Project Title: Split

CIS 4900: Independent Study Project

Jeremy Hernandez

July 24, 2020

Proposed Idea:

Create a fitness tracking application that allows the users to keep track of their daily and weekly workout routines. The user will be allowed to input their various personal workout related information such as weight, age, gender, and type of workout/machine used. Using this information, the application will keep track of the user's workout progress and results. The application is tentatively planned to be created using Swift and Xcode.

How it works:

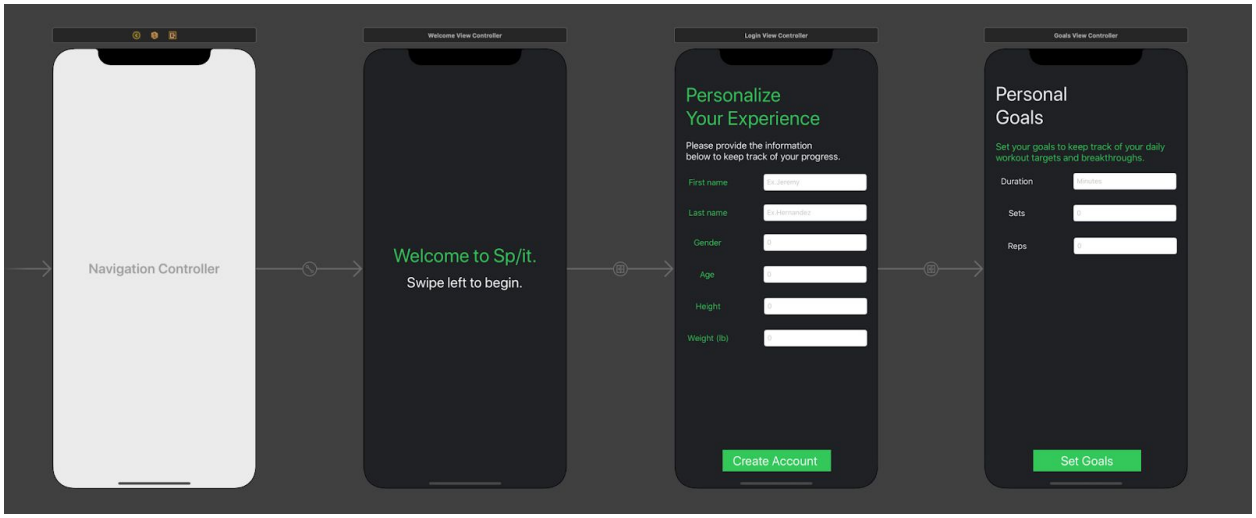
The program was created using Swift 5 and Xcode 11.5. The program is intended to help the user keep track of their weekly and daily workout routines and activities. It provides an intuitive user interface that is easy to understand and use. The application begins by welcoming the user and asking for their personal information to create their account and personalize their experience. It then asks for the user's desired daily workout goals such as desired duration of working time, the number of sets and number of reps. The program uses this information to check if the user has met their daily workout targets. Once the user has finished logging in. He is greeted with the homepage of the application. The homepage is one of the main five pages included in a tab bar. The home page is intended to be an informative page, where the user can learn more about the application, get exercise suggestions and learn about future updates. The second page in the application is the Exercises page. Within this page the user is allowed to scroll and select any one of the thirty plus exercises included with the application. The exercises are neatly presented inside a table and can also be searched by using the search bar located on the top of the table. Once the user has selected a desired exercise, he or she is redirected to the Add Exercises page where they are presented with an image and description of the selected exercise. Also located in this page is the option for the user to enter their workout entry. These

entries include date, duration, number of sets, and number of reps. If the user chooses to add an entry, he or she can view all their registered workouts in the Activity History page.

The Activity History or third page is composed of a table where all the registered exercises of the user are placed. In this page the user is presented with the dates of the days he or she has exercised. If the user taps on any one of those dates, the table will expand and display all the exercises that were registered on that specific date. It will also display the duration of set workout, and the number of sets and reps achieved. The fourth page in the tab bar is the Today at a Glance page. This page provides the user with a quick glance at their daily progress towards their registered workout goals. It is composed of three circular progress bars that show the user's progress towards their duration, sets and reps goals.

