



# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 3

# CLOUDKIT



# CLOUDKIT

- CloudKit  
QuickStart

Table of Contents

Introduction

Enabling CloudKit in Your App

About Containers and Databases

Setup

Enable iCloud and Select CloudKit

Access CloudKit Dashboard

Share Containers Between Apps

Add Containers to an App

Create Custom Containers

Verify Your Steps

Create an iCloud Account for Development

Recap

Creating a Database Schema by Saving Records

About Designing Your Schema

Separate Data into Record Types

Decide on Names for Your Records

Create Records Programmatically

Save Records

Enter iCloud Credentials Before Running Your App

Alert the User to Enter iCloud Credentials

Run Your App

Verify Your Steps

Recap

Using CloudKit Dashboard to Manage Databases

About the Development and Production Environments

Select Your Container

Reset the Development Environment

Create and Delete Record Types

Add, Modify, and Delete Records

Search Records

Sort Records

Recap

Fetching Records

Fetch Records by Identifier

Fetch and Modify Records

Query for Records Using Predicates

Recap

Using Asset and Location Fields

Adding Reference Fields

CloudKit Quick Start

Next

## About This Document

This document gets you started creating a CloudKit app that stores structured app and user data in iCloud. Using CloudKit, instances of your app—launched by different users on different devices—have access to the records stored in the app's database. Use CloudKit if you have model objects that you want to persist and share between multiple apps running on multiple devices. These model objects are stored as records in the database and can be provided by you or authored by the user.



You'll learn how to:

- Enable CloudKit in your Xcode project and create a schema programmatically or with CloudKit Dashboard
- Fetch records and subscribe to changes in your code
- Use field types that are optimized for large data files and location data
- Subscribe to record changes to improve performance
- Test your CloudKit app on multiple devices before uploading it to the App Store, Mac App Store, or Apple TV App Store.
- Deploy the schema to production and keep it current with each release of your app

See [Glossary](#) for the definition of database terms used in this book.

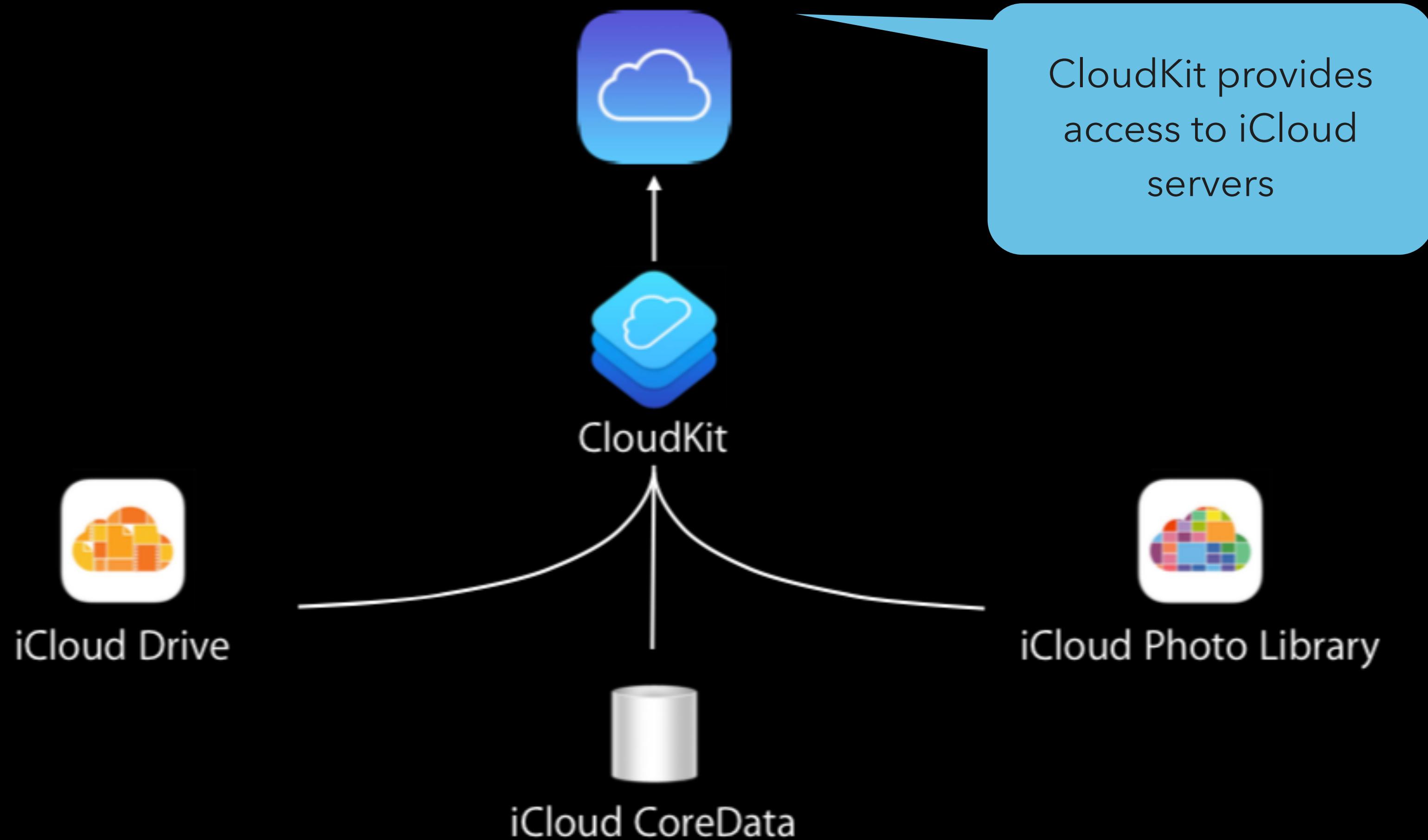
## See Also

The following WWDC sessions provide more CloudKit architecture and API details:

- [WWDC 2014: Introducing CloudKit](#) introduces the basic architecture and APIs used to save and fetch records.
- [WWDC 2014: Advanced CloudKit](#) covers topics such as private data, custom record zones, ensuring data integrity, and effectively modeling your data.
- [WWDC 2015: CloudKit Tips and Tricks](#) explore some of its lesser-known features and best practices for subscriptions and queries.
- [WWDC 2016: What's New with CloudKit](#) covers the new sharing APIs that lets you share private data between iCloud users.
- [WWDC 2016: CloudKit Best Practices](#) best practices from the CloudKit engineering team about how to take advantage of the APIs and push notifications in order to provide your users with the best experience.

The following documents describe how an API you can use to access the same data across multiple devices.

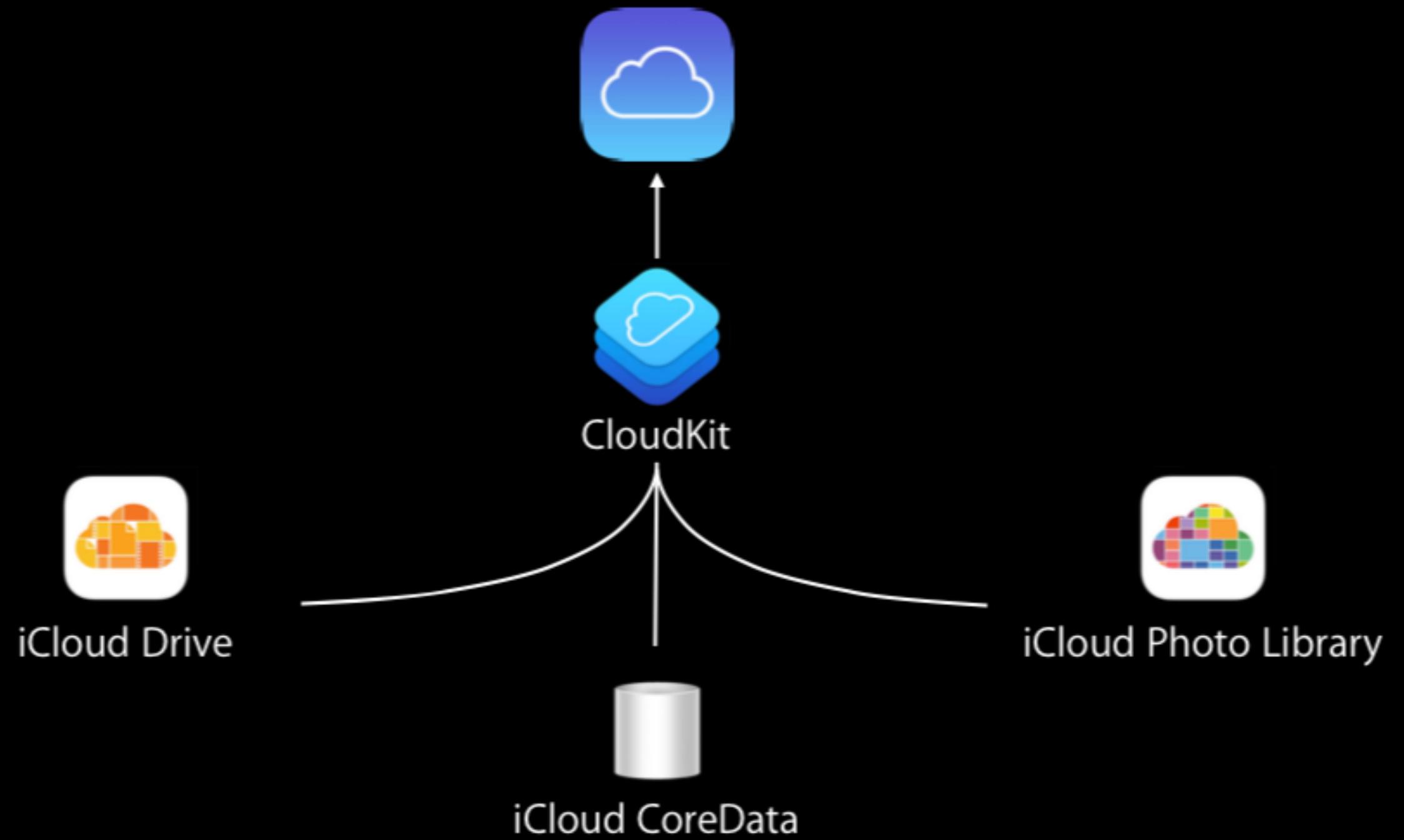
# CLOUDKIT



# CLOUDKIT

- Access to iCloud servers
- Supported on macOS, tvOS, iOS, watchOS and web (JS API)

#1 CONSIDERATION ON CHOOSING  
OVER OTHER SERVICES



# CLOUDKIT

- Free...with limits



# CLOUDKIT

- Public scales with users to PB
- Private db is charged against users quota
  - Permission can make anything private

## Getting Started with CloudKit for free.

CloudKit provides a generous amount of free public storage and data transfer to help you get started. Sign in to the [CloudKit Dashboard](#) to view your quota and project usage.

<b>10 GB</b> Asset storage	<b>100 MB</b> Database storage	<b>2 GB</b> Data transfer	<b>40</b> Requests per second
-------------------------------	-----------------------------------	------------------------------	----------------------------------

# CLOUDKIT

- Nice problem to have



## Capacity scales with your users.

The amount of public storage and data transfer allocated to your apps will grow with every new active user—all for free with very high limits. Calculate the amount of storage you'll gain as your number of active users grows.

