GitHub Followers

App Premise

- The user can enter a GitHub username and retrieve a list of that username's followers.
- The user can search within these followers for a specific follower.
- The user is able to tap on a follower from that list to get more information about that follower.
- The user is able to save favorite username searches so they don't have to type them every time. This is persisted between app launches.

Design

- 100% programmatic UI - No Storyboard
- AutoLayout using NSLayoutConstraints
- No 3rd Party Libraries
- Works with Dark Mode

API

Used the GitHub API

- Followers endpoint - https://developer.github.com/v3/users/followers/
- User info endpoint - https://developer.github.com/v3/users/

Approach

UIKit Components

- TabBarController
- NavigationController
- UICollectionView with Diffable Data Source
- Composition with Child View Controllers
- UIStackView
- SFSymbols

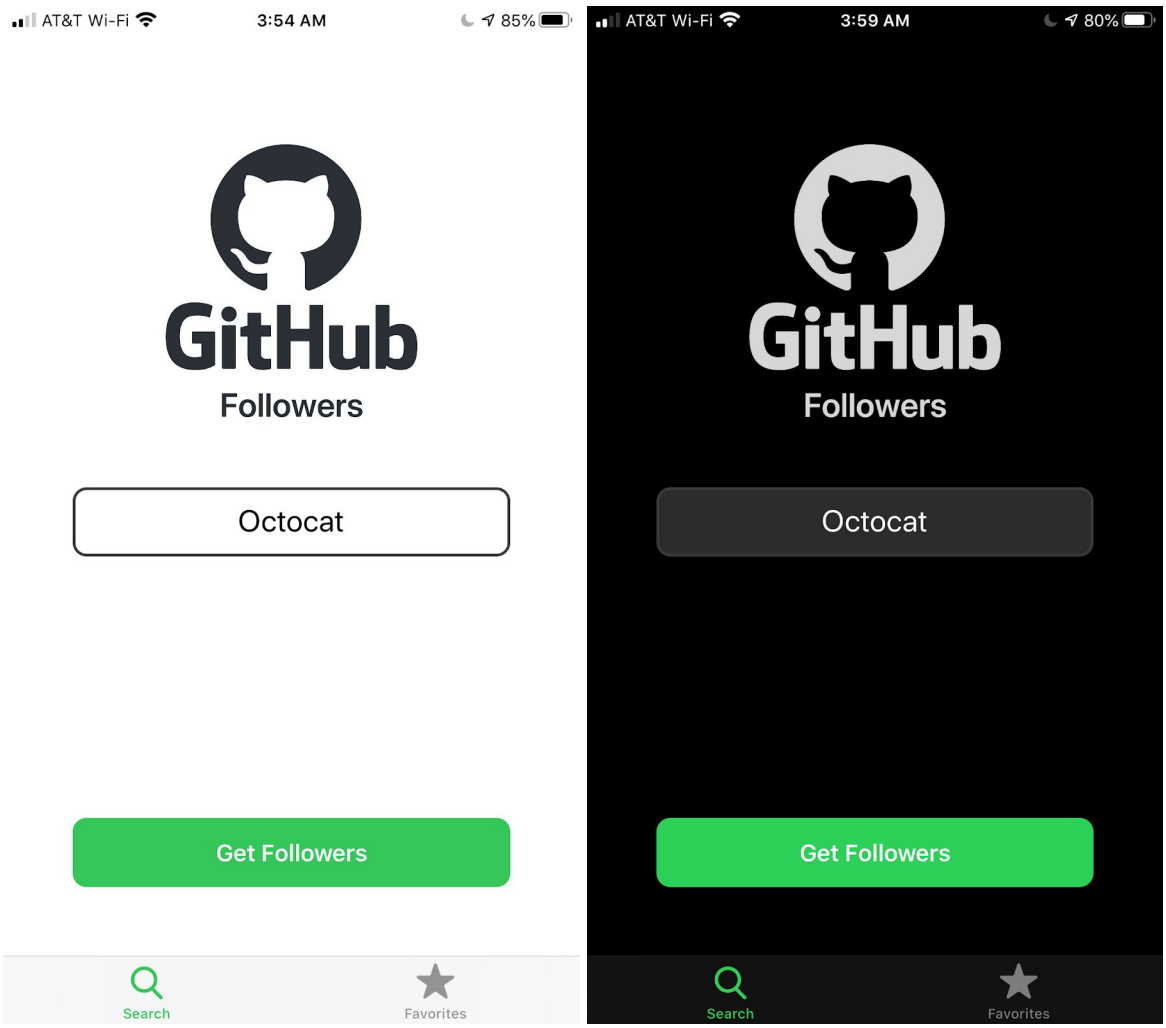
Custom Elements

- Custom Alert Controller
- Empty State View
- AvatarImageView
- Labels
- Buttons
- TextFields

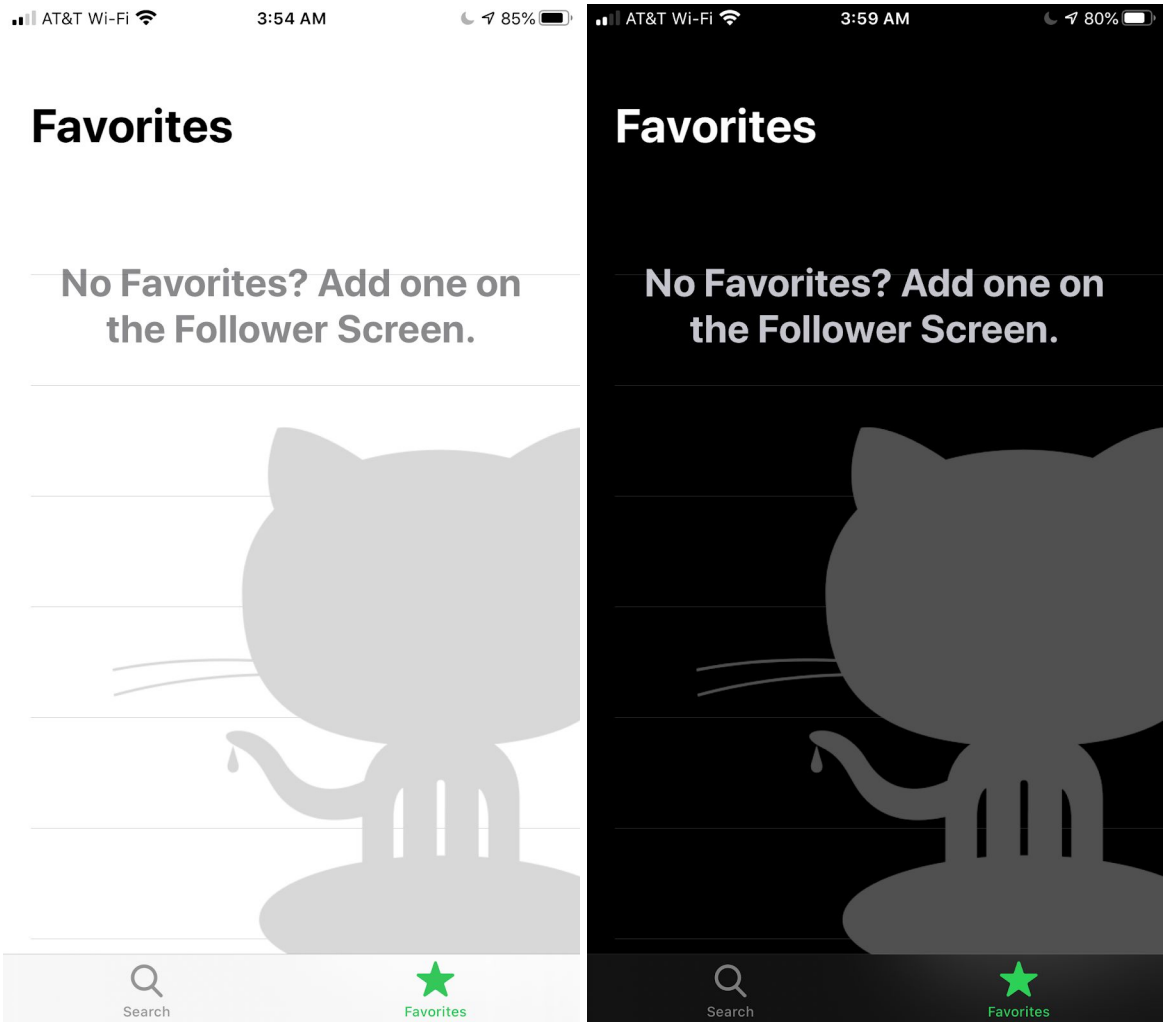
Project available on GitHub https://github.com/heminj/GHFollowers

