Luke Hoffman

Dr. Pranshu Gupta

CS-453-01: Senior Seminar

May 9, 2025

iOS App Documentation

1. Overview

In this document, I present an informal overview of my contribution to the iOS application, some challenges faced, and a few closing notes.

2. Work Completed

For the iOS app project, as someone new to iOS app development, much of the time I spent on the project was in getting the app to build and in understanding SwiftUI enough to make nonbreaking changes.

Given that we did not have the final work from the Database team to integrate, much of our effort was focused on updating the existing screens in the app and implementing the Design team's GUI mockups into the SwiftUI code. My self-assigned tasks were the two reporting pages as they share code and progress on one could translate easily into progress on the other (as I am a SwiftUI and Xcode beginner).

After this, I familiarized myself with the code to understand where to begin styling UI elements. It turned out that the report pages share the flowchartSQL.swift file for their main functionality.

From here, it was lots of SwiftUI documentation and web searching for information on views and modifiers.

I worked on the following files:

- Report.swift
- report_elder.swift
- flowchartSQL.swift

For each, I styled them to correspond to the Design team's mockups and added comments for clarification and to delineate different functional sections of the code (that I could determine).

3. Issues Encountered

Out of the many errors I ran into, the two below gave me the most trouble and ate the most time.

3.1 Package Dependency Errors

After getting a copy of the project on my system, the app would not build due to package dependency errors. The rest of the team figured this out and it will likely be in their reports, but for the sake of documenting my own experience, I break down how I handled it in bullet points below. For reasons beyond me, I had to remove and add the dependencies (to mitigate more build errors) after each time I pulled code.

- Package Dependencies:
- PostgresClientKit version 1.5.0: https://github.com/codewinsdotcom/PostgresClientKit
- BlueSSLService: https://github.com/Kitura/BlueSSLService
- BlueSocket: https://github.com/Kitura/BlueSocket
- Steps Taken:
- In Xcode's Project Navigator, locate the Frameworks folder in the project hierarchy and delete whatever is in by right clicking the item and selecting "delete" (or similar) from the context menu.
- Also in the Project Navigator, click the project name at the top of the hierarchy to open the page of project properties. Find and select the tab "Package Dependencies."
- Click the plus button to begin adding the dependencies.
- From here, you can use the search bar "Search or Enter Package URL" to look up the link to the first package dependency, PostgresClientKit. The app uses version 1.5.0, so if you don't want to try using the latest version, select the "Exact Version" rule from the Dependency Rule dropdown and enter "1.5.0" there as the version to use.
- Click "Add Package," then click "Add to Target" and choose the app as the target (don't pick the tests).
- Click the next "Add Package" button to confirm.
- Repeat the process (but omit the Dependency Rule) for the remaining two dependencies.
- Build the project!

I found a brief article on Medium which I thought was informative about Xcode dependencies and worth mentioning even though it is not entirely relevant to my working solution above; the link is: https://medium.com/@fdchiu/how-to-add-3rd-party-framework-or-library-to-your-xcode-project-466ffa75b2e5

3.2 Type-Check Error

"The compiler is unable to type-check this expression in reasonable time; try breaking up the expression into distinct sub-expressions" — given my newness to SwiftUI this stumped me until I commented out all my changes and re-introduced them bit-by-bit to determine the source of the error. It embarrassingly turned out I was missing brackets.

4. Closing Thoughts and Notes

As I am a complete SwiftUI beginner, the styling could probably be streamlined in its implementation.

Also, I could not get the NavigationView (used in both Reports.swift and report_elder.swift to provide extra information upon button presses) to style. From resources online, e.g.

- https://medium.com/@amitsrivastava115/modify-navigationview-in-swiftui-f305bef9330c
- https://dev.to/shohe/swiftui-change-background-color-of-navigationview-form-3lpf

…it looks like it can be styled, but is beyond my current ability (although I did attempt both methods).

Lastly, the state Picker in flowchartSQL.swift and the Alert that is in both Report.swift and report_elder.swift could be styled to match the rest of the design better, but that was also beyond my ability and time. It appears that styling the Picker involved the global menu style (https://www.swiftyplace.com/blog/swiftui-picker-made-easy-tutorial-with-example). I did not figure out how to style an Alert.

Regardless, now the pages are much more polished in appearance and the code has been clarified with sectional comments.