Stacks and Lists

SwiftUI Homework

What we are going to build

In this homework, we are going to build a SwiftUI app similar to the one shown in the first 13 minutes of this awesome WWDC talk: Introduction to SwiftUI. In the end, you will have a fully-functional app that displays a list of your favorite items. The next homework will continue to build on top of this homework, so remember to keep it around after we complete it. Let's get started!

Git and GitHub

When building the app for the homework, remember to practice using Git and GitHub:

- Create Git repository when you create the Xcode project
- Commit changes you've made to your project
- Push your project to GitHub, so our graders can download it at a later time

If you need a refresher, check out the homework guide: Git and GitHub with Xcode.

Introduction to SwiftUI

In the first 13 minutes of this talk: Introduction to SwiftUI, we are going to learn to display some delicious sandwiches in a list using SwiftUI. For this homework, we will practice building a similar app! Take some time to think about what you'd want to display in the app. You can build this app to display sandwiches, or to display any other items that interest you!

Here are some ideas:

- a list of your favorite colors
- a list of your Instagram friends
- a list of spells in Harry Potter
- a list of Apple Products
- a list of your favorite makeup brands
- a list of your favorite cars
- a list of your favorite TV shows

  ... and many more!

The possibilities are endless. Take some time to think about what you'd like to display in the app!

Data Model

First, create a type that represents the thing you want to display. Put this type in its own file.

Here's the `Sandwich` type in Sandwich.swift from the WWDC talk. What type make sense for your app? What properties should it have? Should it conform to any protocols?

```swift
// Sandwich.swift

import Foundation

struct Sandwich: Identifiable {
    var id = UUID()
    var name: String
    var ingredientCount: Int
    var isSpicy: Bool = false

    var imageName: String { return name }
    var thumbnailName: String { return name + "Thumb" }
}

let testData = [
    Sandwich(name: "Club", ingredientCount: 4, isSpicy: false),
    Sandwich(name: "Pastrami on rye", ingredientCount: 4, isSpicy: true),
    Sandwich(name: "French dip", ingredientCount: 3, isSpicy: false),
    Sandwich(name: "Bánh mì", ingredientCount: 5, isSpicy: true),
    Sandwich(name: "Ice cream sandwich", ingredientCount: 2, isSpicy: false),
    Sandwich(name: "Croque madame", ingredientCount: 4, isSpicy: false),
    Sandwich(name: "Hot dog", ingredientCount: 2, isSpicy: true),
    Sandwich(name: "Fluffernutter", ingredientCount: 2, isSpicy: false),
    Sandwich(name: "Avocado toast", ingredientCount: 3, isSpicy: true),
    Sandwich(name: "Gua bao", ingredientCount: 4, isSpicy: true),
]

```

Definition of Sandwich

> **Note**
>
> Often, having a testData array handy when building your app can help you quickly visualize and debug your app.

Cell View

To display our favorite items in a list, each item should display its information in its own "cell". Go ahead and create a new SwiftUI View using File->New->File..., and select SwiftUI View.

What properties should your cell have? Each cell should display the name of the item, along with any other information that you'd like to show for each item.

When you think your cell view is ready, use SwiftUI Previews to see your view in action.

> **Experiment**
>
> Add some modifiers in your cell view to change the font, font colors, and paddings.
>
> What do you see? Do you like your adjustments? Why or why not?
>
> Which texts should use larger font and primary color? Which texts should use smaller font and secondary colors? Why?

> **Note**
>
> Check out the Human Interface Guidelines for recommendations on designing good user interface.

List

Next, go ahead and create a new SwiftUI View. This View should display all your favorite items in a List.

> **Experiment**
>
> Use Live Preview to run the app live. Can you scroll the List?
>
> What happens if you change the List into a VStack around a ForEach? Can you scroll it now? Why do you think that is?

Add a Title

Try building and running your app in the simulator. If needed, modify your code to show the list in the simulator. We are now running a fully-functional app on an iOS device!

To finish up, go ahead and add a navigation title to our views.

> **Important**
>
> Apply the .navigationTitle() modifier to the List, and not to the NavigationView.

Resume Previews to see that the title is showing up correctly.

> **Experiment**
>
> What happens if we apply the .navigationTitle() to the NavigationView instead? What do you see?

Summary

Congratulations! You've completed this homework, which means that you have built a fully-functional app in SwiftUI! This is a big achievement, yet notice how little code we need to write in order to make this all work. Take a moment to reflect on the skills you've practiced in this homework. What did you like? What did you find frustrating or confusing? Why do you think that is?

Apple announces improvements and additions to SwiftUI at each year's WWDC. If you are interested in learning more, take a moment to explore what SwiftUI has to offer through Apple's SwiftUI Tutorial.

Grades breakdown

Basic breakdown:

- (+20%) Correctly uses Git and GitHub for homework submission
- (+20%) Data model implemented correctly
- (+20%) Cell view displays correctly
- (+20%) List view and title displays correctly
- (+10%) Homework submission contains demo video for graded requirements

Additional Notes:

- (+10% extra credit) if cell view contains custom images
- (-2% each) each type of warning or error shown in Navigator and Debug Console
- Additionally, if submission contains an extra credit demo video, assign extra credits correspondingly

Extra Credits

Here are some small ideas to improve our app further (2% each):

- If you have a color scheme that you'd like to apply to the app, can we apply these colors to UI elements in a consistent and aesthetic manner?
- Can we use Previews to verify that our app looks right in both dark mode and light mode?
- Can we use Previews to verify that our app looks right when the user sets a larger or smaller font size?
- If your items can be filtered into different categories, can we create sections in the list for each category?
- Can we make our app more accessible to everyone by using the accessibility APIs in SwiftUI?
- Can we verify that our app works correctly in VoiceOver?

Here are some more complex ideas to improve our app further (10% each):

- Can we use localization tools with Xcode to make our app available in other languages?
- Can we verify that our app looks right when running on an iPad? Is there anything we need to modify or change?
- Can we adapt our app so that it looks right when running on an Apple Watch?
- Can we adapt our app so that it looks right when running on a Mac?
- If your item contains numbers, dates, or different measurements, can we display these information using the new formatters in Foundation?