**10,000,000**

Active Users

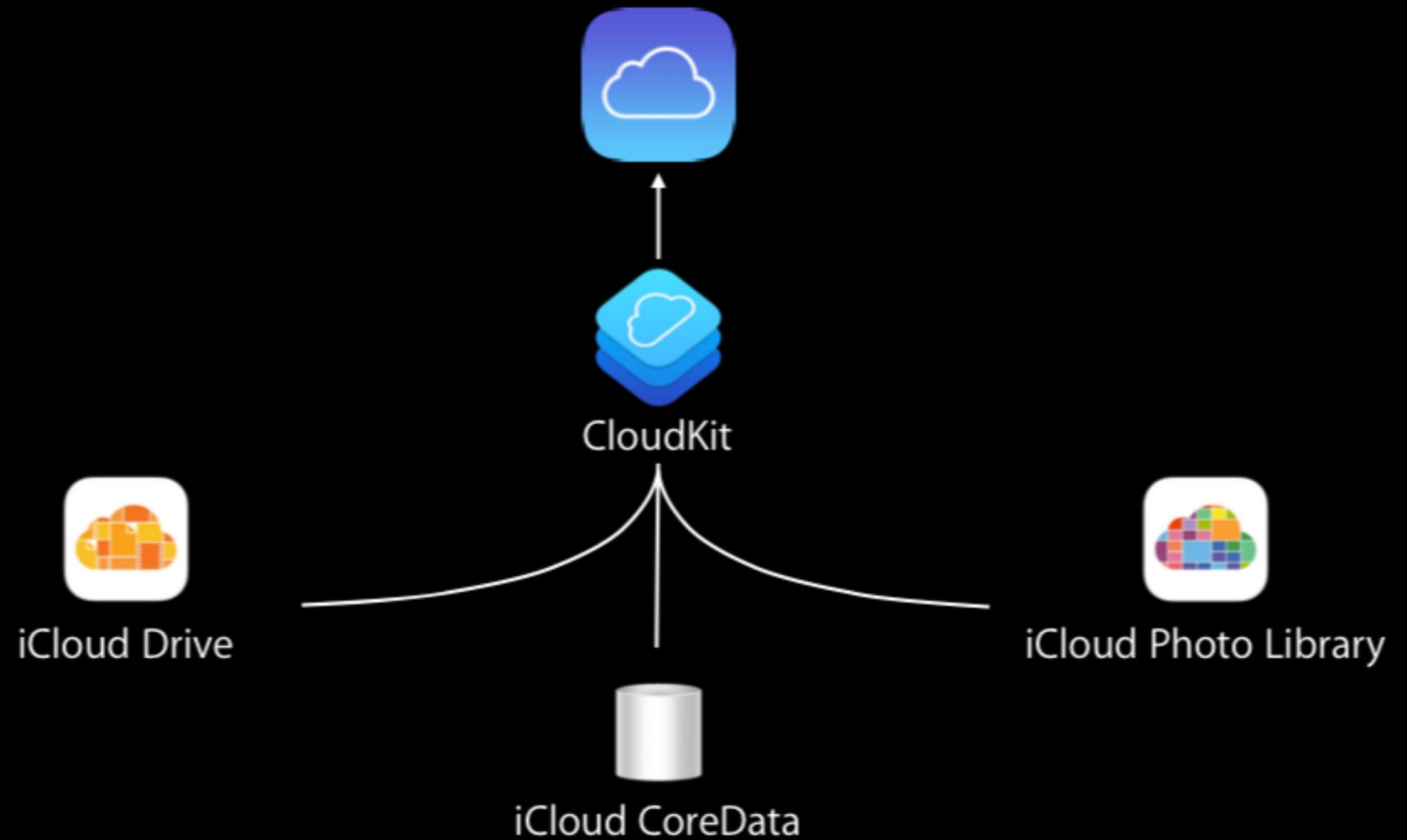
<b>1 PB</b> Asset storage  Based on: 100 MB per user	<b>10 TB</b> Database storage  Based on: 1 MB per user	<b>200 TB</b> Data transfer  Based on: 20 MB per user	<b>400</b> Requests per sec.  Based on: 4 per 100,000 users
--	--	---	---

**\$0**

Total Cost

# CLOUDKIT

- Uses iCloud accounts
  - Logged in accounts used to identify user
  - Not logged in users can have read only anonymous access



# CLOUDKIT

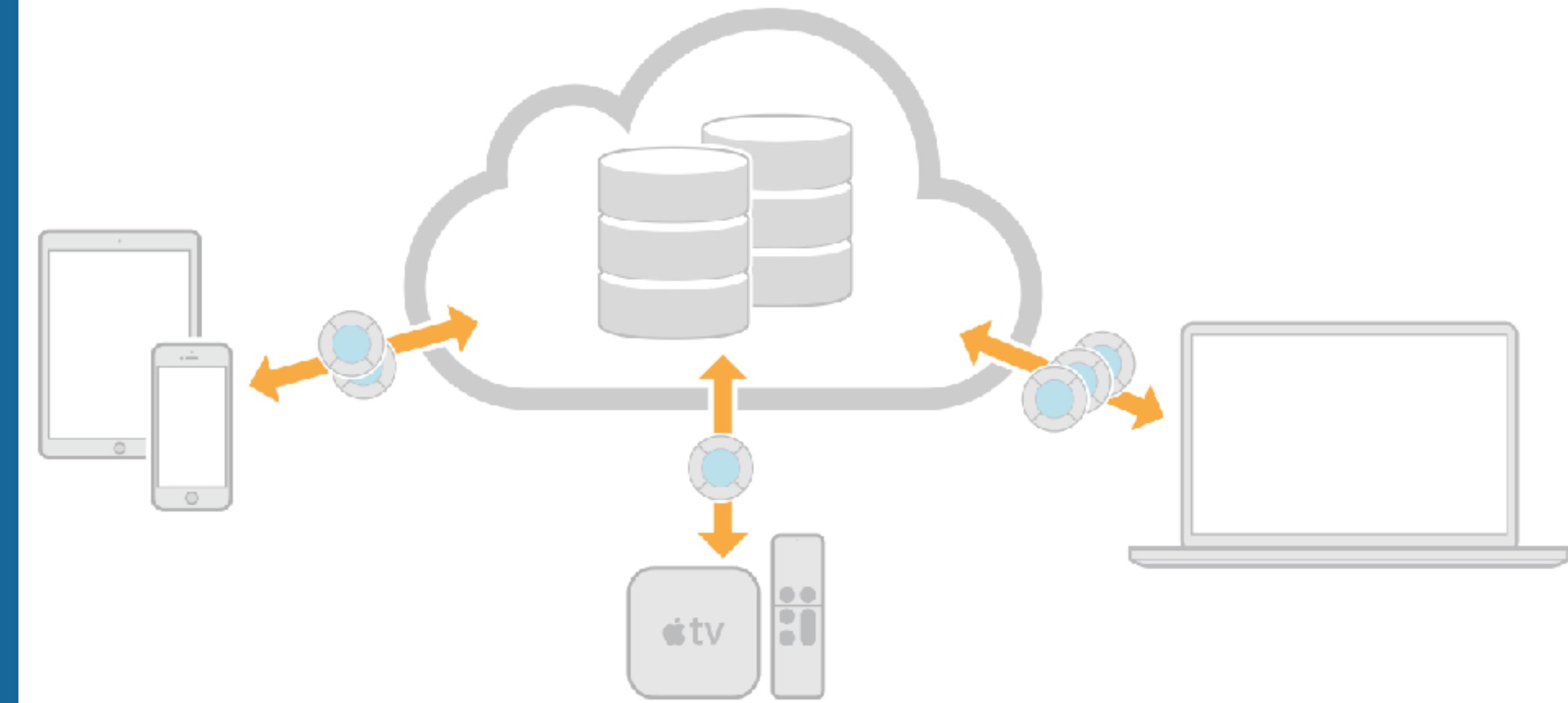
- Databases
  - Public
  - Private
  - Shared



# CLOUDKIT

SUBTITLE

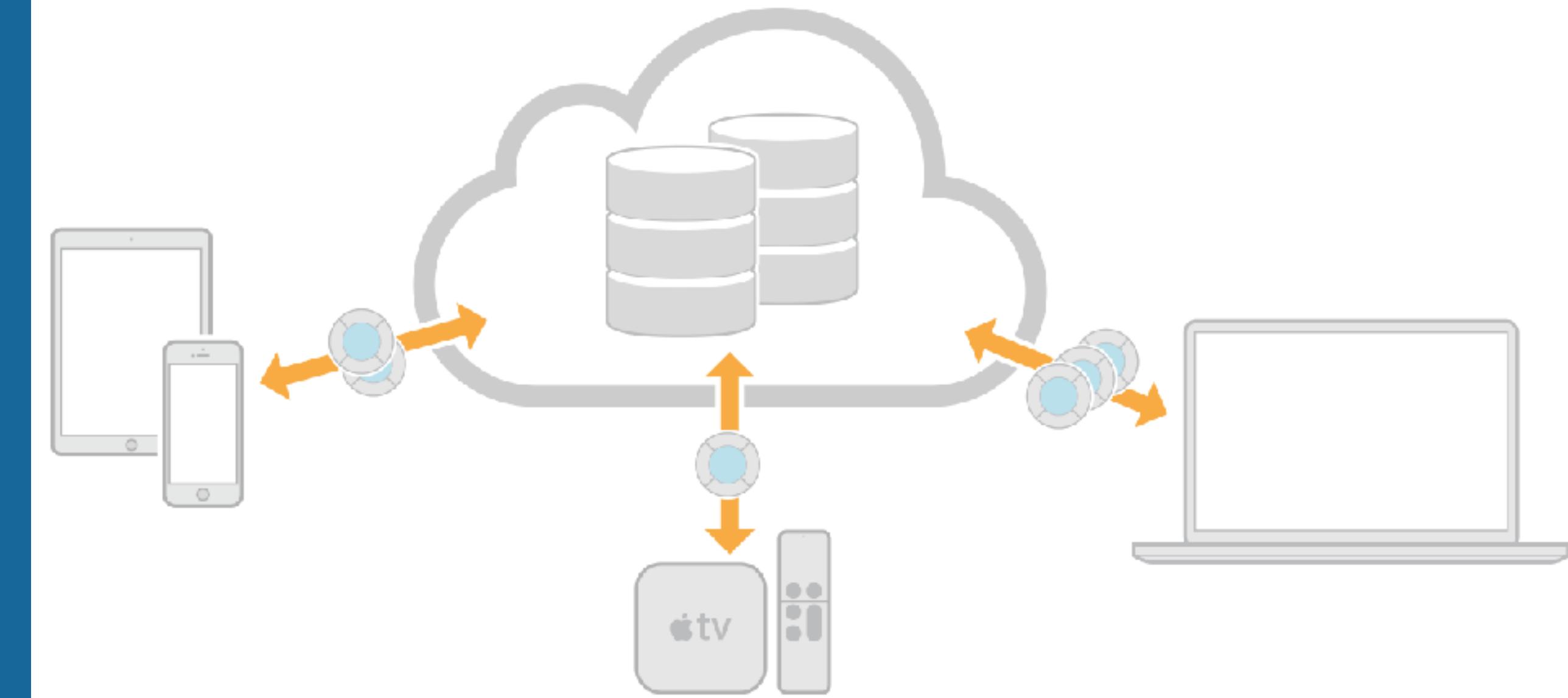
- Structured and bulk data
  - Data with types
  - Blobs



# CLOUDKIT

## CORE OBJECTS

- Containers
- Databases
- Records
- Record Zones
- Record Identifiers
- Shares
- References
- Assets

