type. For example, in an email invite this appears as the subject.
  invite.setMessage("Try this out!\n -\(FIRAuth.auth()?.currentUser?.displayName ?? "")")
  // Title for the dialog, this is what the user sees before sending the invites.
  invite.setTitle("Fire Chat 🔥")
  invite.setDeepLink("https://a795j.app.goo.gl/6SuK")
  invite.setCallToActionText("Install!")
  invite.setCustomImage("http://i.telegraph.co.uk/multimedia/archive/02830/cat_2830677b.jpg")
  invite.open()
}
```

MY DEEP LINK

FIREBASE INVITES

- Customize the appearance, text and link



FIREBASE INVITES

```
private func inviteFinished(withInvitations invitationIds: [Any], error: Error?) {
  if let error = error {
    print("Failed: \(error.localizedDescription)")
  } else {
    print("Invitations sent")
  }
}
```

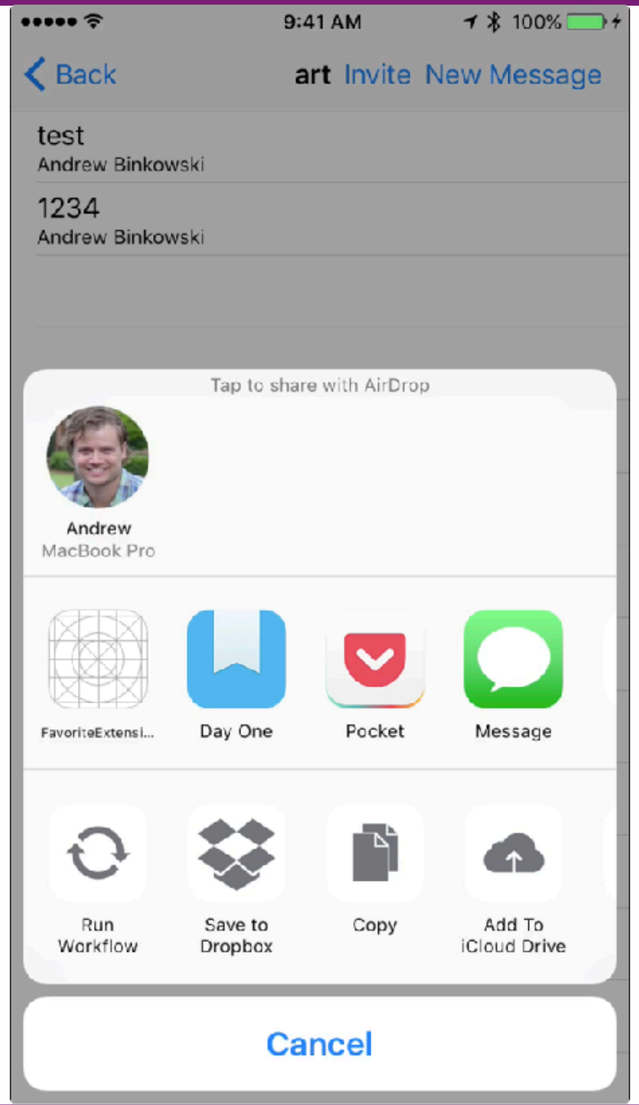
- Callback in case there was a problem

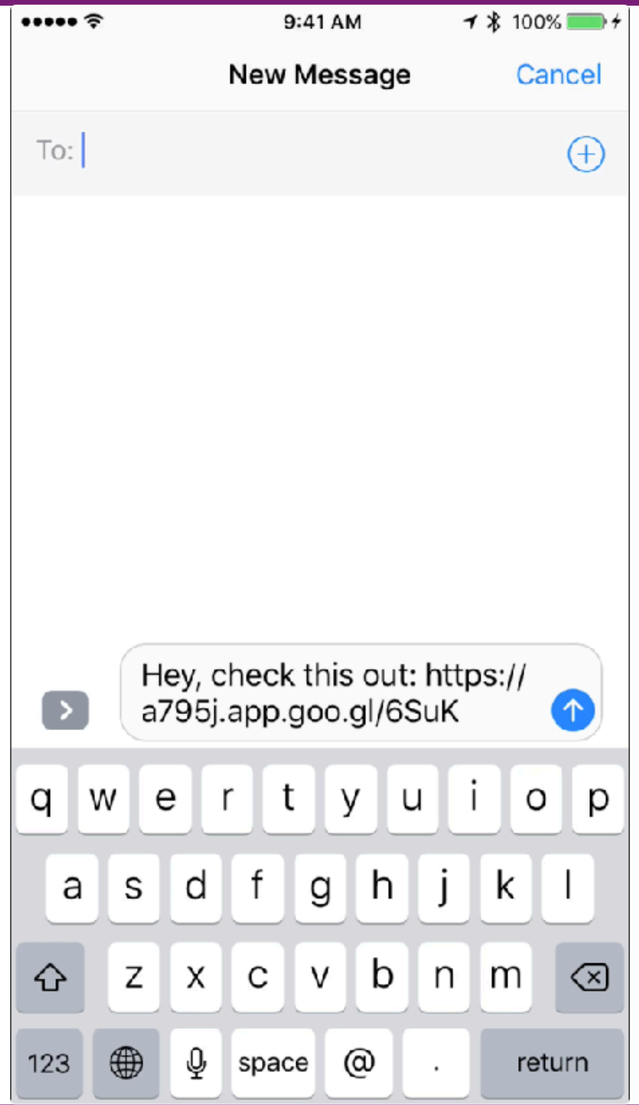
