Take-Home Project

In this take-home assignment, you will contribute to a starter project with two major goals:

- Read all contacts from the system AddressBook and display them in a scrolling list.
- For each contact, compute and display the distance between the contact's address (if any) and the location of the user's iPhone.

Starter Project

This project provides a good starting point for you to work from. It is broken into two targets:

- `HatchContacts` (a Swift package)
  - `ContactManager.swift` - A basic implementation for interacting with the system AddressBook.
  - `Contact.swift` - Provides a `Contact` struct.
  - `ContactImporter.swift` - Loads fake contacts from a `json` file and writes them to the system AddressBook.
  - `LocationManager.swift` - Used to acquire the user's coordinates.
- An Xcode multiplatform iOS app target
  - This target includes several SwiftUI files that make use of `HatchContacts`.
  - `PermissionsView.swift` - This view prompts the user for permission to access `Contacts` and `CoreLocation`.
    - After access is granted, the user's location will be acquired, and PermissionsView.swift will be dismissed.
  - `ContactsView.swift` - A `SwiftUI.List` which displays a single hardcoded contact to serve as an example.
    - Provides a navigation bar button (for iOS Simulators only) which makes use of `ContactImporter` as described above.

Functional Requirements

The code contains two `#warning("TODO: ...)` statements:

1. In `ContactManager.swift`, implement `func loadContacts()` to fetch all contacts.
   - Use Apple's `Contacts` framework to fetch `[CNContact]`.
   - Map each `CNContact` to a `Contact` then display them in a scrolling list.
   - Ensure that the user doesn't have to wait a long time while all the contacts load since they won't all be displayed until the list is scrolled.
2. Implement `protocol DistanceComputable` to compute the distance between the contact's postal address and the device's location.
   - This should be done for each contact where `contact.postalAddress != nil`:
   - `LocationManager` is already configured to fetch the user's location during app start up.
   - Display the distance for each contact.
   - The choice of distance units (meters, kilometers, miles, etc.) is up to you.
   - Consider UI performance, error handling, and other relevant aspects.

Other considerations:

- The user should have some indication in the UI that data is loading or is not available yet.
- Consider ways to improve architecture and testablity. Showcase your knowledge of Protocol Oriented Programming.
- Feel free to change or refactor any part of the code in this project as long as you implement the two requirements mentioned above.
- Although the code provided is in SwiftUI, you are not limited to using SwiftUI exclusively.

Scope limitation

We may discuss these topics in follow-up conversations, but do not spend time on:

- Different layouts per platform (iPhone, iPad, Mac should all use the same layout)
- Handling orientation changes.
- Localization / globalization.
- Permission changes: There is no need to worry about `revoked` or `denied` permission cases.

Documentation

- https://developer.apple.com/documentation/contacts

- https://developer.apple.com/documentation/corelocation/converting_between_coordinates_and_user-friendly_place_names
- https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/LocationAwarenessPG/CoreLocation/CoreLocation.html

Evaluation Requirements

- Please send us a zip file of your completed project. We will be evaluating your code, so make sure you structure your modular components logically and explain interesting bits of code with comments.
- While this code is meant to be evaluated, we do not expect production-quality code. Focus on the key areas.
- We value your time. This should take around 1 hour, but please do not spend more than 2. If it's taking longer, just send us what you have.
- Code should be written using modern Swift and modern layout techniques (such as auto-layout, size classes, respecting the safe area, etc). Use UIKit or SwiftUI.
- Do not use any 3rd party code, rely only on official Apple frameworks and your own original code. (You can search for information online, but ensure that you fully understand all the code. Do not just copy and paste stuff). We may discuss your project in a follow-up session.
- Be ready to explain how your code works, why you made various (technical) design decisions, and how various areas of the code or functionality can be improved or extended.