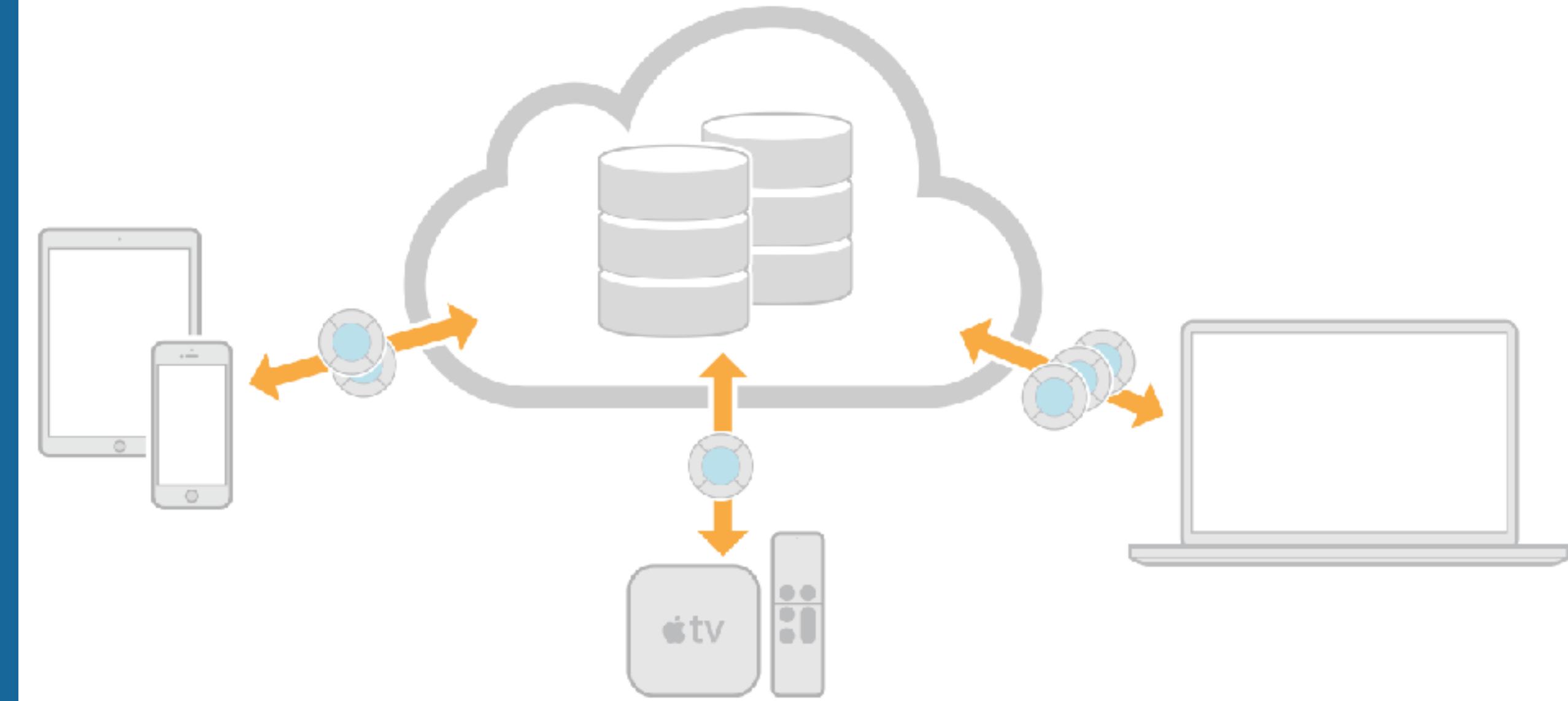


# CLOUDKIT

SUBTITLE

- Transport layer, not local persistence
- You still need to figure out what to do with the data once you have it on your device
  - Cache
  - Mirror

#2 CONSIDERATION ON  
CHOOSING OVER OTHER  
SERVICES



# CLOUDKIT

- CloudKit's place in mobile backends -  
The good
  - Convenient
  - Free
  - Zero authentication
  - Push notifications
  - Proven (Apple actually uses it)



# CLOUDKIT

- CloudKit's place in mobile backends -  
The good
  - Convenient
  - Free
  - Zero authentication
  - Push notifications
  - Proven (Apple actually uses it)



# CLOUDKIT

- CloudKit's place in mobile backends -  
The bad
  - No local storage
  - Platform lock  
(account & device)
  - No native server logic



# CLOUDKIT

SUBTITLE

- CloudKit's place in mobile backends - The bad
  - No local storage
  - Platform lock (account & device)
  - No native server logic

WE WILL DISCUSS HOW THESE MAY BE LESS OF AN ISSUE THAN THEY SEEM



# ENABLING CLOUDKIT IN YOUR APPLICATION

# ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the Xcode interface with a project named "CloudKit-1" selected. The "Signing & Capabilities" tab is active. In the "TARGETS" section, the "CloudKit-1" target is selected. Under the "Signing" heading, the "Automatically manage signing" checkbox is checked, and the "Team" dropdown is set to "University of Chicago (Department of Compu...)" with a provisioning profile "Xcode Managed Profile". The "Bundle Identifier" is set to "mobi.uchicago.CloudKit-1". A note at the bottom says "Add capabilities by clicking the "+" button above." To the left, a yellow callout bubble contains the text "NEED PAID DEVELOPER ACCOUNT OR SCHOOL PROGRAM".

CloudKit-1: Ready | Today at 4:27 PM

CloudKit-1

CloudKit-1

General Signing & Capabilities Resource Tags Info Build Settings Build Phases Build Rules

PROJECT CloudKit-1

+ Capability All Debug Release

TARGETS CloudKit-1

Signing

Automatically manage signing  
Xcode will create and update profiles, app IDs, and certificates.

Team University of Chicago (Department of Compu...)

Bundle Identifier mobi.uchicago.CloudKit-1

Provisioning Profile Xcode Managed Profile ⓘ

Signing Certificate iPhone Developer: Andrew Binkowski (8T47C82J...)

Add capabilities by clicking the "+" button above.

Identity and Type

Name CloudKit-1

Location Absolute

CloudKit-1.xcodeproj

Full Path /Users/tabinkowski/Google Drive/g-Teaching/uchicago.cloud/mpcs51033-2019-autumn/mpcs51033-2017-autumn-code-samples/CloudKit-1/CloudKit-1.xcodeproj

Project Document

Project Format Xcode 9.3-compatible

Organization T. Andrew Binkowski

Class Prefix

Text Settings

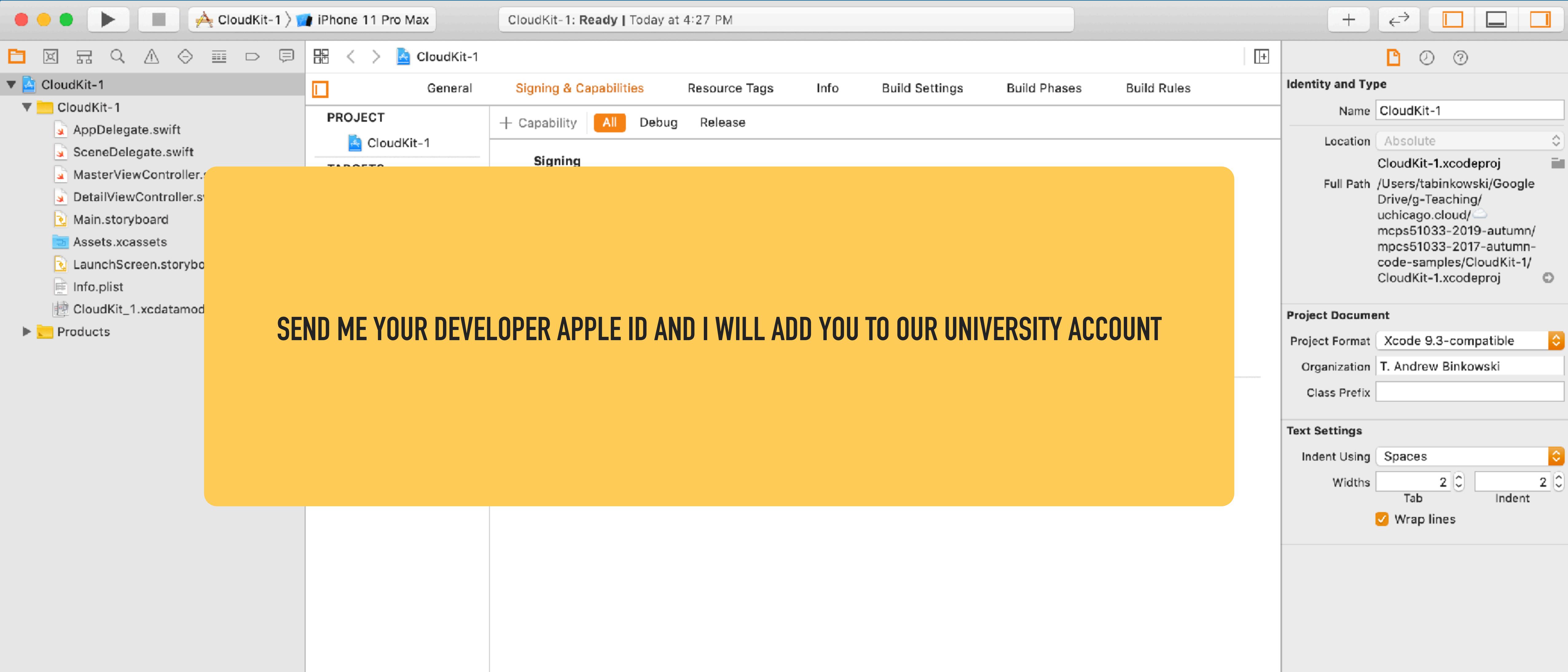
Indent Using Spaces

Widths 2 Tab 2 Indent

Wrap lines

NEED PAID DEVELOPER ACCOUNT OR SCHOOL PROGRAM

# ENABLING CLOUDKIT IN YOUR APPLICATION



# ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the Xcode interface with a project named "CloudyWithAChanceOfErrors". A yellow callout box labeled "NEW CAPABILITY" points to the "CloudKit" capability in the list. Another yellow callout box labeled "CUSTOM CONTAINERS CAN BE SHARED BETWEEN APPS" points to the "iCloud" capability, which is highlighted with an orange selection bar.

**CloudKit**

**iCloud**

iCloud Storage APIs enable your apps to store data and documents in iCloud, keeping your apps up to date automatically. Use iCloud to give your users a consistent and seamless experience across iCloud-enabled devices.

Capabilities

- HealthKit
- HomeKit
- Hotspot Configuration
- iCloud**
- In-App Purchase
- Inter-App Audio
- Keychain Sharing
- Maps
- Multipath
- Near Field Communication Tag R...
- Network Extensions

# ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the Xcode CloudKit dashboard. On the left, there's a sidebar with files like 'CloudKit.storyboard', 'Info.plist', 'CloudKit\_1.xcdatamodeld', and a 'Products' folder. A yellow speech bubble points from the top-left towards the center with the text 'NEW CAPABILITY'. Another yellow speech bubble points from the bottom-left towards the center with the text 'CUSTOM CONTAINERS CAN BE SHARED BETWEEN APPS'. The main area displays provisioning profile and signing certificate information. It shows an error message: 'Status Failed to register bundle identifier. The app identifier "mobi.uchicago.CloudKit-1" cannot be registered to your development team. Change your bundle identifier to a unique string to try again.' Below this are three detailed error messages about provisioning profiles not supporting iCloud or missing entitlements. At the bottom, there are sections for 'Services' (checkboxes for Key-value storage, iCloud Documents, and CloudKit) and 'Containers' (checkboxes for various iCloud.CloudKitOperation types). A '+' button is at the bottom right.

CloudKit.storyboard  
Info.plist  
CloudKit\_1.xcdatamodeld  
► Products

Provisioning Profile Xcode Managed Profile ⓘ  
Signing Certificate Apple Development

Status Failed to register bundle identifier.  
The app identifier "mobi.uchicago.CloudKit-1" cannot be registered to your development team. Change your bundle identifier to a unique string to try again.

Try Again

⚠ Provisioning profile "iOS Team Provisioning Profile: \*" doesn't support the iCloud capability.

⚠ Provisioning profile "iOS Team Provisioning Profile: \*" doesn't include the com.apple.developer.icloud-container-identifiers entitlement.

