CSE118 Summer 2023

Assignment 5

In this assignment you will develop a simple Swift / SwiftUI app to display hierarchical data loaded form a JSON resource to solidify your understanding of:

- Resource Bundles
- JSON Serialization
- Protocols
- Lists & Stacks
- Navigation Views
- Automated UI testing

This assignment is worth 10% of your final grade.

Setup

Download the starter code archive from Canvas and expand into an empty folder. I recommend creating a folder for the class and individual folders beneath that for each assignment.

Start Xcode and open the project: **File → Open**

Select the folder where you expanded the starter code and click **Open**.

Requirements

Basic:

Implement a SwiftUI iOS App that loads Slack-like hierarchical data from the resource `Workspaces.json` and exhibits a user interface with the following navigation:

Workspace and Channel Lists:

Drill-down from Workspace to Channels within a workspace, as demonstrated in class. Only need to show the name of the Workspace or Channel.

Message List:

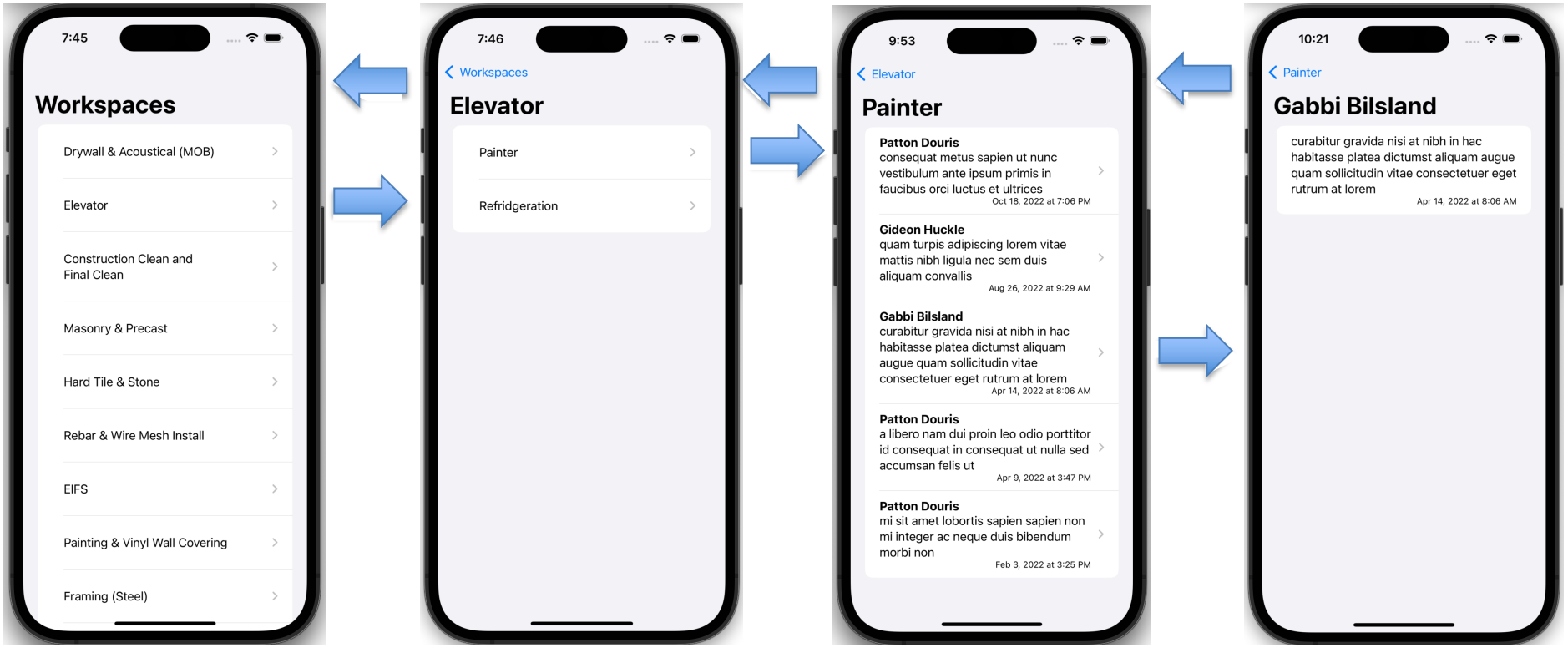
List of messages in a Channel, drill-down to here from the Channel list. Show the poster (Member) name, the content and the date/time the Message was posted in long form, `Oct 18, 2022 at 7:06 PM` for example.

Individual Message:

View of a single Message, drill down to here from the list of Messages in a Channel. The poster (Member) name becomes the navigation title. See later screen shot for an example.