The fifth and final tab in the program is the Profile page. The Profile page will display the user's personal information as well as an infographic line chart that details the user's weight loss progress over time. The weight loss progress graph will update itself every time the user enters a current weight within the settings page. The settings page is located within the profile page in the upper right corner. By selecting the edit button, the user will be presented with the settings page. Within this page, the user will have the option to update their profile information, set new daily goals, update their weight and logout of the app. Any changes done in the setting will be reflected in the app. For example, entering a new weight will cause the weight loss progress chart to gain a new entry and updating the daily goals will cause to Today at a glance progress to change. Finally, if the user decides to select the logout button, he will be notified with the alert that if he or she decides to proceed, all his or her user's information will be deleted alongside their account. If they select yes, all their information is deleted for memory and the application restarts at the welcoming page.

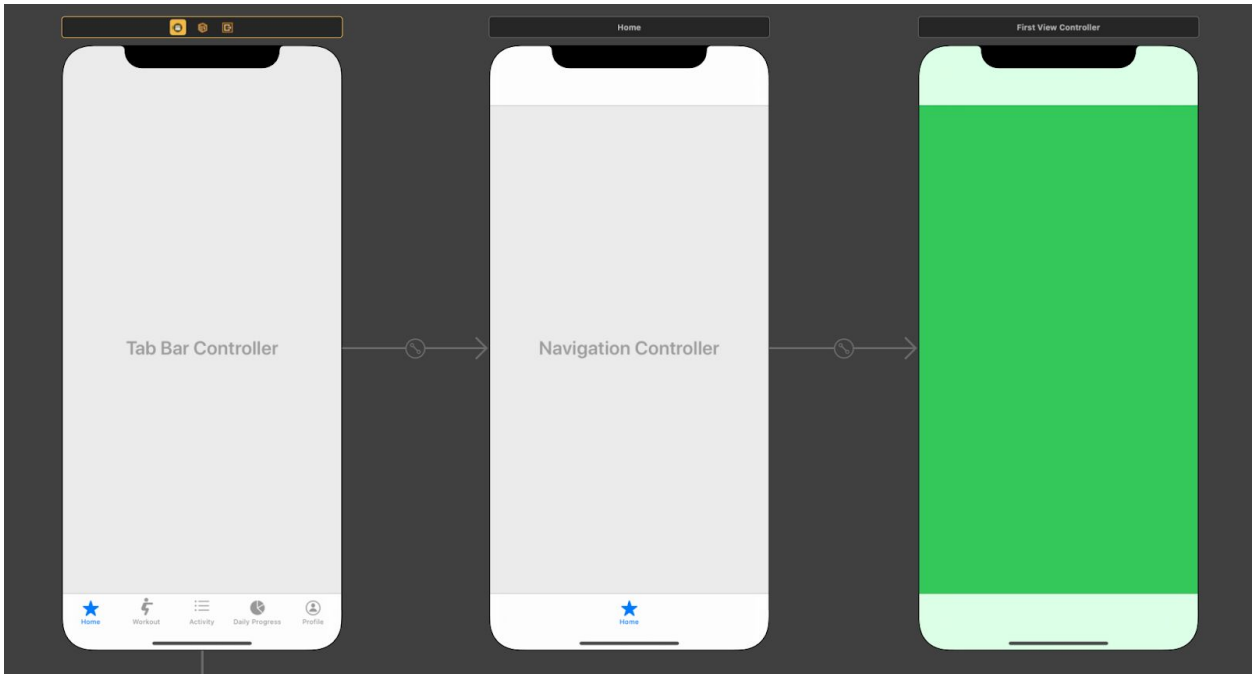
Detail description of components :



