



# Mobile Applications Coursework

University of the West of England

**Name:** Juan Camilo Rodriguez

**Student ID:** 16020551

**Module:** Mobile Applications

**Module Code:** UFCF7H-15-3

**Word Count:** 4389

Access to Component B, stage 1 & 2, please follow this link:

<https://drive.google.com/drive/folders/1u9tyw-i3yN-Js8Jq4Xrf-UsQvqWzG1OQ?usp=sharing>

To access the code, please follow this link:

<https://github.com/JCR21598/RR-SmartScheduler>

# TABLE OF CONTENTS

<b>TABLE OF CONTENTS.....</b>	<b>2</b>
<b>ABSTRACT.....</b>	<b>4</b>
<b>RESEARCH.....</b>	<b>5</b>
PROBLEM DEFINITION & PROJECT GOALS.....	5
Problem.....	5
Research Aim .....	5
LITERATURE REVIEW .....	6
Time Management.....	6
Round Robin.....	7
TIME MANAGEMENT MARKET .....	8
Existing Applications .....	8
Toggl.....	8
Flat Tomato .....	9
Focus Matrix – Task.....	10
Observations for Applications and Market .....	11
IDEA .....	12
<b>REQUIREMENTS.....</b>	<b>13</b>
FUNCTIONAL REQUIREMENTS.....	14
NON-FUNCTIONAL REQUIREMENTS .....	14
<b>METHODOLOGY.....</b>	<b>15</b>
PROJECT SCRUM.....	15
<b>PLANNING &amp; DESIGN .....</b>	<b>16</b>
SYSTEM ARCHITECTURE .....	16
USER INTERFACE MOCK-UPS.....	17

<b>IMPLEMENTATION &amp; TESTING .....</b>	<b>18</b>
EXAMPLE OF APP RUNNING .....	18
STRUCTURE OF PROJECT.....	19
VIEWS INTERACTING.....	20
Passing Data over Views .....	20
Avoiding Views from Stacking.....	20
USER INTERFACE .....	21
Elements Organisation & Constraints .....	21
ROUND ROBIN IMPLEMENTATION.....	22
TESTING REQUIREMENTS & APPLICATION FUNCTIONALITY .....	25
<b>REFERENCES.....</b>	<b>29</b>

## ABSTRACT

This document is the report for **ScheduleSmart** application. An iOS app that takes a different approach when dealing with time management. The way this is achieved is through the use of principles of the CPU scheduling technique, known as Round Robin.

This report covers everything relevant that was done throughout the development of the application. Which includes research, requirements, methodology, design, implementation, and testing.

*Chapter 1:*

# RESEARCH

## Problem Definition & Project Goals

### Problem

Time management has always been a complex problem to solve. This becomes even more of an issue when dealing with multiple variables, such as having many upcoming deadlines at work/university. Sometimes it is hard to determine what to prioritise or how much time should be invested into a piece of work compared to others.

### Research Aim

It is acknowledged that there are time management applications on the market that have been able to help a large audience. Regardless there is no “perfect” app nor technique that can solve everyone’s problems.

The aim of the application is not to create the “perfect” application as that task would be overly complex to achieve, if it is at all even achievable. However, the aim is to create an application that can provide an alternative solution as to how to deal with managing tasks.

Therefore, through the literature review and by learning about time management a new technique is hoped to be found or created. The intentions to have something that is different to what is already available in the App Store.

## Literature Review

This subsection will cover all the content encountered throughout research that is relevant for the report and for the further chapters.

### Time Management

As Brian Tracy (2013) states:

*“Time is the one indispensable and irreplaceable resource of accomplishment. It is your most precious asset. It cannot be saved, nor can it be recovered once lost.”*

Time as the statement suggests is the main factor that makes time management so complex and difficult to solve. This is a variable that cannot be changed as it is a fixed constant that everyone fights against. Additionally, everyone has their own other responsibilities to complete on a daily, weekly, monthly, and yearly basis. All these variables cause everyone to have a unique problem, as to previously it was stated that there is no solution that can consider all these variables at once and solve everyone's problems.

Furthermore, humans are only capable to do a single task at a time. This has been proven by several research papers over and over again that multi-tasking is not feasible (Buser, T. and Noemi, P., 2012) (Chase, J. *et al* , 2012). These papers show that perhaps multi-tasking has a small chance of being possible, however, results demonstrate that it is not at all efficient and/or effective when dealing with anything that is not a menial task. Therefore, this finding suggests that the application that will be developed would have to be one that focuses on a particular task. Instead of having a technique that requires the user to do more than a task at the same time.

Now there is the debate about spending a certain amount of time on a task or doing a task until completed. According to McGraw Centre for Teaching and Learning (2016), which is a Princeton University facility, suggests that due to time being a factor that cannot be controlled whatsoever then individuals should not devote fully to a specific task. Instead spread time for all the responsibilities at hand. The argument/point which is trying to be conveyed by McGraw Centre of Teaching and Learning is that it is difficult to prognosticate the amount of time that will be required to complete a task. Therefore, by spreading time, all responsibilities can be completed to a certain degree completed, this mainly avoids having to rush tasks.

## Round Robin

Throughout researching, stumbled across a CPU process that seems to satisfy the needs to be able to complete the project aim. The technique found is known as *Round Robin Scheduling*, and in this section will be discussed as to how this technique works and the principles behind it.

The way in which this technique works is that there is a CPU that needs to complete a number of operations. The way it completes it is firstly by setting a fixed time to spend on each task, this fixed time is also known as *time quantum*. Then the CPU dedicates that allocated time for each task, not all tasks will be completed by when the time is up, therefore, the task is kept in the queue and will be revisited once the other tasks are visited. What this means is that there is a sort of cycle that is created for all the tasks. But once a task is completed the CPU simply removes it from the cycle, likewise for when a new task wants to be introduced.

Following is a visual representation of how Round Robin works:

**Tasks on  
Queue:**

**T4, T5, T1**

*Time Quantum: 3 seconds*



According to Noon, A., Kalakech, A. and Kadry, S. (2011) and Mishra, M.K. and Rashid, F. (2014) these are some of the characteristics of Round Robin:

- There is performance fairness in regards of time and of the effort of the CPU
- Avoids process starvation, meaning that all tasks will be given enough time, and none should feel as if there was not enough
- Keeps the CPU 100% of the time busy.
- Maximizes Throughput - Achieving a lot of jobs within a period of time
- Worst time turnaround - this is the time it takes to complete a task. This is due to the fixed time which is set.

## Time Management Market

In this section, applications from the App Store are going to be reviewed and at the end will be given the general observations.

