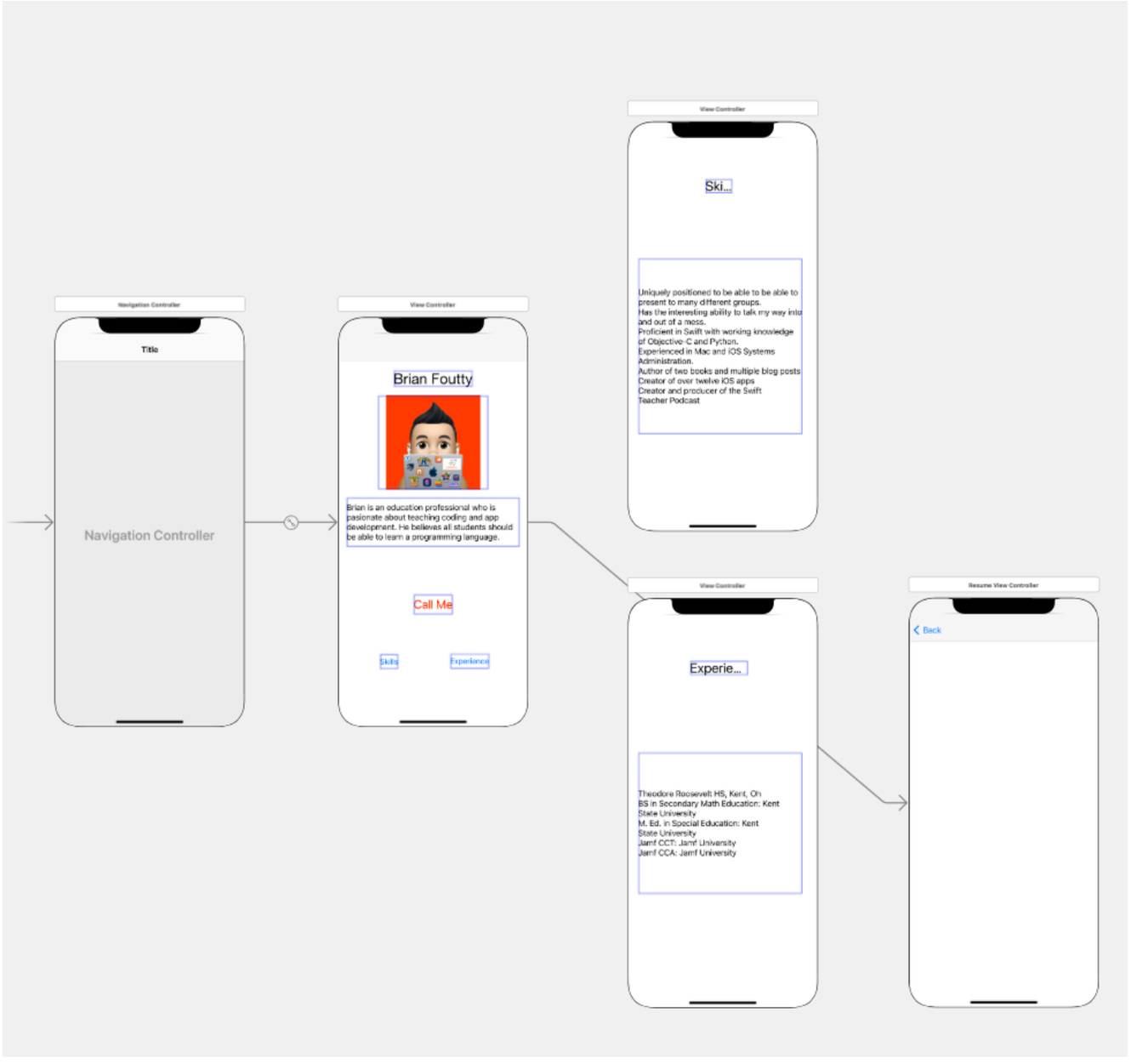
Brian Foutty

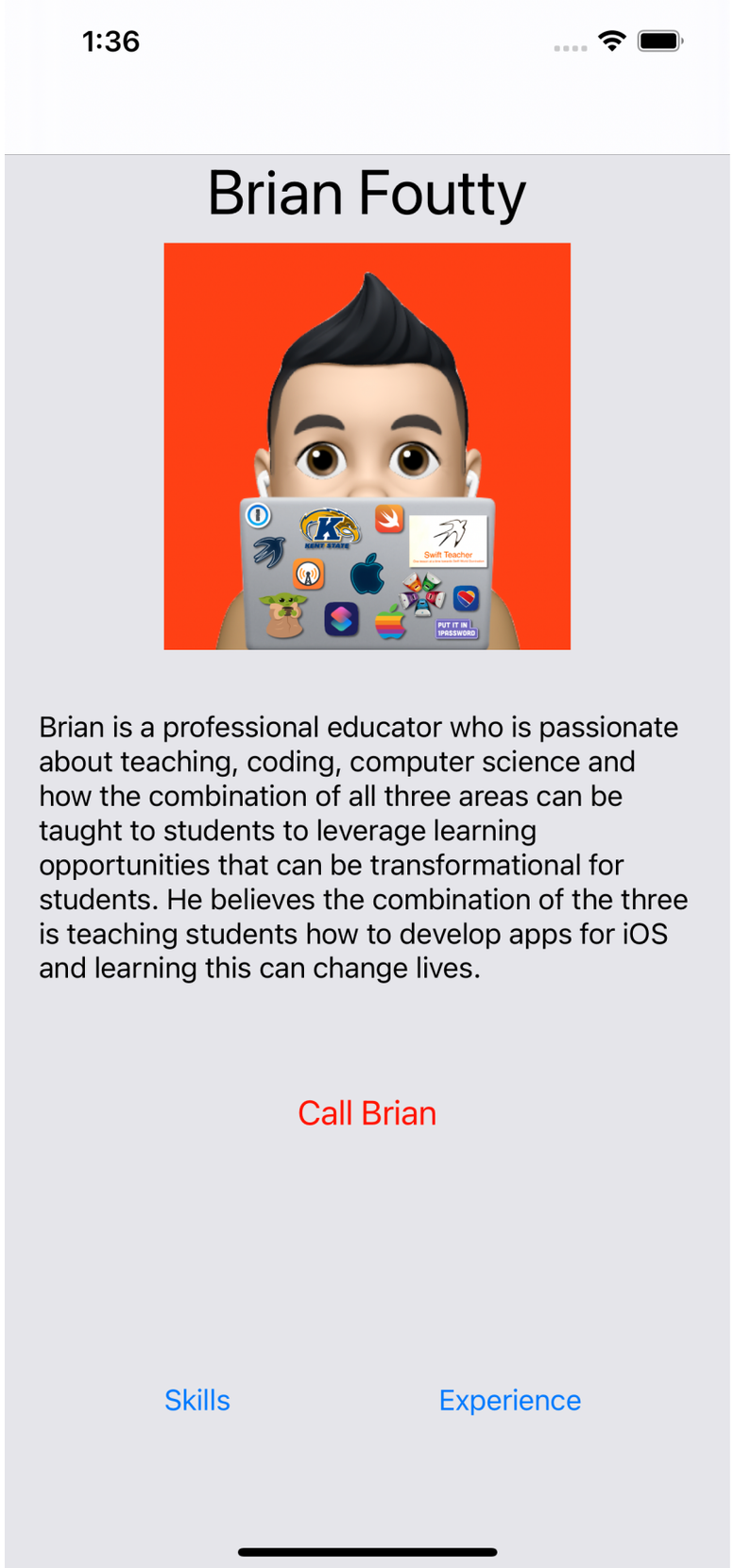
MyResume app project



The Project

Activate, Explore, Apply

This app is an extension of the Hello app project created a part of Apple's Develop in Swift Fundamentals Teacher and Student course. The project can be found in Lesson 2.8: Displaying Data starting on page 368 in the Student book and starting on page 253 in the student book. The general purpose of this project is to review and practice using Optionals in code, creating segues, and making multiple screen apps. Additionally, this project provides me another chance to teach my students to the process of software improvement and iteration by taking some software they created and adding functionality to it. In doing this I am also able to introduce the concepts of software versioning and using GitHub for software creation, versioning, and the proper (and safe) way to iterate on software using branching and merging. If you are going to do this project individually or with your students, I am going to make the assumption that you have access to Mac hardware and, just as important, Apple first-party software such as Keynote, Preview, and Pages.



