tawk.to iOS Developer Test

Your task is to make an app that can fetch GitHub users list and then display the selected user's profile.

PLEASE NOTE - All requirements in all the sections must be met (except the BONUS section - that is optional, you can do all, some or none of the BONUS tasks).

General

1. In the first screen, the app has to fetch GitHub users list, parse it and display in the list (using UITableView or UICollectionView).
2. Selecting a user has to fetch the user's profile data and open a profile view displaying the user's profile data.
3. The design must loosely follow the wireframe (at the bottom of this document) but you must demonstrate a high level of knowledge about best practices of iOS UX and UI principles (e.g. HIGs). The app must look good, work smoothly and the design must follow platform defaults.
4. Own code logic should be commented on.

Generic Requirements

1. Code must be done in Swift 5.2 using Xcode 13 (latest official release), target iOS13.
2. **CoreData** must be used for data persisting.
3. UI must be done with **UIKit** using **AutoLayout.**
4. All **network calls** must be **queued** and **limited** to **1** request at a time.
5. All **media** has to be **cached** on disk.
6. For GitHub api requests, for image loading & caching and for CoreData integration only Apple's apis are allowed (no 3rd party libraries).
7. Use Codable to inflate models fetched from api.

8. Write Unit tests using **XCTest** library for data processing logic & models, CoreData models (validate creation & update).
9. If **functional programming** approach is used then only **Combine** is permitted (instead of e.g. ReactiveSwift).

GitHub users

1. The app has to be able to work **offline** if data has been previously loaded.
2. The app must handle *no internet* scenario, show appropriate UI indicators.
3. The app must **automatically** retry loading data once the connection is available.
4. When there is data available (saved in the database) from previous launches, that data should be displayed first, then (in parallel) new data should be fetched from the backend.

Users list

1. Github users list can be obtained from https://api.github.com/users?since=0 in JSON format.
2. The list must support pagination (*scroll to load more*) utilizing **since** parameter as the integer ID of the last User loaded.
3. *Page size* has to be dynamically determined after the first batch is loaded.
4. The list has to display a spinner while loading data as the last list item.
5. Every fourth avatar's colour should have its (image) colours inverted.
6. List item view should have a note icon if there is note information saved for the given user.
7. Users list has to be searchable - local search only; in *search mode,* there is no pagination; username and note (see Profile section) fields should be used when searching; precise match as well as *contains* should be used.
8. List (table/collection view) must be implemented using at least **3 different cells** (normal, note & inverted) and **Protocols** - meaning that controller (UITableViewController or UICollectionViewController) is unaware of specific cell or data (cell views, models & viewmodels) it has to show as long as those conform to certain protocols. It must be able to display any data (other cells) that would be added later without any modifications - e.g. adding cell with an indicator whether the user is a site admin (“**site_admin**”**)**. Example: `the tableview only knows that cell, it has to display, conforms to` **`AnimalCell protocol`**`, data model conforms` **`AnimalDataModel protocol`** `and it can get the cell to show from object conforming to` **`AnimalCellViewModel protocol`** `providing only itself (the tableView and the indexPath). Then the actual data would consist of all kinds of animals Cats, Dogs, Parrots - all conforming to AnimalXXX protocol.`

Profile

1. Profile info can be obtained from https://api.github.com/users/[**username**] in JSON format (e.g. https://api.github.com/users/tawk).
1. The view should have the user's avatar as a header view followed by information fields (UIX is up to you).
2. The section must have the possibility to retrieve and save back to the database the **Note** data (not available in GitHub api; local database only).