### Existing Applications

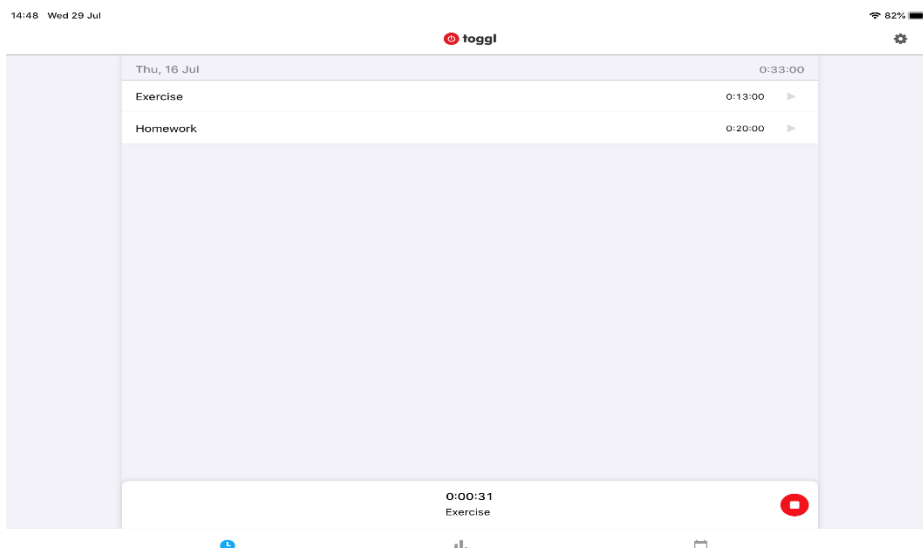
#### Toggl

**Citation:** Toggl OU (2006)

**Market:** App Store

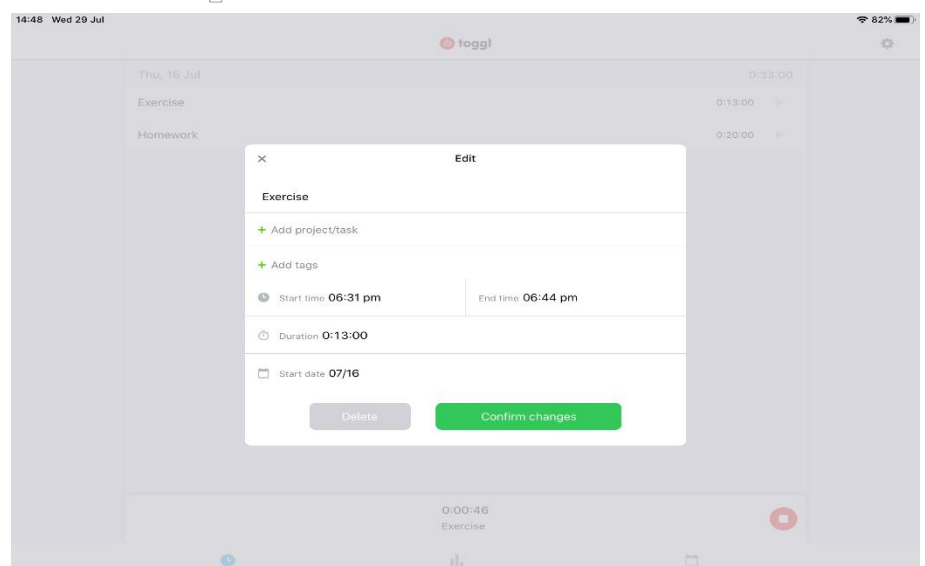
**Description:** A typical and simple time management application that allows users to select their tasks and time. There is no sort of strategy. It simply allows the user to set a disposed time for anything they want to complete.

#### Screenshots:



The home page is the image to the left. Slightly underwhelming page in terms of visualisation and how it delivers the product. It does not deliver that “accomplishment” sensation when performing tasks.

Regardless, the application does come with everything that one would really need to complete tasks and with enough flexibility to create your own goals.





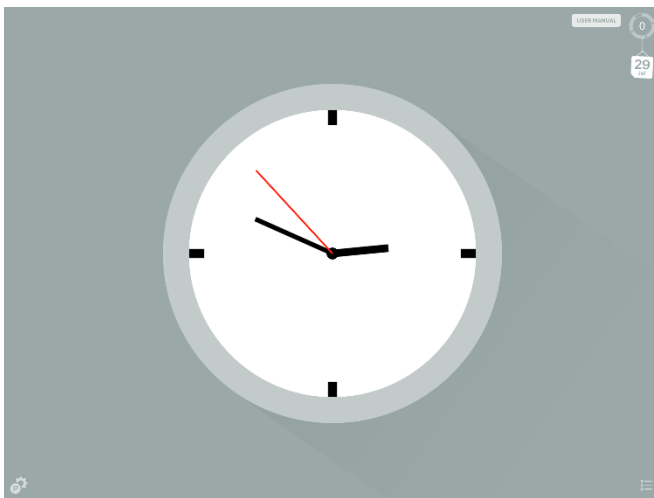
## Flat Tomato

**Citation:** Jain bu (2020)

**Market:** App Store

**Description:** A time management that has incorporated pomodoro technique. Which consists of setting periods of study and having a small break in between tasks. After x number of tasks completed the user is treated to a longer break as a reward.

### Screenshots:

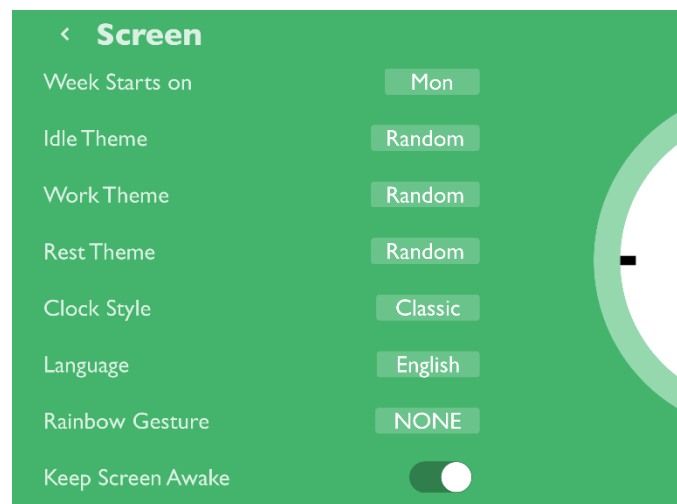


Has a very graphical user interface. With each side of the screen (top, bottom, left and right) slide having a different feature that can be used.

---

Apart, from the stunning graphical interface that the program provides. In addition, it also has a very extensive settings page (by sliding towards the left on the main screen). There the user can change things from screen, notifications, pomodoro, and connections with other applications. Extraordinarily rich application for pomodoro technique application.

---



## Focus Matrix – Task

**Citation:** Denys Yevenko (2020)

**Market:** App Store

**Description:** The task management that uses the Eisenhower matrix, which is another time management technique. The way it functions is by giving each task/responsibility a level of priority and therefore, by visually seeing where each task belongs will make the user complete most likely tasks that are soon and of high priority.

