Welcome!

https://cs342.stanford.edu

cs342-aut1920.slack.com

Santiago Gutierrez
Oliver Aalami
Mike Hittle

Fall 2019
**aim to be done with assign5 by next class**
Assignment #6: Backend

Connect your apps to a GCP backend server

This is the last assignment of the course before your final presentations. We will be integrating your apps with the Google Cloud Platform (GCP), focusing on: authentication, data storage, and simple data visualization.
Overview for today

● Last steps to setup Firestore
● Let’s see some samples of:
  ○ Getting RK data to Firestore
  ○ Getting HealthKit data to Firestore
● We are sharing a web portal for you to use (ReactJS)
● More samples of:
  ○ Getting Files to Cloud Storage
  ○ Xcode 11, iOS 13, & CareKit
● Assignment 6 Overview

https://cs342.stanford.edu

cs342-aut1920.slack.com
Add Firebase to your iOS app
Initialize

1. Register app
   iOS bundle ID: edu.stanford.Project-OSMART
2. Download config file
3. Add Firebase SDK
4. Add initialization code
   To connect Firebase when your app starts up, add the initialization code below to your main AppDelegate class.
   - Swift
   - Objective-C

   ```swift
   import UIKit
   import Firebase

   @UIApplicationMain
   class AppDelegate: UIResponder, UIApplicationDelegate {
       var window: UIWindow?

       func application(_ application: UIApplication,
                       didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
           FirebaseApp.configure()
           return true
       }
   }
   ```

© 2016 Stanford Byers Center for Biodesign
Authentication (sign-in methods)
additional steps for iOS devices

we want you to focus on your apps
we will provide as much source code as we can
Authentication (sign-in methods)
additional steps for iOS devices

Authenticate with Firebase Using Email Link in iOS

You can use Firebase Authentication to sign in a user by sending them an email containing a link, which they can click to sign in. In the process, the user’s email address is also verified.

There are numerous benefits to signing in by email:

- Low friction sign-up and sign-in.
- Lower risk of password reuse across applications, which can undermine security of even well-selected passwords.
- The ability to authenticate a user while also verifying that the user is the legitimate owner of an email address.
- A user only needs an accessible email account to sign in. No ownership of a phone number or social media account is required.
- A user can sign in securely without the need to provide (or remember) a password, which can be cumbersome on a mobile device.
- An existing user who previously signed in with an email identifier (password or federated) can be upgraded to sign in with just the email. For example, a user who has forgotten their password can still sign in without needing to reset their password.

Before you begin

1. Add Firebase to your iOS project. Include the following pods in your Podfile:
Sample Study

everything we have covered so far in action

demo on our GitHub
use our resources to polish your apps and add additional functionality!

Branch with Xcode 11, iOS13, and CareKit 2.0 support:
https://github.com/cs342/CardinalKit-CS342-StudySample/tree/ios13
Building for Digital Health
Biodesign’s CS342/MED253 - Autumn 2019

CardinalKit Sample Study

https://github.com/cs342/CardinaKit-CS342-StudySample
Saving data to Firestore

Set a document

To create or overwrite a single document, use the `set()` method:

```swift
// Add a new document in collection "cities"
db.collection("cities").document("LA").setData(
    {
        "name": "Los Angeles",
        "state": "CA",
        "country": "USA"
    }
) { err in
    if let err = err {
        print("Error writing document: \(err)")
    } else {
        print("Document successfully written!")
    } }"}
```

ViewController.swift
Saving data to Firestore
Follow official documentation https://firebase.google.com/docs/firestore/manage-data/add-data

Set a document

To create or overwrite a single document, use the `set()` method:

```swift
// Add a new document in collection "cities"
db.collection("cities").document("LA").setData(
  "name": "Los Angeles",
  "state": "CA",
  "country": "USA"
) { err in
  if let err = err {
    print("Error writing document: \(err)"")
  } else {
    print("Document successfully written!"")
  }
}
```

ViewController.swift
Saving data to Firestore

Your collection name should have the following format (root-level):

/studies/{bundleId}/users/{loggedInUserId}/...  {whatever/you/want}

For surveys:
/studies/{bundleId}/users/{loggedInUserId}/surveys/{surveyUniqueID}

HealthKit data:
/studies/{bundleId}/users/{loggedInUserId}/healthKit/{healthTypeAndUniqueEntryId}
Saving data to Firestore
Follow official documentation https://firebase.google.com/docs/firestore/manage-data/add-data