Hemin Joshi
NUID - 001407458

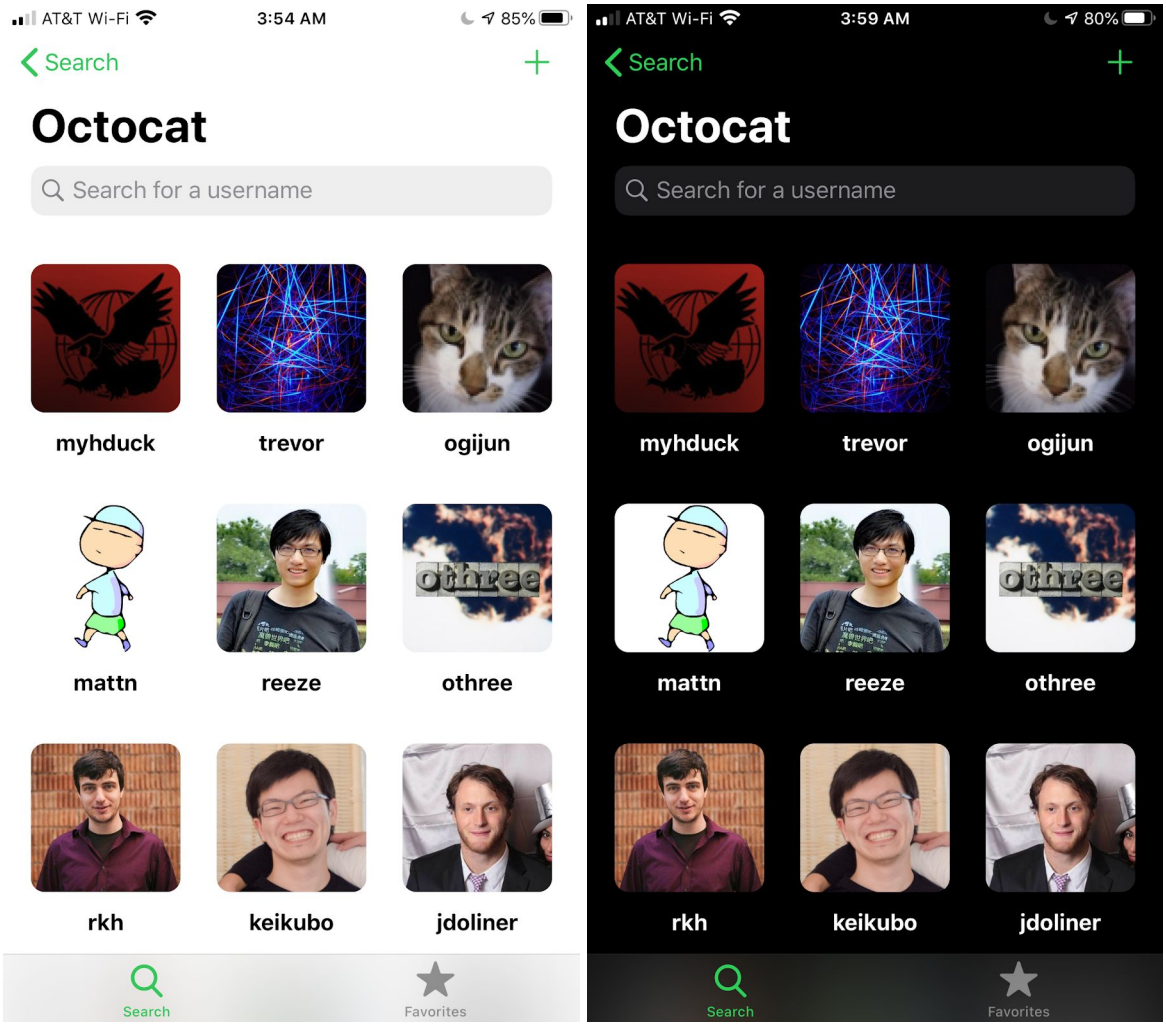
Screens