FIREBASE INVITES

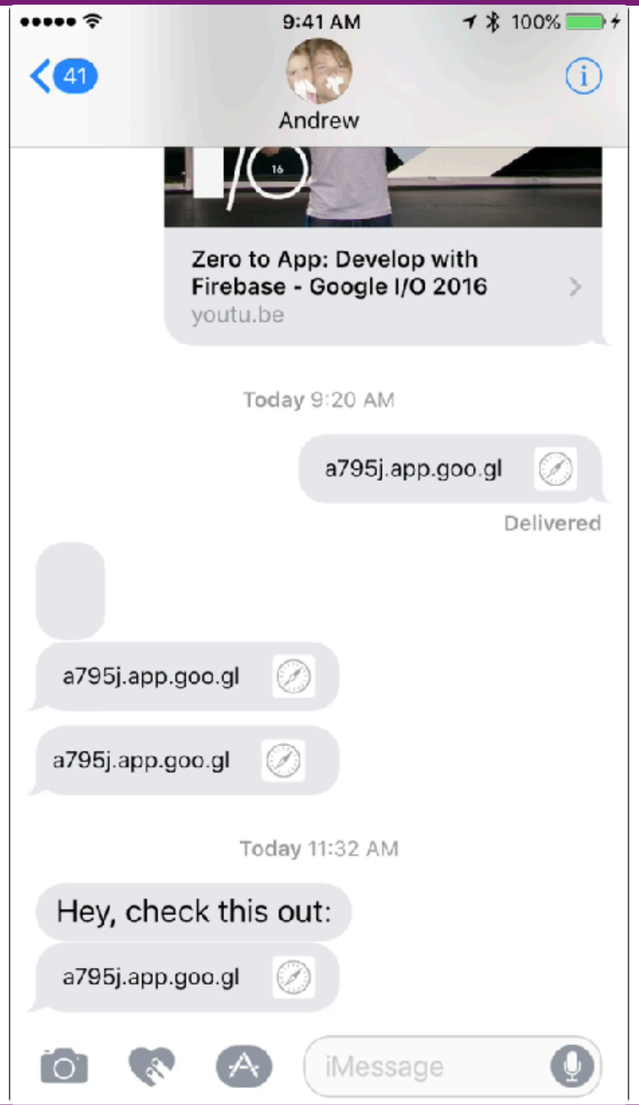
```
// Standard UI way
let myDynamicLink = "LINK_TO_SHARE"
let msg = "Hey, check this out: " + myDynamicLink
let shareSheet = UIActivityViewController(activityItems: [ msg ],
                                          applicationActivities: nil)
shareSheet.popoverPresentationController?.sourceView = self.view
self.present(shareSheet, animated: true, completion: nil)
```

- Use standard UIKit to send a message

REALTIME DATABASE



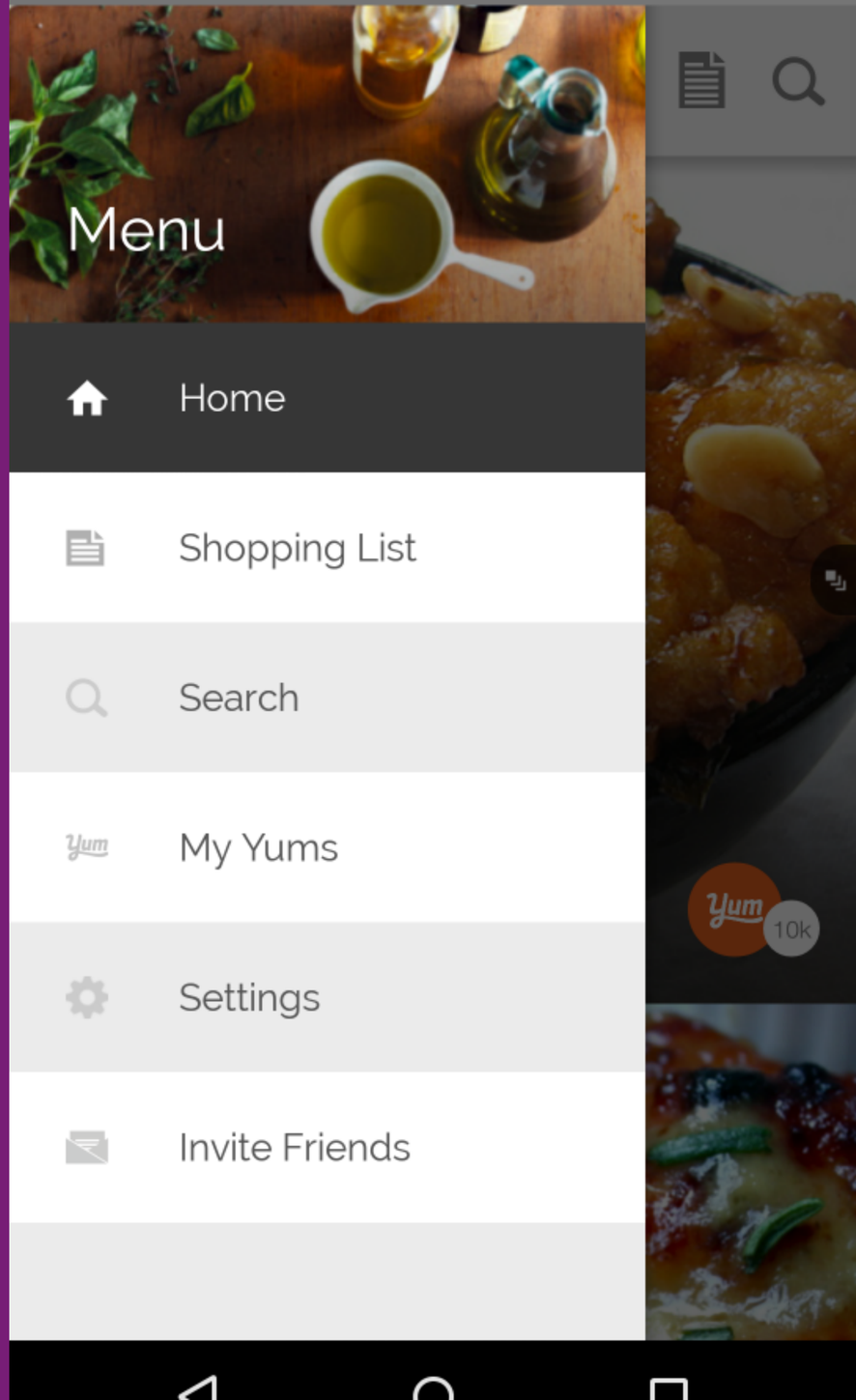




BEST PRACTICES

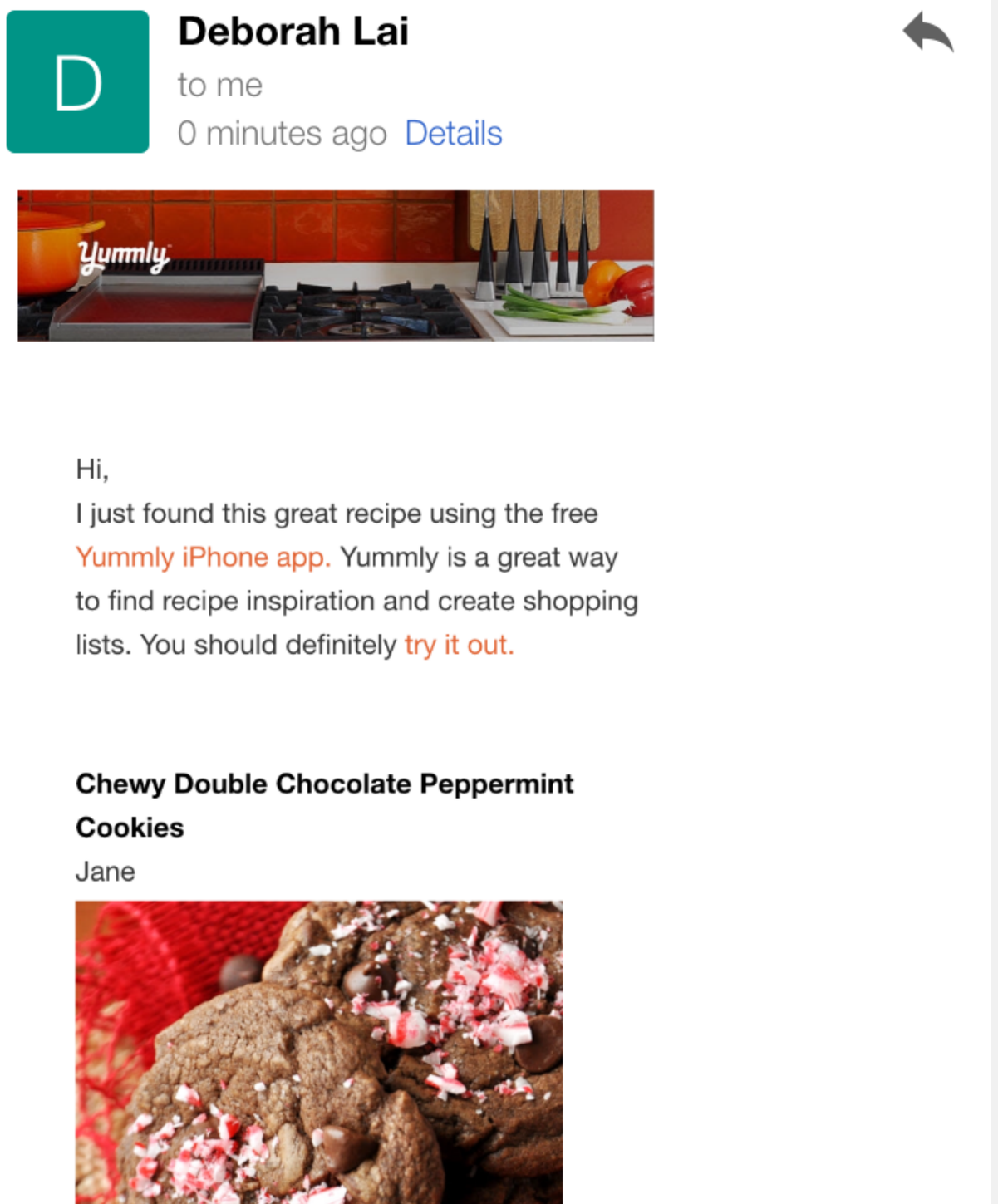
INVITES
BEST PRACTICES

- Make the invite and share options easy to find
- Build a custom share sheet
- Keep the message simple



INVITES
BEST PRACTICES

- Customize SMS invitation message from the sender
- Use HTML to customize email invites
  - Include shared content



INVITES BEST PRACTICES

