Day Planner Application (iOS)

Prepared by Jonathan Franck

07 December 2019

Table of Contents

[I. Project Plan: 3](#_Toc26647923)

[A. Introduction: 3](#_Toc26647924)

[B. Requirements: 3](#_Toc26647925)

[II. Project Documentation: 4](#_Toc26647926)

[III. Program Design: 7](#_Toc26647927)

[A. Algorithms: 7](#_Toc26647928)

[B. Pseudo Code: 7](#_Toc26647929)

[C. System Flowchart: 9](#_Toc26647930)

[D. Program Flowchart: 10](#_Toc26647931)

[IV. Code: 11](#_Toc26647932)

[V. Testing: 16](#_Toc26647933)

[Rubric: 17](#_Toc26647934)

# Project Plan:

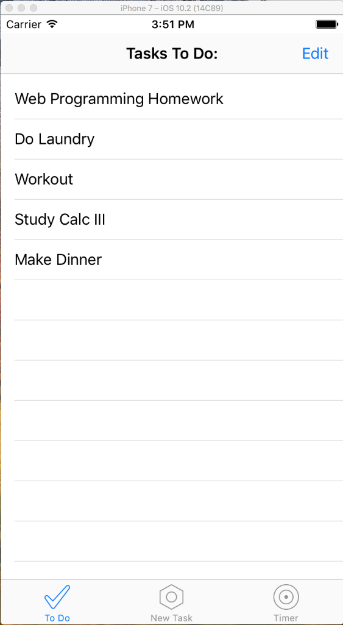
## Introduction:

This report outlines an iOS application called Day Planner. This is an app that is meant to help the user manage their daily schedules. Specifically, the Day Planner app allows users to organize everything they need to complete in an editable to do list. It will also aid in time management, by allowing the user to set a timer that will alert the user when it is complete by sounding alarm, letting them know that their current task should be completed and that it is time to move on to their next goal. This app is for daily use and will be easy to use for anyone with an iOS device.

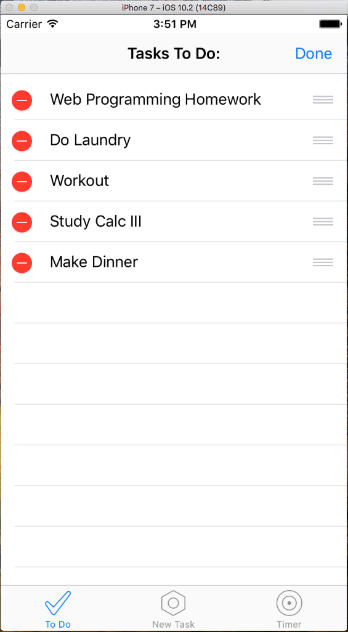
## Requirements:

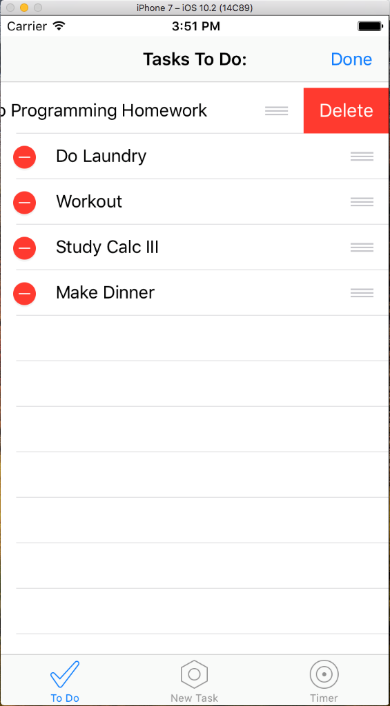
Development of this application has multiple requirements involving hardware, software, personnel, and research. As far as hardware goes, iOS applications can only be developed on Apple Macintosh systems; for this project the systems in Palm Beach Atlantic University’s Computer Science laboratory were used. These machines also require specific software to develop an iOS app; in this project Xcode version 11.2 was used. Being a somewhat simple, yet useful, application, it realistically only entails the work of one developer/programmer, in this case Jonathan Franck. To develop an application such as this, it is necessary to devote several hours into the subject area of Xcode development, with specific focus on Swift syntax and app layout design. This was done throughout the semester through the course CSC 4303 – Mobile Applications Development, which is taught by Dr Machica McClain and utilizes the textbook: **Mobile App Development for iOS and Android** Edition 2.0 by Jakob Iversen and Michael Eierman. There was also a significant amount of time dedicated to tutorials by other programmers, via Youtube, and other resources for iOS developers on StackOverflow.com.