▼ iCloud

Services  Key-value storage  
 iCloud Documents  
 CloudKit

Containers  iCloud.CloudKitOperation  
 iCloud.CloudKitOperation.TodayView  
 iCloud.CloudKitOperation.photoShare

+

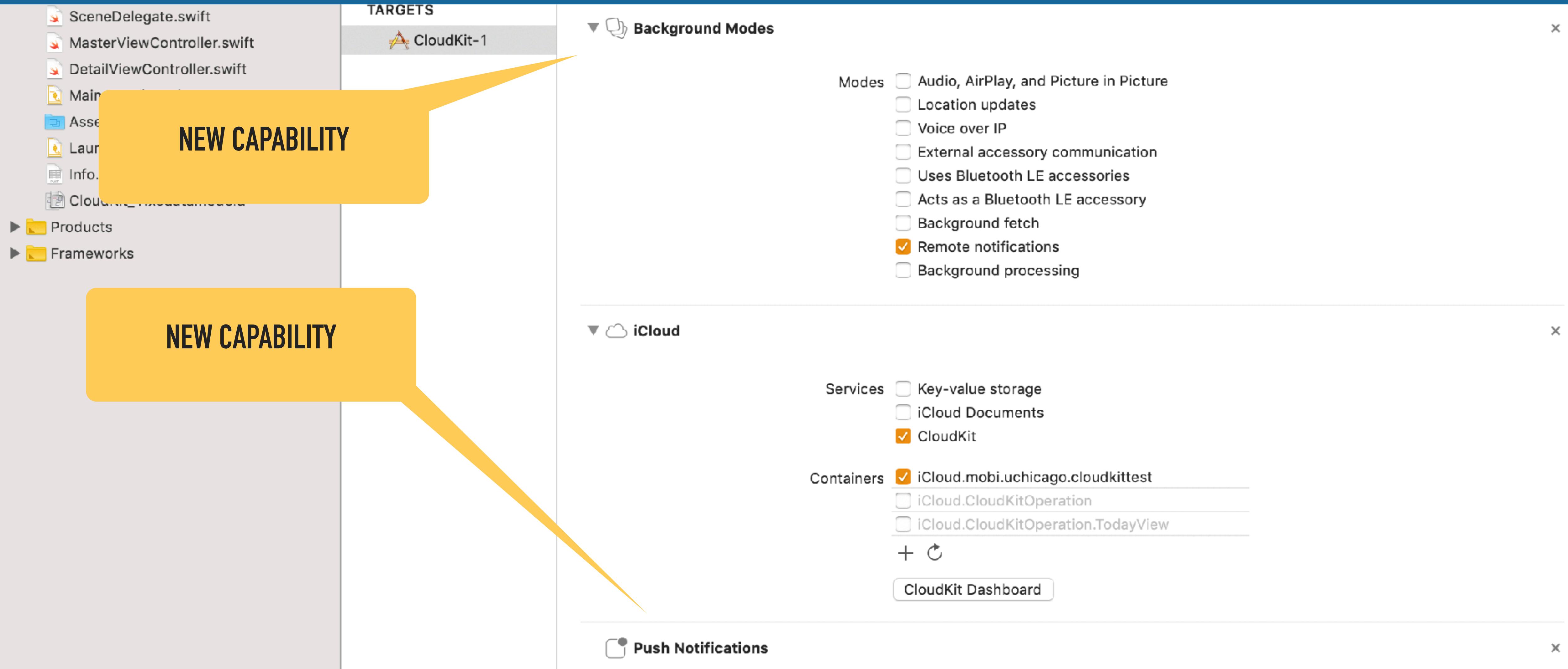
CloudKit Dashboard

+

+ Filter

Filter

# ENABLING CLOUDKIT IN YOUR APPLICATION



# ENABLING CLOUDKIT IN YOUR APPLICATION

The screenshot shows the CloudKit Container Permissions interface. On the left, a sidebar lists various iCloud containers. A yellow arrow points from the 'Container Permissions' section in the main area towards the 'Development' section.

**Container Permissions**

**Development**  
Build and test your schema. You can add, modify, and delete record types in the development environment.

**Data** > **Telemetry** > **Logs**

**Schema** > **Usage** > **API Access**

**Production**  
Schema for production environment.  
but not delete record types in the production environment.

**DEFAULT CONTAINER IS CREATED ON ICLOUD**

Search containers

- Andrew Binkowski
- University of Chicago (Department of Co... ⓘ
- iCloud.CloudKitOperation
- iCloud.CloudKitOperation.Today...
- iCloud.CloudKitOperation.photo...
- iCloud.SpotNews
- iCloud.acham1.Joke-of-the-Day
- iCloud.cloud.uchicago.CloudyWit...
- iCloud.cloud.uchicago.chungan....
- iCloud.cloud.uchicago.chungan....
- iCloud.com.SpotNews
- iCloud.com.Xuefeng.Koala
- iCloud.com.alicechicago.Instawa...
- iCloud.com.bennetth.loopyQ2
- iCloud.com.dcaronson.Photo-Jo...

## ENABLING CLOUDKIT IN YOUR APPLICATION

- Dashboard is only way to access permissions
- Data can be created, manipulated
- View analytic information about your data, users and operations

# iCloud.mobi.uchicago.cloudkittest

Container Permissions

## Development

Build and test your schema. You can add, modify, and delete record types in the development environment.



Data



Telemetry



Logs



Schema



Usage



API Access

## Production

Schema for production use. You can add and modify but not delete record types in the production environment.

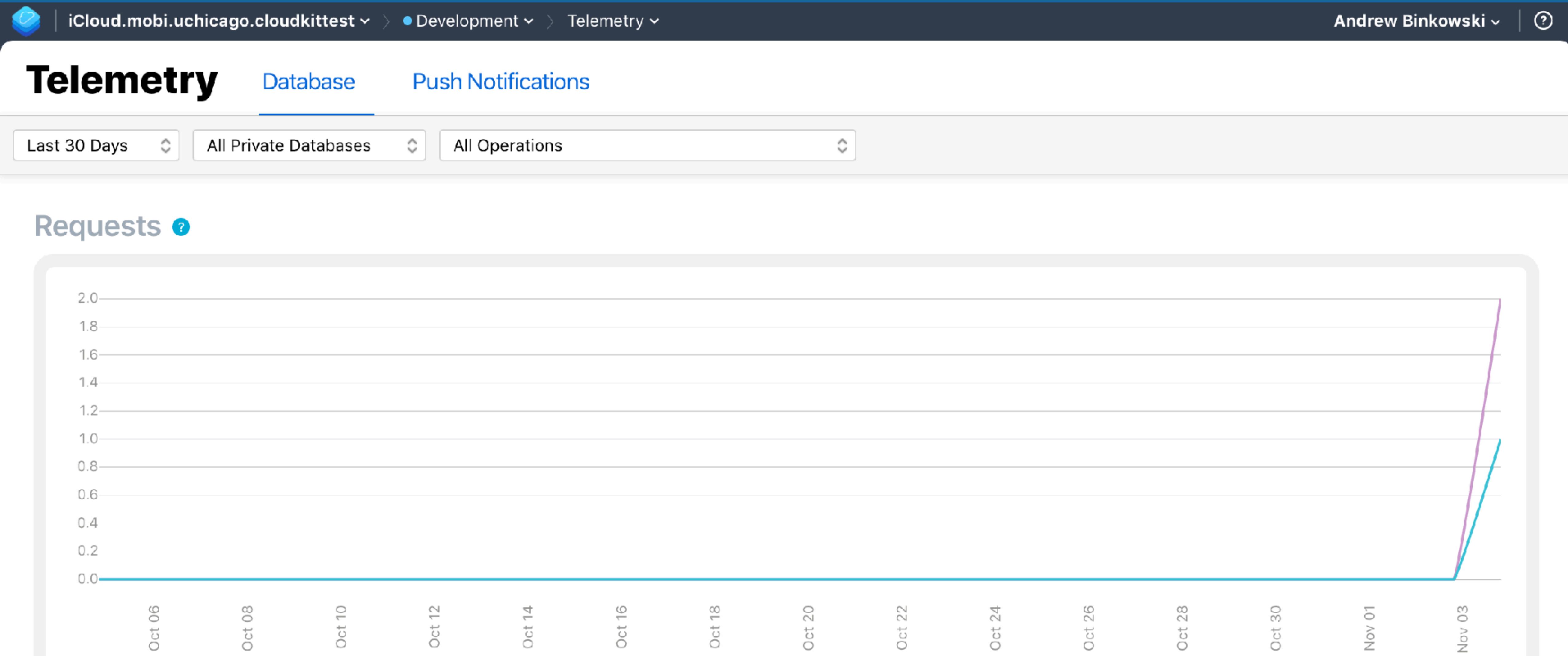
This container has not been deployed to production.

# ENABLING CLOUDKIT IN YOUR APPLICATION

- "Flip a switch" to go to production mode

The screenshot shows the CloudKit Records interface. At the top, there's a navigation bar with the domain "iCloud.mobi.uchicago.cloudkittest", a "Development" dropdown (which is currently selected), and a "Data" dropdown. A tooltip for the "Development" dropdown shows two options: "Development" (selected, indicated by a checkmark and blue dot) and "Production" (indicated by a red dot). Below the navigation, the main title is "Records". The interface includes sections for "Database" (set to "Private Database" and "abinkowski@uchicago.edu"), "Zone" (set to "\_defaultZone"), and "Using" (with tabs for "Query" and "Fetch", where "Query" is selected). Under "Using", there are filters for "Type" (set to "Users"), "Filter" (set to "None"), and "Sort" (set to "None"). On the far right, the word "Query" is partially visible.

# ENABLING CLOUDKIT IN YOUR APPLICATION



# WHEN TO USE CLOUDKIT

# WHEN TO USE CLOUDKIT

- Whenever possible
- When you have a tightly defined problem to solve
- You're developing exclusively for Apple platforms (or willing to do a lot of work)

# WHEN TO USE CLOUDKIT

- Sync is hard
- Lot's of approaches

To be honest I wasn't expecting this complicated an app, this looks like at least a year of coding, maybe even more. Remember the app only syncs a few record types, reading list (bookmarks), history (read articles) and personalisation data like favourite feeds. Implementing syncing is going to be no easy task at all; no wonder we are all clueless after a few hours of WWDC talks! Hopefully some of my research has helped you though.

**malhal**

iOS developer, since the beginning



malhal

13th January 2017

CloudKit

Apple WWDC  
app

## CloudKit Sync Nightmare

So after [previous post on analysing the web requests](#), where we found out the classes involved and sort of when they should be used, we are still clueless as to how to structure our app to enable CloudKit syncing. We know [we shouldn't subclass fetch operation](#) but then how should we gather our information to pass to the

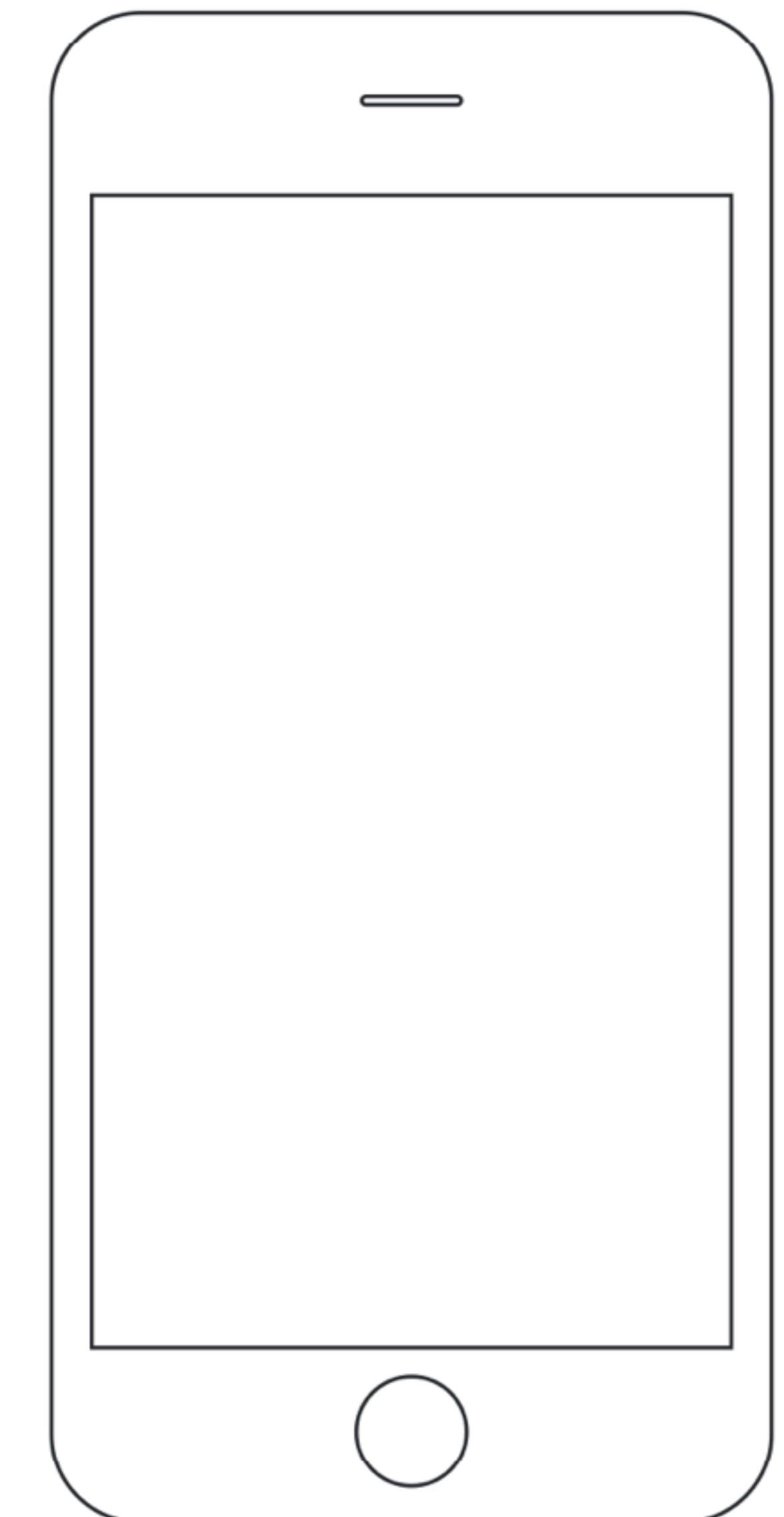
First of all we notice 336 files. 336 files just for the networking and caching, not the UI. Unbelievable, but lets carry on anyway.

