Welcome!

https://cs342.stanford.edu

cs342-aut1920.slack.com
Overview for today

- Assignment #5 overview (HealthKit)
- More on the backend for your apps!
  - Using Google Cloud Platform
- Stanford University Research IT support
- Getting Started with your projects / resources

https://cs342.stanford.edu

cs342-aut1920.slack.com
Assignment #5: HealthKit

Adding health tracking capabilities to your clinical apps.

For this assignment, let’s put what we have learned about HealthKit into practice. Your deliverable is divided into three parts:

1. Enable background delivery for HealthKit activity types. This will add passive patient monitoring functionality to your study.
FHIR serves two technical roles in healthcare IT

FHIR as an **API specification**

FHIR as a **data model**
Patient: "Patient/example_patient"

Procedure: "Appendectomy (Procedure)"

Practitioner: "Dr Cecil Surgeon"

Reason Code: "Generalized abdominal pain 24 hours. Localized in RIF with rebound and guarding"

Follow-Up: "ROS 5 days - 2013-04-10"
Cloud Healthcare API

Cloud Healthcare API offers a robust, scalable infrastructure that integrates with various healthcare data sources using industry-standard protocols to bridge the gap to the Cloud. Currently, Cloud Healthcare API supports FHIR, HL7 v2 and DICOM.

Currently in Beta
Hands-on Lab
How to access the lab?

- [https://explore.qwiklabs.com/](https://explore.qwiklabs.com/)
- Create a New Account using your [@stanford.edu](mailto:@stanford.edu) email
- Login using your [@stanford.edu](mailto:@stanford.edu) email and password
- Select the Course: **CS 342: Building for Digital Health (MED 253)**
- Lab: **Introduction to Healthcare FHIR APIs**

**NOTE:** The lab will not be available after **Tuesday, 19th November**
What is a “backend”?

- database / persistence layer
- server-side logic - (scheduling, pre-processing, triggers)
- ETL - extract/transform/load piping
- authentication
  hardware / compute + networking

https://cs342.stanford.edu

cs342-aut1920.slack.com
Client-Server Model vs Serverless

TRADITIONAL vs SERVERLESS

TRADITIONAL

SERVERLESS (using client-side logic and third-party services)

https://www.gocd.org/2017/06/26/serverless-architecture-continuous-delivery/
TRADITIONAL

VASCTRAC
flow of network requests
March 8, 2017

Compatible Devices

- iPhone
  - Token Stored in Keychain
  - (1) RS256 Token Key
  - (2) Request with RS256 Token Key

- Computer
  - Authentication uses signed cookies
  - https://github.com/vinbase/session
  - (1) Signed Session Token
  - (2) Request with Signed Session Token

Device

Public Server

Username=Password

Web Server

vasctracdev

Public Server

vasctracdev

Database

SQL Server

Microsoft Azure

Golang

kubernetes
What is Protected Health Information (PHI)?

- ~18 identifiers

1. Names
2. All geographical identifiers smaller than a state, except for the initial three digits of a zip code if, according to the current publicly available data from the U.S. Bureau of the Census: the geographic unit formed by combining all zip codes with the same three initial digits contains more than 20,000 people; and the initial three digits of a zip code for all such geographic units containing 20,000 or fewer people is changed to 000
3. Dates (other than year) directly related to an individual
4. Phone Numbers
5. Fax numbers
6. Email addresses
7. Social Security numbers
8. Medical record numbers

9. Health insurance beneficiary numbers
10. Account numbers
11. Certificate/license numbers
12. Vehicle identifiers and serial numbers, including license plate numbers;
13. Device identifiers and serial numbers;
14. Web Uniform Resource Locators (URLs)
15. Internet Protocol (IP) address numbers
16. Biometric identifiers, including finger, retinal and voice prints
17. Full face photographic images and any comparable images
18. Any other unique identifying number, characteristic, or code except the unique code assigned by the investigator to code the data

https://cs342.stanford.edu

cs342-aut1920.slack.com
### Approved Services

This table indicates which classifications of data are allowed on a selection of commonly used Stanford University IT services.