# Project Documentation:

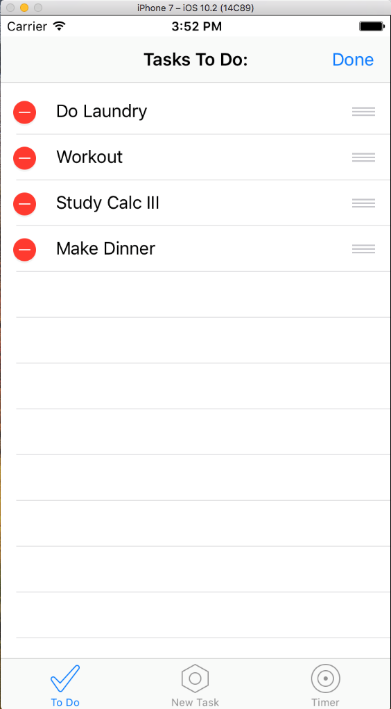


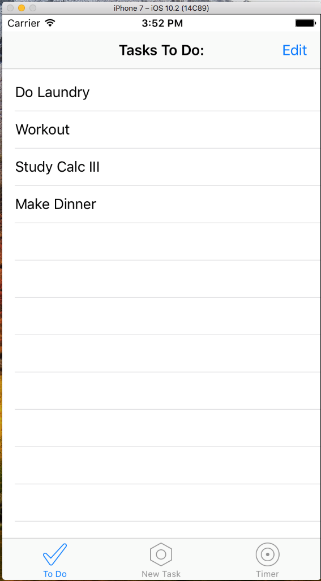
The app opens in the To Do list view. It shows a list of tasks that the user creates and can edit as they are completed. For exhibition purposes, the app is being run on an emulator of iOS 10.2 on an iPhone 7 device.

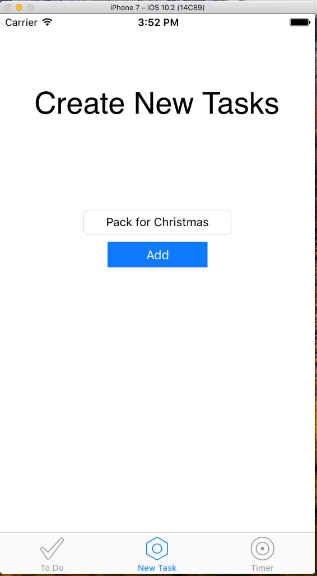
As shown here, when the user taps the “Edit” button, it enables the list to be edited, and the button is replaced with a “Done” button to return the list to its normal state.

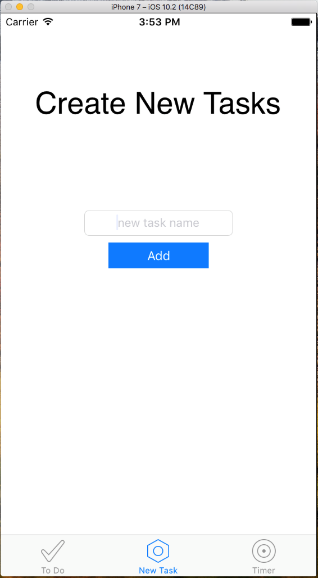
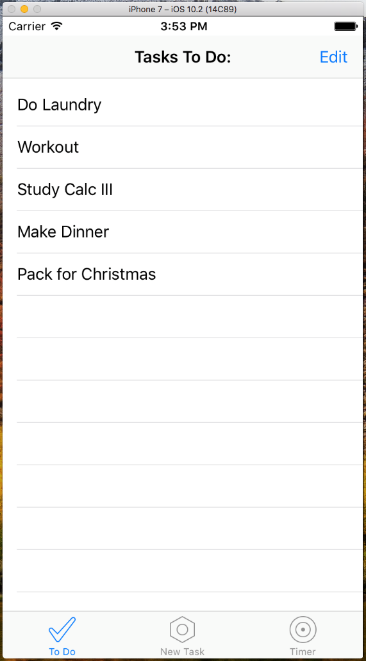


