Your First App - Follow this Step By Step

Links:

1. Preface
2. Setting up your Environment
   1. Github
      1. Create an account
         1. <https://github.com/join>
      2. Create a repository A screenshot of a cell phone

         Description automatically generated
         1. Follow steps in the new repo. You can make it private if you want no one to access it, however, many, if not most, repositories are Public.A screenshot of a cell phone

            Description automatically generated
            1. Do not initialize a README. We will do this later via Terminal.
   2. Now, go to Xcode
      1. Create project following screenshotsA screenshot of a cell phone

         Description automatically generated
   3. A screenshot of a computer

      Description automatically generatedA screenshot of a social media post

      Description automatically generatedA screenshot of a social media post

      Description automatically generated
      1. Be sure to note what file location you saved it in. You need to be able to locate it via “Terminal” or “the command line” in the next step
      2. Now you need to set up your Github in Xcode. (Xcode calls it source control sometimes). First, click on **Preferences** in the dropdown in the upper left of your screen.

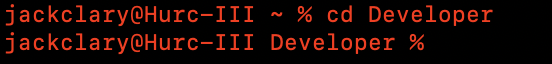
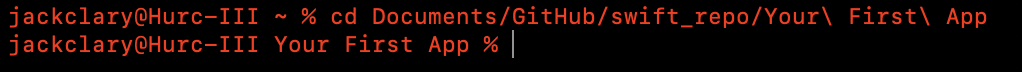
A screenshot of a cell phone

Description automatically generated

* + 1. Now, click “Source Control” in the upper center and make sure you have these boxes checked.A screenshot of a cell phone

       Description automatically generated
    2. Lastly click the little “Git” and type in your information
    3. A screenshot of a cell phone

       Description automatically generated
    4. You will have to type your Github account information later, but it can/will happen a few different times so just have it at the ready.
  1. Now, open Terminal
     1. “**ls**” shows current directory location and its contentsA close up of a logo

        Description automatically generated
     2. “**cd** ‘**filename’**” moves you to said file or directory
        1. As you can see, my “directory” changed from “~” to “Developer”
        2. If you type in “**cd** **..**” it will bring you back to the previous directory.
        3. If you type in “**cd**” it will bring you back to the “~” directory
     3. So, cd to the directory you have your Xcode project saved in
        1. If the title of the directory is unique, you can press the **tab** **key** after you’ve typed in sufficient letters to auto-type the rest. This is useful when there are spaces like in “Your First …” because they require “\ /” in terminal.
     4. Use the following commands in terminal when you are in your directory. You know you are in your directory when it says the correct name to the left of the “%” (E.g. “Your First App” in my case)A screenshot of a cell phone

        Description automatically generated
        1. The link, i.e. <https://github.com/jaclary/test.git> will be different for you based on your github username and the name of the repository. This link can be found if you look at your browser’s search bar when you are on your repository page which will load after you create it in step 2.a.ii.1

1. Your First App Intro - Users can sign-up for your service
   1. After you start the project + initializing your repository
      1. Creating a .gitignore
         1. **“cd”** to the repository location in terminal
         2. “**touch** **.gitignore**”
         3. “**open** **.gitignore**”
         4. This is basically a text file where each line represents a file or files you don’t want being put in your repository.
      2. What to type in it?A screenshot of a cell phone

         Description automatically generated
         1. \*.xcuserstate
         2. \*.xcuserdata
         3. \*.DS\_Store
         4. The “\*” at the beginning means all files that contain the specified string.
      3. How to github?
         1. Read this: <http://think-like-a-git.net>
         2. Terminal vs. Xcode vs. Other Applications
            1. Terminal is more difficult, but more powerful. Look here for a tutorial: <https://rogerdudler.github.io/git-guide/>
            2. Xcode is more simple, but prone to bugs/errors.
            3. Other applications like Github desktop and GitKraken are great, simple ways to deal with source control. For simplicities sake, download Github Desktop and use it.
            4. <https://desktop.github.com>
         3. GitHub Desktop
            1. Follow the instructions when you first download it. Very simple.
            2. For using it,A screenshot of a cell phone