The general process and flow of this project is in the format of an Apple Teacher Portfolio lesson. There are three parts/ phases to the lesson:

1. Activate
2. Explore
3. Apply

Activate

We want our students to activate any prior knowledge on the topic. Since the students have previously made the Hello app, they will all have the starting point of the project. However, the goal of the project is to greatly improve the app and add more advanced iOS features. As a starting point, students should have a resumé made when this part is complete. Here is the activity I use with my students:

1. Open Pages and create a blank document. Create bulleted list of your experiences that best highlight why someone would want to employ you. Add a separate bulleted list of your skills that best highlight why someone would want to employ you.
2. In Pages, create a new document and open multiple resume templates.
3. Choose a template and create a resumé.
4. Open and review the Hello app.
5. Students should now think, pair, share ways they could incorporate parts of their resumé into the Hello app to better introduce themselves to others and potential employers. Students will do this in a whiteboard session as it is done when discussing app features and code reviews.

Explore

Now that students are ready to put their best foot forward and market themselves to a prospective employer we want them to create a web presence in the form of a personal/professional website, online portfolio, or blog so that we can incorporate it into the app.

1. Introduce students to some of the free website/portfolio sites: Wix, Weebly, SquareSpace, Tumblr, or students may build their own.
2. Decide upon what skills, talents, knowledge they want to showcase on their website.
3. Have students build an introductory website that can be expanded to include learning and professional skills.
4. Review in Apple's Developer documentation what can be done to include both a traditional paper resumé and a web-based resumé/website/portfolio.

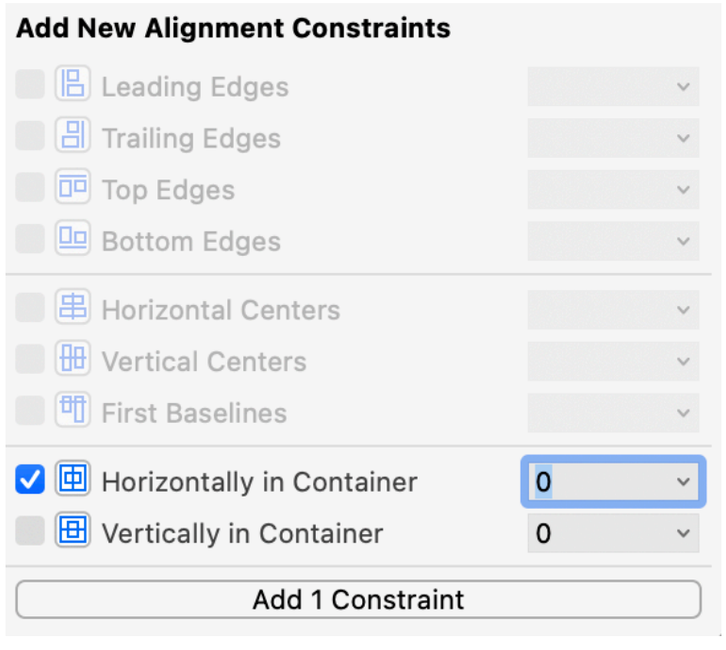
Apply

In this part of the project we build the app. You can find the GitHub repo for this project here:

MyResume GitHub Repo

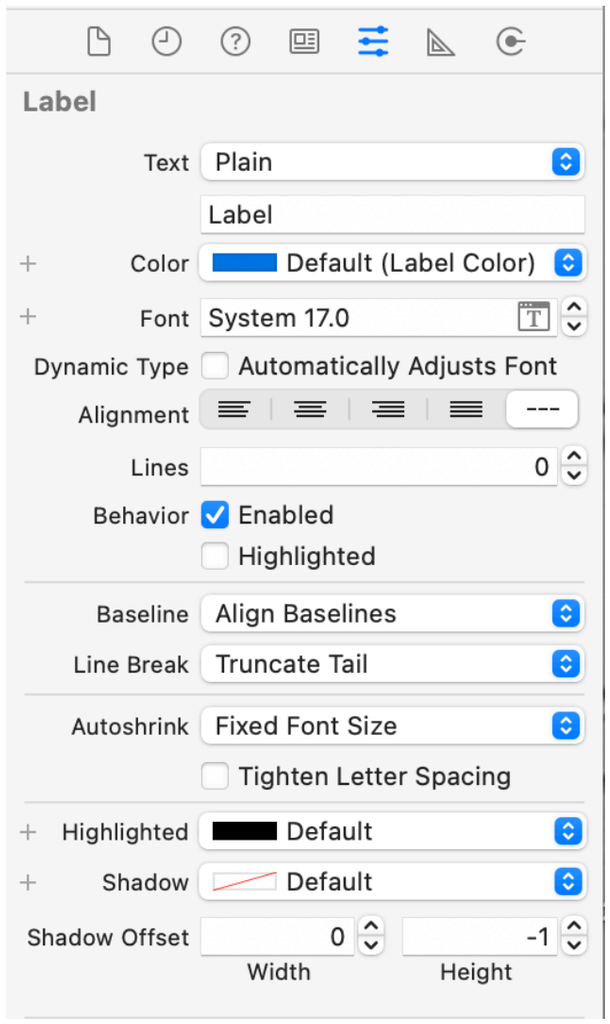
Hello app set up screencast

1. Open the existing Hello app
2. Add a button to the screen that is centered and just under your introduction label. Set the size to 25.
3. Add two buttons under the Call Me button. One button should be “Experience” and one should be “Skills”. Set the font size to 17 for each button.
4. Set the font color for each button if you want to change from the default blue color.
5. Click and drag to select the name label, your picture, and introduction label. In the Menu Bar click Editor > Resolve Auto Layout Issues > Add Missing Constraints.
6. Select the Call Me button. Click Add New Alignment Constraints. Select “Center Horizontally in Container” and click “Add 1 Constraint”.



7. Click and drag to select the Skills label and the Education label. In the Menu Bar click Editor > Resolve Auto Layout Issues > Add Missing Constraints.
8. From the Object Library add two View Controllers in Main Storyboard.
9. Add a label to the top of each of the new View Controllers. Use the alignment guides to center the label near the top of the screen.
10. Change one label to **Experience** and the other label to **Skills** and set the font to be Title 1 for each label.
11. Create a “Show” segue from the Experience button to the View Controller with the Experience label.
12. Create a “Show” segue from the Skills button to the View Controller with the Skills label.
13. Add a label to the the Experience screen and to the Skills screen. For each label, select the label and use the drag handles to stretch the label to the leading and trailing edge of the safe edges of the screen. Now, use the top and bottom drag handles to make label cover about half of the screen. Use the alignment guides to center the label horizontally and vertically.

14. Set the number of lines for each label to zero in the Attributes Inspector.



15. Open the Pages document with the bulleted lists for experiences and skills. Copy the experiences bulleted list and paste it into the label on the Experience screen. Adjust the label size and position to display the text. Now, do the same for the skills bulleted list.