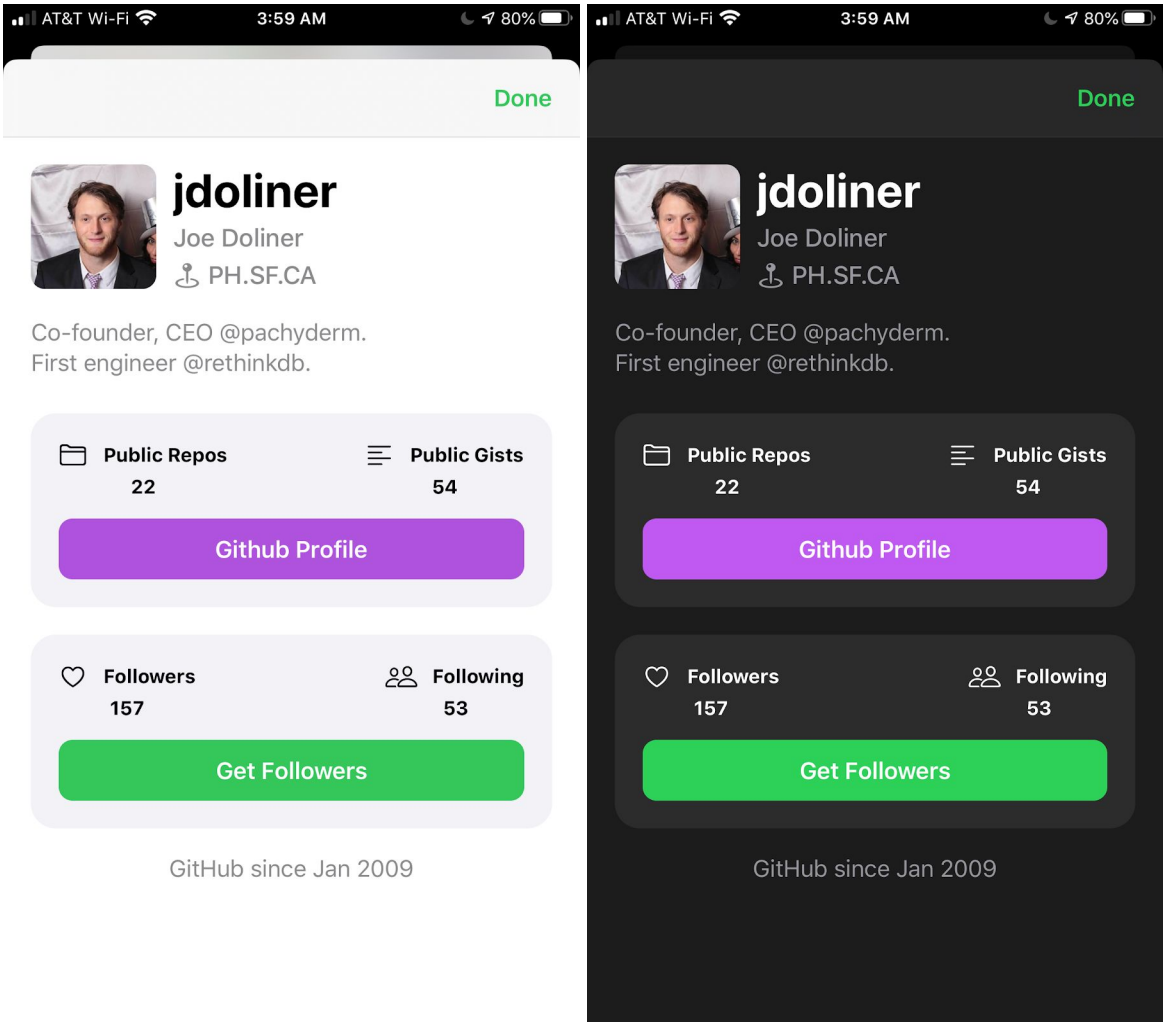
Project available on GitHub https://github.com/heminj/GHFollowers



