Icare

Zeping He

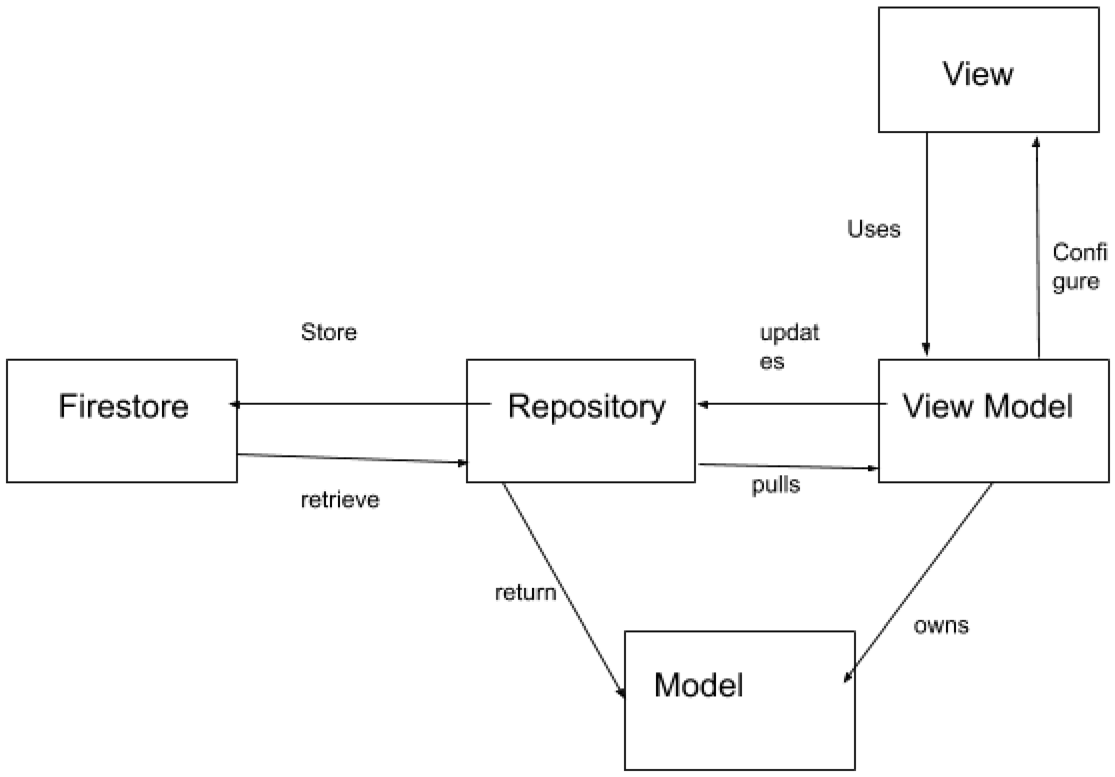
This project’s idea is for tracking and sharing the self-cure experience of a certain disease.

The idea is from the increasing trend of the global pandemic COVID 19. On the Internet, the self-cure experience topics are very attractive, interesting and in some way, it can represent a fight between the virus and individuals and give others faith. However, traditional ways of blogging (ex: making posts on social media platforms) can not express the self-cure experience well, the reason is: un-standardized format of the blog, flipped weight on text and data and less mobility. Un-standardized format means the writing style of blogs have so much difference, some bloggers try to emphasize their feelings while others might want to express their timelines. Flipped weight on text and data represent the main focus point is the data itself, rather than text description. The mobility simply stands for how easy that a user can track and share its data to others.Those three points are the main driving force of the development of this app.

Icare, an swiftUI based IOS app, tracks your cure experience and makes it a standard datapoint to make other users easy to see. It has four features as of the 5/9 --- Use a form to record user’s data (including temp, pain level, symptom and date), a notification system allows users to schedule next input time, a sharing system to handle the sharing and editing demand of users and finally a smart Authorization system make both Anonymous and Signed in with third party possible.

Architecture Description:

This project’s architecture attached below:



Record and Profile Models:

The record and profile struct as both codeable and identifiable structs, the reason for doing that is it’s easier to encode and decode the data to the backend database without explicitly encoding the process. Also, each record and each user profile unique and identifiable by firebase, that requires us to add an unique field to the type, which is ‘id’, we also need to add ‘@DocumentID’ to make that id is used by firebase data collection.

Firestore storage:

The storage is a cloud firestore, it supports the real time updates and a no-sql query. In this project, it handles any request from the repository and pushes them to the cloud server.

Signed-In options:

I think it is unfair to set a login wall before actual app content, if users has no prior knowledge about the actual app, the best way of convince them to login in is have a complete feature experience of the app and tell them if they want to switch device, we can sync your data on that device. So the answer would be use the anonymous signIn then sync the data. Luckily, the handling process isn't much complicated since firebase provides a large variety of sign in options.

Repository:
In this project, the repository serves as the interface of communicating between the view model and firestore, I designed the repository as an ObservableObject class, it is able of several operations including load, update and remove functions. It also needs to be observable because I used all the repositories as Environment variables, which enable them to be accessed along all views and updates in realtime.

ViewModel:
This is the core part of this app, basically each app instance has a global RecordViewModel and sharedData ViewModel. The ViewModel hands the communication task between the repository and Views. Each ViewModel is an Observable Object, with @Published tag attached to the repository and array of lower-leveled viewModels or Views. It will map the data from repository to lower view Cells

View:
The element is constructed by SWIFTUI. I choose swiftui because the communication of data becomes much much convenient than UIKit. It uses @State and @Binding to communicate between each front and back view. Thanks to the Published protocol, every update can reflect and be updated in real time. Also, because it uses SwiftUI, the way of stacking different Views becomes much easier too, we can customize different Navigation View, Navigation bar item just by placing dot (.) under their code.

Line-chart:
I fork and customized an open source data representation [1], This was to use Geometry to get a relative location on the frame. I use this line-chart as a separate view.

Icons:
This project's icon are all from systemIcon available across all apple platform

Difficulties of the project:

SwiftUI, building a swiftUI app from scratch.
I overcame this by searching every possible resources.

Mapping data from firebase to repository and models:
I visit Firebased youtube channel, go through a semi-live stream of building a app using combine and firebase.