CS342/MED253 Building for Digital Health
Lecture 6b: Coding with HealthKit
Overview for today

- In-class coding session.
  - Fetch our starter code for today and practice coding five different types of queries from those that we showed last week
- Assignment #5

https://cs342.stanford.edu

cs342-aut1920.slack.com
S-SMART HK Data Types

These are the data elements that app should be collecting and storing:

**Activity**

```swift
static let stepCount: HKQuantityTypeIdentifier
A quantity sample type that measures the number of steps the user has taken.

static let distanceWalkingRunning: HKQuantityTypeIdentifier
A quantity sample type that measures the distance the user has moved by
walking or running.

static let distanceCycling: HKQuantityTypeIdentifier
A quantity sample type that measures the distance the user has moved by
cycling.

static let pushCount: HKQuantityTypeIdentifier
A quantity sample type that measures the number of pushes that the user has
performed while using a wheelchair.

static let distanceWheelchair: HKQuantityTypeIdentifier
A quantity sample type that measures the distance the user has moved using a
wheelchair.

static let swimmingStrokeCount: HKQuantityTypeIdentifier
A quantity sample type that measures the number of strokes performed while
swimming.
```
In-class Code

Querying Patient Health Data

- Running Queries
  - Sample Query
  - Stats Query
  - Activity Query
  - Health Records
- Background Query
- Looking at Graphs
let hkTypesToRead: Set<HKObjectType> = [
    HKObjectType.quantityType(forIdentifier: .stepCount),
    HKObjectType.quantityType(forIdentifier: .distanceWalkingRunning),
    HKObjectType.quantityType(forIdentifier: .flightsClimbed),
    HKObjectType.quantityType(forIdentifier: .heartRate),
    HKObjectType.characteristicType(forIdentifier: .bloodType)
]

var hkTypesToWrite: Set<HKSAMPLEType> = [
    HKObjectType.quantityType(forIdentifier: .stepCount),
    HKObjectType.quantityType(forIdentifier: .distanceWalkingRunning),
    HKObjectType.quantityType(forIdentifier: .flightsClimbed),
    HKObjectType.quantityType(forIdentifier: .heartRate)
]

func getHealthAuthorization(completion: @escaping (_ success: Bool, _ error: Error?) -> Void) {
    guard HKHealthStore.isHealthDataAvailable() else {
        completion(false, HealthKitError.notAvailable)
        return
    }

    let healthStore = HKHealthStore()
    healthStore.requestAuthorization(toShare: hkTypesToWrite, read: hkTypesToRead) {
        (success, error) in
        completion(success, error)
    }
}
Sample Query
General-purpose. You will be using these most!

Anchored Query
Only return new/modified elements.

Statistics Query
Quickly find the sum, min, max, or avg of a data set.

Activity Query
Move, exercise, and stand data. Three rings.

Document Query
Useful to find health records!

More Information
Observer Query
Notifies when samples change

```swift
func observerQuery() {
    let sampleType = HKObjectType.quantityType(forIdentifier: .stepCount)!

    let query = HKObserverQuery(sampleType: sampleType, predicate: nil) {
        query, completionHandler, error in

        if error != nil {
            // Perform error handling.
            fatalError("unable to run query \(error!.localizedDescription)"")
        }

        // Take whatever steps are necessary to update your app's data and UI
        // This may involve executing other queries
        //self.updateDailyStepCount()

        // If you have subscribed for background updates you must call the completion handler here.
        completionHandler()
    }

    let store = HKHealthStore()
    store.execute(query)
}
```
In-class Code

Querying Patient Health Data

- Running Queries
  - Sample Query
  - Stats Query
  - Activity Query
  - Health Records
- Background Query
- Looking at Graphs
Starter Code on GitHub
https://github.com/cs342

https://github.com/cs342/Lecture6B-HealthKit-Starter
Code a Sample Query with HealthKit
Fake Data 🖥️ [live] quick tutorial
let hkTypesToRead: Set<HKObjectType> = [
    HKObjectType.quantityType(forIdentifier: .stepCount),
    HKObjectType.quantityType(forIdentifier: .distanceWalkingRunning),
    HKObjectType.quantityType(forIdentifier: .flightsClimbed)
]

func query(quantityType: HKQuantityTypeIdentifier, completion: @escaping ([HKQuantitySample], Error?) -> Void) {

    let predicate = HKQuery.predicateForSamples(withStart: Date(), yesterday, end: Date(), options: .strictStartDate)
    let sortDescriptor = NSSortDescriptor(key: HKSampleSortIdentifierStartDate, ascending: false)
    let sampleType = HKObjectType.quantityType(forIdentifier: quantityType)!

    let query = HKSampleQuery(sampleType: sampleType, predicate: predicate, limit: 100, sortDescriptors: [sortDescriptor])
    query.execute { (query, samples, error) in
        guard let samples = samples as? [HKQuantitySample] else {
            completion([HKQuantitySample](), error)
            return
        }
        completion(samples, error)
    }

    let healthStore = HKHealthStore()
    healthStore.execute(query)
}

don't forget to request permissions for the type(s) that you are querying
2.