16. Build and run the app. Students will be able to navigate to either the Experience or Skills screen, but then... Zoinks!! There is no way to return to the Home Screen of the app.
17. Ask students for suggestions. You have two main options.
    1. Add a button to the screen, change the text to Back in the Attributes inspector, create a backToHomeScreen function in ViewController.swift using the UIStoryBoardSegue as its only parameter as was done in Lesson 3.6 and explained on page 412 in *Develop in Swift Fundamentals*. This option is a lot of steps.
    2. Embed the Home Screen in a Navigation Controller - this is method I use with my students.
18. The Navigation Controller on the Home Screen will now cover the name label from the original project. We will later set the title in the navigation bar to be your name programmatically.
19. Build and run. You can now navigate to the Experience and Skills screens and then return back to the app Home Screen.

Steps 2 - 19 screencast

20. Open Main.storyboard. We now want to create an action that will allow the phone to call our number.
21. Open the Assistant Editor.
22. Create an action from the Call Me button by control-dragging from the Call Me button to just under the viewDidLoad method. Name the function something similar to **`callMe`** or **`callMyName`** (where MyName is your name).
23. Create a phoneNumber constant at the top of the class and set it as a String to your phone number.
24. Add this code to the callMe method.

```
// this enables a physical device to call our phone number
if let url = NSURL(string: "tel://\(phoneNumber)") {
    UIApplication.shared.open(url as URL, options: [:], completionHandler: nil)

    // since the Simulator cannot make phone call. This line will print a string
        to the console to confirm the code executed.
    print("The phone is calling my number. Voicemail would be triggered
        automatically if this were a physical device.")
}
```

[Steps 20 - 24 screencast]

25. How can we make it better?
    1. Have the Skills button display our personal webpage, blog, portfolio, etc.
    2. Have the Experience page display our actual resume.

26. Displaying our webpage, blog, or portfolio
    1. Go to the Source Control Navigator and add a new branch and name it "**webportfolio**"
    2. Break the connection from the Skills button to the Skills screen by clicking on the segue that connects the Home Screen to the Skills screen and then delete that arrow.
    3. At the very top of ViewController.swift file above the import UIKit statement, add this statement: **`import SafariServices`**.

```
import SafariServices
import UIKit
```

27. At the top of ViewController class right after the UIViewController declaration add this:

```
class ViewController: UIViewController, SFSafariViewControllerDelegate {
```

28. Create a new function called **`showWebPage()`** and add this code to the function (you should use your website address):

```
func showWebPage() {
    let urlString = "https://www.swiftteacher.org"

    if let url = URL(string: urlString) {
        let config = SFSafariViewController.Configuration()
        config.entersReaderIfAvailable = true

        let view = SFSafariViewController(url: url, configuration:
            config)
        view.delegate = self
        present(view, animated: true)
    }
}
```

29. Open Main.storyboard
    1. Open the Assistant Editor
    2. Connect the skills button to an action to open the webpage by control-dragging from the **Skills** button to under the **`callMyName`** function which is under the **`viewDidLoad()`** method. Name the action **`skillsButtonTapped`**.

```
@IBAction func skillsButtonTapped(_ sender: Any) {

}
```

