Vitality Onboarding Assessment

Clean Architecture in iOS

Introduction

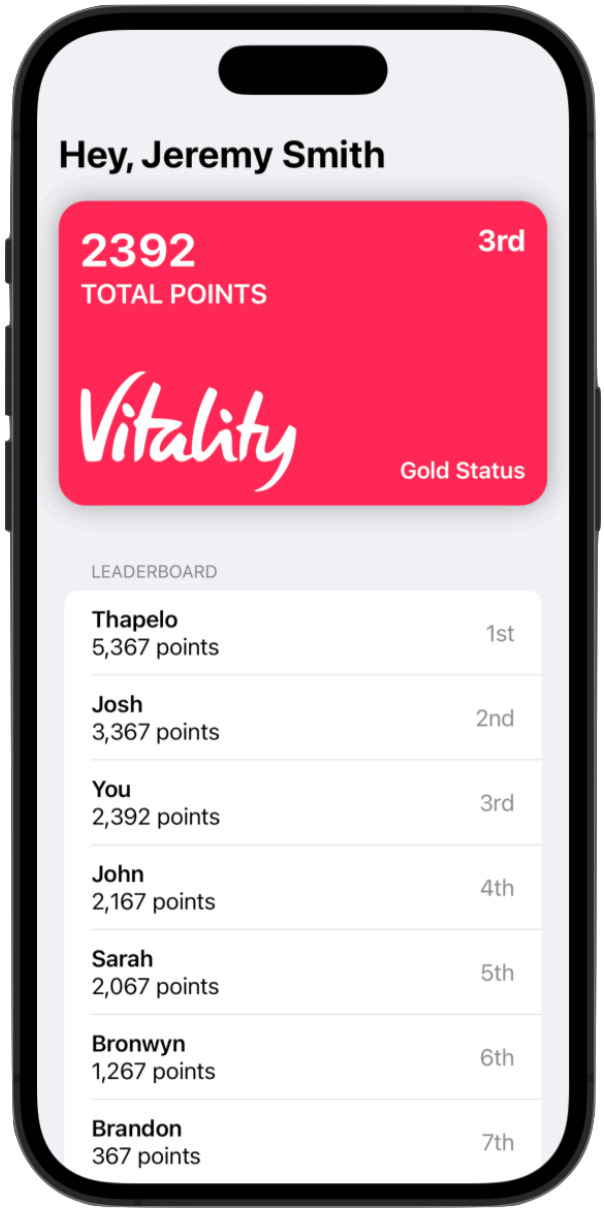
The following technical onboarding assessment will test your knowledge and touch on a few important areas of our app's architecture and certain practices we follow.

The iOS member app makes use of Clean Architecture and is predominantly written in Swift. The app makes use of UIKit especially in Legacy components and features, however we are steadily moving everything over to SwiftUI with our own modular framework, therefore this assessment will focus heavily on SwiftUI as the base UI framework to use.

This document will outline what is expected of you as far as implementation details when completing the assessment. While the main goal is to create a screen mirroring the provided screenshot below, it is important to follow the guidelines outlined in this document in order to adhere to our architectural practices.

What to build

The following screenshot will serve as a guideline as to what's expected from a UI perspective when completing this assessment. The app will need to fetch a points statement from a mocked API, persist this points statement through a gateway and display the data through necessary view models. As mentioned above, while the accuracy of the UI is important, the underlying implementation of how you achieved the final design will be more relevant.



What we give you

API

The main task of this assessment is to build a simple screen that fetches points data from a mock API we provide you. The Mock API simply returns a JSON object with all the data you'll need from a JSON file named **mockPointsStatement.json**. Familiarise yourself with the structure of the file as this will be returned verbatim from the API we provide you in the Networking folder.

Persistence (Realm)

We have included Realm as a dependency in your project, this will serve as the main database layer for your project. We have also provided empty Realm database model structs for you to use to store your data, feel free to add any fields you feel are necessary as well as completely change the models if you see fit.

We have also included a file named **RealmAdapter.swift** which acts as your main interface between Realm, please use this interface for all database transactions and make sure to inject it where needed, not manually instantiate it (see Dependency Injection for more info)

Dependency Injection (Swinject)

Dependency Injection is another crucial part of our app, it will be required that all objects used in view models and/or views should be injected and not manually instantiated. Swinject is included in your project for use as your main DI framework. We have also included a file named **AppDependencies.swift** which will be the main DI container for your app, all dependencies should be bound in here.

Unit Tests (XCTest)

We have provided an empty suite for your unit tests. Your unit tests should test your view model logic in terms of sorting the leaderboard as well as certain string manipulation for components such as the user greeting at the top. Ideally your test coverage should be 75%+.

Key Outcomes

There is no "correct way" to complete this assessment, however there are a few crucial points we will be looking at when reviewing your assessment. They include, but are not limited to the following:

- ★ Does the UI match the provided screenshot and is the UI created in SwiftUI?
- ★ Is the UI accessible?
- ★ Have you used reusable components to build your entire view?
- ★ Are you making use of a Gateway to persist the response you receive from the mocked API?
- ★ Are you using Dependency Injection?
- ★ Are you using initialiser-based injection?
- ★ Are you using view models where necessary?
- ★ Are you making use of domain-level objects or are you using database models directly?
- ★ Is your app correctly sorting the data in the leaderboard?
- ★ Does your project contain unit tests?

Documentation and Help

The following resources may assist you in completing the assessment

Clean Architecture
https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Swinject
https://github.com/Swinject/Swinject

Realm
https://github.com/realm/realm-swift