- Case studies

Fabulous

Fabulous is a research-based app incubated in Duke University's Center for Advanced Hindsight. It helps users to embark on a journey to reset their habits, replacing them with healthy rituals, with the goal of improving health and well-being. Learn how Fabulous achieved growth and increased their share links with Firebase Invites.

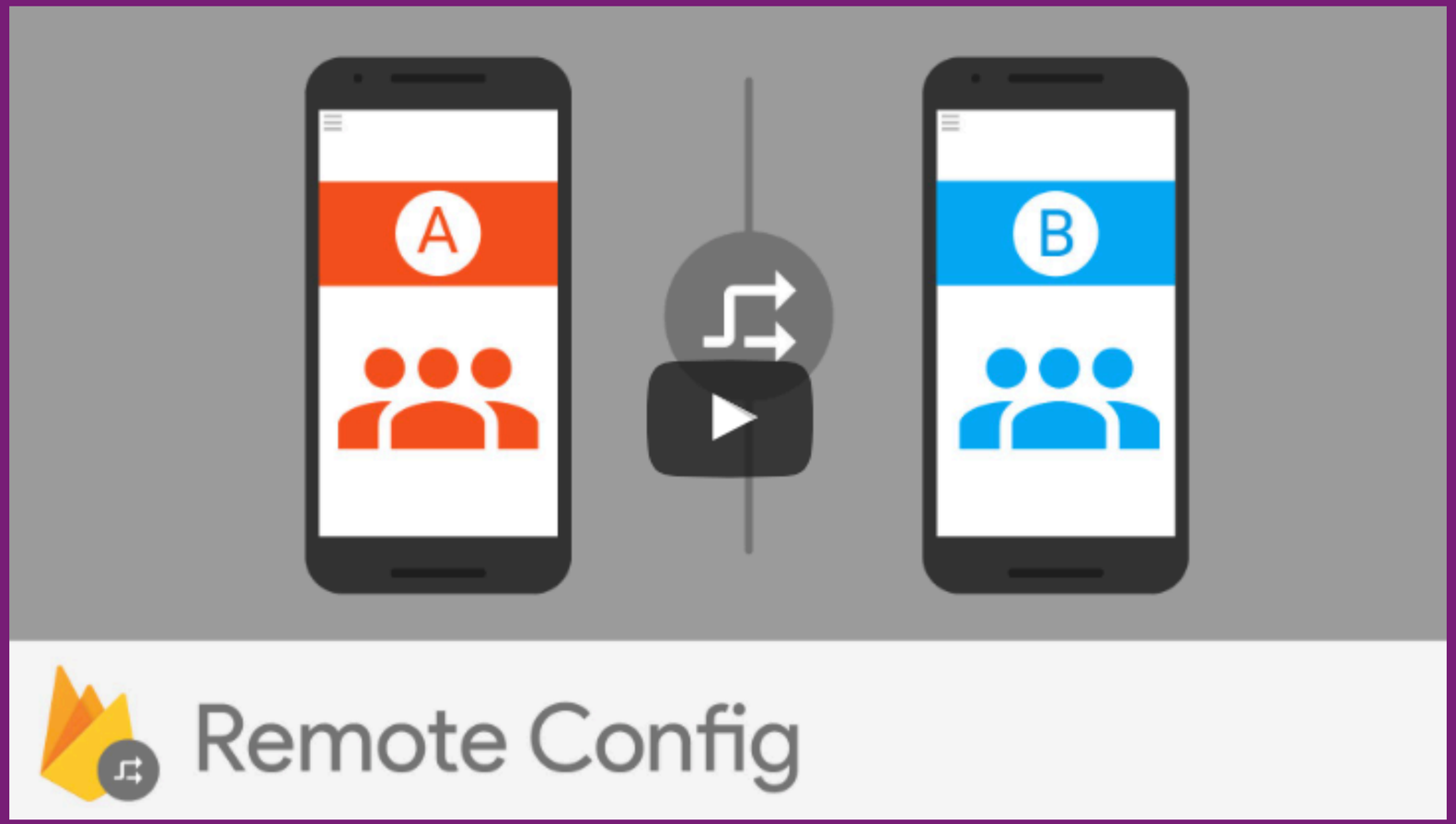
Yummly

Yummly is a food discovery platform that views cooking a meal as a personalized, shareable experience. Yummly wanted to expand its user base and generate awareness on the Android platform, so they turned to Firebase Invites, and found installation rates to be ~60% higher than with other sharing channels. Learn how Yummly achieved this here.

BREAK TIME 😴

FIREBASE REMOTE CONFIG

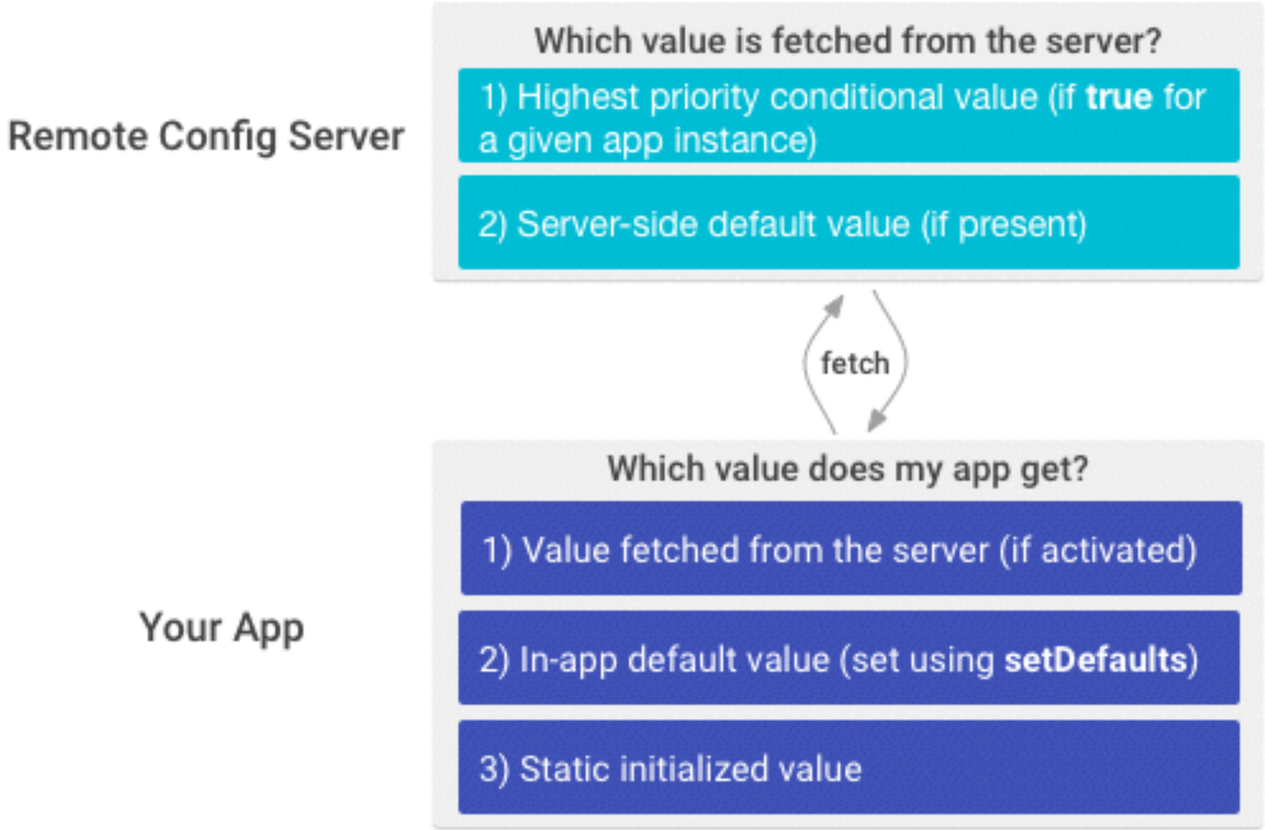
REMOTE CONFIG



- https://youtu.be/_CXXVFPO6f0?list=PLl-K7zZEsYLmOF_07layrTntevxtbUxDL