A lot of the CloudKit related files are prefixed with FCCK, 27 of them! And there are many more like controllers etc. that use these objects. Maybe half the files have a reference to some aspect of CloudKit one way or another.

# ASSIGNMENT 3

# ASSIGNMENT 3

- Picture Journal App
  - Users select a picture and add a caption
  - Meta data generated from picture
  - Share and Today Extensions
  - Sync using CloudKit subscriptions



# ASSIGNMENT 3

Day One



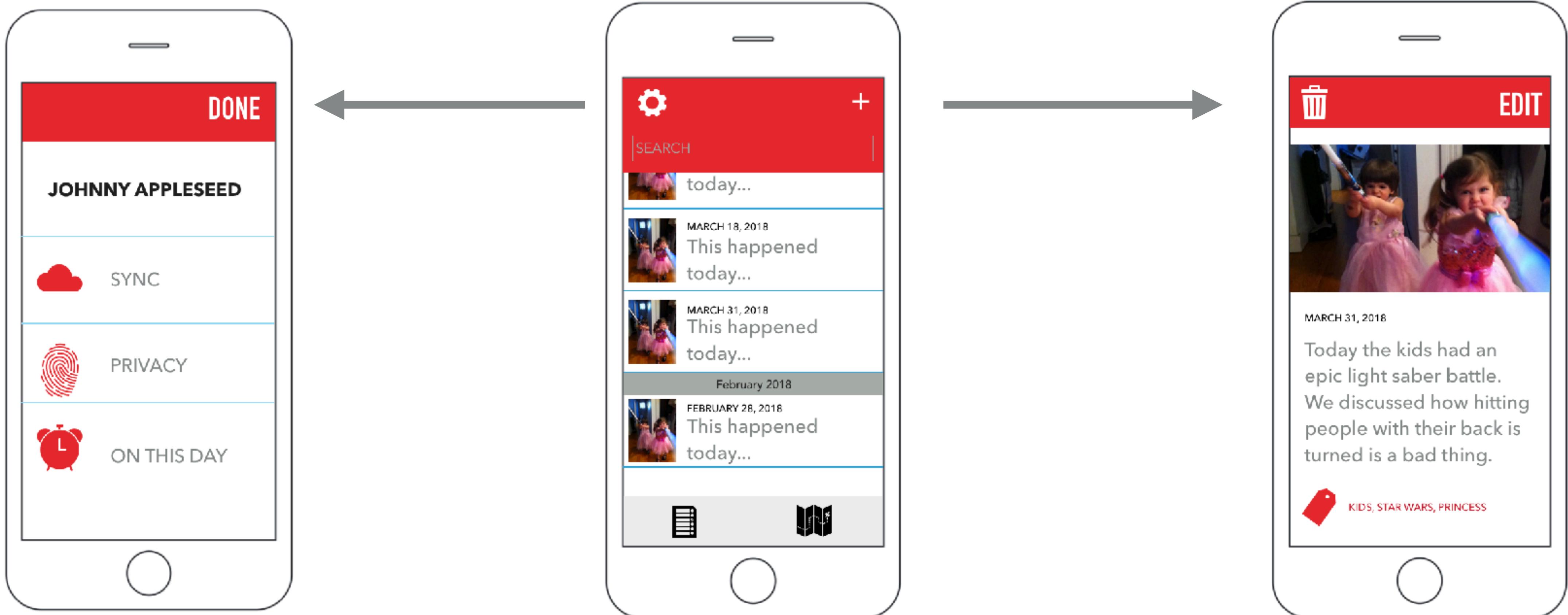
# ASSIGNMENT 3

- Day One
  - Apple Design Award winner
  - Success as a paid app (for a while)
  - Moved to subscription model
  - Move from using Dropbox flat files to iCloud Drive to custom sync solution



# ASSIGNMENT 3

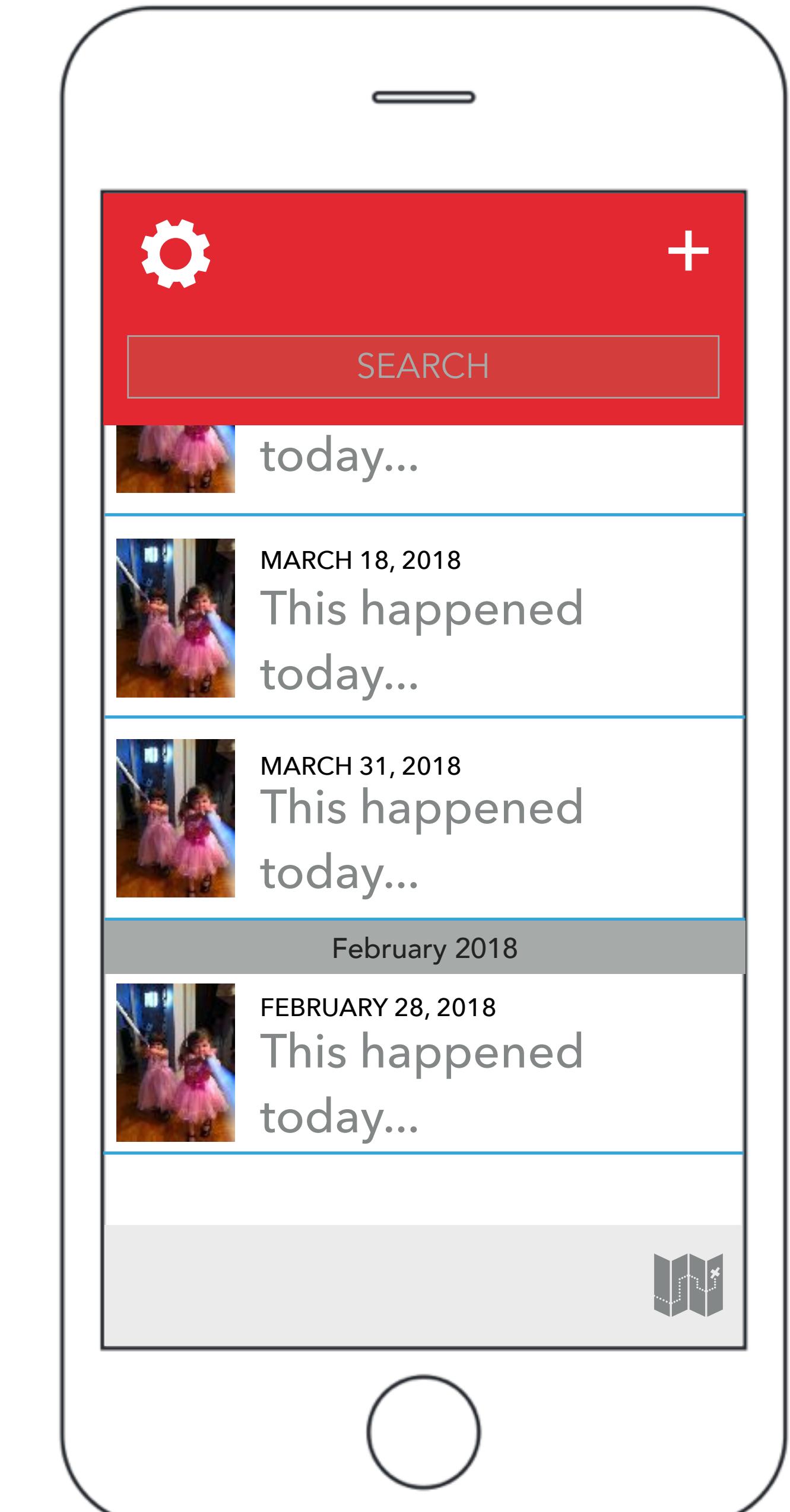
## OVERVIEW



# ASSIGNMENT 3

## HOME SCREEN

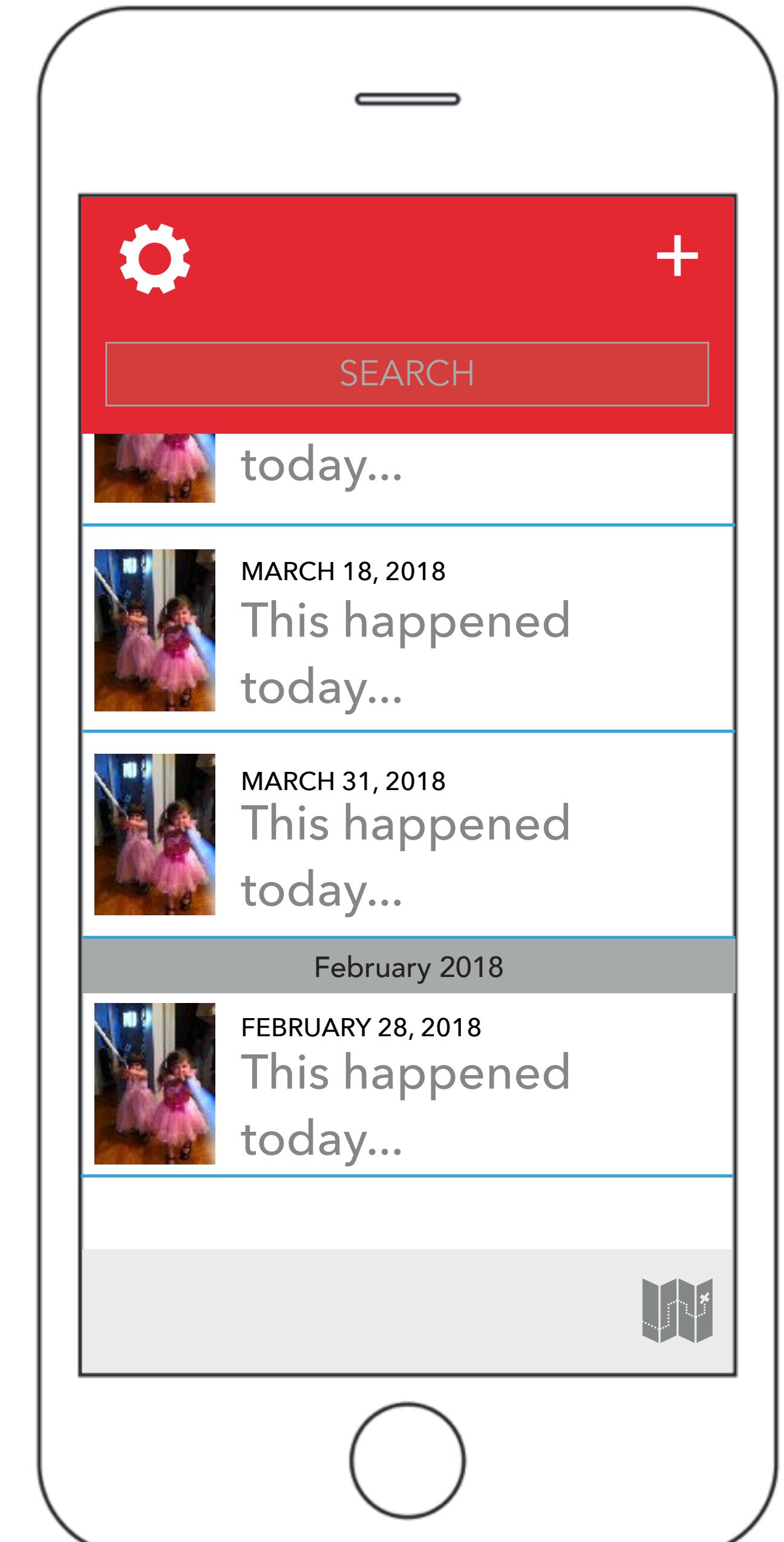
- Home screen is list
  - TableView (or CollectionView) should be segmented by month
  - Cell should display thumbnail, text blurb, and date (can include more if like)
  - Search bar should search by text and tags



