LambdaSchool / **ios-sprint-2-challenge**

No description, website, or topics provided.

1 commit **1** branch **0** releases **1** contributor

Branch: **master** New pull request Create new file Upload files Find file Clone or download

erica first commit Latest commit `221fe36` 2 days ago

Crayons.xcassets	first commit	2 days ago
images	first commit	2 days ago
Crayon.swift	first commit	2 days ago
CrayonModel.swift	first commit	2 days ago
README.md	first commit	2 days ago

README.md

Sprint Challenge: iOS Fundamentals 2 - Color List

This challenge allows you to practice the concepts and techniques learned over the past week and apply them in a concrete project. This Sprint explored iOS Fundamentals. During this Sprint, you studied delegation, auto layout, detail views, segues, and more. In your challenge this week, you will demonstrate proficiency by creating an application that displays crayon colors and allows users to select their favorites.

Instructions

Read these instructions carefully. Understand exactly what is expected *before* starting this Sprint Challenge.

This is an individual assessment. All work must be your own. Your challenge score is a measure of your ability to work independently using the material covered through this sprint. You need to demonstrate proficiency in the concepts and objectives introduced and practiced in preceding days.

You are not allowed to collaborate during the Sprint Challenge. However, you are encouraged to follow the twenty-minute rule and seek support from your PM and Instructor in your cohort help channel on Slack. Your work reflects your proficiency in iOS and your command of the concepts and techniques in this first unit.

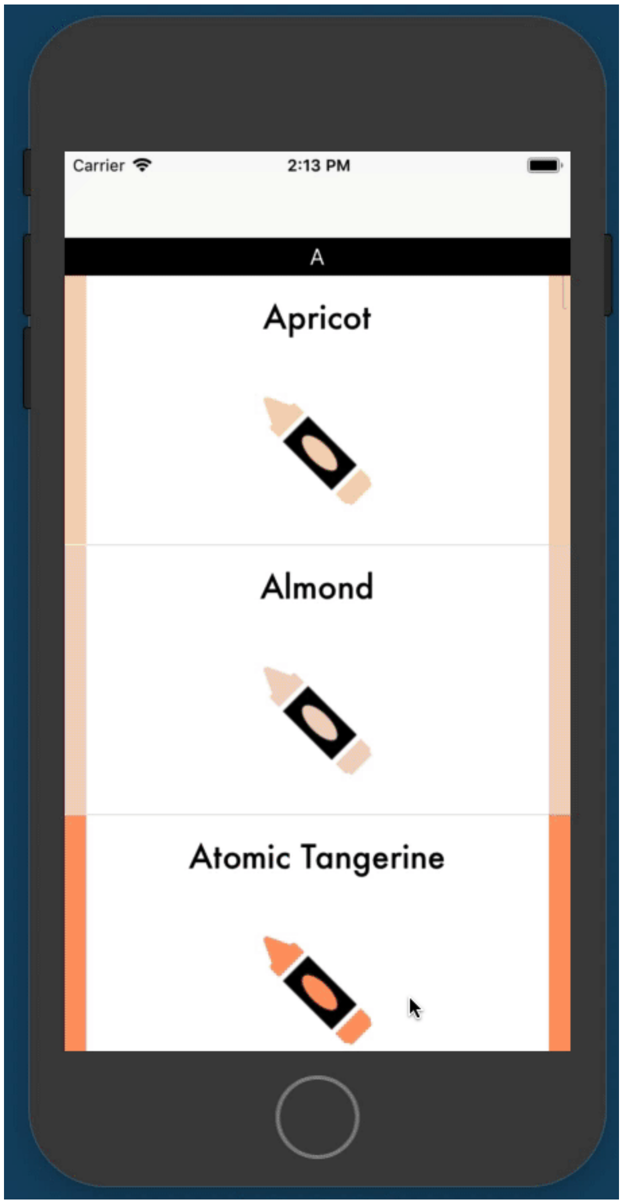
You have three hours to complete this challenge. Plan your time accordingly.

Commits

Commit your code regularly and meaningfully. This helps both you (in case you ever need to return to old code for any number of reasons) and your project manager.

Description

In this challenge, you build an app that displays crayon colors and allows users to select ones they particularly like. Your finished project looks something like this and emphasizes Auto Layout skills:



In meeting the minimum viable product (MVP) specifications listed below, your application will be able to display color details and allow users to "like" their faves using one of two handy buttons that work in sync.

Project Set Up

This repository contains a basic project that includes pre-built classes for a crayon model and an asset catalog with individual crayon art.

Minimum Viable Product

Your finished project must include all of the following requirements:

- A scrolling table view that looks (more or less) like the animated GIF.
- You will include section headers grouped by the first letter of each crayon.

- Cells display color names and crayon images. Each cell is bordered with supporting color swatchs on its edges.
- Each cell leads to a crayon detail view.
- The detail view includes the crayon image, its name, and a swatch on a white background. Behind the white background, the main view is tinted with the crayon color.
- There are like buttons at the top and bottom of the white swatch. Either one can be tapped and they work in sync.
- User likes are stored in the crayon model and are correctly displayed. Hearts do not appear for items that have not been liked, even if the cell is re-used. Hearts appear for every item that has been liked, even when the cell has scrolled off-screen and back.

In your solution, it is essential that you follow best practices and produce clean and professional results. Schedule time to review, refine, and assess your work and perform basic professional polishing including spell-checking and grammar-checking on your work. It is better to submit a challenge that meets MVP than one that attempts to much and does not.

Validate your work through testing and ensure that your code operates as designed.

Stretch Problems

After finishing your required elements, you can push your work further. These goals may or may not be things you have learned in this module but they build on the material you just studied. Time allowing, stretch your limits and see if you can deliver on the following optional goals:

- Add any or all of the color information lines to your detail view. You retrieve RGB color channels by calling:

```
var (r, g, b): (CGFloat, CGFloat, CGFloat) = (0, 0, 0)
crayon.color.getRed(&r, green: &g, blue: &b, alpha: nil)
```

- Adjust your RGB floating point values to display only 3 digits at a time.
- Present RGB integer values ranging between 0 and 255
- Present the RGB integer values as a combined hex value
- Add Sprite Kit emitter nodes to sparkle up the presentation, either randomly in the cells, or in the detail view. Make sure your particles are color coded!
- Replace the background colors of the two like buttons with contrasting colors found 180 degrees around the color wheel so their hues are complementary.
- Add a random animated character into the color swatch in the middle of the detail screen that plays over the background color. You'll want to use a larger swatch than the one in the demo GIF.

Progress Dashboard

Download checked items