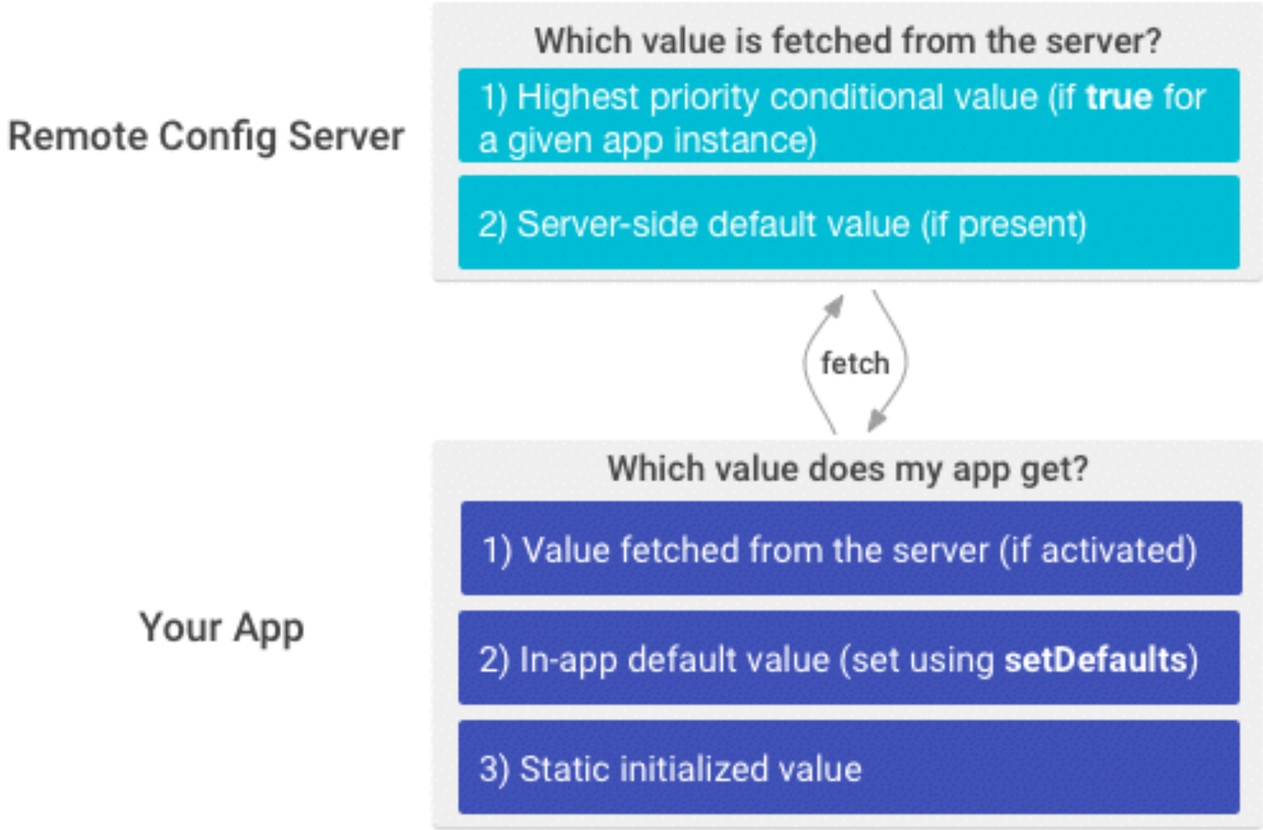
REMOTE CONFIG

- Firebase Remote Config is a cloud service that lets you change the behavior and appearance of your app without requiring users to download an app update



REMOTE CONFIG

- Create values that can be changed in the console to make them appear in your app
- There are plenty of ways to do this...this is just less work



REMOTE CONFIG

- Use cases

Quickly roll out changes to your app's userbase	You can make changes to your app's default behavior and appearance by changing service-side parameter values. For example, you could change your app's layout or color theme to support a seasonal promotion, with no need to publish an app update.
Customize your app for segments of your userbase	You can use Remote Config to provide variations on your app's user experience to different segments of your userbase by app version, by Firebase Analytics audience, by language, and more.