# ASSIGNMENT 3

## HOME SCREEN

- Map screen will show locations from entries
  - Tapping on entry will show all entries in that location via location based query
  - Show back on main home screen



# ASSIGNMENT 3

## ADD/EDIT SCREEN

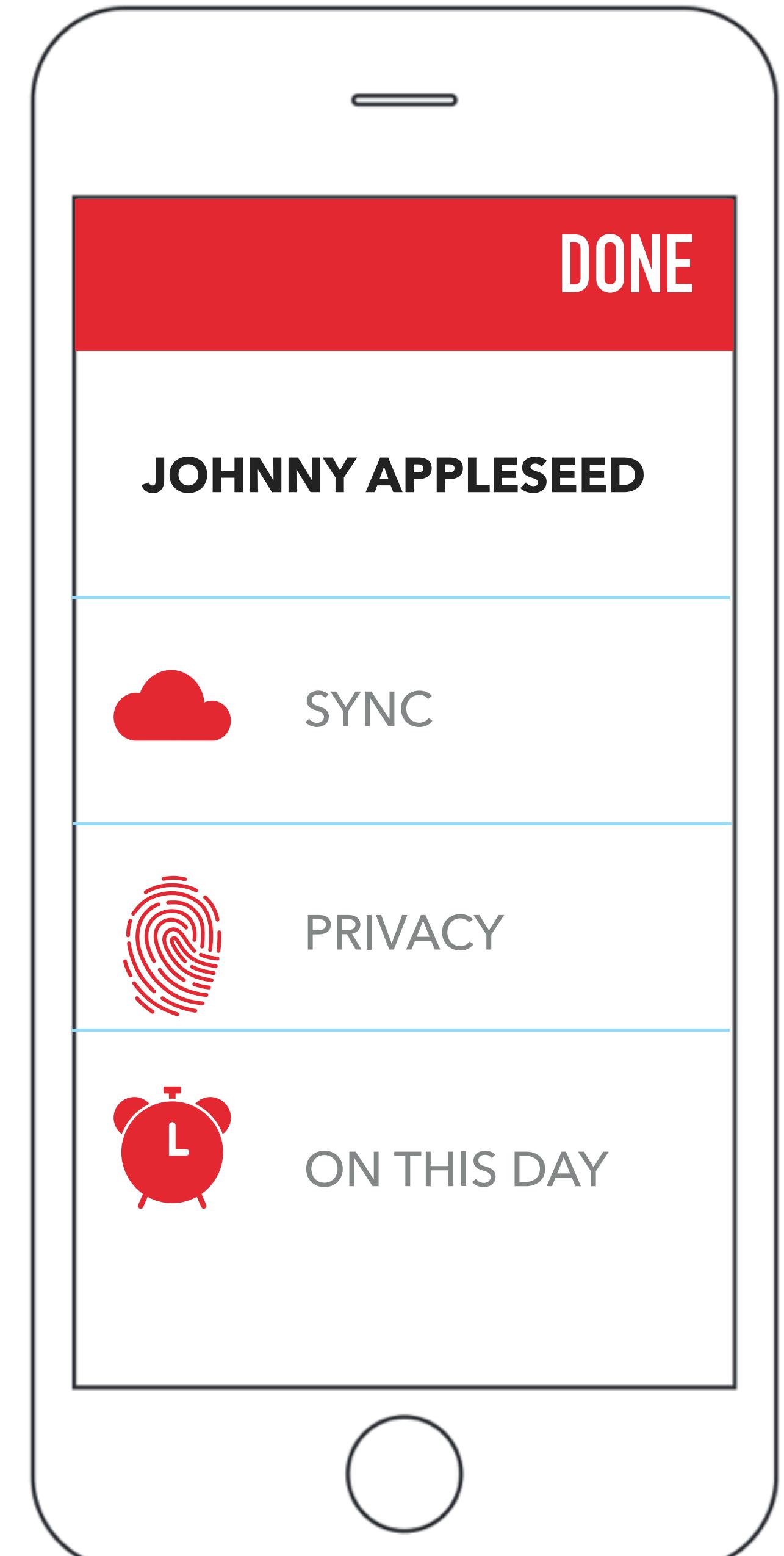
- User is in app and selects add a new entry
  - Entry has a photo and caption text, tags (user generated)
  - Meta-data is automatically generated
    - GPS, Date: From Photos framework
    - Weather: <https://darksky.net/dev>
  - Tags: Vision, NLP framework
  - Stored as a custom struct and serialized to disk as JSON
  - Everything can be edited (including date)



# ASSIGNMENT 3

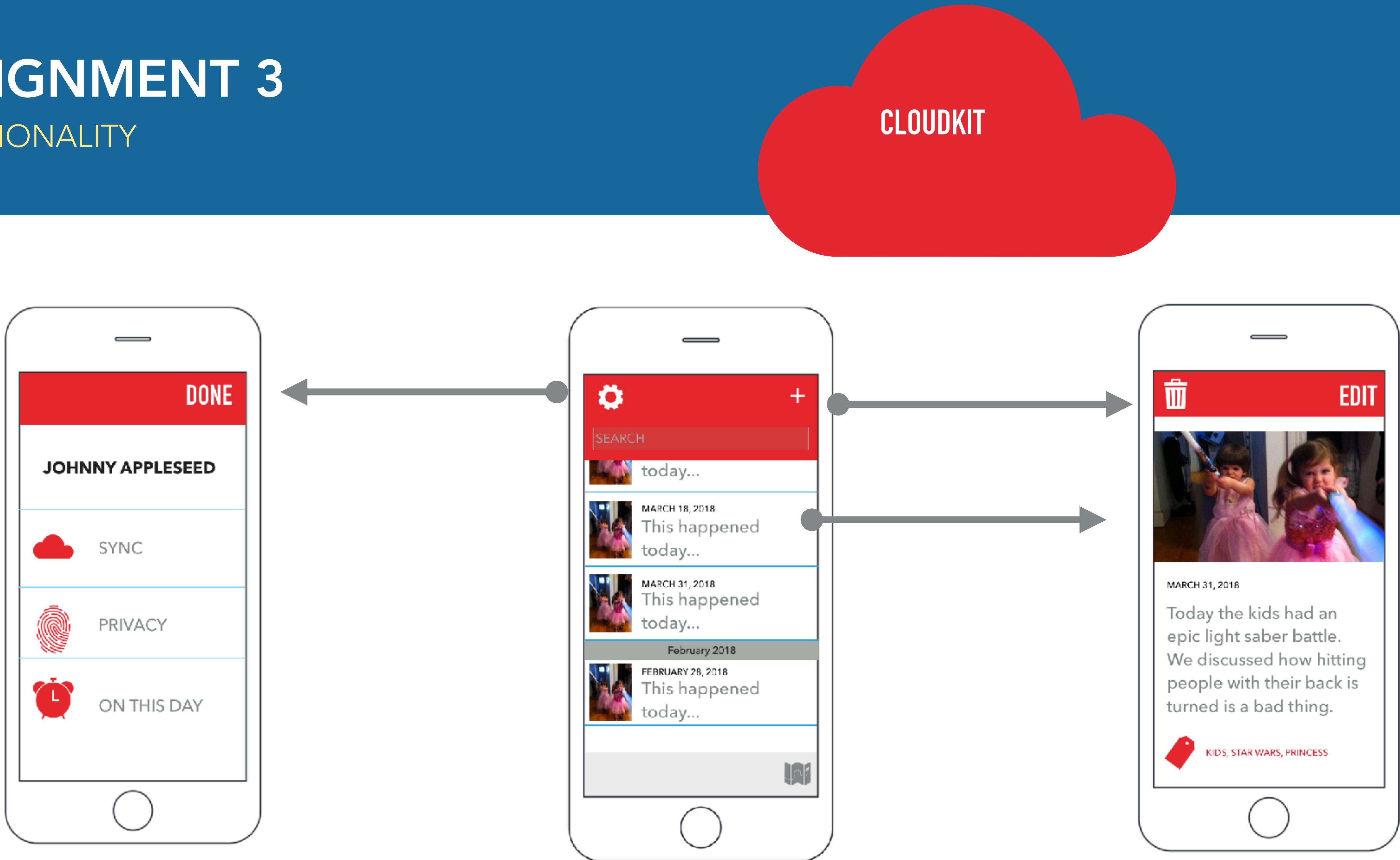
## SETTINGS SCREEN

- Settings screen
  - Show iCloud user name
  - Allow user Sync (reset local cache); provide good feedback
  - Set a passcode
  - Bonus Point (on this day notification)



