GoodOnes Thought Process

For the introduction screen, I was thinking of having hovering text over the main view with the photos explaining the swiping using the ephread/Instructions CocoaPod, but the prompt specified a separate view. Therefore, I went with a sheet presented at the beginning in the style of an Apple welcome screen. I feel this elegantly does the job and allows for the photos to load in time before the user gets to the main screen.

For the photo presentation, I went for a hovering card view in the middle of the view with a rotatable transition for swiping. I thought this would allow for the nicest animation for a user to have when dismissing or favoriting. As a user swipes a photo right to favorite, a green rectangle with the text “Favorite” pops up on the card showing what is happening to the card. As a user swipes a photo left to dismiss, a red rectangle with the text “Dismiss” pops up on the card.

The cards are also nicely stacked on each other – only 4 at a time with some y axis offset – to indicate that there are more photos to select. I decided to only fetch 30 photos at once – hardcoding the n value since the prompt does not specify leaving that number to the user. If needed, I could use a number picker or input view to get the number desired by the user on the welcome screen before fetching.

I use gesture state of the photo card to tell whether a card is being dismissed or favorited and use that information to perform updates to the view itself. I was thinking of adding a hue of green and red for favoriting and dismissing respectively on the sides as the user swipes the cards, but I focused my time more on trying to get the Google Photo connection working.

In order to make sure that the app remembers what photos have been processed, I have kept two UserDefault variables’ endDate and begDate. I keep track the date of the last photo to be swiped on and check to see if it is earlier than the begDate or later than the endDate. If either, then update the corresponding value. The PHFetchOptions use these variables to check what photos to fetch. Then, I only fetch the ones outside of this range.

Once all the photos are swiped on, a confetti animation pops up with text signifying that all the photos of that session have been dealt with.

I have tried to use Deivitaka’s GPhotos Github as a CocoaPod or as a Package Dependency, but neither were working. Therefore, I added his files manually to my project in hopes of the structures and classes becoming accessible to me, which they finally did. I also had to configure the app to Firebase in order to be able to use the GCP API. I was able to ask the user to sign in to GPhotos, but for some reason, I am having issues with getting the photos after authorizing access. It kept giving an authentication error even though I had a token that was printing out perfectly fine.

I was going to start using AlamoFire to send the API calls directly, but I had to stop for two days during the week because my family and I went somewhere secluded to celebrate my parent’s anniversary, so I haven’t had the necessary time to try that.

I am experienced with using APIs in mobile apps and can make this work with more time, but I had to take a break in the middle. Also, the project is much smaller without the Firebase Cocoapod – which I added for the firebase connection to Google Photos API, so part 1 is very efficient while part 2 is filled with attempts of mine. The project was a blast to work on and hopefully I can do/make more for your team!