See following page for navigation flow between views.

 Copyright © 2022-2023 David C. Harrison.



Workspace List | Channel List | Message List | Individual Message

Copyright © 2022-2023 David C. Harrison.

Advanced:

Modify the Workspace and Channel lists so items have the following additional information displayed, and write tests to demonstrate the accuracy of your implementation:

Channels:

- The number of Messages
- The number of unique posters (Members) across all Messages
- When the most recent message was posted

Workspaces:

- The number of Channels
- The number of unique posters (Members) across all Messages in all Channels
- When the most recent Message across all Channels was posted

Where the most recent message indicator follows these rules:

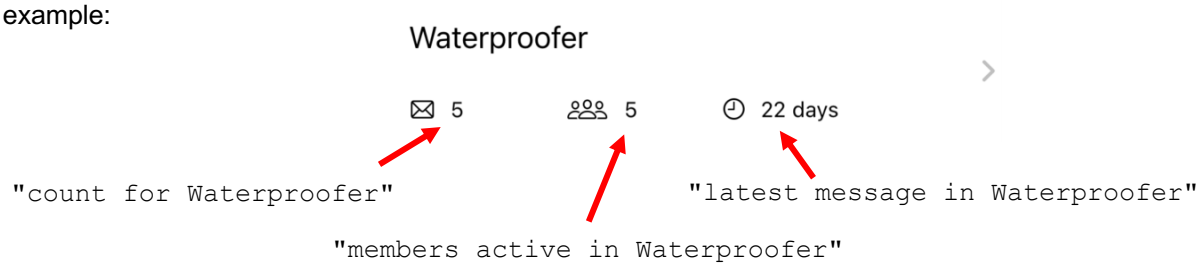
- No messages: Blank
- Less than a minute ago: `N secs`
- More than one minute but less than one hour: `N mins`
- More than one hour but less than one day: `N hours`
- More than 24 hours ago: `N days`

View types and accessibility labels:

Numeric indicators must be disabled `TextField`. Accessibility identifiers are dynamic, based on the Workspace / Channel being show:

count for <workspace or channel name>
members active in <workspace or channel name>
latest message in <workspace or channel name>

For example:



Finally, lists of message in channels should sorted by date posted, most recent first.

 Copyright © 2022-2023 David C. Harrison.

Stretch:

Your App should exhibit 100% class, method, line, and branch coverage when the provided Basic tests and your tests for Advanced (if any) are executed.

What steps should you take to tackle this?

Start by defining serializable Structures for the Workspace, Channel, Message, and Member data types.

Next load the data from the provided JSON as a resource.

Next create a Workspace list to satisfy the Basic requirement, then link it to a Channel list, then link the Channel list to a Message list and finally to an individual message screen.

Alternatively, you can start with example data in your previews then work on loading the JSON resource.

Once you are passing all the Basic tests with 100% code coverage, move on to the Advanced requirement writing tests as you go.

Note that completing the Basic requirement with 100% code coverage is worth 60% on this assignment. Take care when implementing Advanced functionality to pay close attention to your coverage.

How much code will you need to write?

A model solution that satisfies all requirements has approximately 500 lines of code, including tests for the Advanced requirement.

Grading scheme

The following aspects will be assessed:

1. (100%) **Does it work?**

    a. Basic (40%)
    b. Advanced (40%)
    c. Stretch (20%)

2. (-100%) **Did you give credit where credit is due?**

    a. Your submission is found to contain code segments copied from on-line resources or created by code generation tools and you failed to give clear and unambiguous credit to the original author(s) in your source code You will also be subject to the university academic misconduct procedure as stated in the class academic integrity policy. (-100%).

    b. Your submission is determined to be a copy of a past or present student's submission. (-100%)

    c. Your submission is found to contain code segments copied from on-line resources that you did give a clear an unambiguous credit to in your source code, but the copied code constitutes too significant a percentage of your submission:

        - < 25% copied code No deduction
        - 25% to 50% copied code (-50%)
        - > 50% copied code (-100%)

 Copyright © 2022-2023 David C. Harrison.

What to submit

On the console (PowerShell on Windows), navigate to the folder you extracted the starter code into and run the appropriate command to create the submission archive:

Windows:

```
$ Compress-Archive -Path . -DestinationPath Assignment5.Submission.zip
```

Linux:

```
$ zip -r Assignment5.Submission.zip *
```

Mac:

```
$ zip -x "*.DS_Store" -r Assignment5.Submission.zip *
```

**** UPLOAD** `Assignment5.Submission.zip` **TO THE CANVAS ASSIGNMENT AND SUBMIT ****

 Copyright © 2022-2023 David C. Harrison.