Code a Stats Query with HealthKit
func totalSteps(completion: @escaping (Double?, Error?) -> (Void)) {
    let calendar = NSCalendar.current
    let now = Date()
    let components = calendar.dateComponents([.day, .month, .year], from: now)
    let startDate = calendar.date(from: components)!
    let endDate = calendar.date(byAdding: .day, value: 1, to: startDate)
    let sampleType = HKQuantityType.quantityType(forIdentifier: .stepCount)!
    let predicate = HKQuery.predicateForSamples(withStart: startDate, end: endDate, options: .strictStartDate)
    let query = HKStatisticsQuery(quantityType: sampleType, quantitySamplePredicate: predicate, options: .cumulativeSum) {
        query, result, error in
        if error != nil {
            completion(nil, error)
            return
        }
        var total = 0.0
        if let quantity = result?.sumQuantity() {
            let unit = HKUnit.count()
            total = quantity.doubleValue(for: unit)
        }
        completion(total, error)
    }
    healthStore = HKHealthStore()
    healthStore.execute(query)
Code an Activity Query with HealthKit
func activityRings() {

// Create the predicate for the query
let summariesWithinRange = HKQuery.predicate(forActivitySummariesBetweenStart: startDate, end: endDate)

// Build the query
let query = HKActivitySummaryQuery(predicate: summariesWithinRange) { (query, summaries, error) -> Void in
    guard let activitySummaries = summaries else {
        guard let queryError = error else {
            fatalError("*** Did not return a valid error object. ***")
        }

        // Handle the error here...
        return
    }

    // Do something with the summaries here...
    //activeEnergyBurned, appleExerciseTime, appleStandHours, activeEnergyBurnedGoal, appleExerciseTimeGoal, appleStandHoursGoal: HKQuantity
}

// Run the query
let healthStore = HKHealthStore()
healthStore.execute(query)
Retrieve Health Records with HealthKit
Fake Data [live] quick tutorial
Retrieve Health Records with HealthKit

```swift
let hkTypesToRead: Set<HKObjectType> = [
    HKObjectType.clinicalType(forIdentifier: .allergyRecord)!,
    HKObjectType.clinicalType(forIdentifier: .immunizationRecord)!
]

func healthRecordAllergies() {
    guard let allergyType = HKObjectType.clinicalType(forIdentifier: .allergyRecord) else {
        fatalError("*** Unable to create the allergy type ***")
    }

    let allergyQuery = HKSampleQuery(sampleType: allergyType, predicate: nil, limit: HKObjectQueryNoLimit, sortDescriptors: nil) { (query, samples, error) in
        guard let actualSamples = samples else {
            // Handle the error here.
            print("*** An error occurred: \
                 (error?.localizedDescription ?? "nil") ***")
            return
        }

        let allergySamples = actualSamples as? [HKClinicalRecord]
        // Do something with the allergy samples here...
    }

    let store = HKHealthStore()
    store.execute(allergyQuery)
}
Run a background query
Run a background query

```swift
func observerQuery() {
    let sampleType = HKObjectType.quantityType(forIdentifier: .stepCount)!

    let query = HKObserverQuery(sampleType: sampleType, predicate: nil) {
        query, completionHandler, error in

        if error != nil {
            // Perform error handling.
            fatalError("unable to run query \(error!.localizedDescription)"")
        }

        // Take whatever steps are necessary to update your app's data and UI
        // This may involve executing other queries
        //self.updateDailyStepCount()

        // If you have subscribed for background updates you must call the completion handler here.
        completionHandler()
    }

    let store = HKHealthStore()
    store.execute(query)
}
```
modify the line graph view
In-class Code

Querying Patient Health Data

- Running Queries
  - Sample Query
  - Stats Query
  - Activity Query
  - Health Records
- Background Query
- Looking at Graphs
Attendance

- [https://tinyurl.com/cs342-oct31-attendance](https://tinyurl.com/cs342-oct31-attendance)

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

[cs342-aut1920.slack.com](https://cs342-aut1920.slack.com)