GitHub Example: [link]
Accessing data to Firestore


Get a document

The following example shows how to retrieve the contents of a single document using `get()`:

```swift
let docRef = db.collection("cities").document("SF")

docRef.getDocument { (document, error) in
    if let document = document, document.exists {
        let dataDescription = document.data().map(String.init(describing:)) ?? "nil"
        print("Document data: \(dataDescription)"
    } else {
        print("Document does not exist")
    }
}
```

ViewController.swift
Saving data to Firestore
Follow official documentation https://firebase.google.com/docs/firestore/manage-data/add-data

Your collection name should have the following format (root-level):

/studies/{bundleId}/users/{loggedInUserId}/...  {whatever/you/want}

For surveys:
/studies/{bundleId}/users/{loggedInUserId}/surveys/{surveyUniqueID}

HealthKit data:
/studies/{bundleId}/users/{loggedInUserId}/healthKit/{healthTypeAndUniqueEntryId}
Backend Architecture

Firebase
Web SDK

Cloud Identity
Authentication

SDK & tools

and more!

Google BigQuery

Cloud Storage

Firestore
NoSQL Database
Sample Webapp

see some of your app data in action!
Building for Digital Health
Biodesign’s CS342/MED253 - Autumn 2019

CardinalKit ReactJS App (+Firebase/GCP)

https://github.com/cs342/CardinalKit-CS342-Web
Sample Webapp 📐
see some of your app data in action!

If a team member has experience with JavaScript (/ReactJS) and wants to modify the portal, we will award extra credit
Sample Webapp

see some of your app data in action!

If a team member has experience with JavaScript (ReactJS) and wants to modify the portal, we will award extra credit

- Show & download HealthKit data
Consent
visual templates with best practices and a transparent process

Surveys
collect feedback and patient-reported outcomes

Active Surveys
measure body activity using iPhone sensors
motor activities, fitness, cognition, speech, hearing, hand dexterity, and vision

More
login flows, passcode creation, charts, etc.
Active Tasks

Active tasks invite users to perform activities under partially controlled conditions while iPhone sensors are used to collect data. For example, an active task for analyzing gait and balance might ask the user to walk a short distance, while collecting accelerometer data on the device.

Predefined Active Tasks

ResearchKit™ includes a number of predefined tasks, which fall into seven categories: motor activities, fitness, cognition, speech, hearing, hand dexterity, and vision. Table 1 summarizes each task and describes the data it generates.

<table>
<thead>
<tr>
<th>CATEGORY</th>
<th>TASK</th>
<th>SENSOR</th>
<th>DATA COLLECTED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Activities</td>
<td>Range of Motion</td>
<td>Accelerometer Gyroscope</td>
<td>Device motion</td>
</tr>
<tr>
<td></td>
<td>Gait and Balance</td>
<td>Accelerometer Gyroscope</td>
<td>Device motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pedometer</td>
</tr>
<tr>
<td></td>
<td>Tapping Speed</td>
<td>Multi-Touch display</td>
<td>Touch activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accelerometer (optional)</td>
<td></td>
</tr>
<tr>
<td>Fitness</td>
<td>Fitness</td>
<td>GPS Gyroscope</td>
<td>Device motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pedometer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Heart rate</td>
</tr>
<tr>
<td></td>
<td>Timed Walk</td>
<td>GPS Gyroscope</td>
<td>Device motion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Pedometer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Location</td>
</tr>
<tr>
<td></td>
<td>Spatial Memory</td>
<td>Multi-Touch display</td>
<td>Touch activity</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Accelerometer (optional)</td>
<td>Correct answer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Actual sequences</td>
</tr>
</tbody>
</table>
Timed Walk

Your description goes here.
This activity measures your lower extremity function.

Assistive device

Use the same assistive device for each test.

Do you wear an ankle foot orthosis?

Yes
No

Do you use an assistive device?
Tap here to select an answer.

