Meme v2.0 - Udacity iOS Nanodegree

by Ryan Collins Project Link: http://techrapport.com/Meme-Me

GitHub: https://github.com/TechRapport/Meme-Me

Build information

- Version: 2.0
- xCode: Version 8
- iOS: 10.0
- Swift: 3.0
- iPhone: 4S or later

This app was created as a project for the Udacity iOS Nanodegree. Before creating this app, I took a month long course on UIKit Fundamentals through Udacity. This course firmed up my working knowledge of Apple's UIKit framework, which is used extensively for UI design, control and navigation.

The app offers the following features:

- Create a Meme by taking or selecting a photo
- Add text to the top and bottom
- Change the color, font and size of the text
- Share and Save your Meme
- View saved Memes in a UITableView and a UICollectionView
- View a saved Meme
- Edit a saved Meme
- Delete saved Memes

See the extra credit features and what I intend to add next at the end of this article.

Technologies and best practices used

Some of the more notable technologies and best practices I used in this app are listed below. I also wrote about some of these subjects in my blog in the Swift development section.

- UIKit
    - UINavigationControllers
    - UITableViews
    - UICollectionViews
        - UICollectionViewFlowLayout
        - UICollectionViewCell
- MotionKit
- Delegation
- MVC (Model View Controller)
- NSNotifications
- Extensions

As I stated above, I developed this application using several important design paradigms and best practices. Most notably, I used Model View Controller (MVC) extensively. I created a Meme struct model object, a font attributes struct model and a MemeCollection struct model for storing and processing Memes. At this point, the app does not use Core Data to persist the data, but it will in the next version.

All model, view and controller components are completely separate of one another and Model and View objects only talk to one another through the view controller. I used delegation & data source methods and key-value observation to insure this. It will be very easy to utilize Core Data and networked data within this app since the Model is abstracted away from the View & View Controllers.

I used optionals and unwrapped them when necesarry. To avoid bugs, I only used implicitly unwrapped optionals on values that I knew would never be null, otherwise I was safe and unwrapped optionals.

I created several View Controllers for controlling navigation, writing to the model and updating the display. I also utilized extensions, NSNotifications, the delegate design pattern and more. I really enjoyed using Swift 2.0 and found that I was able to add a lot of functionality very easily using extensions.

I used storyboard on many occasions, but also built some of the user interface in code. I found that doing both helped to cement the ideas and to understand what is

going on behind the scenes. I used segues throughout my project, most of the time configured in storyboard, but also in code. I used unwind segues to unwind to the main view after editing a saved meme.

Git & GitHub

Although I have been using Git and other version control systems for many years, I have never used Github professionally because my professional work has been private. I used best practices when using Git and Github. I used separate branches for the two versions, wrote commit messages that were clear and concise, used Git to debug and I chose logical points to commit.

Extra Credit

In order to receive extra credit and to receive Exceeds Specifications, I added additional functionality to v2.0 of Meme-Me. Some of the functionality that I added are listed below.

Color and font picker, which both show as UIPopoverViews.

The color picker is really great and allows the user to select a color by gliding their finger over a group of colors, each of which shows up inside of a magnifying glass, showing the user which color they selected. As the color is picked, the font is updated in the view and the color/font attributes are stored with that instance of the Meme, so the user can edit the meme at a later time.

Shake to reset

I added a shake to reset feature, which greatly improved the experience by allowing the user to shake the phone to reset the font back to default. I did this using a single extension, which extends UIViewController.

Class extensions

As I was developing this app, I realized that my code was getting a bit complex, especially after I added delegate functions for the text fields, UITableViews, UICollectionViews, and other UIKit compononents. In order to make the code more readable and manageable, I once again used UIViewController extensions in order to add the delegate functions for various UIKit components. I find that this is good practice as long as you contain the extensions in one file, preferably in the same file of the view controller that utilizes them. I would recommend utilizing extensions for basic delegate functionality that will be shared throughout your app.

Collection View Features

I added some custom functionality to the collection view in this app. I added the ability to select a Meme in the collection view to edit it, add a new Meme and I added functionality to delete multiple memes at once. To handle this, I tracked an array of indecies that corresponded to the selected Memes. When the delete button is pressed, the view controller sorts the array, then asks the model to delete the Memes at the index collected in the array, deletes the corrseponding cells and then updates the view.

Other

To make this project more professional, I added a launch screen with a small loading indicator, following Apple's iOS Human Interface Guidelines. I also made the interface look good while still following their guidelines.

I added helpful text to show the user what each view does and I used icons (see below for credit) to show what each button does. Each icon used has all size classes in the xCode Assets file, so they will show nicely on any iPhone.

Wrap up

All in all, this project helped me to learn a tremendous amount about developing for the iOS platform. I feel that I have mastered UIKit, that I can do just about anything on the iOS platform now that I know how to navigate the user documentation, and I feel that I am building on my programming knowledge by utilizing best practices. I am also taking the Stanford iOS development courses, which complements this course tremendously. I am extremely confident that I will be developing professionally for the iOS platform in no time.

More features

Well, you'd think that I would be done with this app, but I am not. In the next iteration, I will be including Core Data to persist data, Flickr images and randomly generated Memes. Keep a watch out for the next revision as I plan to release it to the App Store.

A special thanks goes to: "Christian Zimmermann" who wrote the code for the SwiftColorPicker. You can view his GitHub page here: https://github.com/Christian1313/iOS_Swift_ColorPicker.

Please take a look at my blog to learn about how Udacity has helped me succeed: Tech Rapport's Blog

Credit for the app icon goes to Udacity.

Icons made by Zurb from www.flaticon.com is licensed by CC BY 3.0
Icons made by Picol from www.flaticon.com is licensed by CC BY 3.0
Icons made by Freepik from www.flaticon.com is licensed by CC BY 3.0
Icons made by Freepik from www.flaticon.com is licensed by CC BY 3.0