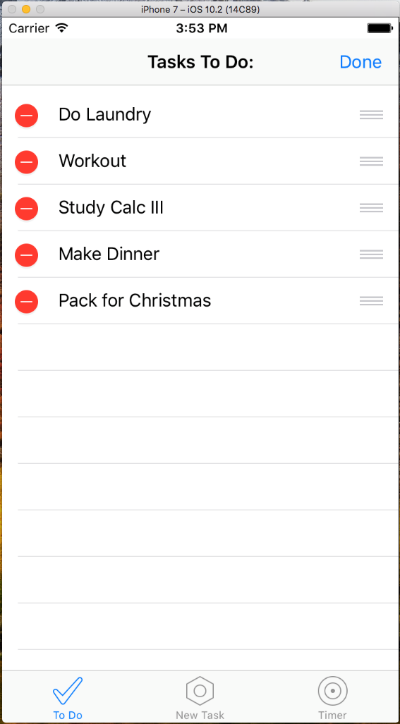
As displayed in these two screenshots, the user is able to remove completed or unwanted items from the list by tapping the red minus button and then confirming by tapping “Delete.”

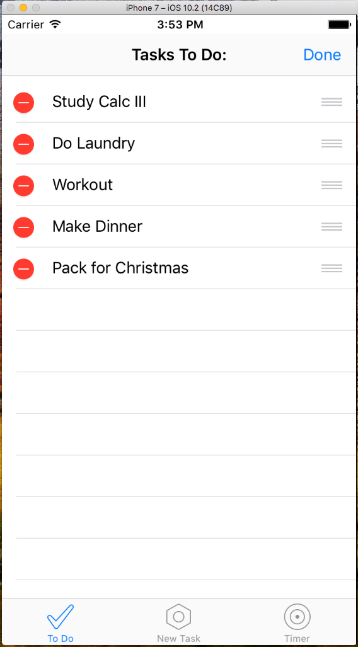


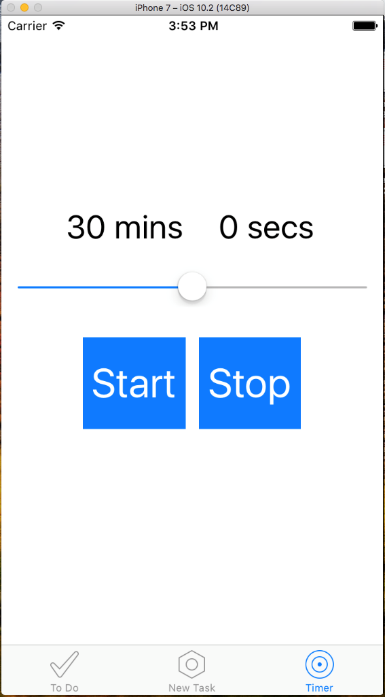
The app also features a navigation bar at the bottom of the screen for easy use. By tapping the “New Task” bar item, the user can move from the To Do list-view, to a page where they can add new tasks to the list. In this instance, the user adds the task “Pack for Christmas” by typing it into the text box and pressing the “Add” button. The user then navigates back to the To Do list-view to confirm that the task was added.