Run A/B tests to improve your app	You can use Remote Config random percentile targeting with Firebase Analytics to A/B test improvements to your app across different segments of your userbase so that you can validate improvements before rolling them out to your entire userbase.

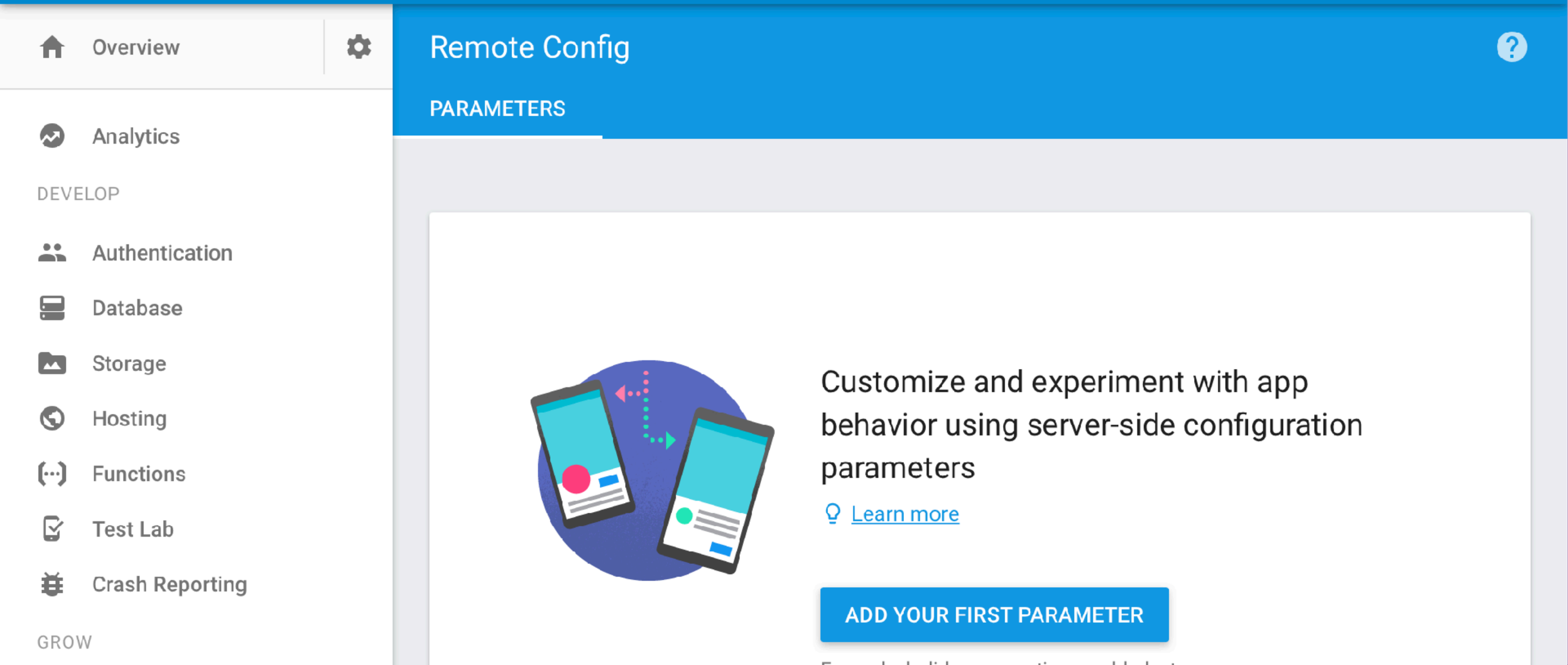
REMOTE CONFIG

- Note the following policies
  - Don't use Remote Config to make app updates that should require a user's authorization. This could cause your app to be perceived as untrustworthy.
  - Don't store confidential data in Remote Config parameter keys or parameter values. It is possible to decode any parameter keys or values stored in the Remote Config settings for your project
  - Don't attempt to circumvent the requirements of your app's target platform using Remote Config.

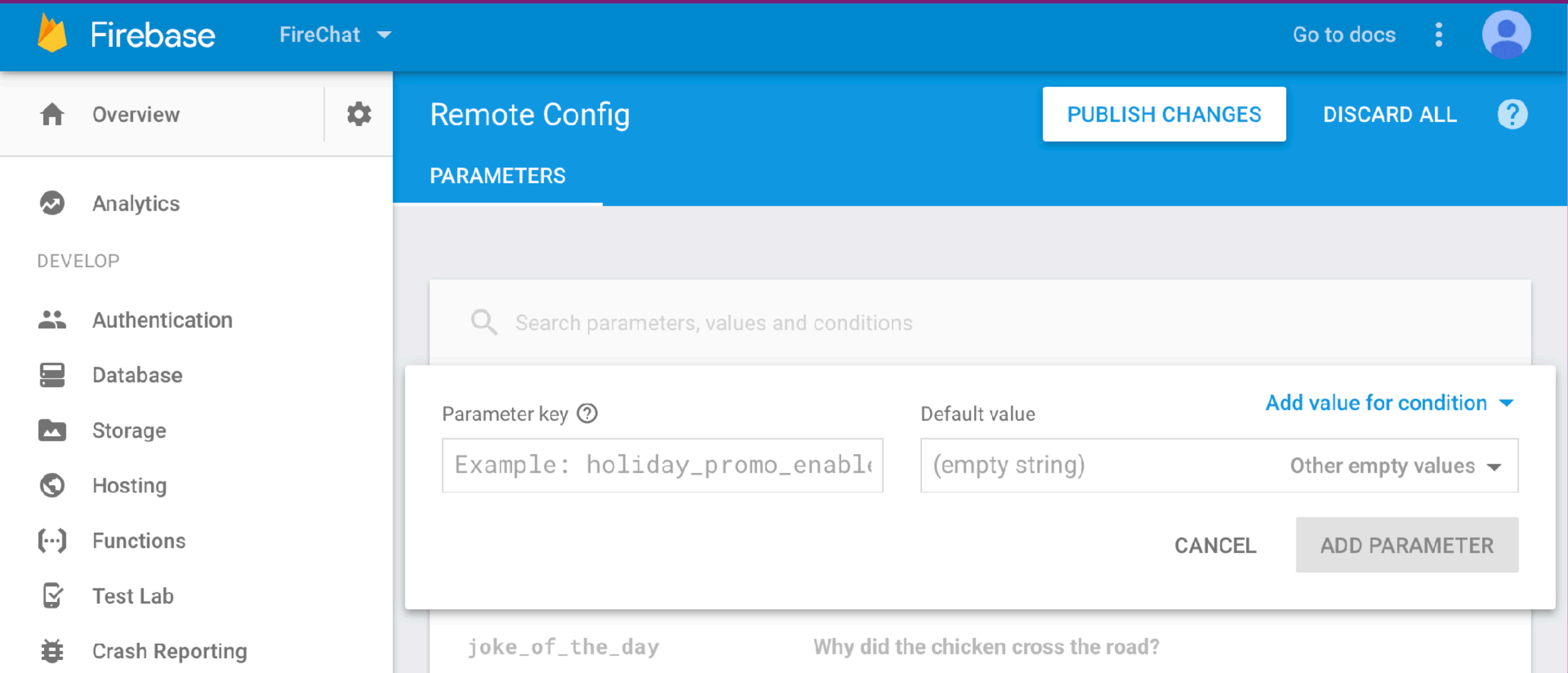
REMOTE CONFIG