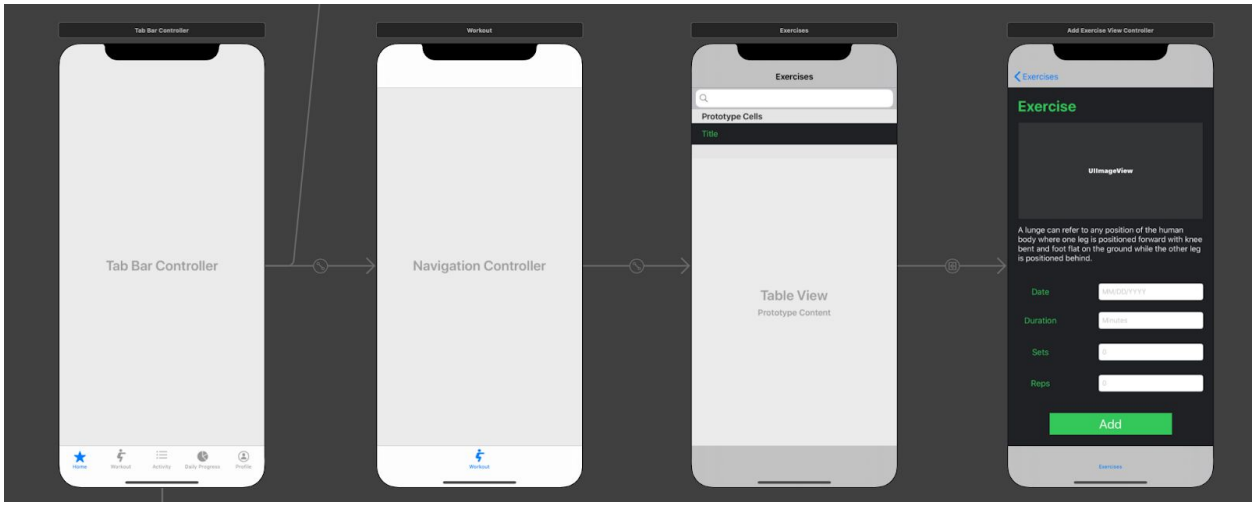
Welcome View Controller: This is a simple view controller intended to be the welcoming page of the application. This page checks whether or not the user is already logged in to the application. If not, it welcomes the new user and guides them to create an account. If the user is logged in, it will skip these steps and take the user to the firstViewController which is also known as the home page. It does this by instantiating the UITabBarController and then executing a push to the First View Controller. In order to check whether or not the user is logged in, the application checks a bool value store in UserDefaults. UserDefaults is described in a section below.

Login View Controller: The login View Controller is the page where the user get's to create his account for the application. It's composed of a series of UILabels that describe the set of fields the user can fill with his or her information. The user can input the information through UITextField outlets. These UITextField objects allow for string input and if combined with a UIPickerView, they allow the user to make a selection from a pre-existing dataset. These types of input are used to get the user's first/last name, height, gender, age and weight. Also located on this page is a UIButton, the create Account button which notifies the program when to save the user's input into persisted data and then directs the user to the Goals View. More precisely, this data is stored in a User object which is then stored in UserDefaults.The User class is described in a section below. Also stored in this controller is the loggin Bool value to determine if the user has created an account and logged in. This value is also stored in UserDefaults.

Goals View Controller: The Goals View Controller is sort of an extension of the login view controller. It is intended for the user to enter his daily goals for the amount of time spent exercising or duration, the number of sets, and reps completed. This page was also created by using UILabels and UITextfields. It is structured very similar to the previous page. Thus, Once the user has selected the set Goals button, the information provided will be saved into the User object stored in UserDefaults. Then the program instantiates the UITabBarController and redirects the user to the First View Controller.



