Welcome to HealthKit Health Tracker Tutorial!

By Wesley Harrison and Jerod Muilenburg

HealthKit in SwiftUI

HealthKit is Apple's framework for managing health data securely on iOS devices (HIPPA Compliant). It allows apps to read, write, and display health-related data like heart rate, step count, and calories burned. This tutorial will focus on understanding the **HealthKit API**—you'll learn how to request authorization, fetch health data, and handle key HealthKit concepts.

Getting Started

To follow this tutorial, ensure you have the following:

- **Device**: Use a real iPhone to test (HealthKit has limited functionality on simulators).
- **XCode**, and **SwiftUI** knowledge.

Setup

1. **Enable HealthKit**:
   Go to your project's **Signing & Capabilities** tab. Add the **HealthKit** capability.
2. **Permissions**:
   - Open your `Info.plist` file and add:

```
<key>NSHealthShareUsageDescription</key>
<string>We need access to your health data to display insights.
</string>
```

HealthKit Basics

1. Request Authorization

To access HealthKit data, your app must first request authorization from the user.

```
import HealthKit

class HealthKitManager {
    private var healthStore = HKHealthStore()
```

```
    func requestAuthorization(completion: @escaping (Bool) -> Void) {
        // Figure out the data types you want to read into your app here,
and define them.
        let typesToRead: Set<HKObjectType> = [
            HKObjectType.quantityType(forIdentifier: .heartRate)!,
            HKObjectType.quantityType(forIdentifier: .stepCount)!,
            HKObjectType.quantityType(forIdentifier: .activeEnergyBurned)!
        ]

        // Request authorization from iOS to access Health data.
        healthStore.requestAuthorization(toShare: nil, read: typesToRead) {
success, error in
            DispatchQueue.main.async {
                if let error = error {
                    print("Authorization error: \
(error.localizedDescription)")
                }
                completion(success)
            }
        }
    }
}
```

2. Fetching Data

Once authorized, you can query HealthKit for specific data types (e.g., heart rate or step count).

```
func fetchData(for sampleType: HKSampleType, completion: @escaping
([HKQuantitySample]?) -> Void) {
    let query = HKSampleQuery(
        sampleType: sampleType,
        predicate: nil,
        limit: HKObjectQueryNoLimit,
        sortDescriptors: [NSSortDescriptor(key:
HKSampleSortIdentifierStartDate, ascending: false)]
    ) { query, samples, error in
        DispatchQueue.main.async {
            if let error = error {
                print("Error fetching data: \(error.localizedDescription)")
                completion(nil)
                return
            }
            completion(samples as? [HKQuantitySample])
        }
    }
```

```
    healthStore.execute(query)
}
```

You can use this method to fetch various data types. For example:

Heart Rate:

```
let heartRateType = HKObjectType.quantityType(forIdentifier: .heartRate)!
fetchData(for: heartRateType) { data in
    if let samples = data {
        for sample in samples {
            let value = sample.quantity.doubleValue(for:
.count().unitDivided(by: .minute()))
            print("Heart Rate: \(value) bpm")
        }
    }
}
```

Step Count:

```
let stepCountType = HKObjectType.quantityType(forIdentifier: .stepCount)!
fetchData(for: stepCountType) { data in
    if let samples = data {
        for sample in samples {
            let value = sample.quantity.doubleValue(for: .count())
            print("Steps: \(value)")
        }
    }
}
```

3. Handling Errors and Permissions

Always check if HealthKit is available before making API calls.

```
if !HKHealthStore.isHealthDataAvailable() {
    print("HealthKit is not available on this device.")
}
```

If authorization is denied, prompt the user to enable it in Settings:

```
if !success {
    print("User did not grant permission. Ask them to enable HealthKit in
```

```
Settings.")
}
```

Summary

This tutorial introduced the basics of integrating HealthKit into your SwiftUI app. Main points...

Authorization: Always request permission for specific data types.
Data Fetching: Use `HKSampleQuery` to retrieve health data.
Error Handling: Check device compatibility and permissions.

For more features or observing live updates, check the HealthKit documentation.

Feel free to refer to the provided project files for more details!