               Description automatically generated
            3. The blue represents the current repository.
            4. The red is the current branch (
         4. Pull Requests
            1. FLESH OUT
   2. How do you Xcode?
      1. “cmd+r” runs your app in a virtual apple device
      2. “cmd+b” builds the app, but doesn’t run it.
      3. Preview
      4. FLESH OUT
   3. What are these initial files?
      1. AppDelegate
         1. Runs when your app starts, specifically the first function of the class runs (func application(\_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool)
         2. More information: <https://dev.to/theobendixson/comment/4g8j>
      2. SceneDelegate
         1. Read here (also gives good information on AppDelegate): <https://www.donnywals.com/understanding-the-ios-13-scene-delegate/>
      3. ContentView
         1. ContentView is your firs View, the technical word for a screen on your app. It is initialized, or started, as a empty view with the words “Hello World” in the center of the screen.
      4. Assets.xcassets
         1. This is a folder that will hold assets like images. We won’t go into detail here.
      5. LaunchScreen.storyboard
         1. This is the “splash screen” or the window that briefly appears as the app loads.
      6. Info.plist
         1. This topic has a lot involved. We will discuss this when we add dependencies to the app.
   4. SignUpView
      1. What are views in SwiftUI?
         1. <https://developer.apple.com/documentation/swiftui/view>
         2. VStack
         3. HStack
         4. Group
         5. Text
         6. TextField
      2. @State
         1. Storing user data
         2. Create a user class? (maybe throw all signed up users in a list that displays after they sign
      3. NavigationView?
   5. SignUpViewModel
2. Let’s code!
   1. Start by opening the SignUpViewTest.swift. You should see the below.A screenshot of a cell phone

      Description automatically generated
   2. To showcase how the “body” section of the page works, try duplicating line 14. Do you get the following error?A screenshot of a cell phone

      Description automatically generated
      1. You get this error because the body needs to return 1 thing (it does so implicitly, hence no “return Text(“Hello World”)), and now it is returning 2 things. But what if you want it to say Hello World twice? Stacks!
   3. If you wrap the two Text() in a stack, like vertical stack, or VStack, you should get the followingA screenshot of a cell phone

      Description automatically generated
      1. See how they’re vertically aligned? If you wanted them horizontal, you can use a horizontal stack or HStack. A screenshot of a cell phone

         Description automatically generated
         1. As a side note, if you are running the preview (option + cmd + return and then option + cmd + p) and you click on a section of the code, like the HStack here, the preview will highlight that portion of the screen.
   4. Now let’s begin the real fun. We like to define features as User Stories because they allow us to be flexible in the way we implement our solutions. For this story, lets define it like this:
      1. Our user opens the app and can type in their name.
   5. To do this we are going to embed or wrap our text, that tells the user what to do, with a TextField, that allows the user to type in information. FLESH OUTA screenshot of a cell phone

      Description automatically generated
   6. Now that we can type in a name, lets present that name back to the user. Since we stored the name in the variable aptly titled “name” we can display it like this:A screenshot of a cell phone

      Description automatically generated
   7. However, this looks terrible. To begin adding some design lets add a few more variables, things like email, and some styling.
      1. The styling we will add simply showcases some simple tools you have at your disposal like the alignment parameter in the VStack (and in the other stacks), fontWeight, font, foregroundColor, textFieldStyle, and padding.A screenshot of a cell phone

         Description automatically generated
   8. Now that we’ve done this for our first name, lets do it for all the new variables including last name, email, and password. A screenshot of a cell phone

      Description automatically generated
   9. As you can see, we HStack each textField so that the image appears in-line, or horizontally, with the textfield.
   10. Now that we can enter our information, let’s make it so that after we do, it appears at the bottom of the screen. To do this, we need to create a variable to tells us whether or not we should show that part of the screen. 
   11. Now, let’s attach that variable to a Button, so that when we press it, it shows the screen. A screenshot of a cell phone

       Description automatically generated
       1. Essentially what this does: creates a button where the action of pressing on it causes the showScreen variable to “toggle()” or switch from it’s current state of false to true (or vice versa).
   12. By pressing it the your information section will appear. A screenshot of a cell phone

       Description automatically generated