### Screenshots:

### The Eisenhower Method

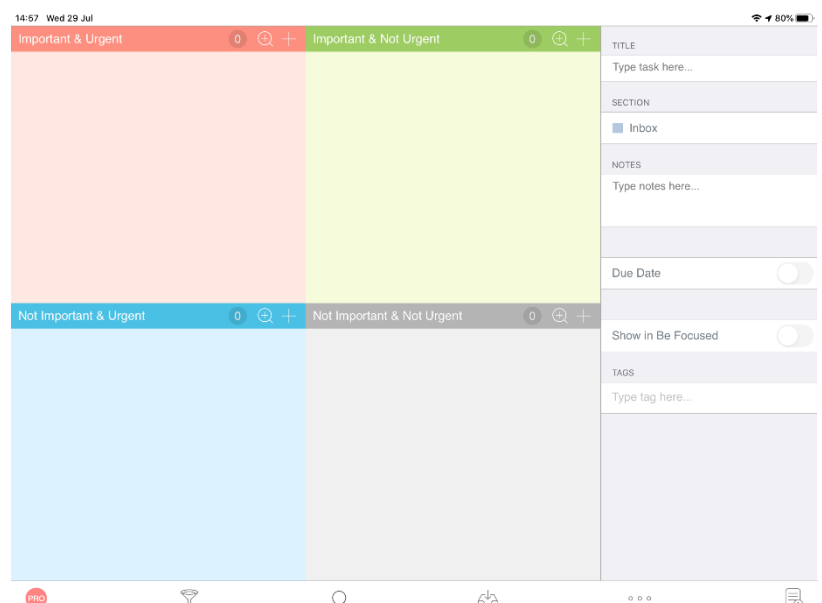
The Eisenhower method helps you organize duties and tasks in order to figure out priorities. Pick a task and ask yourself two questions: "Is it urgent," and "Is it important?" Put the task into the proper quadrant.

- **Not urgent and not important:**  
Don't do these, if you spend a lot of time here, stop doing it and start spending time in the 'not urgent and important' zone.
- **Urgent and not important:**  
Avoid these as much as possible. When you're interrupted, handle it as fast as possible.
- **Urgent and important:**  
Do these, when done, spend time to think about how to deal with the situation in the future.
- **Not Urgent and important:**  
While not urgent, all your available time should go to this quadrant.

	Urgent	Not Urgent
Important	<i>Emergencies</i> <i>Deadlines</i> <i>Some calls</i>	<i>Exercise</i> <i>Vacation</i> <i>Planning</i>
Not Important	<i>Interruptions</i> <i>Distractions</i> <i>Other calls</i>	<i>Trivia</i> <i>Busy work</i> <i>Time Wasters</i>

The first view of the application. Providing an introduction as to how the Eisenhower technique is used. Useful for individuals that have never managed used it. Therefore, informative and well thought out view for the users.

However, remarkably similar to Toggle app, the user experience and the user interface of the application is slightly underwhelming. In this case it is a bit less bland, but the only useful view is the first one as the rest don't really provide any important input for the user.



There are other applications on the market. However, they follow similar principles and ideas to the ones discussed in this section. The ones demonstrated here are the most popular based on the strategy they opt to use.

## Observations for Applications and Market

Here are some observations that were made during app review:

- Time management market is relatively saturated with many time management applications.
- However, many of the applications fail to deliver an application that satisfies the consumers' needs or the user experience and/or user interface lacks
- Out of all the applications reviewed, the application that seems to have delivered the best experience and for my personal needs (due to its flexibility) was Flat Tomato.
- Out of all the techniques, the technique that seems to have the better applications overall is "pomodoro". The other techniques do not have an application that can hit the levels of pomodoro applications

## Idea

Based from the above literature review and critical analysis of the applications and market, this is the proposed idea for the application.

**Idea:** *The application is to create a time management method using the principles from Operating System Round Robin (RR) scheduling algorithm.*

Apart from this being a slightly new idea that is being brought to the application market. It is also believed to be an application to be more applicable for real life situations. What this means is that many things that is done on a daily basis requires time to complete. However, as McGraw Centre for Teaching and Learning (2016) suggested, real-life events are different to what is expected to be achieved within a time period, as there are external factors that can clash and cause disruption.

That being said what this application is trying to achieve is:

- Deliver equal effort to all tasks. Therefore, all tasks will have a standard, instead of some being better than the rest of the tasks that are completed.
- It is acknowledged that there are tasks that humans try to avoid as in some sense cause annoyance/boredom. Therefore, this forces individuals for even that type of work to put in a fair amount of time.

Chapter 2:

## REQUIREMENTS

Requirements are an important step when developing an application. They determine what is desired to have within it. Regardless, a common problem that is faced regularly in software development is *time*. This factor is not always in favour and in some cases not the entire ideal functionalities come into the final product due to this reason.

Therefore, it is of importance to order the desired functions in terms of what is desired more over other features using a **requirement prioritisation technique**. A quite common technique is known as MoSCoW, and it had four different levels to determine the importance of features. This is how Achimugu P. *et al* (2014) defines each level:

<b>Must</b>	<i>Requirements are not negotiable; the failure to deliver these requirements would result in the failure of the entire project</i>
<b>Should</b>	<i>Features that would be nice to have if at all possible</i>
<b>Could</b>	<i>Features that would be nice to have if at all possible but slightly less advantageous than the "S" (Should)</i>
<b>Won't</b>	<i>These requirements are not unimportant, but they will definitely not be implemented in the current software project. They may, at a later stage, be created.</i>

It is also important to split the different functions into serviceable (functional requirement) and how to measure the service features quantitatively and qualitatively (non-functional requirement). These are the definitions proposed:

<b>Functional Requirements (FR)</b>	<i>Functional requirements specify the functions of the system, how it records, computes, transforms, and transmits data (Lausen, S., 2002)</i>
<b>Non-Functional Requirements (NFR)</b>	<i>Non-functional requirements describe the nature and limitations on the project instead of its functionality, also this term describes the non-behavior aspects and attributes of the system including usability, portability, security, understandability, reliability, and modifiability. In general, the non-functional requirements highlight the requirements that describe "how good" the software (Hudaib, A. <i>et al</i>, 2018)</i>

## Functional Requirements

ID	Priority	Description
FR-01	Must	The application will bring a new way to be able to manage time
FR-02	Must	The application will notify to the user when it is time to go to the next duty
FR-03	Must	The user being able to add to the application the tasks they would want to complete
FR-04	Must	The user being choosing the amount of time to use for all tasks
FR-05	Should	The user being able to edit the task for round robin

## Non-Functional Requirements

ID	Priority	Description
NFR-01	Must	The styling of the application following current trends
NFR-02	Should	The interface should be easy enough that the user should be able to understand easily how the application works and its flow
NFR-03	Could	The application being adaptable and work in a variety of iOS devices
NFR-04	Could	The application having the ability to orientate in any direction
NFR-05	Wont	The project to be entirely documented

Chapter 3:

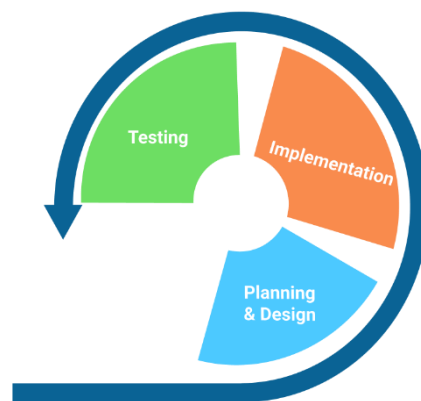
## METHODOLOGY

There are several methods as to how a program could be developed. It has been learnt in the industry that the approach used really comes down to the task undertaken. Generally speaking, Software Development Life Cycle (SDLC) can be categorised into two groups: *Traditional Development* and *Agile Development*.