<table>
<thead>
<tr>
<th>STANFORD SERVICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio and Video Conferencing: BlueJeans, WebEx, Zoom</td>
</tr>
<tr>
<td>Backups: CrashPlanProe</td>
</tr>
<tr>
<td>Calendar: Office 365</td>
</tr>
<tr>
<td>Cardinal Fax</td>
</tr>
<tr>
<td>Cardinal Print</td>
</tr>
<tr>
<td>Cloud Infrastructure: AWS</td>
</tr>
<tr>
<td>Cloud Infrastructure: Microsoft Azure, Google Cloud Platform</td>
</tr>
<tr>
<td>Content Management: Drupal (Stanford Sites), Wordpress</td>
</tr>
<tr>
<td>Database Hosting: MySQL</td>
</tr>
<tr>
<td>Document Management: Box and Office 365 OneDrive</td>
</tr>
<tr>
<td>Document Management: Google Drive and Google Shared drives</td>
</tr>
<tr>
<td>Document Management: Google G Suite: Docs, Sheets, Forms and Slides</td>
</tr>
<tr>
<td>Document Management: Google G Suite: All others (Sites, Photos, etc...)</td>
</tr>
</tbody>
</table>

**HIPAA-Compliant** + **BAA agreement in-place**

source:
https://uit.stanford.edu/guide/riskclassifications
GCP Available Services:
https://cloud.google.com/security/compliance/hipaa/
**What is an SDK?**

- **Software Development Kit**
- In our case, the Firebase SDK is a collection of tools (pre-packaged functions and methods) that let’s us easily code using Firebase.
- [https://firebase.google.com/docs/ios/setup](https://firebase.google.com/docs/ios/setup)
```swift
Auth.auth().sendSignInLink(toEmail: email, actionCodeSettings: actionCodeSettings) {
    (error) in
    if let error = error {
        print(error.localizedDescription)
        completion(false)
    } else {
        completion(true)
    }
}
```
SERVERLESS
(using client-side logic and third-party services)

db.collection("users").addDocument(data: {"eID": eid, "userID": user.uid, "lastActive": Date().ISOStringFromDate()})

db.collection(stanfordRITBucket + "/surveys").document(identifier).setData(surveyDataJson)
This document does not exist, it will not appear in queries or snapshots.
Serverless Architecture

• Because Managing Servers...
  • sucks
  • is expensive
  • is inefficient
  • a distraction
  • less secure
  • not necessary
Backend Architecture

- Firebase
  - SDK & tools
- Cloud Identity
  - Authentication
- NoSQL Database
  - Firestore
- and more!
  - Google BigQuery
  - Google Cloud Storage
HIPAA Compliance on Google Cloud Platform

This guide covers HIPAA compliance on Google Cloud Platform. HIPAA compliance for G Suite is covered separately.

Disclaimer

This guide is for informational purposes only. Google does not intend the information or recommendations in this guide to constitute legal advice. Each customer is responsible for independently evaluating its own particular use of the services as appropriate to support its legal compliance obligations.

Covered Products

The Google Cloud BAA covers GCP’s entire infrastructure (all regions, all zones, all network paths, all points of presence), and the following products:

- AI Platform Training and Prediction
- App Engine
- Cloud AI Notebooks
- Cloud Armor
- Cloud AutoML Natural Language
- Cloud AutoML Tables
- Cloud AutoML Translation
- Cloud AutoML Vision
- BigQuery
- BigQuery Data Transfer Service

Resource:
https://cloud.google.com/security/compliance/hipaa/
**TRADITIONAL**

![Diagram of TRADITIONAL network flow]

## VASCTRAC

**flow of network requests**  
March 8, 2017

---

**Compatible Devices**

- **iPhone**
  - Token Stored in Keychain
  - (1) RS256 Token Key
  - (2) Request with RS256 Token Key