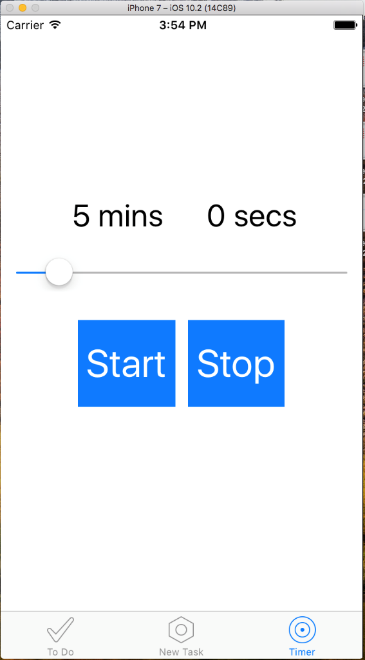


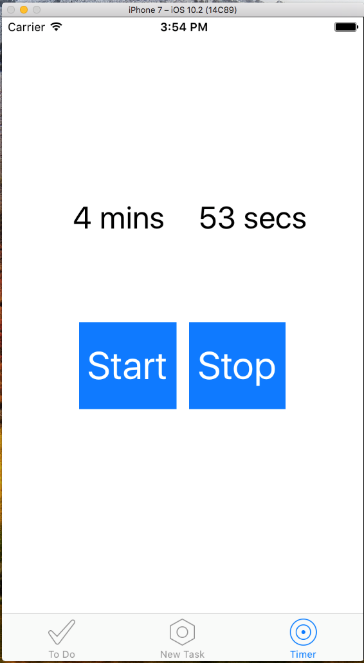


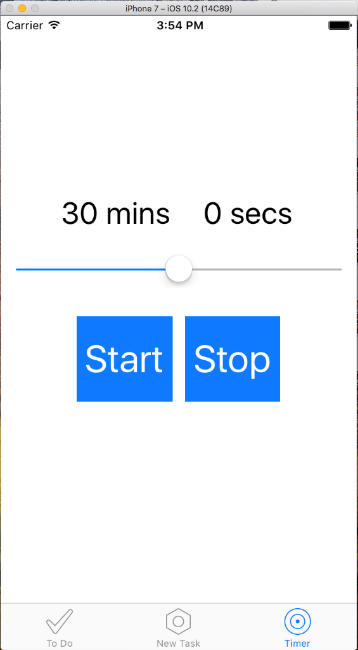
Back on the To Do list-view screen, the user can also decide to re-arrange the order of the tasks by pressing “Edit” again, and then tapping and dragging the items by the three lines that appear next to each.

Here, the user prioritized the task “Study Calc III” and moved it to the top of the list.

The third, and final view in the Day Planner application is the Timer view. It displays a counter for minutes and seconds, a slider bar, and “Start”/”Stop” buttons. The slider, which controls the amount of time it counts to ranges from 0 to 60 minutes in 1-minute intervals.

In this example, the user selects 5 minutes for their timer and presses “Start” to begin the count-down.



As can be seen, the timer can be canceled at any point by pressing “Stop.” If, however, the timer gets down to 0 mins 0 secs, the alarm will sound until the user ends it by pressing “Stop.”

# Program Design:

## Algorithms:

This app requires several algorithms for it to work properly, however most of them are fairly simple. Here I will list a few of the main algorithms and their purposes:

* The first algorithm will be Edit(). This will enable/disable editing on the To Do list-view when the “Edit” button is pressed, and will change the text for it to “Done”/”Edit” respectively.
* Another important algorithm will be addEntry(), which will take the user’s input and add it as a new task on the To Do list-view.
* There will also be a minSlider() algorithm to change the value of the minutes label and the variable holding the number of minutes, appropriate to the current value of the slider on the Timer view.
* One of the most essential algorithms will be the counter(). Counter() will control the timer itself by decrementing the seconds/variables as if it is an actual clock, but in reverse. It will need to decrement the minute variable by one and increment the seconds by fifty-nine after every minute passes. It will also have to sound an alarm once the timer reaches zero minutes and zero seconds.

## Pseudo Code:

Here I will outline some of the pseudo-code that I based the final application off of:

global array ToDoList = [“task1”, “task2”, “task3”, etc…]

Function addEntry(textbox text)

If textbox != empty

Then add text to ToDoList[]

Refresh list-view

Clear textbox

Function Delete(ToDoList[element])

If editing style = delete swipe

Then remove element from ToDoList[]

Refresh To Do list-view

Var minutes = slider value ( 0 to 60)

Var seconds = 0

Function counter()

If seconds = 0 && minutes = 0

Then end time and play alarm

Else If seconds > 0

Then seconds -= 1

Else

Then minutes -= 1 and seconds = 59

Function Start()

Start timer

Hide slider

Disable “Start” button

Function Stop()