```
target 'FireChat' do
  # Comment the next line if you're not using Swift and don't want to use dynamic frameworks
  use_frameworks!

  # Pods for FireChat
  pod 'Firebase/Core'
  pod 'Firebase/Database'

  pod 'Firebase/Auth'
  pod 'GoogleSignIn'

  pod 'Firebase/Crash'

  pod 'Firebase/Storage'
  pod 'Firebase/Invites'

  pod 'Firebase/DynamicLinks'

  pod 'Firebase/RemoteConfig'

  pod 'SDWebImage'

end
```

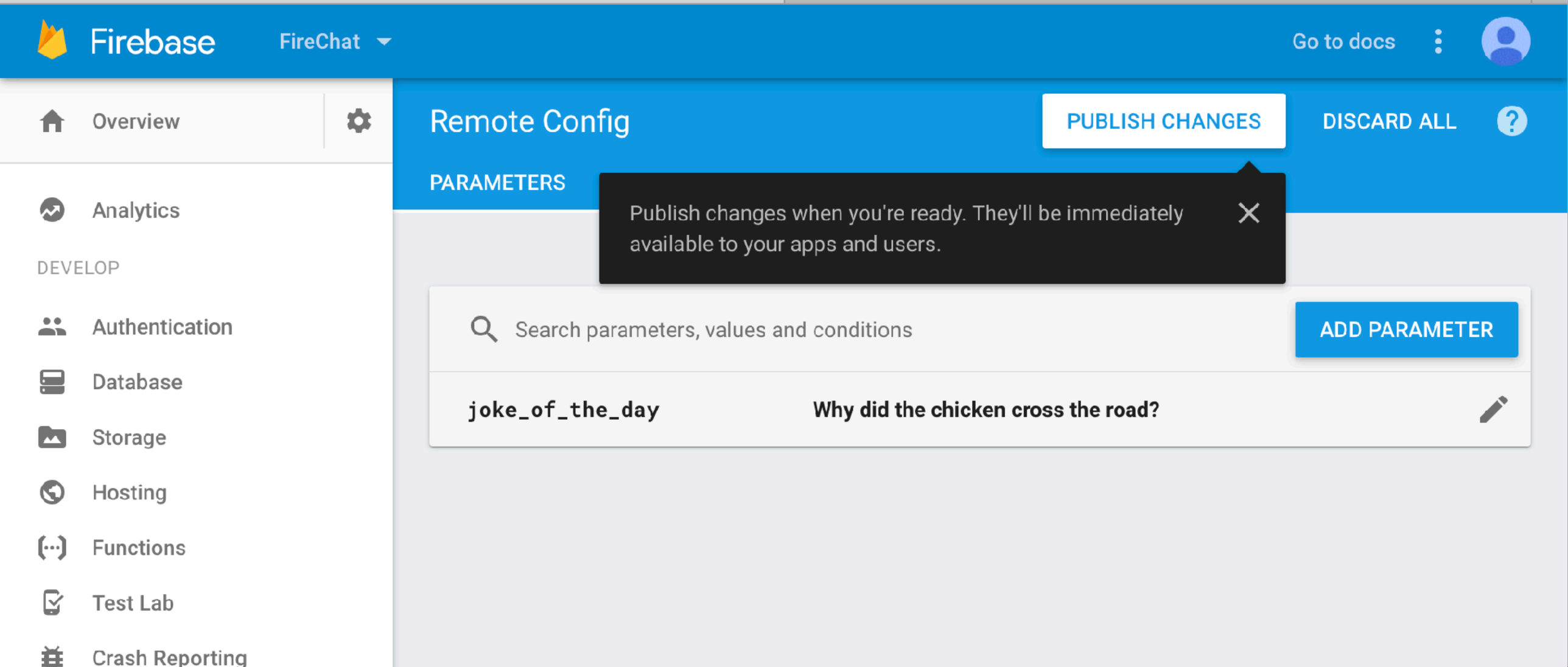
REMOTE CONFIG



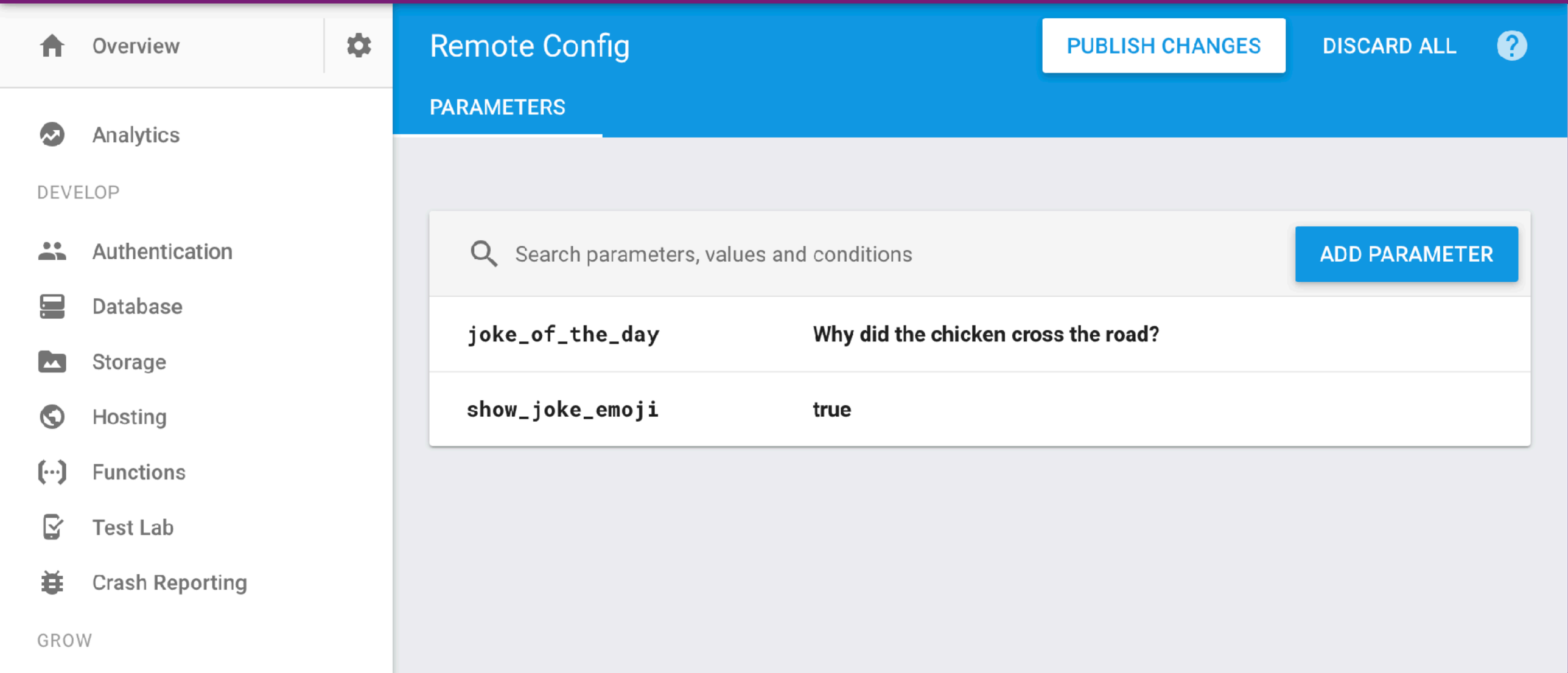
REMOTE CONFIG



REMOTE CONFIG



REMOTE CONFIG



REMOTE CONFIG

- Set default values
  - Programatically
  - Include a plist
- Fetch latest content
- Activate Fetched values

