iOS Final Project: Fitness Planner

Nicholas Zustak

12 December 2016

Technical Aspects

Fitness Planner was made to hold a user's entire weeks workout in an app, and provide a convenient way to see the day's workout and check off exercises as they're completed. The information is be sorted, from outwards in, by weekday, then muscle group, then exercise name, then number of sets and number of reps.

In the genesis of Fitness Planner, I had to decide on a data type to hold all workout information. For better or for (probably) worse, all data for all workouts ended up being held in workouts.plist, which holds a data structure of the form:

```
[ String : [ String : [ String : [Int] ] ] ] =
[ Day : [ Muscle Group : [ Exercise : [#Sets, #Reps] ] ] ]
```

The idea was to have all the data conform to this protocol, so a singleton class WorkoutInformation.swift could hold all information that was ever needed at any moment and be accessed and changed as needed.

This data type proved problematic, and I was unable to edit actual exercise names or delete them within the app. Yet, after all this, the app will let the user edit the number of sets and the number of reps for each exercise. So, I made it such that every single day holds every muscle group and nearly every possible exercise for that muscle. If Monday is Legs day, then just edit the number of sets for all other muscle groups to be zero.

Lastly for the extra features of this app, a progress bar allows the user to see how many of the available sets for the day he has already completed, visually. Each day, the number of set to complete is calculated by referencing the Calendar package with the

workout information. As sets are completed, the bar fills up. Once all sets are done, the bar turns green and you are done for the day!

The other feature is a rest timer. It is a simple start, pause, and reset timer that leaves it up to the user to see how long they've been resting. After 30 seconds, the timer begins to change color to a deepening yellow to pressure him to continue with the workout.

Challenges

The dictionary of dictionaries of dictionaries of ints created problems that were often and numerous. To access individual pieces of data, all pieces were optionals and had to be unwrapped; if any piece of code was written to require a value that wasn't there, the program would crash. This was easy enough to fix as the bugs presented themselves; however this resulted in data not being very manipulatable.

Moreover, this data structure prevented (in the timeframe) being able to edit the table views and table view cells. Going off of our previous notes and online resources, every attempt to edit the table view in the hierarchy did not work. I believe all would have been fine if the data were separated into multiple different files, perhaps having only 1 nested dictionary. Moreover, data persistence did not work; in the future, I would have all workout information be stored permanently by importing "NSCoding".

When working in the storyboard within Xcode, I encountered a few problems. First and absolutely foremost, all relationship "show" segues created between view controllers refused to have a "Back" button. To this day I have no idea what is wrong;

every show segue created for previous assignments automatically created a navigational back button at the top of the destination view. For that matter, no navigational controls ever worked at the top of the view. I could never get “Edit”, “Cancel”, or any sort of natural controls to show up when they needed to.

To solve the back button issue, I created manual “Back” buttons at the bottom of each view that simple rewinded the segue. One issue from this (that still exists) is that most views will rewind back to the view from first tab, even if they were originally accessed from views on different tabs.

Another challenge was trying to implement the progress bar as a constant view. The progress bar information is held in another singleton class that derives UIView, and when trying to put it on every other view, the progress bar only drew once. That is, after completing more and more sets, the progress bar never changed. I realized this was happening because the CG Rect being passed to the draw method inside the class was of size 0, 0 after the initial drawing. This was solved by putting the progress bar on its own tab, and having all information it uses accessed through WorkoutInformation on every call to draw, instead of being stored to class variables.

Final Thoughts

The Fitness Planner app was an educational experience, and solidified everything I’ve learned in the semester and taught me a few more things. Given that I created the workouts.plist myself, I will use this app for my workouts and in the future. I’m excited to tackle another iOS project soon with all the knowledge I’ve gained.