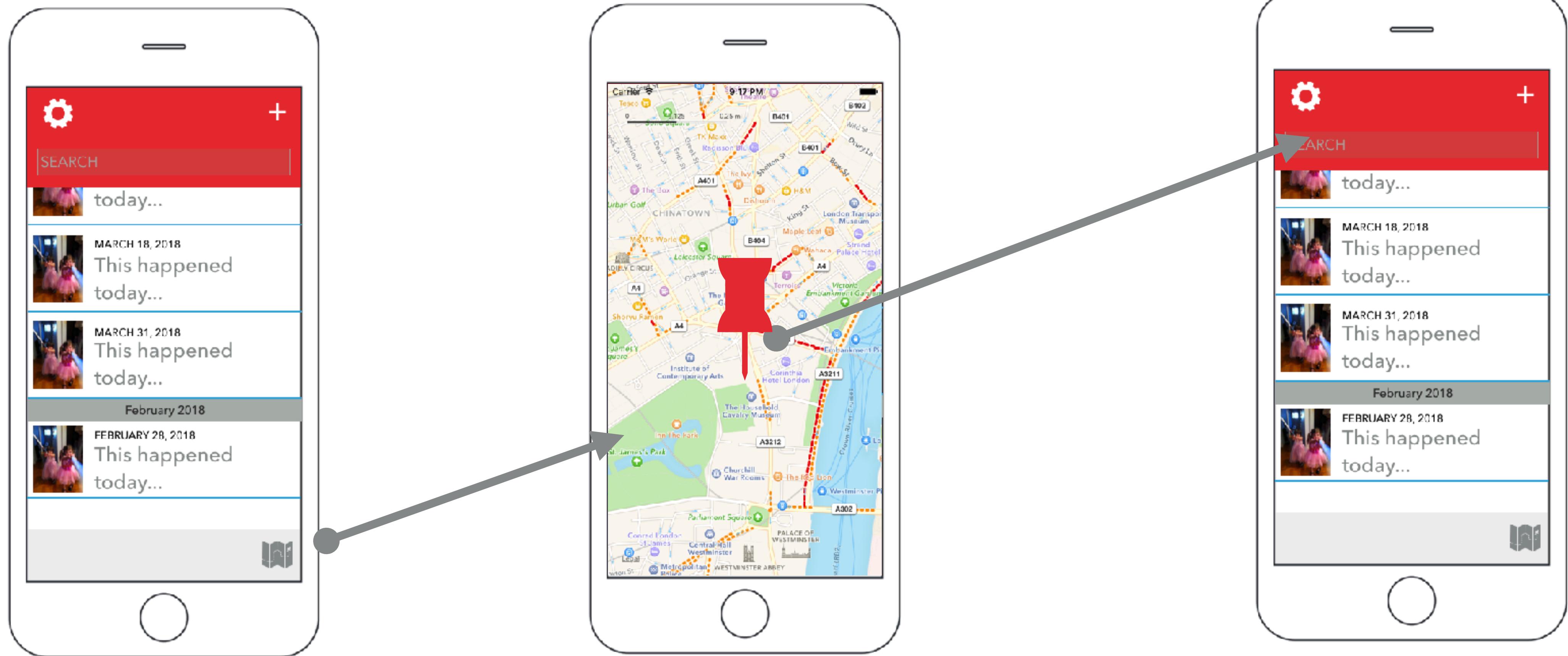
# ASSIGNMENT 3

## FUNCTIONALITY



# ASSIGNMENT 3

## FUNCTIONALITY

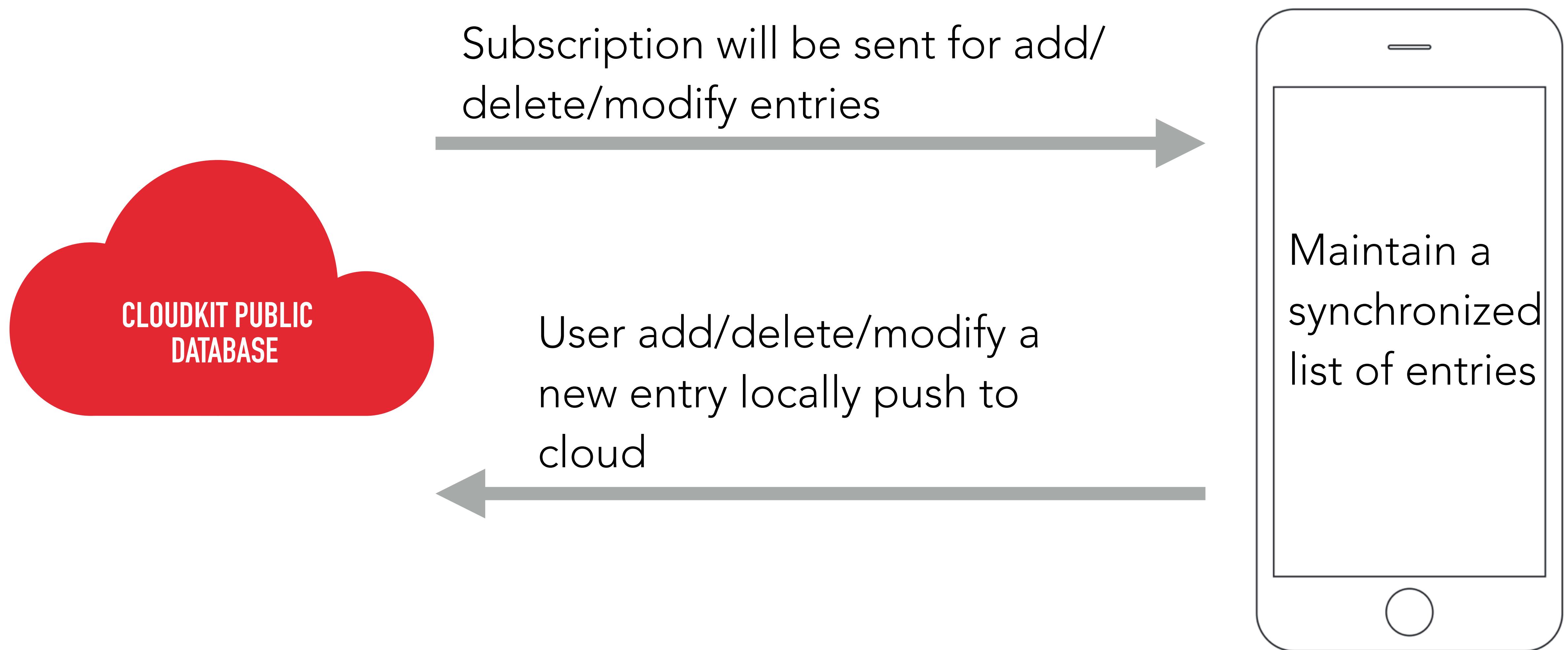


# ASSIGNMENT 3

- Sync is a hard problem
  - Local caching
  - Mirroring
  - Platform services (Firebase, Realm, not CloudKit)
- We are going to use a simple and free strategy using CloudKit public database



# ASSIGNMENT 3



# ASSIGNMENT 3

On app launch

- └─ Fetch changes
- └─ Subscribe to future changes

On push from CloudKit

- └─ Fetch changes



# ASSIGNMENT 3

## OFFLINE-FIRST APPROACH

- Local change
  - Save to local data
  - Mark `synced = false`
  - Update UI (for user's sake)
  - Push to CloudKit
  - On success `synced = true`
  - Save the cloudKit `recordId` to local object

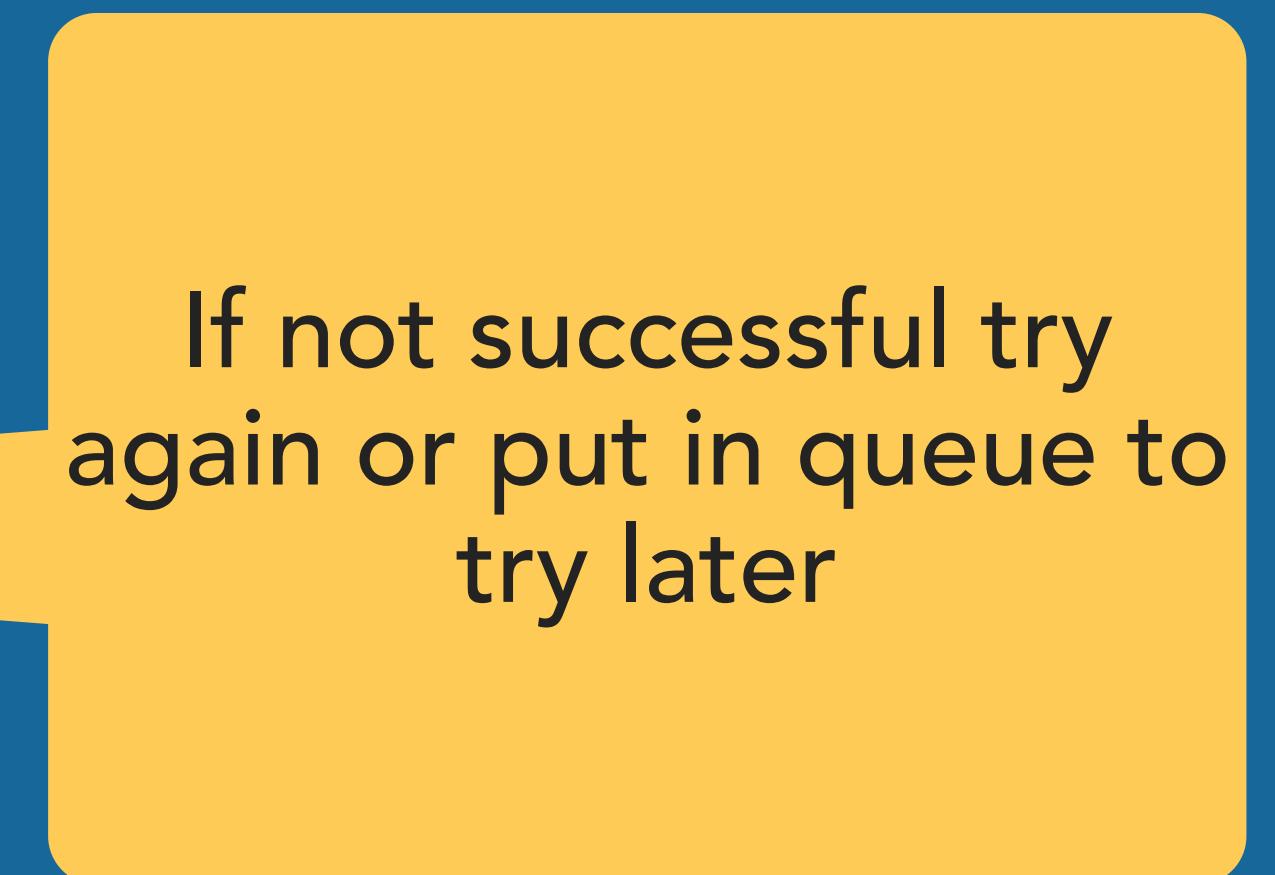


If not successful try again or put in queue to try later

# ASSIGNMENT 3

## OFFLINE-FIRST APPROACH

- Remote change received via subscription
  - Find local object that matches (recordId)
  - Mark `synced = false`
  - Update local object to match changes
  - Update UI (for user's sake)
  - On success `synced = true`



If not successful try again or put in queue to try later

# ASSIGNMENT 3

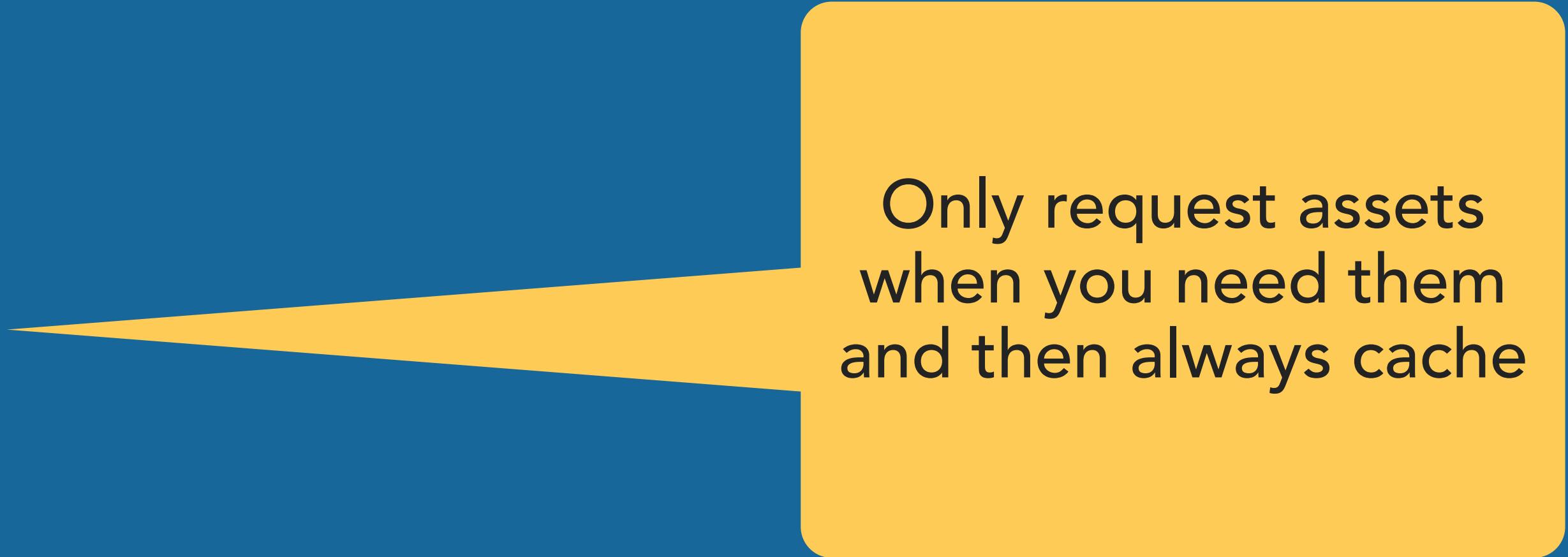
## SYNC FLOW

- Assumptions that make our simple sync approach plausible
  - Journal is single user (not many potential conflicts)
  - Sync data is just text (don't request images until we need them)
  - We can restore the database in a reasonable amount of time
  - Probably not that much data (~1 entry a day)

# ASSIGNMENT 3

## IMAGES

- Cache images
  - NSCache or Cache directory
  - They will be on iCloud if needed
- Ideas for performance
  - Resize images
  - Create thumbnail cache for table