```
import Foundation
import Firebase

class RemoteConfigManager {

  static let sharedInstance = RemoteConfigManager()
  let remoteConfig = FIRRemoteConfig.remoteConfig()

  private init() {
    activateDebugMode()
    loadDefaultValues()
  }

  func loadDefaultValues() {
    let appDefaults: [String: NSObject] = ["joke_of_the_day": "Default Joke" as NSObject]
    remoteConfig.setDefaults(appDefaults)
  }

  ///
  func fetchCloudValues() {


    let fetchDuration: TimeInterval = 0
    activateDebugMode()
    remoteConfig.fetch(withExpirationDuration: TimeInterval(fetchDuration)) { (status, error) -> Void in


      if status == .success {
        print("Config fetched!")
        self.remoteConfig.activateFetched()



      } else {
        print("Config not fetched")
        print("Error \(error!.localizedDescription)")
      }
      print("Joke of the day: \(self.remoteConfig["joke_of_the_day"].stringValue!)")
    }
  }


  func activateDebugMode() {
    let debugSettings = FIRRemoteConfigSettings(developerModeEnabled: true)
    remoteConfig.configSettings = debugSettings!
  }
```

REMOTE CONFIG

- Careful of throttling

```
import Foundation
import Firebase

class RemoteConfigManager {

  static let sharedInstance = RemoteConfigManager()
  let remoteConfig = FIRRemoteConfig.remoteConfig()

  private init() {
    activateDebugMode()
    loadDefaultValues()
  }

  func loadDefaultValues() {
    let appDefaults: [String: NSObject] = ["joke_of_the_day": "Default Joke" as NSObject]
    remoteConfig.setDefaults(appDefaults)
  }

  ///
  func fetchCloudValues() {


    let fetchDuration: TimeInterval = 0
    activateDebugMode()
    remoteConfig.fetch(withExpirationDuration: TimeInterval(fetchDuration)) { (status, error) -> Void in


      if status == .success {
        print("Config fetched!")
        self.remoteConfig.activateFetched()



      } else {
        print("Config not fetched")
        print("Error \(error!.localizedDescription)")
      }
      print("Joke of the day: \(self.remoteConfig["joke_of_the_day"].stringValue!)")
    }
  }


