Todd Bruss
iOS / macOS Developer
9840 Park Springs Court
Charlotte, NC 28210
E-mail: todd@starplayrx.com
Phone: 704-641-9300
Github: https://github.com/starplayrx
www: https://starplayrx.com

Big Nerd Ranch iOS Code Challenge

Thank you for allowing me to take this coding challenge. Here is my walk through of the app.

I studied the read me file, took notes of what needed to be fixed and proceeded to run the mock API. Using Xcode and the Simulator, launched the app to see if there were any problems with compiling. The Mock API tended to crash each time when the App was compiled and executed in the Simulator. Running the Mock API's shell's script using sudo fixed this issue. Ran some calls to the Mock API in Safari and everything checked out.

Took some time getting to know how the app was composed and noticed it followed an MVC, Model View Controller design pattern.

In the Model, the PostMetadata was missing some fields. Added Id and Summary. Added Author and its Name, Image and job Title to the PostMetadata Codable.

From my notes, decided to get sorting to work. That was the lowest fruit. Created a sort routine for by author name, sort by title and a sort by publish date both ascending and descending. I made use of Switch Case statements which worked really well for the sorting and grouping enums' arguments.

Now that we got that sorted out, let's move to the Display.

The read me gave a choice of a Xcode Interface Builder file (.XIB) or a storyboard for the cell summaries. Since a UICollectionView was already used in the app, I decided to layout the cells for the main view using .XIB files with matching Class files. I registered the XIBs in the view's Controller and all is well. Cells where used for both the section headers, used for the group's titles and the post's summaries.

The original layout was not attractive and we had this rich metadata. A whole new look and feel was added to the app and not only was the authors name, date and summaries added to the XIB file, but the author's job title and picture was added too. The photos were square and mathematically that's ideal for making them round, which is trendy too and that was also done in code.

The next phase involved making the app more performant. We already knew we had one problem with Archibald Sweet's giant post that take over 2 seconds to load.

What do we do with Sweet's large post? This network issue could happen again, so let's add a loading message to the title of the next screen and put in a spinner in the middle. Once the data is loaded, show Sweet's post! This gives the user a clear indicator that we are working on it and this issue will work in any slow network situation.

The original app was loading all the posts on each tap and was filtering the data afterwards. Since know the Id of the post which is dependable according our Docs, we can just load the post that was tapped with that Id.

The data was also waiting to load before running the segue to the post's view. This blocked the UI and made the app feel less responsive. I decided to trigger the segue first, load the data the background and present it when ready.

The next task is getting the groups to work. For grouping items, a Dictionary was utilized. One caveat with Dictionaries is order. Sets were also used to store the key or name of each group. Sets don't keep the order either! What's going on here? What Sets do provide is speed and uniqueness. This ensures no duplicate authors or months are stored for the

groups. To fix the ordering issue for both Sets and Dictionaries, after the Set is created, I converted the Set to an Array and sorted that array. Great, now we can layout our group using the sorted array as our template!
After each group is created, I decided that's a good place to complete any additional sorting that needs to take place. Each Array inside of each group is sorted accordingly to our original 3 sort subroutines. I used the same grouping algorithm for both groups by author and groups by month/year.

Now that the grouping and sorting is working together in harmony, which was no small feat, I went back to the main view and added gray background to the header and used semi bold type. The spacing of each "card" or cell was tightened up. Cell separators were simulated using the background of the cell and adding a pixel gap between each card. Giving each summary some nice separation with little effort.

I tested the sorting several times and fixed any bugs along the way. It helped to keep track of the last sort call for author and published dates to simulate a hierarchy between moves. This helps prevent mishaps from happening when sorting is used in conjunction with the groupings. We want our app to look good and work good too!

Unfortunately, I did not have enough time to write unit or UI tests. What I would do in this situation is while the app is out for code review, approval and is awaiting changes, I would write the unit and UI tests.

Where to go from here? Color would be nice for the app. My layout is more of a newspaper style. I would research the customer's branding and color schemes and add that to the app. Another approach might to do redesign the App using Swift UI and add in some nice animations.