End timer

Minutes variable and label = 30

Seconds variable and label = 0

Slider reset to 30

Stop playing alarm

Unhide slider

Enable “Start” button

## System Flowchart:

Start

Alarm

Touchscreen input

Finish

Current view

Touchscreen input

Change view

Edit Tasks

Timer

## Program Flowchart:

**Start**

Move item

Delete item

Start Timer

Set slider

Stop Timer

Time runs out

Input text

Add item

Clear text

Done Editing

Edit list

Timer view

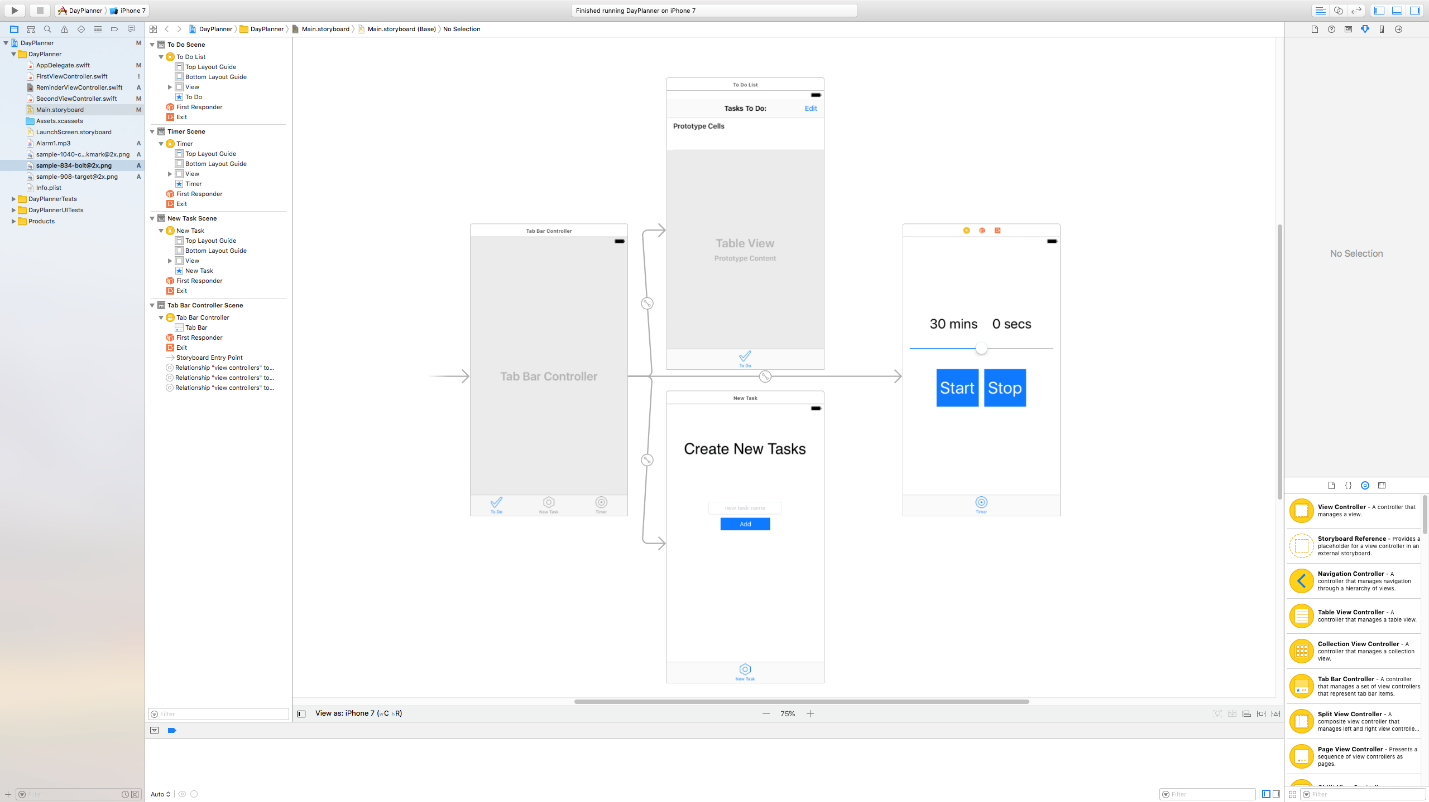
New Tasks view

To Do list View

**Finish**

# Code:

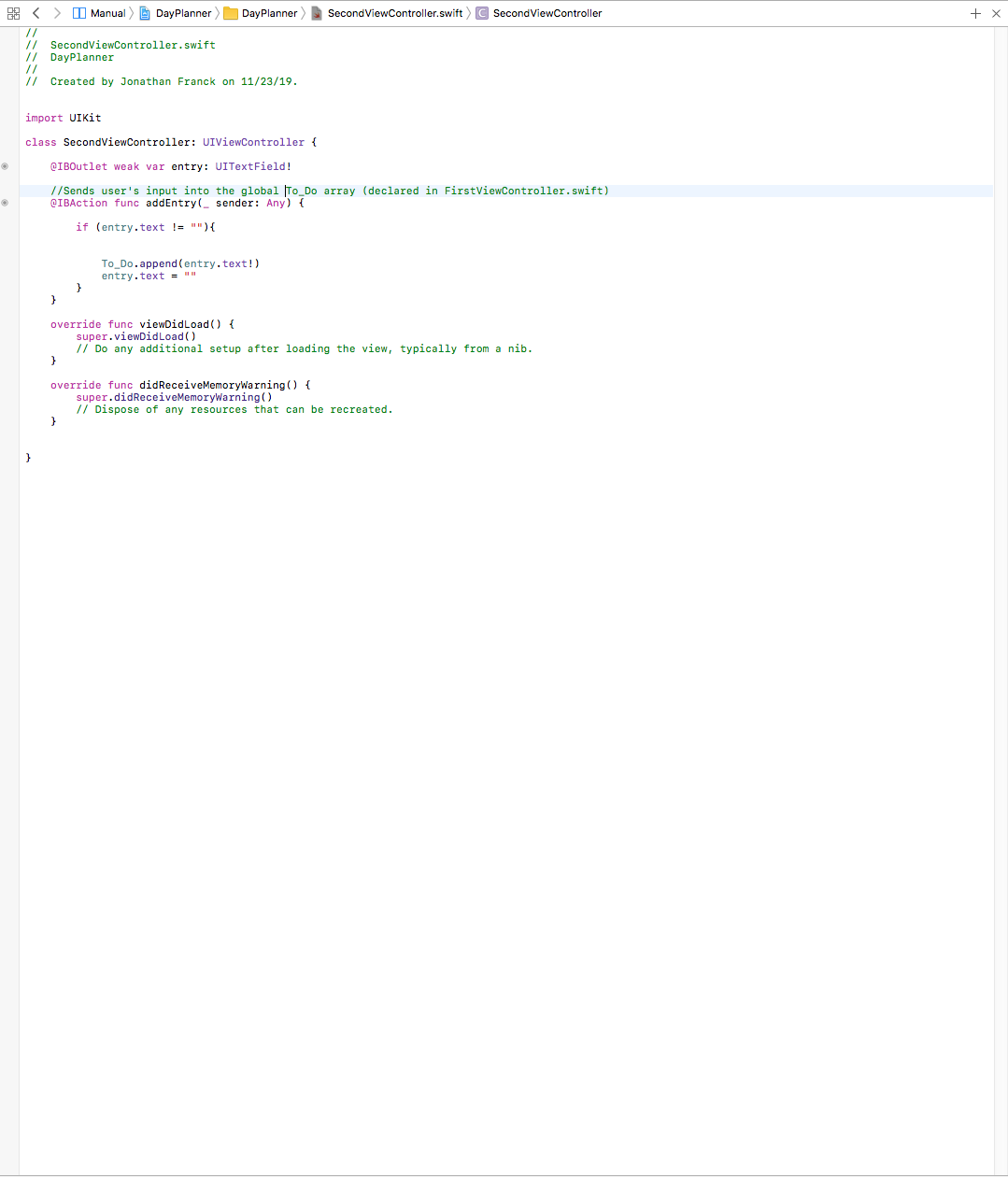
Here I will exhibit and discuss the actual code for the Day Planner application:

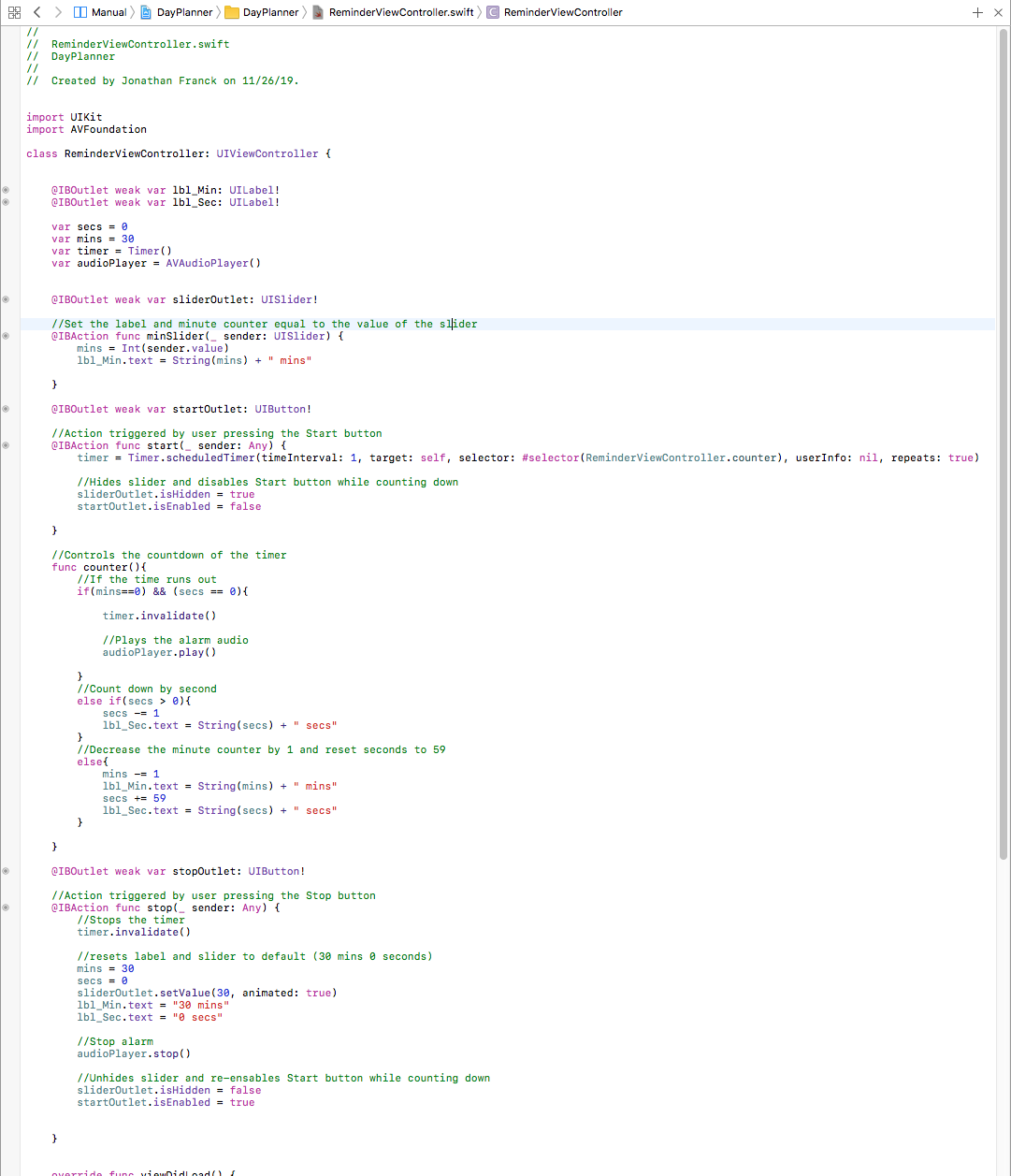


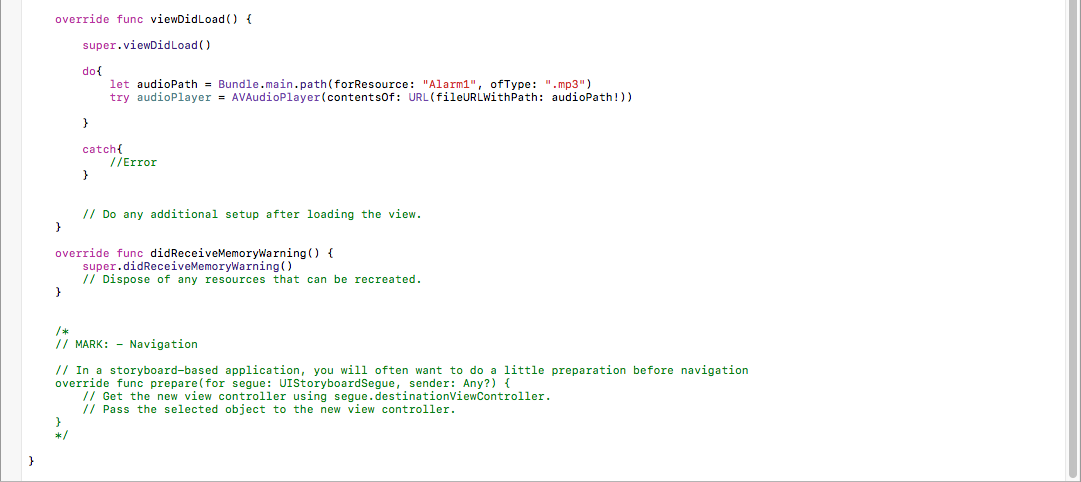
This window is called the Main Storyboard. It shows the layout of every view of the application and the interaction between them. As shown, there are three views, each of which can be accessed from the navigation bar on the bottom of every view. It also shows the individual elements of each view. The First view has a toolbar containing a label and a button, another label, and a table view, which shows one prototype cell; this is the To Do view. The second view simply has a label, a text box, and a button and makes up the New Task view. The final view (Timer) has two labels (mins and secs), a slider, and two buttons.



This is the FirstViewController.swift file, and it contains the code that controls the To Do view. It includes the global array that stores the values of the To\_Do list, as well as functions that perform various processes on this view. These include populating the table, making the table editable, deleting tasks from the To\_Do list, reordering the tasks, and refreshing the table.

This is the SecondViewController.swift file, which regulates the New Task view. The main function in this file is addEntry(), which is triggered by the “Add” button, and checks if there is text in the textbox. If there is, it gets added to the To\_Do list from the FirstViewController.swift file and clears the textbox.





This is the ReminderViewController.swift file, which operates the Timer view. It has variables for the minutes and seconds, and a timer and audio object. It also has several functions, including one that sets the labels and variables equal to the value of the slider. Other functions in this file are the start function, which begins the timer, the counter function to regulate the time and play the audio file when the time runs out, as well as a function to stop the timer and audio. In the viewDidLoad function, there is an extra do statement, which specifies the audio file to play and makes sure it can be connected to (try/catch statement).