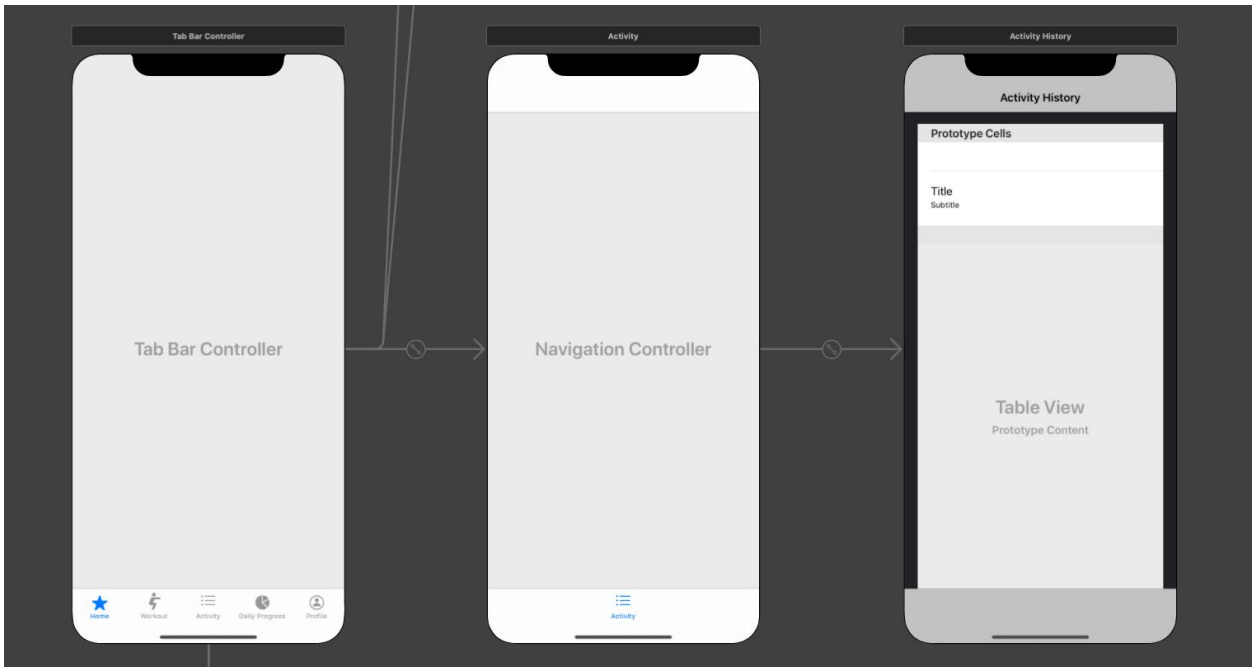
First View Controller: The first view controller is intended to be an informative page or homepage of the application. It is mainly composed of a CardSlider object and an array of items. This controller imports the CardSlider library in order to create the CardSlider object and display an array of the type Item in a card slider view. Item is a struct that defines the structure of each card slider item. These items are cards the user can slide thought and expand to view different information. This animation and interface was achieved by installing the CardSlider from Cocoapods.