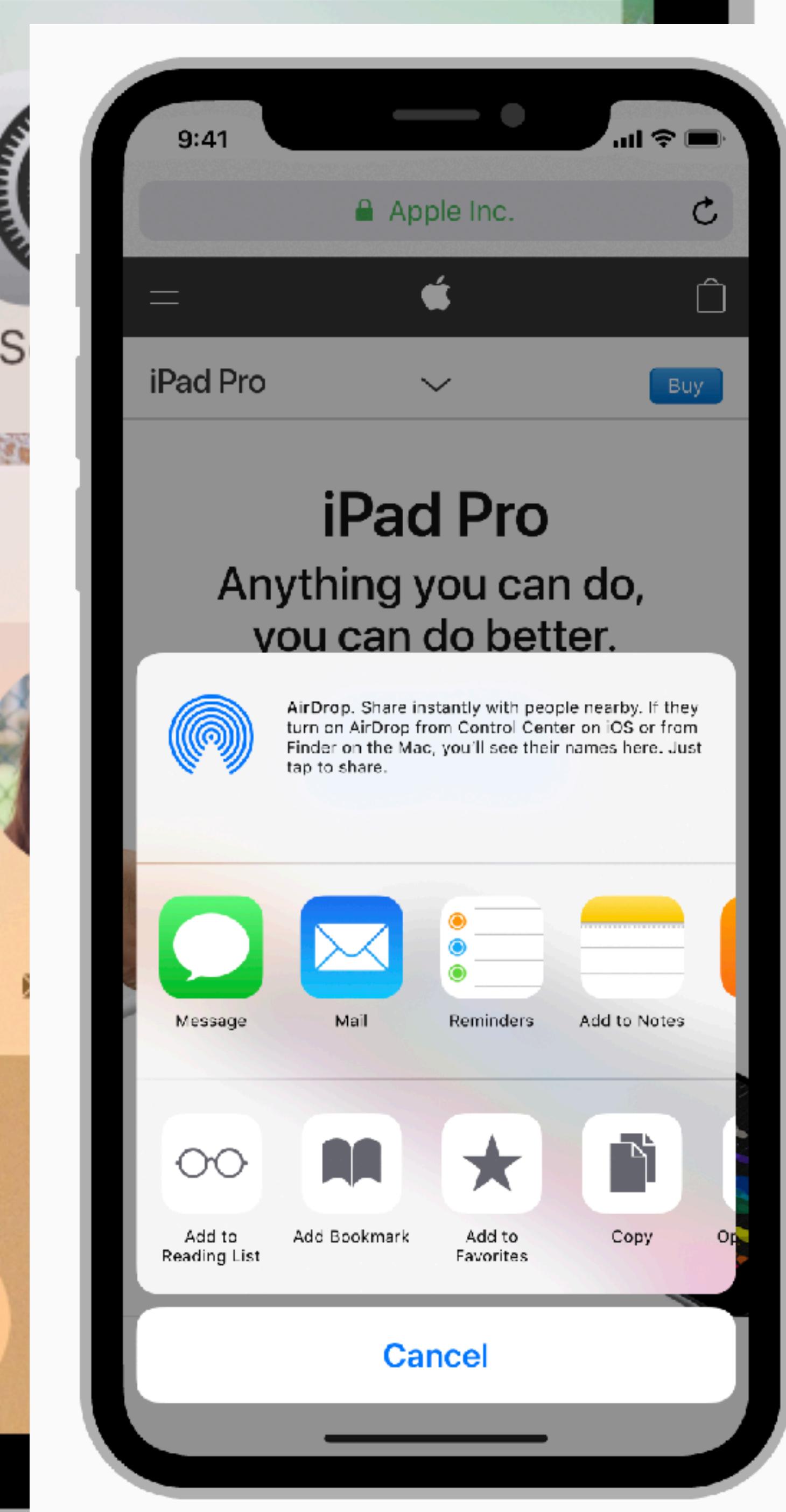
Only request assets  
when you need them  
and then always cache

# RECOMMENDED WORK FOR THIS WEEK

- Entry input, persist, and edit
  - Select photo(s); metadata extraction
  - Input text
  - Input tags
- Sync with CloudKit
  - Subscriptions
  - Dirty flag
- Notifications
- TableView screen
  - Searchbar with query
  - Section headers
  - Map point to search

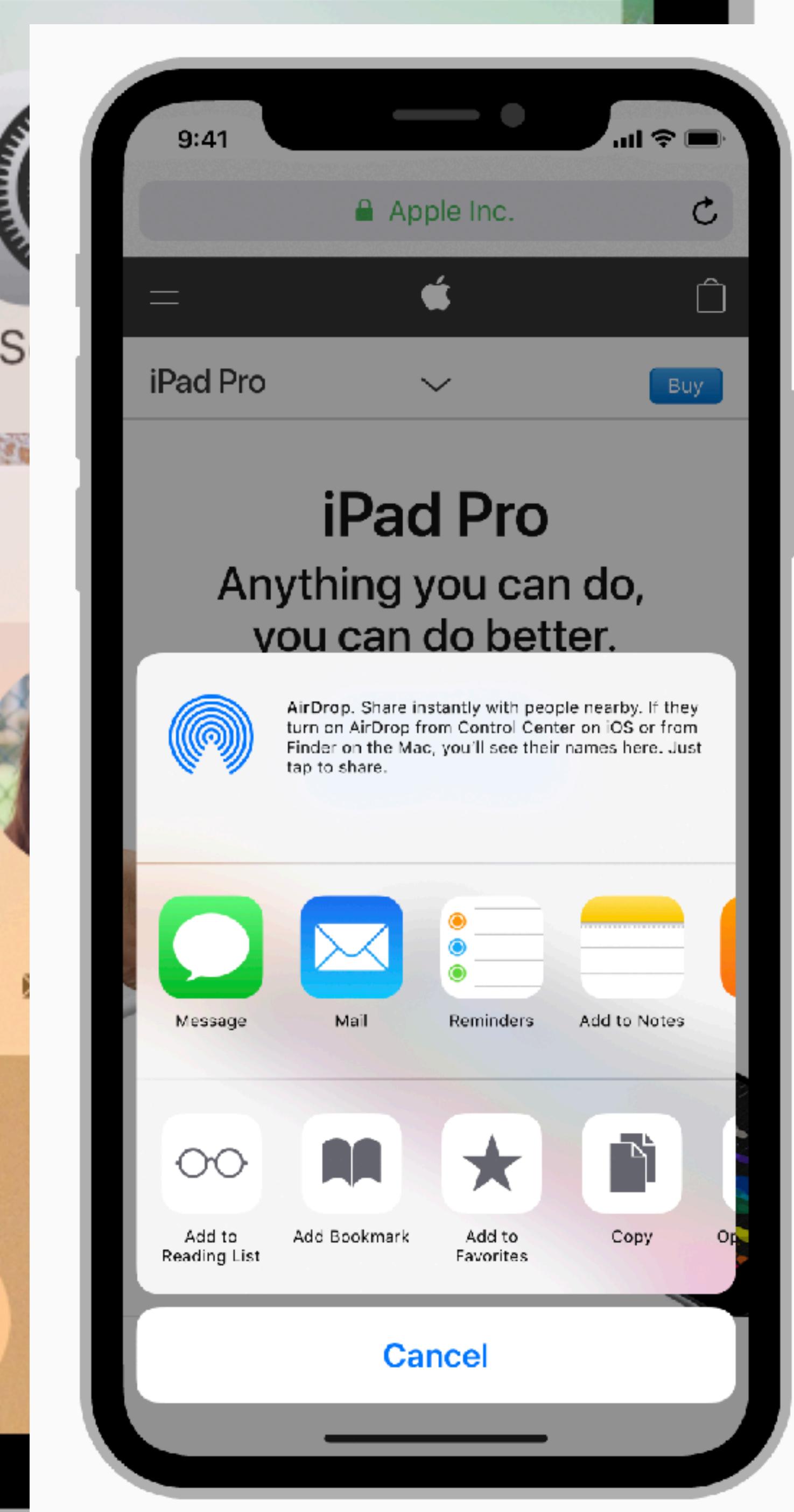
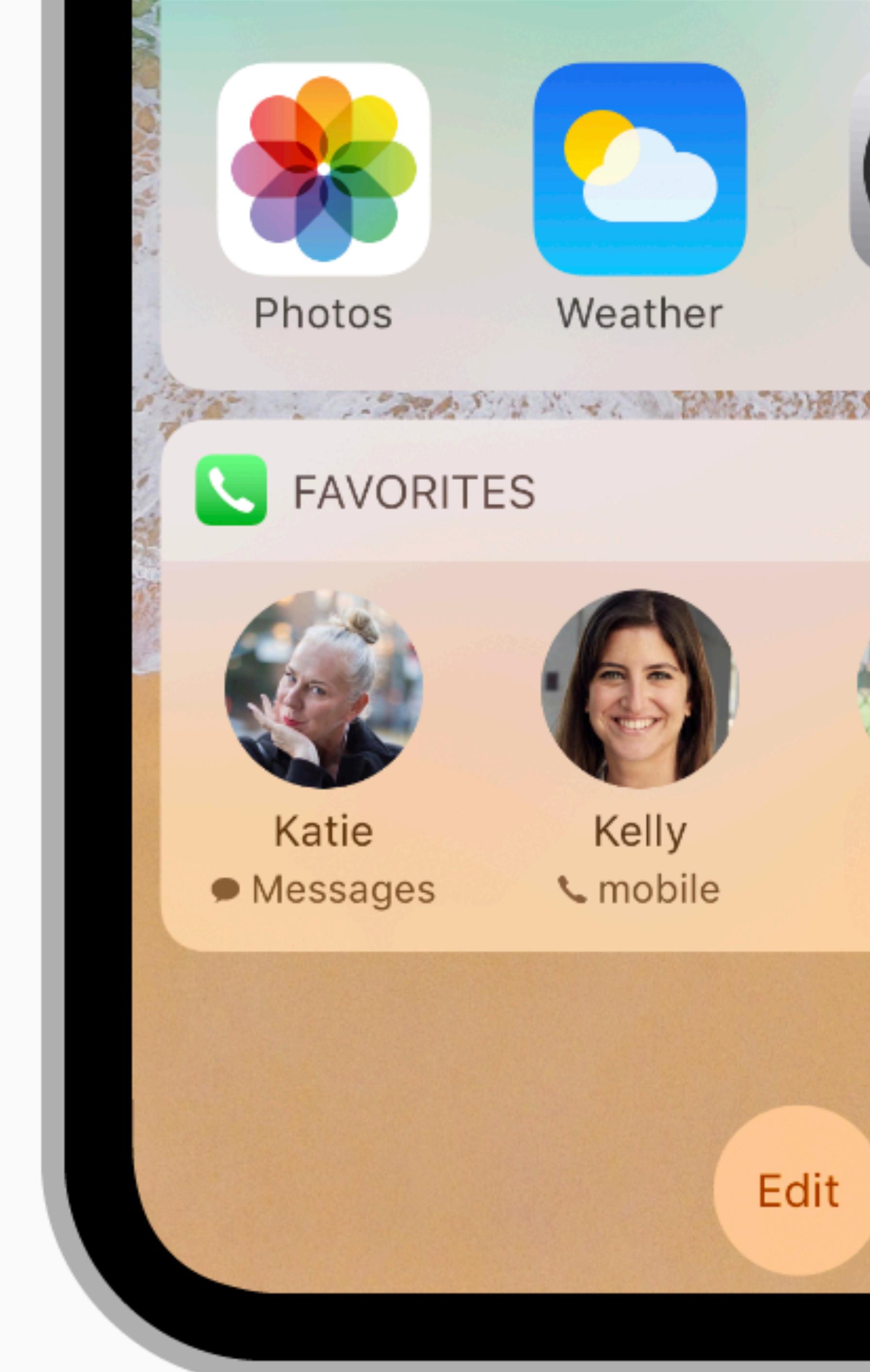
# GOING OVER NEXT WEEK

- Extensions
  - Today
  - Share
- Embedded Frameworks
- App Groups



# GOING OVER NEXT WEEK

- 3D Touch
- Vision Framework
- Background Tasks





# ADVANCED iOS APPLICATION DEVELOPMENT

---

MPCS 51032 • SPRING 2020 • SESSION 3