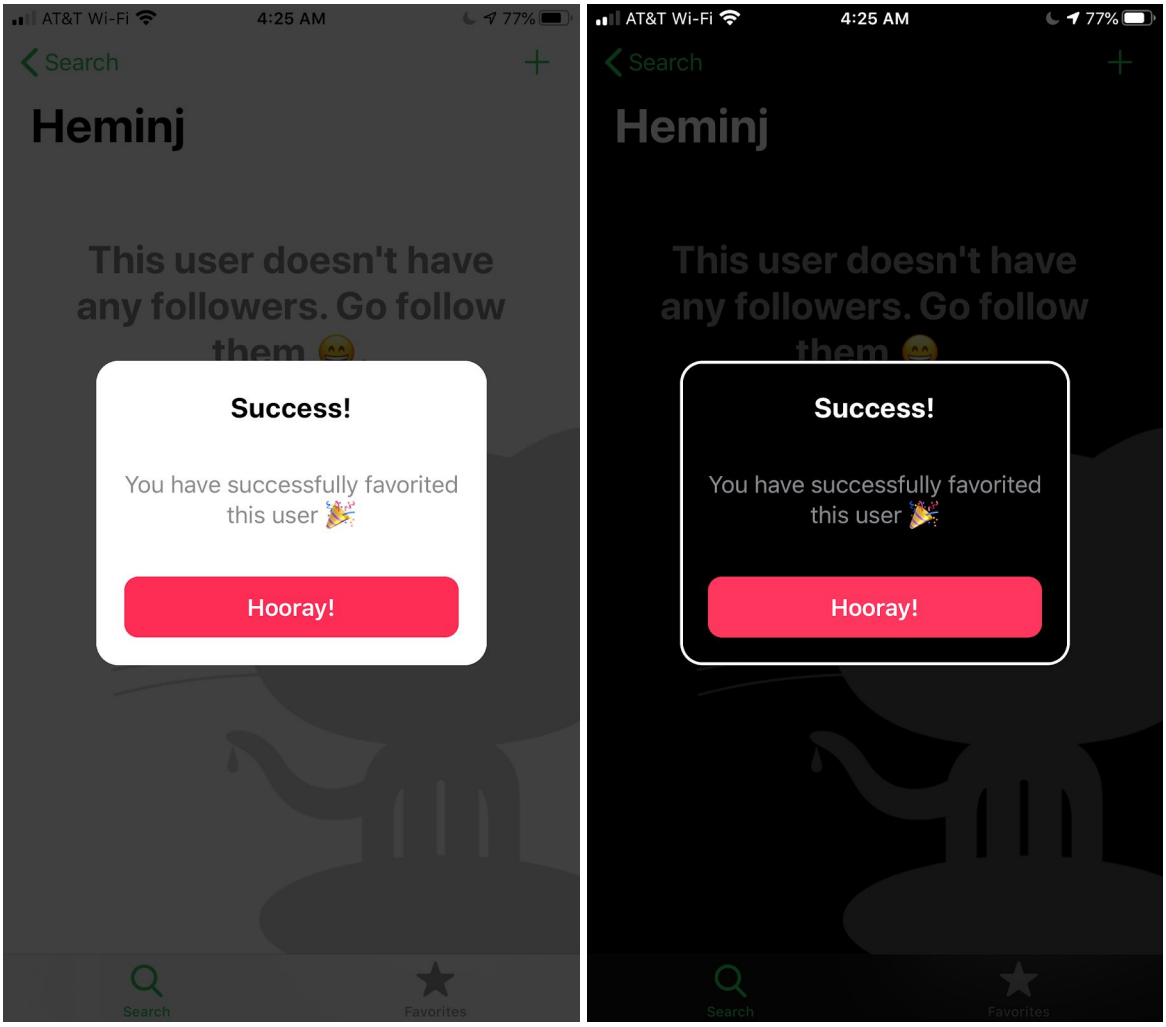
Project available on GitHub https://github.com/heminj/GHFollowers