  func activateDebugMode() {
    let debugSettings = FIRRemoteConfigSettings(developerModeEnabled: true)
    remoteConfig.configSettings = debugSettings!
  }
```

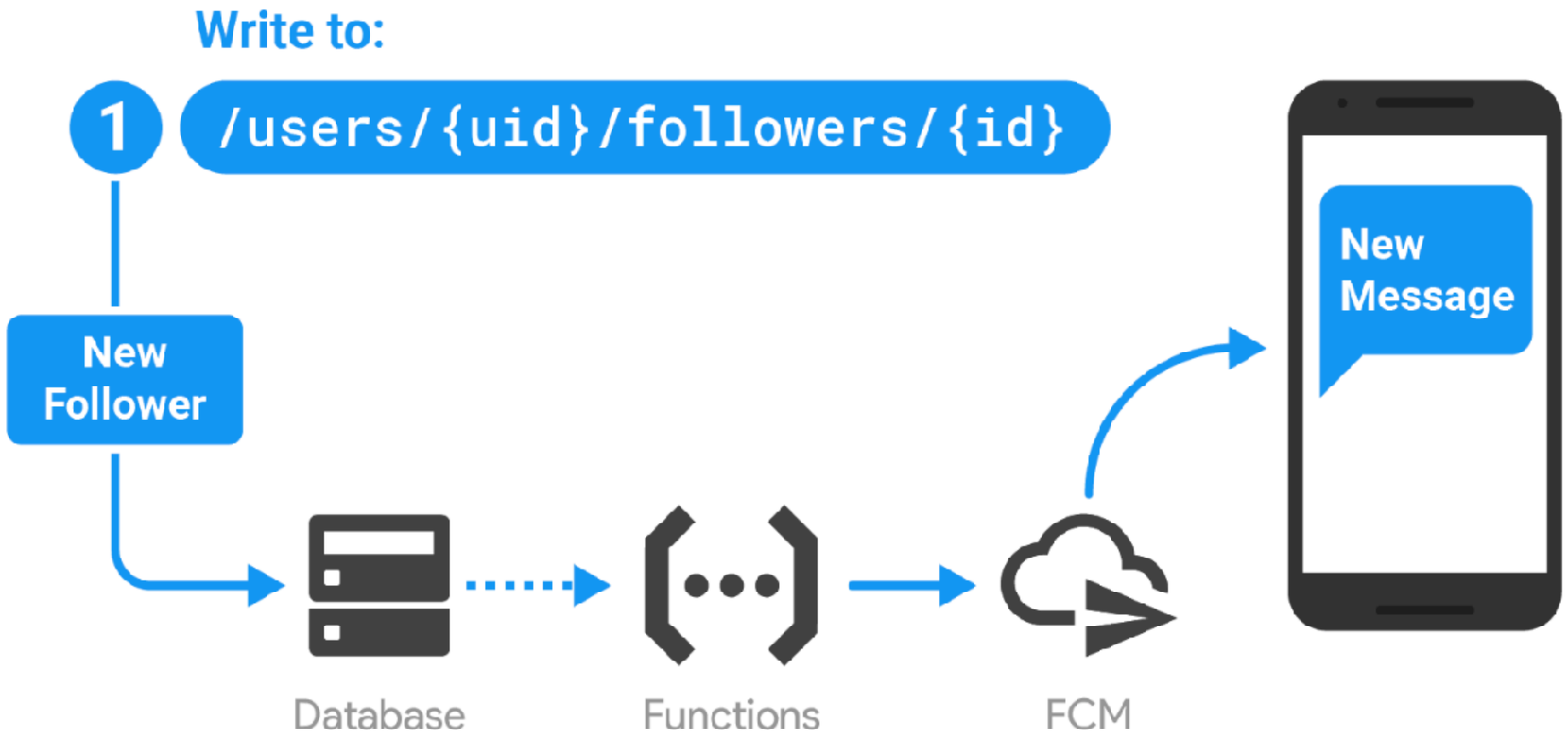
FIREBASE CLOUD FUNCTIONS

CLOUD FUNCTIONS

- https://youtu.be/vr0Gfvp5v1A

CLOUD FUNCTIONS

- One requirement for the assignment, but we will review it next week



ASSIGNMENT 3

ASSIGNMENT 3

- Part 1
  - Firebase tutorials

- Part 2
  - Develop a group messaging application using Firebase

PART 1

ASSIGNMENT 3 - PART 1

- Ray Weinerlich Tutorial to build a grocery app
- https://www.raywenderlich.com/139322/firebase-tutorial-getting-started-2

Firebase Tutorial: Getting Started

Attila Hegedüs on September 19, 2016

Tweet Like

Update note: This tutorial has been updated for iOS 10 and Swift 3 by Attila Hegedüs. The original tutorial was written by David East.

Firebase is a mobile-backend-as-a-service that provides several features for building powerful mobile apps. Firebase has three core services: a realtime database, user authentication and hosting. With the Firebase iOS SDK, you can use these services to build powerful apps without writing a single line of server code.

The realtime database is one of the most unique features of Firebase.

Learn the fundamentals in the Firebase tutorial.

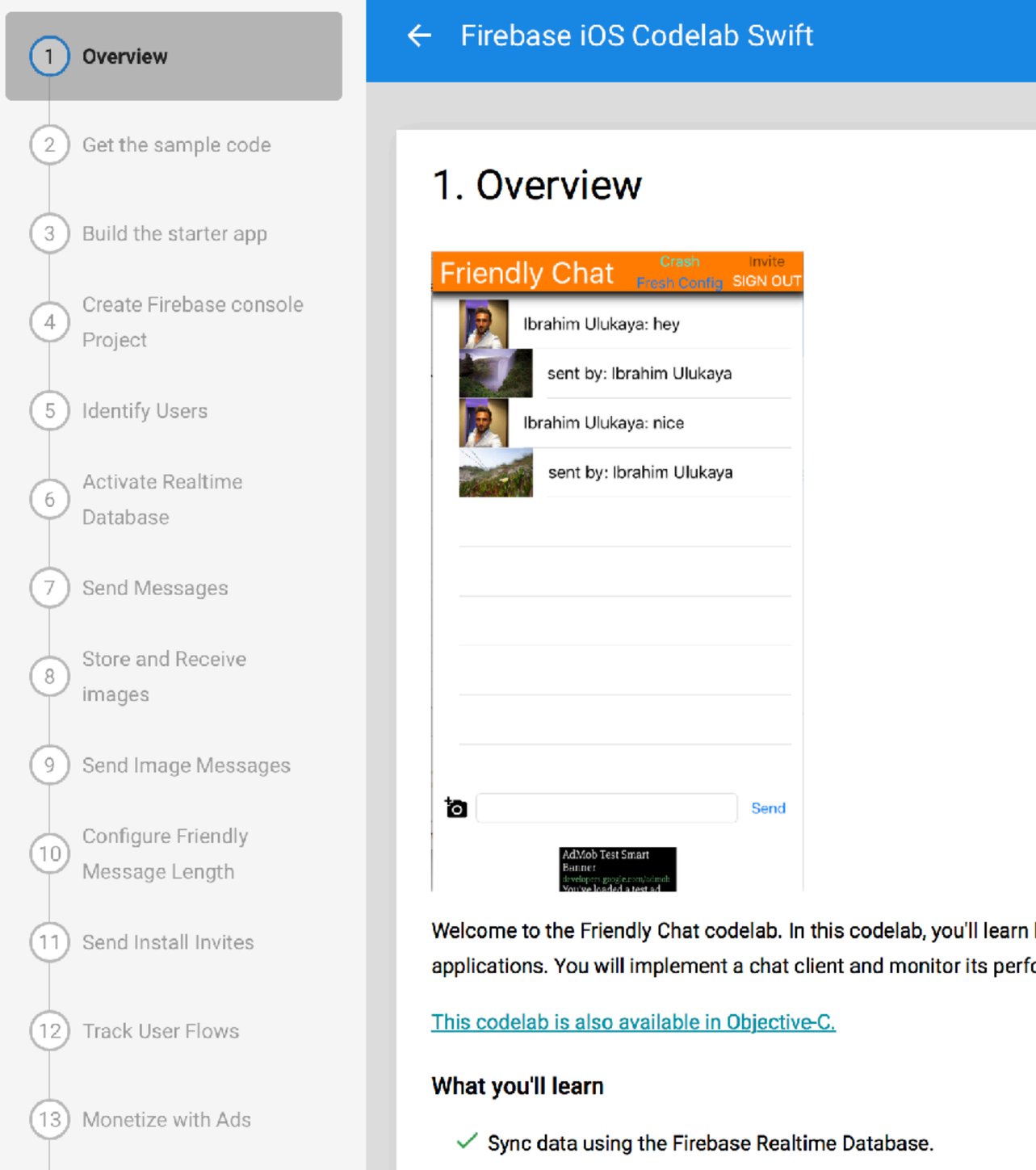
Ever used pull-to-refresh to fetch new data? You can forget about it with Firebase.

When a Firebase database updates, all connected users receive updates in realtime. This means your app can stay up to date without user interaction.

In this Firebase tutorial, you'll learn all the fundamentals of Firebase by making a collaborative groce list app called **Grocr**. When items get added to the list they'll appear instantly on any user's devices, you're not going to stop there. You'll tweak **Grocr** to work offline, so the list stays in sync even with a spotty grocery store data connection.

ASSIGNMENT 3 - PART 1

- Firebase codelab tutorial to build a (very sparse) chat application (that doesn't quite work as advertised)
- https://codelabs.developers.google.com/codelabs/firebase-ios-swift/#0



PART 2

Assignment 3 Part 2
The World Needs Another Chat App

In this part of the assignment, you will create a group chat application that takes advantage of many features available through the Firebase platform. The application will incorporate common functionality of group chat applications to allow users to send messages to each other.

Overall Application Behavior

All users will have to be authenticated with Google authentication for the application. After signing in, users should be presented with a list of groups that they belong to.

ARCHITECTING ASSIGNMENT 3

THE UNIVERSITY OF CHICAGO

MPCS 51033 • SPRING 2017 • SESSION 5

BACKENDS FOR MOBILE APPLICATIONS