The SDLC that this project will use to be able to develop the application is by using Agile Development, more specifically will be using Scrum to develop the application. This will allow for flexibility and adaptability for the task. As well as to continue to develop with more modern techniques.

### Project Scrum

The main aspect that this project wants to utilise from Scrum are the sprints. This will allow for the application to be developed in smaller portions. However, Scrum is a methodology that has more principles than actual techniques as to how tasks should be done. Therefore, the following image demonstrates the steps taken for each sprint.



SCRUM Sprint Step	Description
Planning & Design	This section focuses on planning and designing to help the implementation. In other words, all sketches, diagrams, mocks-ups that are relevant for the application. Chapter 4 covers this section.
Implementation	Overviews the key points of the application development. Portions of the code will be shared to demonstrate how such sections were developed.
Testing	This stage tests the requirements set for the application; the ones covered in Chapter 2.

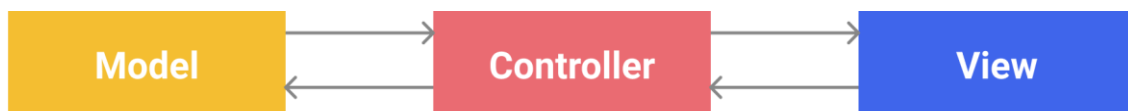
*Chapter 4:*

## PLANNING & DESIGN

### System Architecture

The application will follow **Model-Controller-View (MVC)** design pattern to be able to develop the application. This will provide scalability, organisation, and maintainability for the application if further development is desired. This architecture is divided into three stages:

- **Model:** This area is where all the data and the logical operations occur. In addition, if there is a database, then this area is also in charge of communicating with it.
- **Controller :** This part is what links the model with the view. It receives the data that the user inputs on the user interface and sends it to the model. Likewise, the model can perform some operations and send data to the view.
- **View:** This is about everything that has to do with any graphical/viewable component, in other words anything that is in the User Interface (UI)



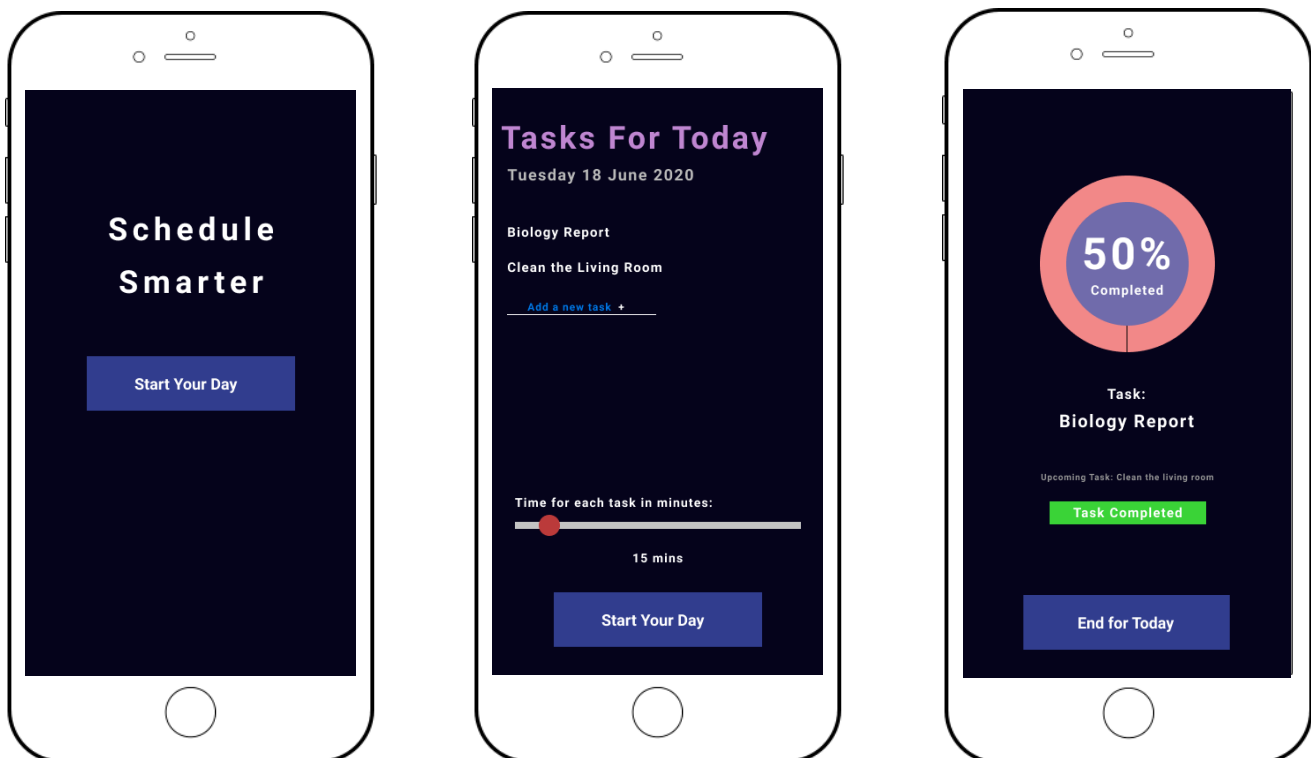
What this architecture promotes is mainly for the User Interface does not have important logical operations in which malicious people can take advantage of and possible steal data.



## User Interface Mock-ups

The following are mock-ups of the desired application:

1. **Start Page** – This is the page initialises with. In theory when a user reaches this page means that they are ready to initialise their day/responsibilities. Therefore, proceeds to click the “Start Your Day” button.
2. **Task Page** – Here the individual will add and submit the tasks the he/she has to complete for the day. In addition, they adjust the fixed number of minutes that they would like to set for each task.
3. **Timer Page** – This page contains the main purpose of the application. Contains a timer to indicate how much is either left for the next task or how much has been completed of the current one. Below indicates what the following task is so that they can prepare for such activity. Additionally, there is a button that can be pressed for when the current task is complete and needs no more iterations. Finally, at the end of the page there is a button which the user can press for when they have finished for the day or have finished all the tasks.