- **Computer**
  - Authentication uses signed cookies
  - https://github.com/golang/sessions
  - (1) Signed Session Token
  - (2) Request with Signed Session Token

---

**Device**

- **Web Server**
  - Username+Password
  - (1) Web UI
  - (2) vasctracdev

- **Public Server**
  - (1) TID 1.3

---

**Database**

- **SQL Server**
  - Encrypted resources from authenticated user
  - (1) vasctracdev
  - (2) TID 1.3

---

**Microsoft Azure**

- **Golang**
  - kubernetes
S-SMART Study

- Firebase SDK & tools
- Cloud Identity Authentication
- Cloud Firestore NoSQL Database
- and more!
  - Google BigQuery
  - Google Cloud Storage

CARE-IT Study

- Firebase SDK & tools
- Cloud Identity Authentication
- Cloud Firestore NoSQL Database
- and more!
  - Google BigQuery
  - Google Cloud Storage
S-SMART Study

- Firebase SDK & tools
- Cloud Identity Authentication
- and more!
- Google BigQuery
- Google Cloud Storage
- Cloud Firestore
- NoSQL Database

CARE-IT Study

- Firebase SDK & tools
- Cloud Identity Authentication
- and more!
- Google BigQuery
- Google Cloud Storage
- Cloud Firestore
- NoSQL Database
Stanford UIT:

- Defines how services can communicate with each other, securely, so you don’t have to.
- Already understands your system, which means your studies get expedited DRA review.
- Helps you get started with premium GCP services at a low-cost — and helps you scale up when needed.
What is a DRA?

- Data Risk Assessment
- Collaboration between Information Security Office (ISO) and the University Privacy Office (UPO) - unsung heros
  - Usually involves a lawyer
  - Usually involves an engineer or other tech expertise
- Required by the IRB
  - More people requiring care and fewer people paying into system
- Thorough review of the data you collect and methods of storage and transfer
  - Data flow diagram
  - Form, documentation submission
  - Interview/Meeting(s)
- Takes 2 weeks to ∞
Create *one* GCP/Firebase Instance per team
this will be used for your own personal testing only

1. Go to [http://firebase.google.com](http://firebase.google.com)
2. Click “Go To Console” (top right)
3. “Add Project” named SSMART, or CARE-IT
4. Disable Google Analytics (not in BAA)
5. “Create Project” & wait 30 sec...
6. “Continue” and under “develop,” click on ”database” -> ”create”
7. Select “test mode” (RIT will manage rules)
8. Deploy to location “us-east-1” and click “done.”
Authentication (sign-in methods)
enable the email/password provider WITH passwordless sign-in
SERVERLESS
(using client-side logic and third-party services)

Firebase
SDK & tools

```swift
Auth.auth().sendSignInLink(toEmail: email, actionCodeSettings: actionCodeSettings) { (error) in
    if let error = error {
        print(error.localizedDescription)
        completionHandler(false)
    }
    completionHandler(true)
```
SERVERLESS
(using client-side logic and third-party services)

Firebase
SDK & tools

Cloud Identity
Authentication
Password-less Login (Google Identity)

1. Patient goes through onboarding & consent
2. Patient inputs their email address
3. Check your email to finish log-in
Password-less Login
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.

https://firebase.google.com/docs/auth/ios/email-link-auth
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

we will follow official Firebase documentation a couple slides later to implement this
https://firebase.google.com/docs/auth/ios/email-link-auth

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:
Link your app to this project
“Project Overview” -> “iOS”
Add Firebase to your iOS app
enter your iOS bundle ID

1. Register app
   - iOS bundle ID: com.company.appname
   - App nickname (optional): My iOS App
   - App Store ID (optional): 123456789
   - Register app

2. Download config file
3. Add Firebase SDK
4. Add initialization code
5. Read the Get Started Guide for iOS
Add Firebase to your iOS app
enter your iOS bundle ID