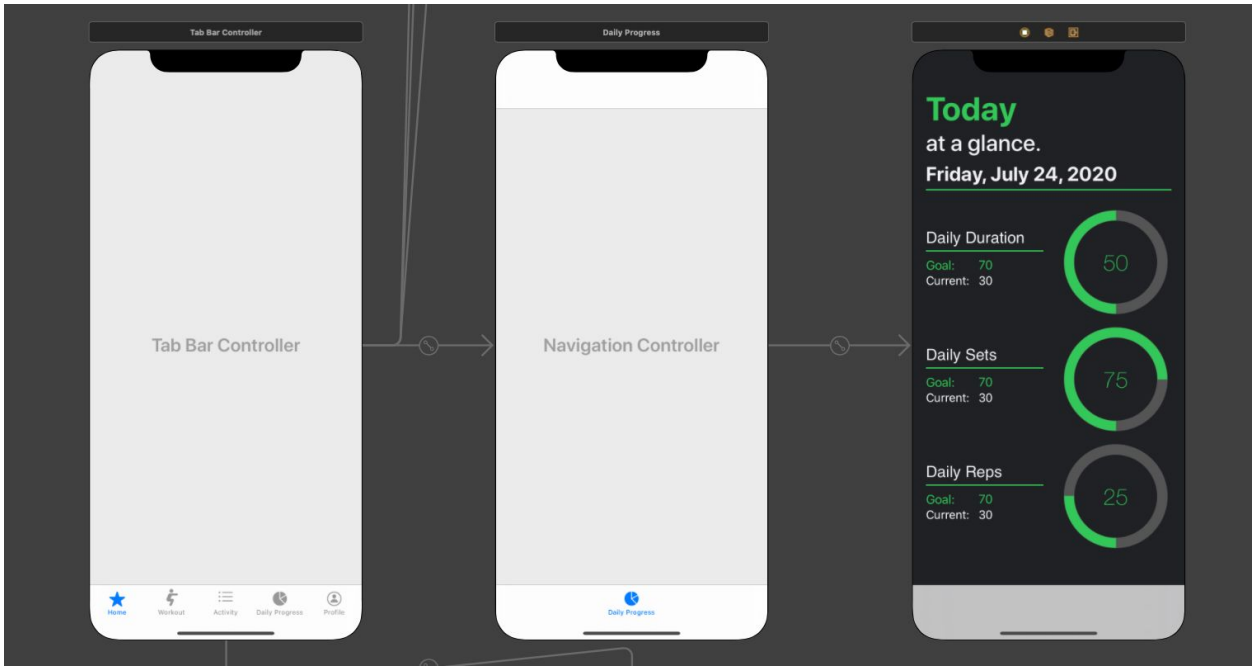
Second View Controller: The second view controller shows a table with the list of exercises the user can select and then register as completed exercise. The view controller is composed of a UISearchBar and a UITableView where a string array with the list of exercises is displayed. The user is allowed to search the tableView for exercises by their name. This view controller was the first challenging part of the project since it involved the creation of a search bar. Creating the filter to find what the user was searching was very interesting to learn and create. This view controller also has a segue that allows the name of the exercise to be passed from this view controller to the Add Exercise View Controller. This is done in order for the user to be able to select a cell in the tableView or exercise and then be redirected to the Add Exercise View Controller.

Add Exercise ViewController: The Add Exercise view controller was created to display the selected exercise from the second view controller and display more information about that set exercise. This page is composed of various UILabels and UITextField and one UIButton and UIImageView. The UILabels are used to display the name and description of the exercise as well as the names of the different options the user has to enter his or her workout logging. The user has the option to enter the date completed, the duration of the workout, quantity of sets, and reps completed. This information is received through the UITextField objects which were modified to only expect number inputs and receive a date picker object. I used a UIDatePicker object to be able to allow the user to pick a specific date from the calendar. In order to save the data imputed, I created an add button with a UIButton. When the user selects the button all the information is checked to make sure the user entered something and if not include some default

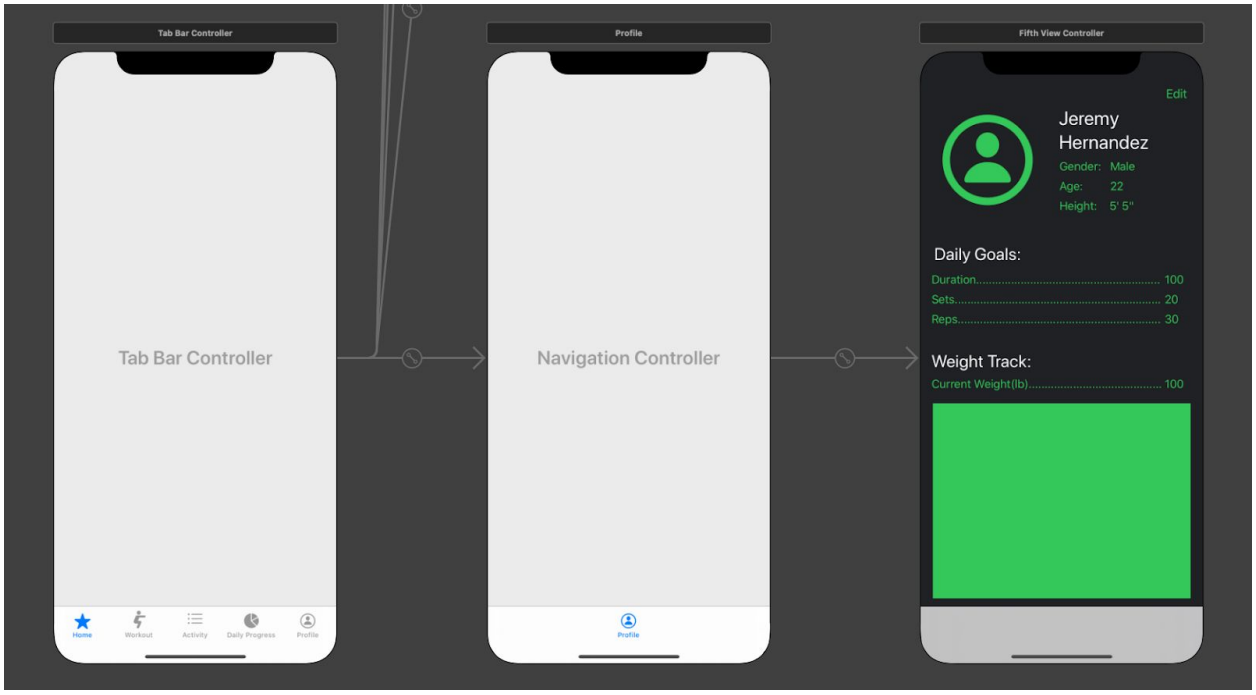
values. Finally, the data is stored in persistent storage within an array of type Exercise. This is done in order for the application to store the data entered even after the user has exited the application. The Exercise class will be described in a section below.