1

START PAGE

2

TASK PAGE

3

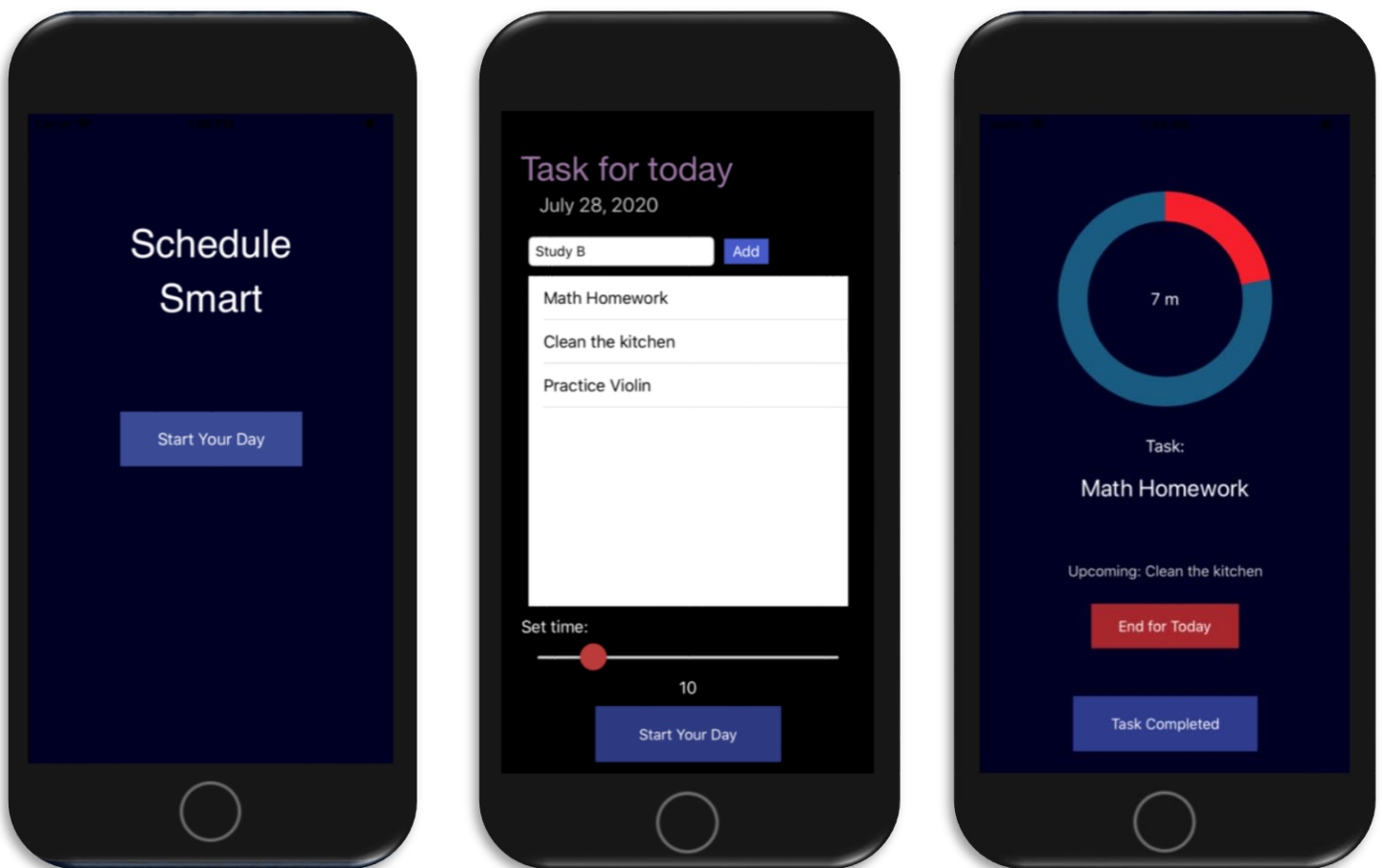
TIMER PAGE

*Chapter 5:*

## IMPLEMENTATION & TESTING

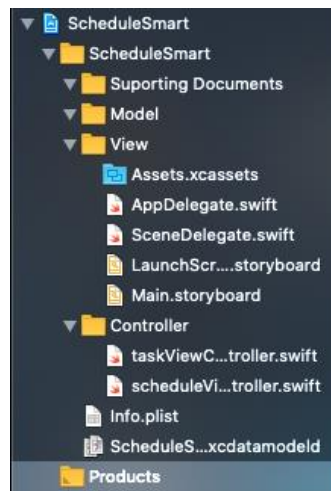
### Example of App Running

The following images are screenshots of using the application, simply to give a visual aid as to how it works:

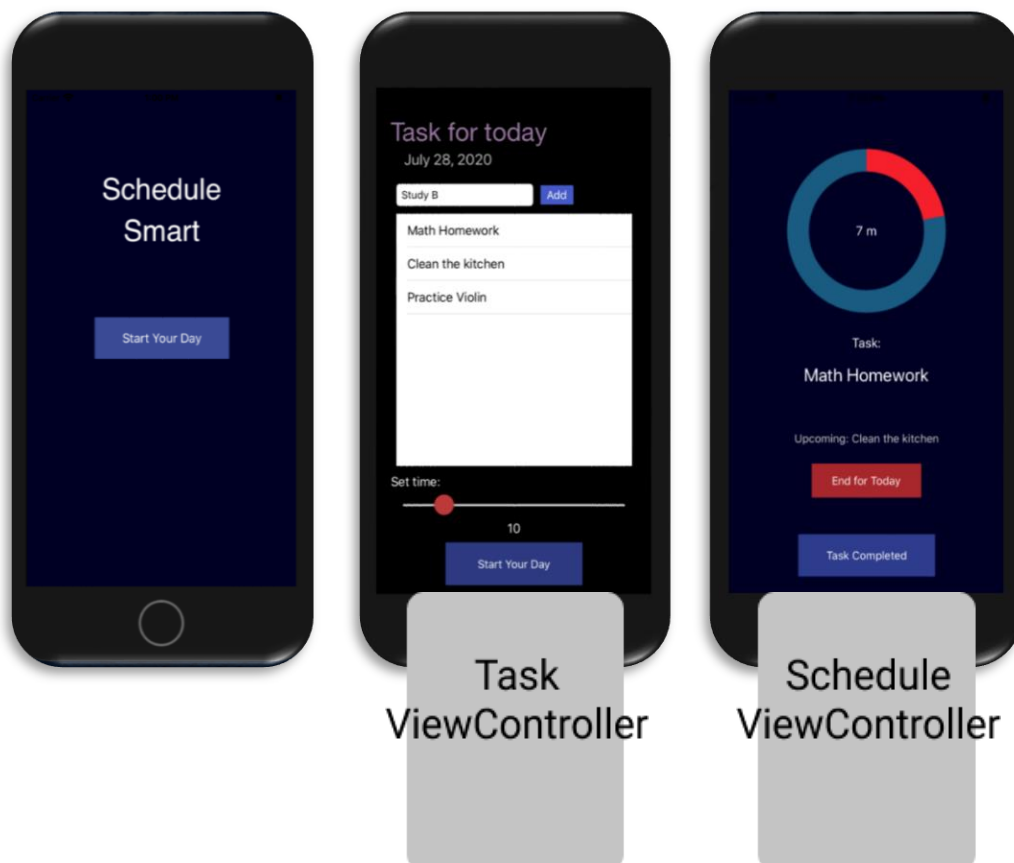


## Structure of Project

As mentioned in *Chapter 4, System Architecture*, the program followed an MVC architectural design. Below is an image illustrating the directory arrangement that allowed think ahead and implement the program in this format.



There are the views that area available and the view-controllers that control that particular view. In essence, the MVC architectural pattern is followed in order to create a more organised and maintainable program.



## Views Interacting

### Passing Data over Views

- The views can communicate with other views through the use of segue relations. The way the segues, apart from moving from one view to another is by passing data through them as well.

```
// Passing the data set by the user over the segue - tasks and time
override func prepare(for segue: UIStoryboardSegue, sender: Any?) {

    // to be able to access attributes from scheduleViewController
    let vc = segue.destination as! scheduleViewController

    // settings those attributes from other viewController to the values obtained on this view
    vc.finalCycleTasks = taskList
    vc.finalCycleTime = Int(timeLabel.text!)


}
```

### Avoiding Views from Stacking

- Another feature that was used with segues is the unwind feature. Which allows to not stack the views on-top of each other. Which would create an inefficient

```
    // If there is no more tasks to do overall
    if finalCycleTasks.isEmpty{
        self.performSegue(withIdentifier: "unwindReset", sender: self)
        return
    }

    // To be able to go to previous views
    @IBAction func unwindReset(_ sender: UIStoryboardSegue){ //Return to main menu function
        self.viewDidLoad()
    }
```

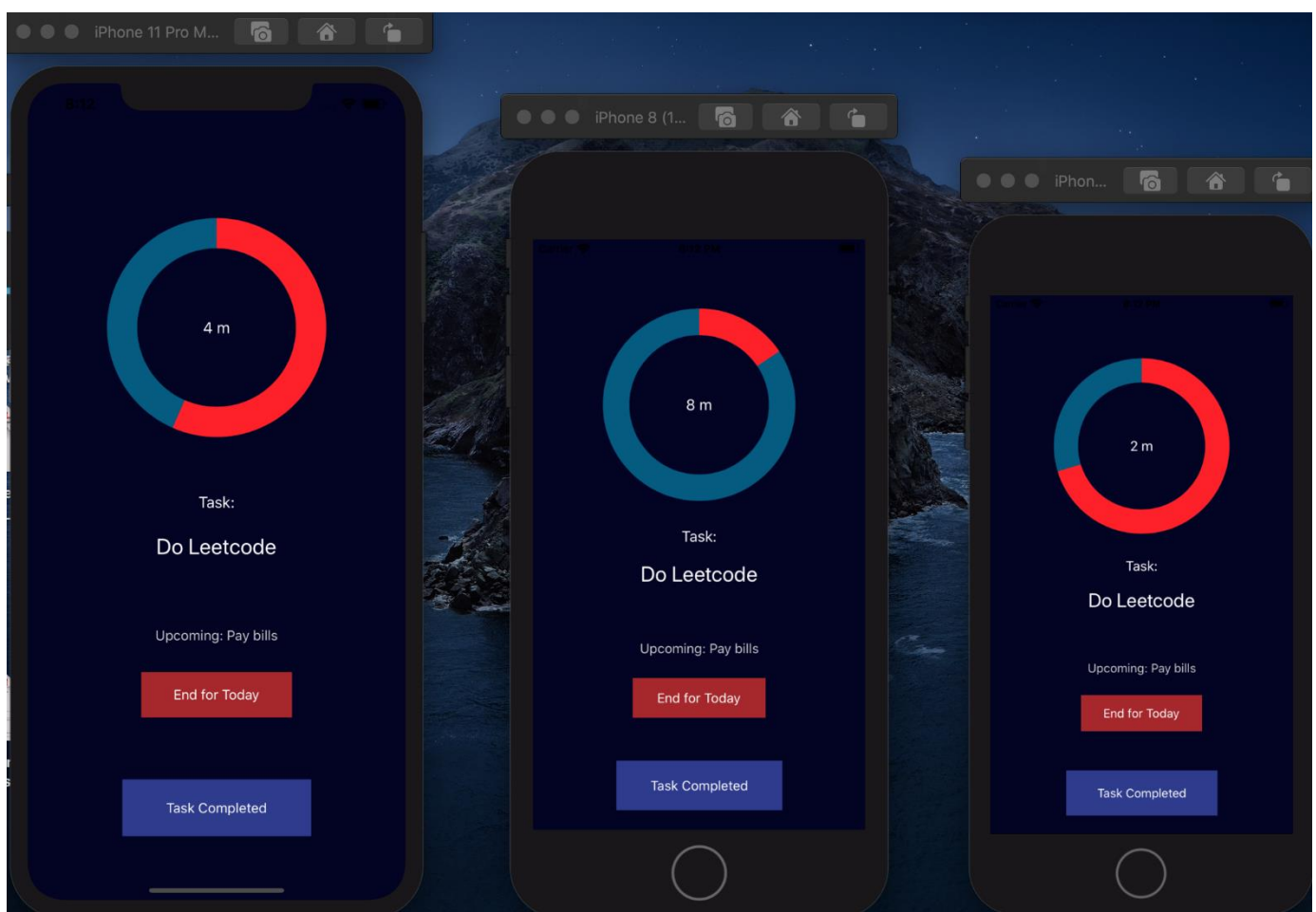


## User interface

### Elements Organisation & Constraints

It was meticulously planned as to how the elements (labels, table-view, buttons, and more) were going to be placed to be able to provide a good user experience and at the same time being able to set constraints that would allow multi-iPhone display.

Constraints were a massive priority for this application as the aim was to be able to display this application on multiple iPhones. In addition, fixed heights, widths, and other similar properties were not given fixed values. Instead they were provided with percentage values and ratios to be able to adapt to different screens and pixel densities.



**iPhone 11 Max Pro**

**iPhone 8**

**iPhone SE**

From the above image it can be seen how the constraints and multipliers are taking effect in the application. Refer to the application as it is difficult to cover as to all the constraints that were implemented

## Round Robin Implementation

In this section, it will be reviewed how the actual implementation of the Round Robin principles were developed in the project. Such implementation can be found in the “*ScheduleViewController*” file.