One more thing. I did some brainstorming on a predictive AI. This idea takes what is visible on the screen and automatically preloads and caches the post's content. Then when the user clicks on long post chances are it is already loaded or is on its way. These predictions could then be saved in memory by the Id and its contents would be the title and body (or what the clients wants). To take that a step further and store it using User Defaults or Core Data for offline reading.

iOS Software Engineer | Swift | Objective-C
Super HiFi
2019 - 2020

Worked on Music AI services for LA based music startup Super HiFi.
Developed offline gapless playback macOS database for a client.
Created an exclusive iOS radio player for Sonos Radio used internally by Sonos and Super-HiFi executive teams.
Consumed REST endpoints with Codable, JSON Decodable/Encodable, Serialization & QuickType.io
Using AVQueuePlayer, created an audio stream emulator designed to test customer content.
Utilized UICollectionView, UITableView, UIKit, reusable cells, dynamic UI and OOP for UI/UX.

Web App Developer | Swift | JavaScript
SignUpGenius, a Lumaverse Technologies Company
2014 - now

Presented an iOS proof of concept app demonstrating the company's APIs using Swift.
Invented the Theme Builder. It allows the user to change the look and feel of their sign ups.
Engineered drag & drop image uploads, image resizing and pixel sharpening.
Developed a new graphical user experience for the New Wizard.
Created an interactive user portal page for our Metric Reports App used by corporate employees.
Programmed a location manager tool that allows customers to add locations to their Sign Ups.
UI/UX, Front End JavaScript, Back End ColdFusion Script and MicroSoft SQL.

iOS Imagineer | Swift | Objective-C
StarPlayrX
2014 - now

Invented StarPlayrX, a fully accessible third party Streaming SiriusXM Radio player.
Created CameoKit, an embedded HLS framework that does all the heavy lifting for StarPlayrX.
Designed UFO Emoji, a side scroller arcade style video game for iOS.
Developed Space-Bar, a futuristic 2D breakout game that has a 1 player air hockey look and feel.
Programmed an experimental 3D ARKit game that uses art on a physical wall to create a 3D breakout game.
Created CloneToolX, backup and restore software for macOS 10.15.
Forged Cat-Woman shell script that allows Mac Pro 2008 3,1 to run macOS Catalina.
Engineered a private framework that can natively control the volume of any AirPlay 2 audio device.

Mac Developer, Designer, Founder | AppleScript | Objective-C
NiceMac
2002 - 2014

Invented StarPlayr for Mac, a revolutionary streaming Sirius Radio Application.
Reverse Engineered Sirius and XM's flash web players and private REST APIs.
uSirius StarPlayr for iOS, featured Album Art, Lyrics, and Song Purchases with Amazon and Apple.
Produced several Mac Radio players: SiriusMac, StarLight, StarLightXM, StarPlayr v1-3.
Invented a virtual jukebox that would automatically tune to the user's favorite artists or songs.
Designed the UI/UX/Graphics for uSirius StarPlayr for iPhone 1G, and StarPlayr for Mac and Windows.
Interviewed with financial blogging and iPhone news sites about uSirius StarPlayr.
Maintained NiceMac's websites, graphic design and marketing. Trademarked the StarPlayr name.
Coined the slogan, "Don't be a Slacker, be a Star Player."
Produced presentation materials for a private meeting with Sirius Satellite Radio.
Created CloneTool, a Darwin based backup utility to backup and restore Macs and Hackintoshes.
Made FreeStepX, step and repeat graphics software that used Macromedia FreeHand for print automation.

Workflow Automation Developer | AppleScript | PostScript
Sherwin Williams Graphic Arts Division
1999 - 2002

Invented Imposition Pro, a fully automated graphic workflow system for offset printing.
My software sported a user-friendly drag and drop interface that was way ahead of its time.

Education

Gaston College

AFA - Graphic Design
GPA 4.0
Dean's List