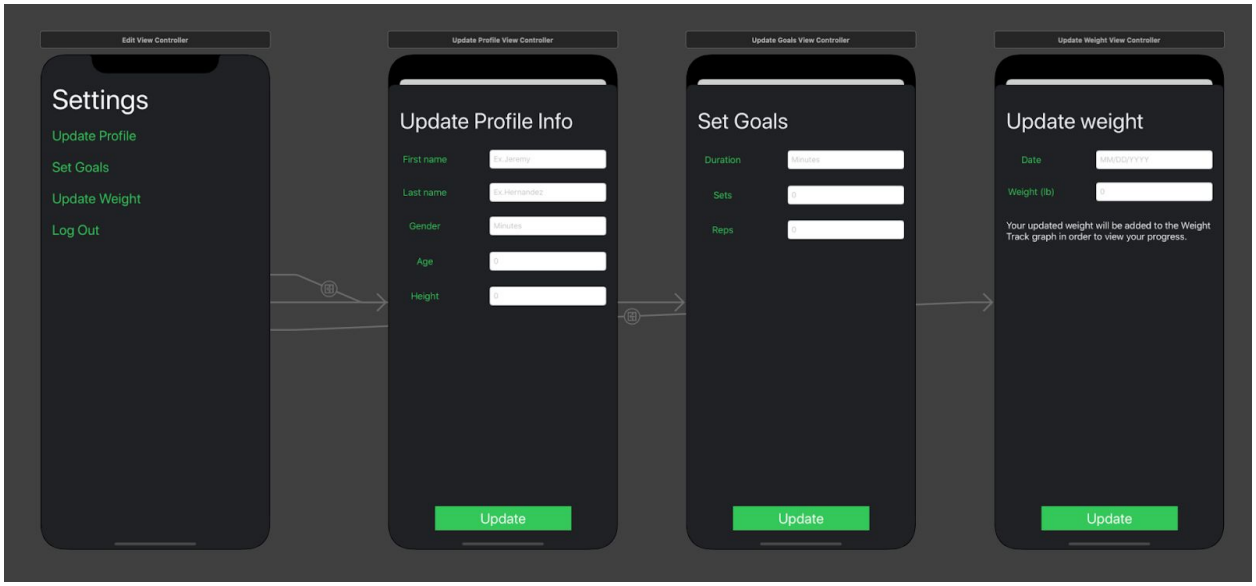
Third View Controller: Arguably one of the most difficult pages to create in my project, the third view controller was created to display the exercises registered by the user in a custom UITableView. I created an expandable/collapsible TableView with a TableViewCell inside of it that displays the dates where the user completed a workout as well as all the exercises completed on that specific date. In this controller, I created two arrays, one of type Exercise and the other of type Section. This was done in order to decode the persisted array and store the exercises completed on the same date within the same section of the table. I made sure the dates in the table were presented in ascending order by sorting the saved array by dates. A particularly difficult aspect of building this page was figuring out how to create the expandables cells and how to separate the array into different sections. I was able to do this by using various for loops and conditional statements to separate the Exercises array into sections that group all the exercises of a specific date together. I also created a UIBarButtonItem in the navigation bar to allow the user to delete all the exercises if desired.