    3. Call **`showWebPage()`** inside of the **`skillsButtonTapped`** function.

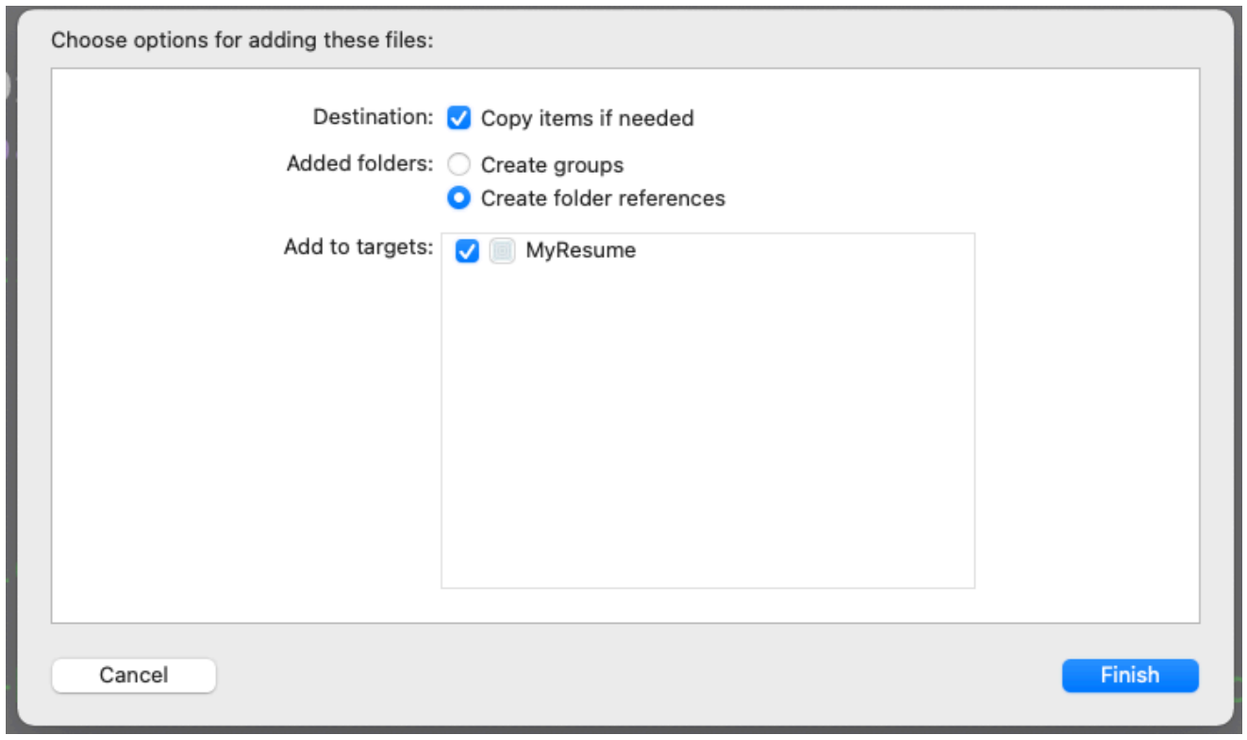
```
@IBAction func skillsButtonTapped(_ sender: Any) {
    showWebPage()
}
```

30. Build and run. Tap the Skills button to verify that the correct webpage loads.
31. If everything works as expected:
    1. Commit the changes and create the remote branch in the process.
    2. Right-click the "**Main**" branch and click the "**Checkout**" button.
    3. Right-click on the "**webportfolio**" branch and choose "Merge "**webportfolio**" into "**main**"". Then click the "**Merge**" button.
    4. In the Menu bar, click **Source Control > Push**. This will send the changes just merged into the main branch to the main branch on GitHub or your chosen repository hosting service.

Steps 26 - 31 screencast

32. How can we make it better?
    1. Have the Education button load a pdf version resume we previously made in the Activate lesson.
    2. Add the share button to the pdf so that any prospective employer can email or message the resume to themselves.

33. Loading a PDF Resume from the apps internal storage and add the share button.
    1. Go to the Source Control Navigator and add a new branch and name it "**resume**".
    2. Open Pages and export your resume from Pages as a PDF.
    3. Add your resume to your project by dragging it into Project Navigator under the *assets.xcassets* folder. "**Copy files if needed**" and "**Folder references**" must be selected as show below.



    4. Break the connection from the Experience button to the Experience screen by clicking on the segue that connects the Home Screen to the Experience screen and then delete that arrow.
    5. Open the Object Library add a View Controller in Main Storyboard.
    6. Create a "**Show**" segue from the Experience button to the new View Controller.
    7. Add a new Cocoa Touch Class file to your project. **File > New > File > Cocoa Touch Class** and click "**Next**". Set your new file to be a subclass of UIViewController and name your file **ResumeViewController** and click "**Create**".