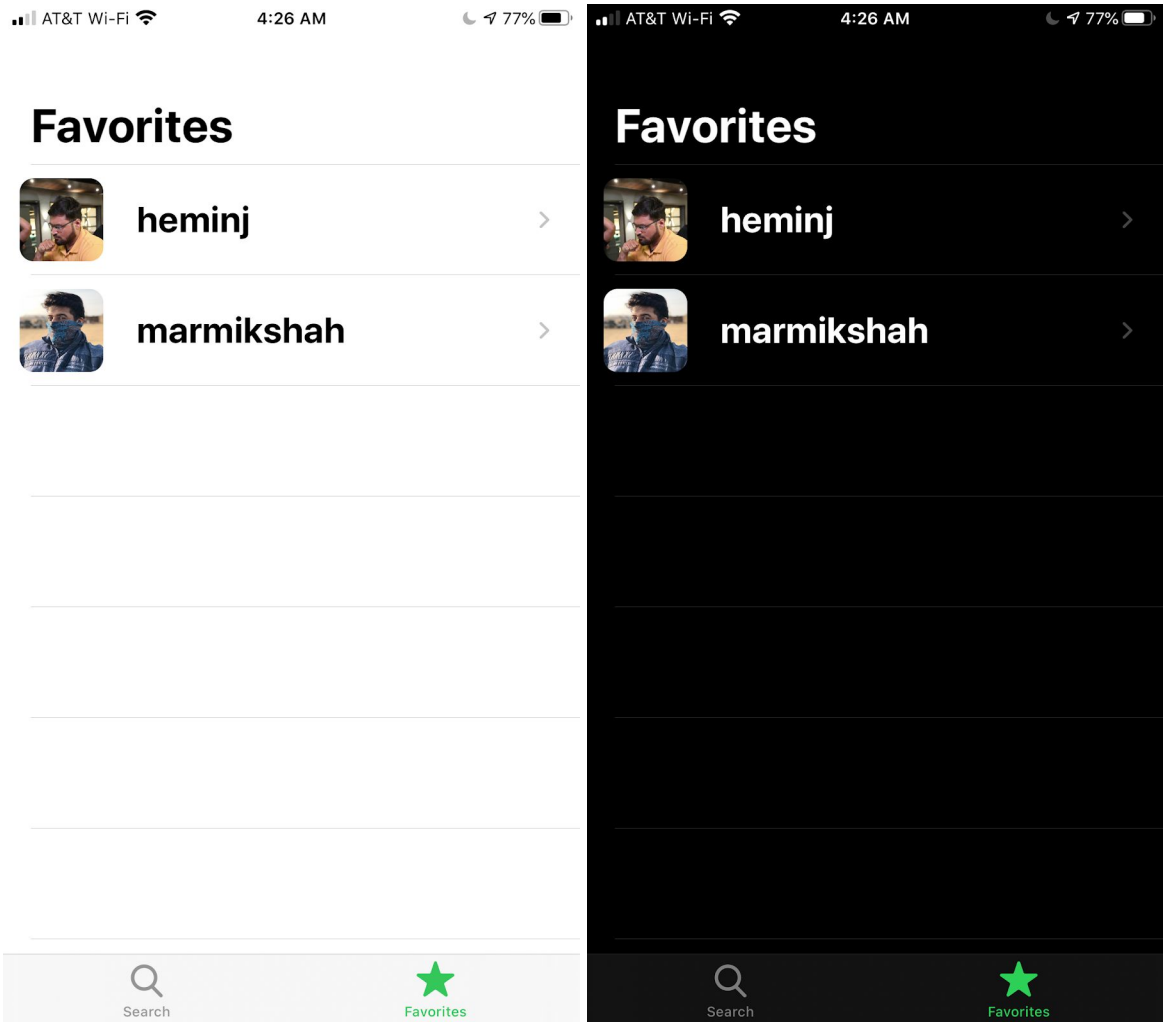
Project available on GitHub https://github.com/heminj/GHFollowers