Fourth View Controller: The fourth view controller is intended to be a sort of a quick glance of the user's daily progress to their established goals. It is composed of three MBCircularProgressBarView objects that detail the process of the user's duration, sets and reps recorded. Also present in this controller is a series of labels to describe the different goals and current process as well as the date of the year. In order to get the goals and current process, I decoded the persisted User object and the array of Exercises in disk.



Fifth View Controller: The fifth view controller is the profile page. This page was created to display the user's personal information as well as their weight progress. This page is composed of multiple UILabels, one LineChartView, one UIImageView and one UIbutton. This page was also one of the most complicated pages to create mainly because of the procedure required to create the line chart graph. Although not present in the current version of the application, this page used to allow the user to access their photo library on the device and add their own profile image. Unfortunately, I had to remove this feature due to some complications with autoLayout constraints, but I left the commented code in the file with the intention to add the feature as a future update. In order to compensate for removing the feature I added a UIImageView object with the icon of a user to showcase where the user image would have gone. In order to create the line chart, I imported the Chart library and dependencies from Cocoapods. I then created the LineChartView object and deconded the persisted Weight array to create the graph based on the weights the user has entered in the program. The graph displays the weights registered by the user throughout the dates they were recorded. Also present in this page, is an object of the class ChartMaker, which was created to allow the user to see a marker with the weight and date of the data entry in the line chart. Finally, this page has and edit UIButton that allows the user to view the settings of the application



Edit View Controller: The edit view controller was created to store the settings page of the application. In this view controller the user can select whether they want to update their profile information, daily goals, weight or log out of the application. The buttons in the view controller are embedded in a stack view and when selected the user will be redirected to the respective settings page. If the user decided to log out, I created a UIAlertController in this controller that verifies that the user does indeed want to logout. If the user selects to do so, all persisted data will be deallocated from UserDefaults and the application will perform a segue to the welcome page.

Update Profile View Controller: The Update Profile View Controller is composed of multiple UITextField objects that are used to store the user input. Three of these UITextFields were combined with UIPickerViews in order to allow the user to select from a predefined dataset. This page is very similar to the login View Controller. Finally, this page has a UIButton that is used to confirm the changes done by the user and update the persisted User object in memory. When completed this page performs an exit segue into the fifth view controller.

Update Goals View Controller: The Update Goals View Controller is very similar to the Goals View Controller, it uses the same methods as such as using UITextFields that only allow integer inputs. This controller decodes the User data and updates it with the new user goals. When completed this page performs an exit segue into the fifth view controller.

Update Weight View Controller: The Update Weight View Controller allows the user to update their weight by entering their new weight and the date of recording. This is also achieved by using two UITextFields that are combined with a UIPickerView and UIDatePicker for weight and date respectively. The new weight is added to the persisted Weight array in order to reflect the changes across the application. When completed this page performs an exit segue into the fifth view controller.

Main.StoryBoard: This is a visual representation of the user interface of the Plit application.

Classes created for this project:

Exercise: This is the main class used in this project. It was created to store the information related to the exercises completed by the user. This class adopts the NSCoding protocol and implements its methods in order to allow the object to be encoded or decoded for archiving and saving to disk. It stores the name of the exercise, the date completed, the duration, the number of sets and number of reps for that specific exercise.

User: The second most import class in the project is the User class. This class stores the user related information such as first/last name, age, gender and height. Also stored in this class is the user set goals for duration, sets and reps. This class also adopts the NSCoding protocol and implements its methods in order to allow the object to be encoded or decoded for archiving and saving to disk.

Weight: This class was created separated from the user class since it is intended to be saved as an array rather than a single instance. It stores a weight and the date of it's recording. It is also NSCoding protocol and implements its methods like the previous two classes.