The way this problem was tackled was different to perhaps what a more conventional method would use. This is mainly because as it can be seen in the mock-up of the application, there was a dynamic timer bar wanted to be included.

The way this was achieved is through the use of in-built timer. Worth noting that a timer is used instead of a while loop due to:

- Not being easily to change labels dynamically during a loop statement
- So that the timer bar is updated quickly, for better UX

```
timer = Timer.scheduledTimer(timeInterval: 0.01, target: self, selector: #selector(updating), userInfo: nil, repeats: true)
```

What this is doing is being called every 0.01 seconds and it is calling the “*updating*” method. Here is the snippet of the method, and it being annotated as to how it functions:

```
132 @objc func updating() {
133
134     /// When there is only a task to do on the list
135     if finalCycleTasks.count == 1{
136         nextTaskLabel.text = "Only this task pending"
137     }
138
139
140
141
142     // This is considering there is more than a value in list and have reached end of task time
143     if Int(count) >= self.finalCycleTime*60{
144
145         /// Reset values
146         count = 0.0
147         timeCirclePercentage = 0.0
148
149
150         if currentIndex + 1 <= finalCycleTasks.count - 1{
151             currentIndex += 1
152             checkNextTask()
153         }
154         else{
155             currentIndex = 0
156             checkNextTask()
157         }
158
159         currentTaskLabel.text = (String(finalCycleTasks[self.currentIndex]))
160
161         return
162     }
163 }
```

Checks if there is a task left, to provided appropriate message

Checks if the current time is more than the one that was set by the user.

If so, then reset and check if the there is another item on the task of list for both the current task and the upcoming one.

```
164
165 // Time label
166
167 /// Display seconds - when <= 60 seconds
168 if (finalCycleTime*60)-Int(count) <= 60{
169     countdownLabel.text = String((finalCycleTime*60)-Int(count)) + " s"
170
171 }
172 /// Display mins - when > 60 seconds
173 else{
174     countdownLabel.text = String(Int(self.finalCycleTime*60-(Int(count)))/60) + " m"
175 }
176
177
178 // Update values
179 timeCirclePercentage = Double(count)/(Double(self.finalCycleTime)*60.0)
180 count += 0.01
181 updatingCircle()
182 }
183
```

Checks how much time is left based upon the time set by the user.

If the time is less than a minute left, then the time changes from minutes to seconds.

The key point there is that the function updatingCircle is called to changes the amount of degrees filled in the circle.

Since this is dependent of time then it is continuously moving. Also, due to the time call being low, then it delivers a smooth process.

As seen from the above code, the code presented bases itself with the program functioning with time. However, from the mocks, it can also be seen that user also has the ability to disrupt the time and tasks. Therefore, code for this was also created for when this occurs as this lead initially to many problems regarding as to indexing incorrectly, going out of range, not starting the cycle correctly and many more problems.

The button that caused this was the one labelled as “*Task Completed*”, here is the annotated code:

```
// Button that removes the current task from the cycle
@IBAction func taskEndButton(_ sender: UIButton) {
    finalCycleTasks.remove(at: currentIndex)

    // Need to perform checks once removed item

    // If there is no more tasks to do overall
    if finalCycleTasks.isEmpty{
        self.performSegue(withIdentifier: "unwindReset", sender: self)
        return
    }

    // By removing one, the whole list is dropped by one
    if currentIndex == (self.finalCycleTasks.count){

        currentIndex -= 1
        nextTaskLabel.text = "Upcoming: \(self.finalCycleTasks[0])"
    }
}
```

If the list is empty, then go back to the previous view

When an item is removed, then the index drops by one

```
// For when its the last item in the list
else if currentIndex == (self.finalCycleTasks.count - 1){
    nextTaskLabel.text = "Upcoming: \(self.finalCycleTasks[0])"
}
// If there are no problems with removing the task then continue as normal
else{
    nextTaskLabel.text = "Upcoming: \(self.finalCycleTasks[currentIndex+1])"
}
currentTaskLabel.text = (String(finalCycleTasks[self.currentIndex]))

count = 0.0
}
```

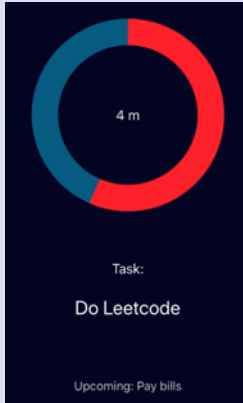
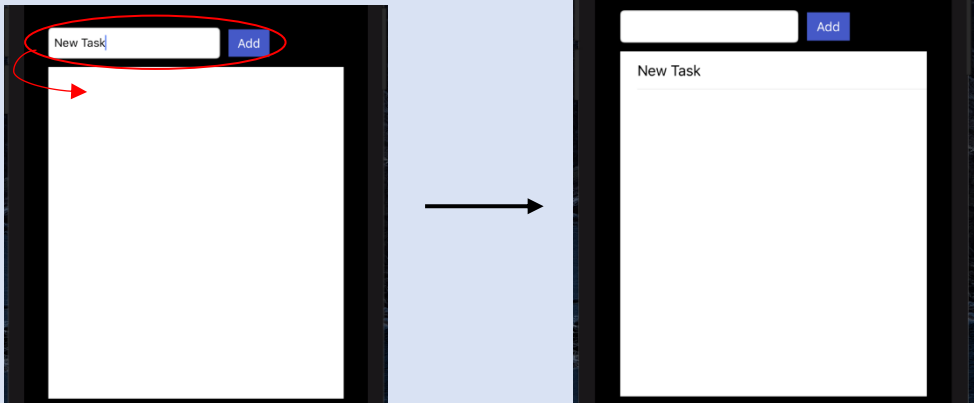
If removing item from the end, then make so that the upcoming label is directly correctly


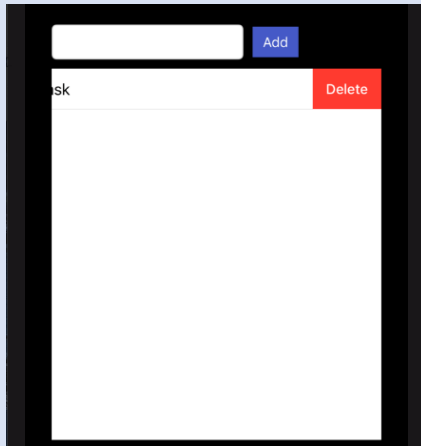
This means that there was no problem when it came to the indexing of the current and upcoming task

This is the main implementation for Round Robin principles. This file was a lot about considering many of the scenarios that could happen with the timer and labels. It is believed that all the possible outcomes and reactions were considered, this will be further explained and proven in the following section which goes into more in-depth testing that was conducted for this application.



