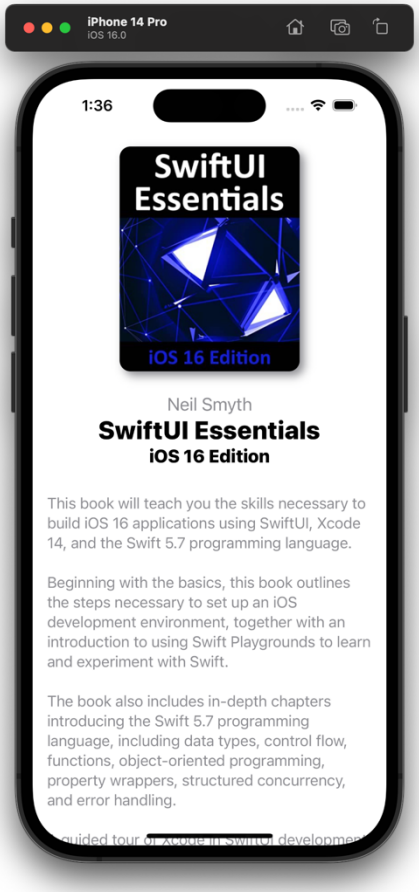
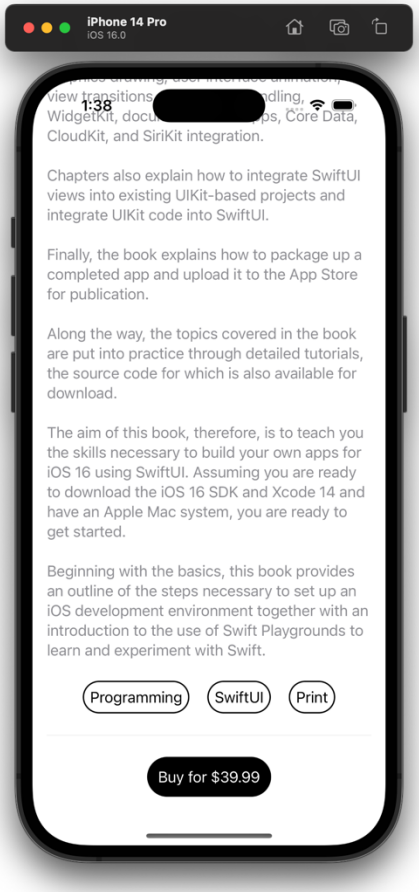
Assignment 1 – 100 points

Use what we've learned over the last few classes to design a single-screen user interface in SwiftUI. The SwiftUI components required for this assignment will include `Image` views, `Text` views, `Shape` views, vertical and horizontal stacks, `ScrollView`s, `Divider`s and a number of different view modifiers.

The User Interface

Your job in this assignment is to try to recreate the user interface shown in the images below:





The images show the UI running on the iPhone 14 Pro simulator. As you can see, there will not be sufficient vertical space to display all of the required elements onscreen at once, but we can ensure that the app will still work by wrapping the UI elements in a `ScrollView`.

The App

Create a new project for a SwiftUI app named `Books`. Most of the coding for this assignment will involve creating your UI in `ContentView.swift`, but there are a couple of other things that you should do first.

1. Defining the Model

Create a new Swift file in your project called `BookModel.swift`. In the file, define a model structure called `Book` that conforms to the protocol `Identifiable`. Give it the following properties (in the order listed):

- A property named `id` that should be initialized to `UUID()`.
- A `String` property named `image`.
- A `String` property named `authors`.
- A `String` property named `title`.
- A `String` property named `edition`.
- A `String` property named `description`.
- An array of `String` property named `categories`. Your program code should assume that this array will contain one to three strings.
- A `Double` number property named `price`.

This is kind of overkill for this assignment, but it's typical of apps that we will write later in the semester.

2. Defining Some Data

Download the resources for Assignment 1, unzip the archive, and drag (or add) the file `BookData.swift` and into your project's bundle. Make sure that the checkboxes for "Destination:" / "Copy items if needed" and "Add to targets:" are checked (they should be by default, but check them if they are not). This will provide you with some hard-coded data that can be displayed in your UI.

This is a fast-and-dirty way of embedding some data in our app; however, it is not a particularly good *way to do it. We will learn better techniques for dealing with data as the semester progresses.*

To access the hard-coded data from `ContentView.Swift`, declare the following property as part of the `ContentView` structure:

```
let book = bookData
```

The `Book` object `book` will contain all the data that you will need to display in your view. You can access the individual properties of the object using standard dot notation, e.g. `book.title`, `book.image`, `book.authors`, etc.

3. Adding the Image Assets

The resources folder for this assignment also includes two images. Drag `cover.jpg` into the `Assets.xcassets` asset catalog so that it is added as a new Image Set. Add the 1024x1024 app icon provided (`books-1024.jpg`) to the AppIcon entry in the asset catalog. Note that with Xcode 14, you no longer need to provide more than a single size app icon.

Hints

- A good frame size for the cover image is 200 x 250. The image in the screenshots above is displayed with rounded corners and a drop shadow.

- Rather than hard-coding three category tags in a horizontal stack, use a `ForEach` loop to generate the appropriate category tags based on the number of elements in the `categories` array. That way your code will work for a book with one or two category tags, not just three.

- As shown, the `ScrollView` does not have visible scroll indicators.

- The “Buy for <price>” button can simply be a `Text` view with no button functionality, since we’re not attempting to implement purchasing a book in this assignment. Use a `NumberFormatter` to format the book’s price.

- You are not required to recreate the UI perfectly in terms of fonts, spacing, etc. Just do your best to make it look similar and ensure that all of the required UI elements are present.

- When submitting your assignment, make sure that you compress and upload the top-level app folder and **all** of the sub-folders and files that it contains. Submitting a single file like `Books.xcodeproj` or one of the app’s sub-folders is not sufficient – if you do so, the TA will not be able to grade your assignment.