Timed Walk

Find a place, preferably outside, where you can walk for about 109 yd in a straight line as quickly as possible, but safely. Do not slow down until after you’ve passed the finish line.
Tap Next to begin.
Collect the Data

The data collected in active tasks is recorded in a hierarchy of ORKResult objects in memory. It is up to you to serialize this hierarchy for storage or transmission in a way that’s appropriate for your application.

For high sample rate data, such as from the accelerometer, use the ORKFileResult in the hierarchy. This object references a file in the output directory (specified by the outputDirectory property of ORKTaskViewController) where the data is logged.

The recommended approach for handling file-based output is to create a new directory per task and to remove it after you have processed the results of the task.

Active steps support attaching recorder configurations (ORKRecorderConfiguration). A recorder configuration defines a type of data that should be collected for the duration of the step from a sensor or a database on the device. For example:

- The pedometer sensor returns a CMPedometerData object that provides step counts computed by the motion coprocessor on supported devices.
- The accelerometer sensor returns a CMAccelerometerData object that provides raw accelerometer samples indicating the forces on the device.
- A CMDeviceMotion object provides information about the orientation and movement of the device by combining data collected from the accelerometer, gyroscope, and magnetometer.
- HealthKit returns sample types, such as heart rate.
- CoreLocation returns location data (combined from GPS, Wi-Fi and cell tower information).

The recorders used by ResearchKit’s predefined active tasks always use NSFileProtectionCompleteUnlessOpen while writing data to disk, and then change the file protection level on any files generated to NSFileProtectionComplete when recording is finished.

http://researchkit.org/docs/docs/ActiveTasks/ActiveTasks.html
Collect the Data

The data collected in active tasks is recorded in a hierarchy of ORKResult objects in memory. It is up to you to serialize this hierarchy for storage or transmission in a way that's appropriate for your application.

For high sample rate data, such as from the accelerometer, use the ORKFileResult in the hierarchy. This object references a file in the output directory (specified by the outputDirectory property of ORKTaskViewController) where the data is logged.

The recommended approach for handling file-based output is to create a new directory per task and to remove it after you have processed the results of the task.

Active steps support attaching recorder configurations (ORKRecorderConfiguration). A recorder configuration defines a type of data that should be collected for the duration of the step from a sensor or a database on the device. For example:

- The pedometer sensor returns a CMPedometerData object that provides step counts computed by the motion coprocessor on supported devices.
- The accelerometer sensor returns a CMAccelerometerData object that provides raw accelerometer samples indicating the forces on the device.
- A CMDdeviceMotion object provides information about the orientation and movement of the device by combining data collected from the accelerometer, gyroscope, and magnetometer.
- HealthKit returns sample types, such as heart rate.
- CoreLocation returns location data (combined from GPS, Wi-Fi and cell tower information).

The recorders used by ResearchKit's predefined active tasks always use NSFileProtectionCompleteUnlessOpen while writing data to disk, and then change the file protection level on any files generated to NSFileProtectionComplete when recording is finished.

http://researchkit.org/docs/docs/ActiveTasks/ActiveTasks.html
Storage
Store and retrieve user-generated files like images, audio, and video without server-side code
Get started

Learn more

- How do I get started?
  View the docs

- How does Storage work?
  View the docs

- What can Storage do for me?
  Learn more
Add Cloud Storage to your app

Ensure the following dependency is in your project’s Podfile:

```ruby
pod 'Firebase/Storage'
```

Run `pod install` and open the created `.xcworkspace` file.
For storage:
/studies/{bundleId}/users/{loggedInUserId}/storage/{whatever/you/want}/{fileName}
Sample Webapp

see some of your app data in action!

If a team member has experience with JavaScript (ReactJS) and wants to modify the portal, we will award extra credit:

- Show & download HealthKit data
- Download content from Storage
Assignment #6: Backend

Connect your apps to a GCP backend server 🍼

This is the last assignment of the course before your final presentations. We will be integrating your apps with the Google Cloud Platform (GCP), focusing on: authentication, data storage, and simple data visualization.

Due Dec 2nd (BEFORE final presentations)
Attendance

- Please be on time so we can start class promptly!
- [https://tinyurl.com/cs342-attendance](https://tinyurl.com/cs342-attendance)

[https://cs342.stanford.edu](https://cs342.stanford.edu)

[cs342-aut1920.slack.com](mailto:cs342-aut1920.slack.com)