## Testing Requirements & Application Functionality

Test ID	Requirements Tested	Test Description	Result
Testing Functional and Non-Functional Requirements			
001	FR-01	This test is a reflective analysis on the application and whether or not the aims that were set in Chapter 1 were delivered	<b>Pass</b> It is believed that the aim of the project was accomplished as it was proven in the research that this application was something that was not seen on the mark
002	FR-02	Whether the application provides the service that it is for the next task	<b>Pass</b> The application visually through the timer bar or and dynamic labels indicate to the user when it is next task. Additionally, for specials circumstances there are pop-ups
	<b>Evidence:</b> 		
003	FR-03	The user being able to add the number of tasks that they would want to include into the Round Robin.	<b>Pass</b> The application uses a text-field and a button to populate a table-view with the data/tasks that the user wants to complete.
	<b>Evidence:</b> 		

004	FR-04	The user is able to choose the time they want to set for the time quantum	<b>Pass</b> User is able to set any time between 1 minute to 60 mins. Allows great range of time for the user to choose. Slider and label are linked top
	<b>Evidence:</b> 		
005	FR-05	Allowing the user to be able to modify the tasks that they want to user for the cycles.	<b>Pass</b> User is able to modify the table-view and decide before submitting the tasks they want to use
	<b>Evidence:</b> 		
006	NFR-01	The application should follow styling/design that is modern	<b>Pass</b> It is acknowledged that this requirement is subjective. However, when comparing it was used considered the ways in which such methods implemented their method for time management. Therefore, there was an attempt to follow the trend being used.
007	NFR-02	The system being easily understandable to the common person	<b>Pass</b> An objective was to make an easily comprehensible system to create a good user experience. With that in mind the system uses top-to-bottom layout, which meaning that the content of each view is read in a chronological manner. Additionally, the words used are one syllable and no extending over 2 words.
008	NFR-03	Application having the possibility to work in a range of iOS devices.	<b>Pass</b> The device does work in a range of devices. As it was covered in the <i>Elements Organisation &amp; Constraints</i> , a wide variety of constraints and non-fixed values were used to

			permit elements of the field to move and take different spacing depending on the device.
009	NFR-04	Application being able to be used in different orientations	<b>Not Considered</b> Time was running short once arriving to this requirement. Regardless, more planning would have need to be able to develop the landscape orientation. Therefore, this is a consideration for future work
010	NFR-05	Documentation formally the usage of the application, along with how key snippets of code works	<b>Partial Considered</b> It was established that full documentation was for future work. Regardless this report as such is a partial document that can help to some degree understanding the ap

#### Testing taskViewController functionality

011	-	Adding a task	<b>Pass</b>
012	-	Removing a task	<b>Pass</b>
013	-	Setting time between 0 and 60	<b>Pass</b>
014	-	If task and time inputted -> goes to next view	<b>Pass</b>
015	-	If task inputted but not time -> pop-up preventing to move to next view	<b>Pass</b>
016	-	If time in inputted but no tasks -> pop-up preventing to move to next view	<b>Pass</b>
017	-	Sending time and tasks list input to scheduleViewController	<b>Pass</b>

#### Testing scheduleViewController functionality

018	-	scheduleViewController receiving all the input from the previous view	<b>Pass</b>
019	-	Timer starting from the time established	<b>Pass</b>
020	-	Timer transitioning from minutes to seconds	<b>Pass</b>
021	-	Timer resetting once task cycle completed	<b>Pass</b>
022	-	Timer working correctly when cycle whole cycle is completed.	<b>Pass</b>
023	-	Timer working properly when there is only a task in the list	<b>Pass</b>

024	-	Timer bar showing a visual representation of the time left	Pass
025	-	Timer bar moving visually representing the correct amount of time left visually	Pass
026	-	Time bar resetting once task cycle is completed	Pass
027	-	List iterating through once a task cycle is completed	Pass
028	-	List cycling back to the start when a whole cycle is completed	Pass
029	-	Upcoming task label displaying +1 if there is an item left in whole cycle	Pass
030	-	Upcoming task label displayed the first task of list if current task index is on the last index	Pass
031	-	If user completes a task, then button removes task from list	Pass
032	-	If there are no more items in the list, then go back to table view	Pass
033	-	If user is done for the day and wants to click on “End for Today” button, goes to table view	Pass

## REFERENCES

Achimugu, P., Selamat, A., Ibrahim, R. and Mahrin, M.N. (2014) A Systematic Literature Review of Software Requirements Prioritization Research. *Information and Software Technology* [online]. 56 (6), pp. 568-585. [Accessed 16 July 2020].

Buser, T. and Noemi, P. (2012) Multitasking. *Experimental Economics* [online]. 15, pp. 641-655. [Accessed 22 July 2020].

Chase, J., Topp, R., Smith, C.E., and Cohen, M.Z. (2012) Time Management Strategies For Research Productivity. *Western Journal of Nursing Research* [online]. 35 (2), pp. 155-176. [Accessed 22 July 2020].

Denys Yevenko (2020) Focus Matrix (1.4) [computer program] Available from: <https://apps.apple.com/gb/app/focus-matrix-task-manager/id1107872631> [Accessed: 17 July 2020]

Hudaib, A., Qasem, M.A.H., Masadeh, R.M.T. and Alzagebah, A.I. (2018) Requirements Prioritisation Techniques Comparison. *Modern Applied Science* [online]. 12 (2), pp. 62-80. [Accessed 16 July 2020].

Jain bu (2020) Flat Tomato (15.5) [computer program] Available from: <https://apps.apple.com/gb/app/flat-tomato-time-management/id719462746> [Accessed: 17 July 2020]

Lausen, S. (2002) *Software Requirements Styles and Techniques*. Great Britain: Biddles Ltd.

McGrow Centre for Teaching and Learning (2016) *Principles of Effective Time Management for Balance, Well-being and Success*.

Mishra, M.K. and Rashid, F. (2014) An Improved Round Robin Cpu Scheduling Algorithm with Varying Time Quantum. *International Journal of Computer Science, Engineering and Applications* [online]. 4 (4) [Accessed 25 July 2020].

Noon, A., Kalakech, A. and Kadry, S. (2011) A New Round Robin Based Scheduling Algorithm For Operating Systems: Dynamic Quantum Using the Mean Average. *Ijcsi International Journal of Computer Science Issues* [online]. 8 (3), pp. 224-229. [Accessed 29 July 2020].

Skills for Learning (2018) *Guide to time management* [online]. Wolverhampton: University of Wolverhampton. [22 July 2019]. Available at: <http://www.wlv.ac.uk/skills>.

Toggl OU (2006) Toggl (2.17.1) [computer program] Available from: <https://apps.apple.com/gb/app/toggl-time-tracker-work-log/id1291898086> [Accessed 15 June 2020]

Tracy, B. (2013) *Time Management*. United States of America: American Management Association.