8. Switch back to Main.storyboard. Select your newest View Controller. Click the Identity Inspector in the Inspector Pane. In the top section, Custom Class, start typing Resume and then hit return once autocomplete fills in the name ResumeViewController (you will probably only have to type “Re” and then you will see ResumeViewController in the box.
9. At the very top of ResumeViewController.swift file above the import UIKit statement, add this statement:

```
6  //
7  import PDFKit
8  import UIKit
```

34. At the very top of ResumeViewController.swift file above the import UIKit statement, add this statement:

```
let resume = Bundle.main.url(forResource: "MyResume_project_example",
    withExtension: "pdf")
```

35. Add this code to the **`viewDidLoad()`** just under **`super.ViewDidLoad()`**:

```
let pdfView = PDFView()
```

36. Add this code to the bottom of **`viewDidLoad()`**:

```
pdfView.translatesAutoresizingMaskIntoConstraints = false
view.addSubview(pdfView)



pdfView.leadingAnchor.constraint(equalTo:
    view.safeAreaLayoutGuide.leadingAnchor).isActive = true
pdfView.trailingAnchor.constraint(equalTo:
    view.safeAreaLayoutGuide.trailingAnchor).isActive = true
pdfView.topAnchor.constraint(equalTo:
    view.safeAreaLayoutGuide.topAnchor).isActive = true
pdfView.bottomAnchor.constraint(equalTo:
    view.safeAreaLayoutGuide.bottomAnchor).isActive = true
```

37. Add this code to the bottom of **`viewDidLoad()`**:

```
guard let path = Bundle.main.url(forResource:
    "MyResume_project_example", withExtension: "pdf") else { return }

if let document = PDFDocument(url: path) {
    pdfView.document = document
}
```

38. Build and run. Tap the Experience button to ensure that your resume loads and is viewable. We are going to add the share button programmatically. To do this programmatically we are going to use Objective-C functions. In Swift we can do this by bridging to Objective-C and using the **`@objc`** keyword before the **`func`** keyword. Create an Objective-C Swift function called **`shareTapped()`** by using this code:

```
@objc func shareTapped() {

    }
```

39. Add this code to the **`shareTapped()`** function:

```
guard let document = resume else { return }

let message = "This is Brian Foutty's resume. It looks great. Doesn't
    it make you want to offer him the job."



let view = UIActivityViewController(activityItems: [document,
    message], applicationActivities: [])

view.popoverPresentationController?.barButtonItem =
    navigationItem.rightBarButtonItem

present(view, animated: true)
```

40. Now that we have created the function that will give the share button its functionality. We will create the share button in code. Add this code towards the top of the **`viewDidLoad()`** right under the **`pdfView`** constant:

```
// adds the share button
navigationItem.rightBarButtonItem = UIBarButtonItem(barButtonSystemItem: .action, target: self,
    action: #selector(shareTapped))
```

41. One last finishing touch. We can programmatically add a title to the Navigation Bar so that reinforces who we are so that our name resonates with the interviewer:

```
title = "Enter your name here"
navigationController?.navigationBar.prefersLargeTitles = false
```

42. Build and run. Tap the Experience button, ensure your resume loads, tap the share button to determine the share button functions as it should.
43. If you are happy with the updated version of the Skills screen:
    1. Commit the changes to the resume branch.
    2. Right-click the "**Main**" branch and choose "**Checkout**". Click the "**Checkout**" button.
    3. Right-click on the "**resume**" branch and choose "**Merge** "**resume**" into "**main**". Then click the "**Merge**" button.
    4. In the Menu bar, click **Source Control > Push**. This will send the changes just merged into the main branch to the main branch on GitHub or your chosen repository hosting service.
44. This updated version of the app is now complete. Congratulations!!

Steps 33 - 43 Screencast

Hello GitHub Repo