# Testing:

For this application I chose to use desk checking to test it. Since it is such a small/ simple program, this method made it quite easy and quick to test. Every time I completed a step in the project, I checked to see if there were any errors before running it. I also walked through the code checking the logic of all of the functions I used. The main function that I needed to test, and fix was the counter() function in the ReminderViewController.swift file. Because the timer counts down two independent variables, I had to make sure that the seconds incremented properly when it was time for the minute to decrement. I also made sure that the timer checked if it was out of time before decrementing the seconds. Otherwise, the seconds would have gone down to negative one before the timer stopped and the alarm played.

Throughout the development process of this application, and especially at the completion, I ran the app and analyzed the reaction to test data. I continuously ran and re-ran the emulator, entering data and testing the functionality of all elements of the app. At the very end, I compared the application to the test plan, and it met all the original requirements of the plan.

# Rubric:

Software Engineering [CSC 3213]

**Software Development Project Rubrics**

|  |  |
| --- | --- |
| Project Topic (Approved) | **Marks / Points** |
| 1. Project Plan | 4 |
| 2. Project documentation | 3 |
| 3. Program Design |  |
| (a) Algorithm | 3 |
| (b) Pseudocode | 3 |
| (c) System flowchart | 5 |
| (d) Program flowchart | 5 |
| 4. Code |  |
| (a) Project source code  (a) Program documentation | 4  2 |
| (b) User documentation  (c) Source code documentation | 3  3 |
| 5. Testing |  |
| (a) Beta Testing / Prototyping  **OR**  (b) Walk through (Desk checking) | 5 |
| **TOTAL (40 MARKS)** |  |