Project available on GitHub https://github.com/heminj/GHFollowers



Project available on GitHub https://github.com/heminj/GHFollowers



Project available on GitHub https://github.com/heminj/GHFollowers

UICollectionView with Diffable Data Source

Diffable data source automatically calculates the difference between the old and new data source. Then it is able to apply the changes with a state of the art animation. DataSource has a hash value of your sections and each of the items in the CollectionView. It takes a snapshot of your UI before any changes are made, when the changes are made it takes a new snapshot and animates between the snapshots very smoothly.

Composition with Child View Controllers

The project uses Child View Controllers on the UserInfo screen. Use of UIViewController helped to access all the view *LifeCycle Methods* which helps in a dynamic UI where views are coming in and out. ViewControllers are also very *Self Contained* as they can not only present a UI but also have the logic to drive the UI in and of itself. As we know they can be pushed onto a nav stack, they do provide a *Flexible Context*. The entire composition is very contrived as this being an educational project allowed tinkering.

ARC, Memory Leaks, Capture Lists and NSCache

Every time a new instance of a class is created, ARC allocates a chunk of memory to store information about that instance. This memory holds information about the type of the instance, together with the values of any stored properties associated with that instance. Additionally, when an instance is no longer needed, ARC frees up the memory used by that instance so that the memory can be used for other purposes instead. This ensures that class instances do not take up space in memory when they are no longer needed. If there is a strong reference between two objects, even if one object goes out of scope, the other object will still have a strong reference to it and that will cause a memory leak. The fix for this will be to make one of the objects a *weak var*. In our app, while making a network call, a *self* in the closure of the network call, a strong reference is created between the NetworkManager and the ViewController which is made *weak* with the help of Capture Lists of *[weak self]*. Because of this, our app is AppStore ready as memory leaks are a reason to get rejected by the automated AppStore submission process. The Avatars are also cached using *NSCache*.

Threads

Multiple threads can improve an application’s perceived responsiveness and real-time performance. Whenever doing a background task like a network call or persistence in a concurrent queue, using *DispatchQueues,* the UI is updated on the Main thread asynchronously.

Future Scope

Complete GitHub credentials including the public repositories, commits, etc.

References

Lecture notes, Apple WWDC, Ray Wenderlich, Sean Allen, LBTA, PaintCode, NSDateFormatter

Project available on GitHub https://github.com/heminj/GHFollowers