- Display Name: Project-SSMART
- Bundle Identifier: edu.stanford.Project-SSMART
- Team: Stanford University

Automatically manage signing
Provisioning Profile: Xcode Managed Profile
Signing Certificate: iPhone Developer: Santiago Gutierrez (FW5Z9F3...
Add Firebase to your iOS app
enter your iOS bundle ID and “register app”

1. Register app
   - iOS bundle ID: com.company.appname
   - App nickname (optional): My iOS App
   - App Store ID (optional): 123456789
     - Register app

2. Download config file
3. Add Firebase SDK
4. Add initialization code
5. Read the Get Started Guide for iOS
Add Firebase to your iOS app
Copy the .plist config file to your project (root)

- Add Firebase to your iOS app
  - Register app
    - iOS bundle ID: edu.stanford.Project-SSMART
  - Download config file

  Move the GoogleService-info.plist file you just downloaded into the root of your Xcode project and add it to all targets.
Stanford UIT:

- Defines how services can communicate with each other, securely, so you don’t have to.
- Already understands your system, which means your studies get expedited DRA review.
- Helps you get started with premium GCP services at a low-cost — and helps you scale up when needed.
Add Firebase to your iOS app
Using CocoaPods (package manager)

CocoaPods is built with Ruby and is installable with the default Ruby available on macOS. We recommend you use the default ruby.

Using the default Ruby install can require you to use `sudo` when installing gems. Further installation instructions are in the guides.

```
$ sudo gem install cocoapods
```

We also have a Mac app for CocoaPods. It only gets major releases ATM though.

https://cocoapods.org/
Add Firebase to your iOS app
Using CocoaPods (package manager)

Register app
iOS bundle ID: edu.stanford.Project-SMART

Download config file

Add Firebase SDK

Google services use CocoaPods to install and manage dependencies. Open a terminal window and navigate to the location of the Xcode project for your app.

Create a Podfile if you don’t have one:

```
$ pod init
```

Open your Podfile and add:

```
# add pods for desired Firebase products
# https://firebase.google.com/docs/ios/setup#available-pods
```

Save the file and run:

```
$ pod install
```

This creates an .xcworkspace file for your app. Use this file for all future development on your application.

Add initialization code

Read the Get Started Guide for iOS
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
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
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!
Building for Digital Health
Biodesign’s CS342/MED253 - Autumn 2019

CardinalKit Sample Study

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

Set a document

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

```
// 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  📧
Accessing data to Firestore
Follow official documentation https://firebase.google.com/docs/firestore/manage-data/add-data

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")
    }
}
```
Authentication (sign-in methods)
additional steps for iOS devices

Additional setup for email links
https://firebase.google.com/docs/auth/ios/email-link-auth

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:
Backend Architecture

- Firebase SDK & tools
- Cloud Identity Authentication
- Google BigQuery
- Google Cloud Storage
- Firestore
- NoSQL Database

and more!
Our last guest lecture of the quarter!
Upcoming Tuesday (Nov 12)

Jonathan Wilt
Chief Technology Officer
innovationOchsner

Jonathan is the Chief Technology Officer of innovationOchsner, an innovation lab and accelerator founded by Ochsner Health System in 2015 to reimagine and revolutionize the experience and delivery of healthcare in a way that dramatically enhances quality and access, decreases cost, and improves patient satisfaction and engagement and caregiver efficiency. He leads the product development team and is responsible for overall technology strategy, analytics and data integration. Wilt started his career as a technical engineer at Epic Systems where he acted as a technical manager and legacy data conversion advisor for Epic implementations. He later founded a boutique software consulting company that focused on EMR optimization, data integration, and maintainability. His company started working with Ochsner Health System in 2011, and he joined Ochsner full time in 2014 as the AVP for the newly created Ochsner Center for Innovation. He holds a B.S. in Physics from the University of Notre Dame.
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](cs342-aut1920.slack.com)