Submit your work

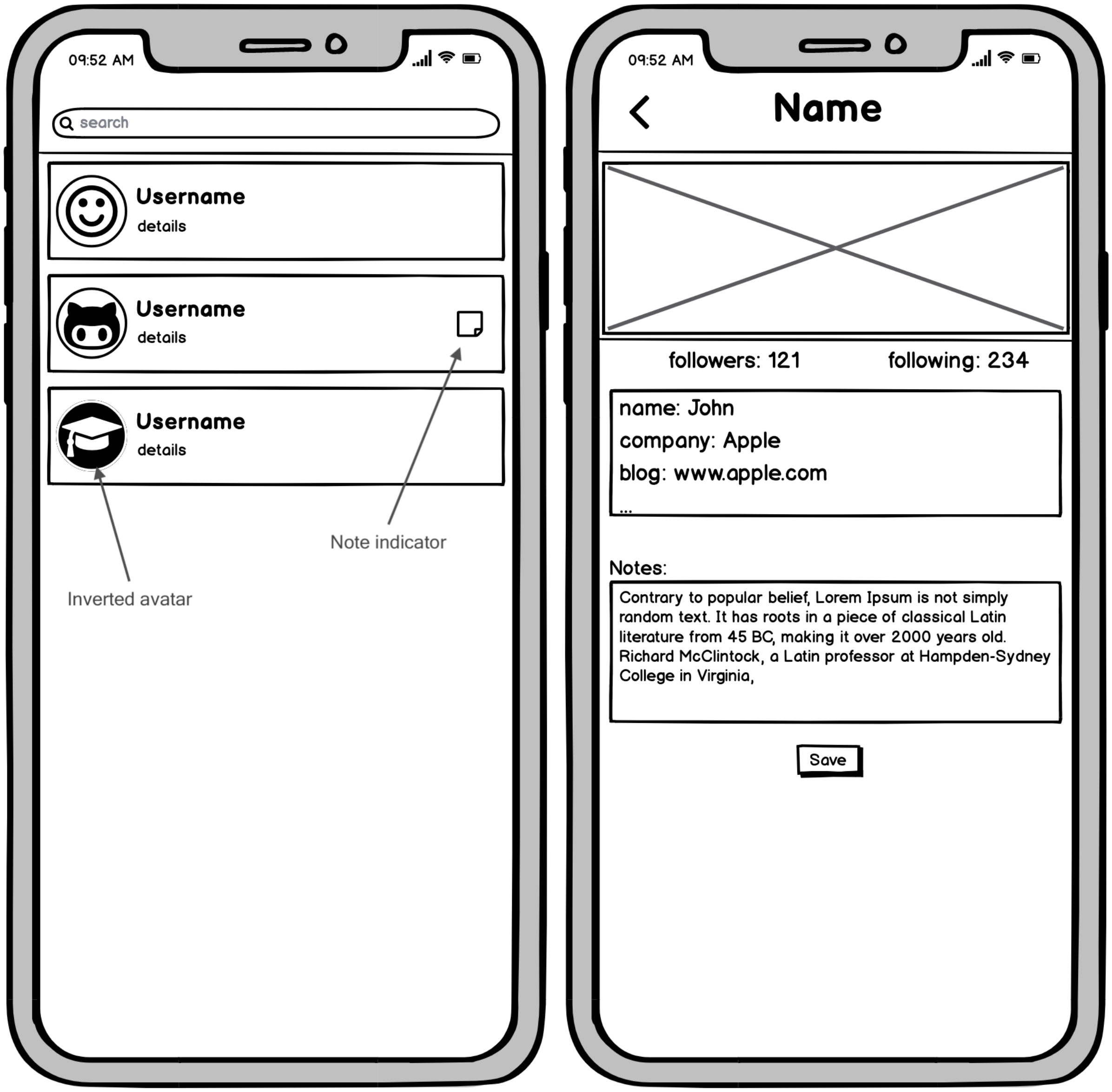
1. Make sure code is running on both - simulator and real device.
2. Push up to your repo. The repository should be publicly accessible - must not require any invitation to be accepted or registrations.
3. Reply back to the initial email with the link to your repository.
4. Any 3rd party code that is being used has to be bundled with the code even if the project is using any code dependency manager.
5. Please indicate which bonus tasks (see next section) have you completed.

BONUS

1. Empty views such as list items (while data is still loading) should have Loading Shimmer aka **Skeletons** ~ https://miro.medium.com/max/4000/0*s7uxK77a0FY43NLe.png **resembling** final **views**.
2. **Exponential backoff** must be used when trying to reload the data.
3. Any data fetch should utilize **Result types**.
4. CoreData stack implementation must use **two managed contexts** - 1.main context to be used for reading data and feeding into UI 2. write (background) context - that is used for writing data.
5. All CoreData **write** queries must be **queued** while allowing one concurrent query at any time.
6. Coordinator and/or MVVM patterns are used.
7. Users list UI must be done in code and Profile - with Interface Builder.
8. Items in users list are greyed out a bit for seen profiles (seen status being saved to db).
9. The app has to support **dark mode**.

Wireframe

wireframe.png



name: John

company: Apple

blog: www.apple.com

...

Notes:

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia,