Alex Banning

October 20, 2024

HW2 Document

**Various Approaches to Storage Management on iOS**

There are several different approaches to storage on iOS. For small user data like settings, profile information, etc., the most common method is to use UserDefaults. This is a built-in framework for small data storage (mostly text and Boolean variables). If the app requires bigger storage, there are a number of directories available for use through the FileManager Apple API. The FileManager contains many subfolders that are obfuscated for security purposes, but they can be accessed through:

FileManager.default.urls(for: “Directory”, in: .userDomainMask),

where “Directory” is one of the default categories (like caches, documentDirectory, photosDirectory, etc.

All of these directory options are containerized within the app, so if the user deletes the app, then this data disappears as well, and Apple provides guides to how developers should use these directories when creating an iOS app.

Another option for storage is SwiftData, which is a new framework that builds on Apple’s CoreData but is specific to Swift. This is a great option for managing large or structured data with much less code than FileManager. UserDefaults simply cannot handle the large data that SwiftData can. SwiftData also automatically manages the persistence of the app state in the background, while FileManager does not and UserDefaults requires slightly more work to handle edge cases and other setbacks.

Finally, third party storage like cloud services or a server could be used to manage the data in an iOS app, either through HTTP requests or other means. These services can offer even more computing power than SwiftData and FileManager but often cost the developer significantly to pay for such services, especially as the app grows in scale.

**Pros and Cons of Each Storage Method for MeetMeHalfway**

|  |  |  |
| --- | --- | --- |
| **Type of Storage** | **Pros** | **Cons** |
| UserDefaults | * Simple and effective for small data * Lots of tutorials available for help * Keeps track of states in the app (if the user closes the app, the state remains) * Easy to use (iOS handles most of the bulk code) * Fast access * Could be used for default addresses in my app | * Only small data can be saved here * Limited amount of storage available in this category * Can’t manage relationships between data * Difficult to come up with key-value pairs for the data I need to use in this app (besides basic information like name/address) |
| FileManager | * Full control over where and how data is saved and loaded * Flexible data storage (raw images, JSON blobs, raw text, custom formats) * Larger data storage * Can store as much data as you want (limited by physical storage on device) * Could be used for storing entire “trips” in my app | * Manual management – lots of code to write * No built-in structures, developer has to create * Doesn’t support relationships between different data * No automatic syncing * Lots of code to write and edge cases to think about |
| SwiftData | * Designed for structured data and more complexity * Integrated with SwiftUI * Automatic data persistence * Querying support * Automatically managed schema in the background * Could make app state persistence really easy to manage (don’t want to restart the search for a meeting point and nearby results) | * Steep learning curve for beginners * Not very intuitive * Must watch tons of tutorials to understand * Probably don’t need all the capabilities of this service in my app * Probably overkill |
| Third Party Services | * Lots of services available online that could make building my app easier * Quick querying and API requests through cloud services that are pre-built and tested * Very powerful scaling tool when user base increases | * Costs money * Difficult barrier to entry with limited experience * Overkill for my needs * More reliance on outside services that could fail or go down without my control (although unlikely, I want my app to be self-contained) |