Section: This is struct, but it is stored with the rest of the classes in order to be located more easily. This is a small struct created to store the sections for the UITableView in the ThirdViewController. This structure stores a date, an Exercise array and an expanded bool value. The Exercise array stores the exercises that are registered under the same date and the bool value is used to identify whether the tableView expands or collapses the section in the tableView.

ChartMarker: This class was included for the Charts Cocoapods library. It allows the user to select an entry within the LineChart in the Profile page and view a marker. The marker contains the weight and date related to that specific entry in the chart.

ExpandableHeaderViewDelegate: This class was created to set up the expandable/collapsible UITableView in the ThirdViewController. This class contains the settings and protocols needed in order to create the UITableView.

Additional Technologies used:

UserDefaults: UserDefaults is an interface to the user's defaults database, where you store key-value pairs persistently across launches of your app. I used this technology, in order to store whether or not the user was logged in as well as the list of exercises and data entered by the user. I used the NSCoding protocol, which enables an object or array of objects to be encoded or decoded for archiving and saving to disk. By using the NSKeyedArchiver class, I was able to encode the object into an architecture-independent format that can be stored in a file. In order to decode it, I used the NSKeyedUnarchiver class which decodes the data in an archive and creates a set of objects equivalent to the original set.

Cocoapods: CocoaPods is a dependency manager for Swift and Objective-C Cocoa projects. Cocoapods facilitates the way developers add, remove and update their libraries. It saves the developer time when dealing with application dependencies. For this project, cocoapods was very helpful because it allowed me to install three libraries that enabled me to scale my project more elegantly. The names of this librarie were CardSlider, MBCircularProgressBar and Charts. Their use is mentioned above in the description for first, fourth and fifth view controllers. .

Unsplash: Unsplash is a website dedicated to sharing stock photography under the Unsplash licenses. I used Unsplash to find and use the images for the exercises and the homepage without any complications. Unsplash makes it easy to use stock photography without the hassle of copyright issues.

Adobe Illustrator: Adobe Illustrator is a vector graphics editor developed by Adobe Inc. I used illustrator to design and create the logo for the application. It wanted to create a logo that reflected the style and theme of the application. The process of creating the logo was a learning experience, but also a fun one.

What I've learned:

When faced with the decision of which technology and language to use in order to bring my idea to life, I choose to go with the technology and language I knew the least about. I did this in order to challenge myself, maximize my learning experience and expand my skill set. At the time, I had barely learned the basics of Swift and let alone the fundamentals of how to build a mobile application. Before this project, I had never built any application outside the realm of a command line interface. Thus, having the opportunity to create my own mobile application and learn this new technology meant a great deal to me.

I began my work by researching workout tracking applications to get an idea of the interface and the tools consumers look for in workout tracking and gym applications. Once I knew what I wanted to include and discard in my project, I sketched out my ideas and designed a basic draft of my application. I then began to research Swift coding tutorials and learn the intricacies of the programming language. I started by learning how to build basic user interfaces by following along youtube tutorial videos. Once I learned the basic in's and out's of the language and Xcode, I began doing more challenging applications such as creating a notes app, a reminder app and finally a matching card game. I became fascinated by the way Swift and Xcode facilitates the process of creating graphical user interfaces and connecting those interfaces to running code. I had learned about the Modal View Controller (MVC) design pattern in college, but it wasn't until now that I clearly saw the importance of this architecture pattern when designing and creating an application. This project showed me the importance of understanding the concepts of data structures and knowing which one is better suited to meet my storage demands. It also pushed me to learn how to persist data in memory and in databases. I learned the advantages and disadvantages of using different technologies such as core data and UserDefaults when storing data. This opportunity also worked as an excellent review of the fundamentals of Object Oriented Design given that the majority of my project was built using the concepts of abstraction, inheritance, polymorphism and encapsulation.

In essence, this project was really exciting to create because it felt like the culmination of everything I have learned in my degree at University of South Florida. It stretched my fundamental knowledge of programming as well as my problem solving skills. It challenged me

to strive beyond my understanding of Computer Science and acquire new skill sets and knowledge.