



# SOFTWARE PROJECT TECHNICAL REPORT: GYM BUDDIES

BSHC4 | Mobile Stream | NCI | May 2017

## Abstract

Technical report documentation for the Gym Buddies iOS application. This document outlines key technical details of the Gym Buddies application. Content includes research, requirements, use cases, data requirements, analysis and design, testing, usability, GUI design and layout and class diagrams.

Mark Kirby – X13114484  
X13114484@student.ncirl.ie

## Declaration Cover Sheet for Project Submission

### SECTION 1 *Student to complete*

<b>Name:</b> Mark Kirby
<b>Student ID:</b> X13114484
<b>Supervisor:</b> Eamon Nolan

### SECTION 2 **Confirmation of Authorship**

*The acceptance of your work is subject to your signature on the following declaration:*

I confirm that I have read the College statement on plagiarism (summarized overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

**Complete the sections above and attach it to the front of one of the copies of your assignment.**

## **What constitutes plagiarism or cheating?**

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a particular piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

## **Penalties for Plagiarism**

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

1	Executive Summary .....	8
2	Project Introduction & Background/Research .....	8
2.1	Introduction & Aims .....	8
2.2	Project Background: Customer Survey.....	9
2.3	Project Background: Competition Research.....	10
2.4	Research Analysis & Conclusions .....	10
2.5	Technologies/Resources .....	11
3	User Requirements Definition .....	11
3.1	Product Perspective.....	11
3.2	Product Functions .....	11
3.3	User Characteristics .....	12
4	System Architecture.....	13
4.1	Operating Environment.....	13
4.2	General Constraints.....	13
4.3	Assumptions & Dependencies .....	13
4.4	Architecture Diagram.....	14
5	System Requirements Specification .....	14
5.1	External Interface Requirements.....	14
6	Data Requirements.....	15
6.1	User Data .....	15
6.2	Workout Program Data .....	15
7	Functional Requirements .....	16
7.1	User.....	16
7.2	Create an Account.....	16
7.3	Login.....	16
7.4	Submit Exercise Program .....	16
7.5	View Existing Programs .....	17
7.6	Rate a Program.....	17
7.7	Save a Program to Personal List .....	17
7.8	Edit Personal Program .....	17
7.9	Send Message.....	17
7.10	Find Gym Nearby .....	18
8	Non Functional Requirements .....	18
8.1	Portability.....	18
8.2	Reliability.....	18
8.3	Ease of Use .....	18
8.4	Speed.....	18
8.5	Size.....	18
8.6	Privacy .....	19
8.7	System Stress Management .....	19
9	Use Cases.....	20
9.1	Use Case Index .....	20
9.2	Use Case #1 – Create Account.....	21
9.3	Use Case #2 – Login.....	23
9.4	Use Case #3 – Submit Program .....	25
9.5	Use Case #4 – View Program .....	27
9.6	Use Case #5 – Rate Program .....	29
9.7	Use Case #6 – Save Program.....	31
9.8	Use Case #7 – Edit Program .....	33
9.9	Use Case #8 – Send Message.....	35

9.10	Use Case #9 – Find Gym .....	37
10	Storyboard/Class Diagram .....	39
11	Analysis & Design Specification .....	39
12	Analysis & Design Introduction .....	39
12.1	Purpose.....	39
13	General Overview & Design Guidelines .....	39
13.1	Assumptions, Constraints & Standards.....	40
14	Architecture Design .....	40
14.1	Logical View.....	40
14.2	Hardware Architecture.....	41
14.3	Software Architecture .....	42
14.4	Security Architecture .....	43
14.5	Communication Architecture.....	44
14.6	Performance .....	44
15	System Design .....	45
15.1	Database Design & Data Conversions .....	45
15.2	Application Program Interfaces .....	46
16	Implementation .....	47
16.1	Register New User .....	47
16.2	Sign in Registered User .....	48
16.3	Post New Exercise Program.....	48
16.4	Loading Exercise Programs from Database into Table View.....	49
16.5	Save Public Program to Personal List.....	50
16.6	Message Handling & Retrieval.....	50
16.7	Centralized Navigation and Signing Out .....	52
17	GUI Design & Layout.....	53
17.1	App Icon.....	53
17.2	App Launch Screen.....	54
17.3	Sign in Screen .....	55
17.4	Register Screen .....	56
17.5	Home Screen.....	57
17.6	Post Program Screen .....	58
17.7	All Programs Screen .....	59
17.8	My Programs Screen.....	60
17.9	Community Chat Screen .....	61
18	Testing & Usability.....	62
18.1	5 Second Test.....	62
18.2	Think Aloud test .....	63
18.3	Trunk Test.....	63
18.4	Usability Testing: Heuristic Evaluation.....	64
18.5	Testing: Conclusions.....	66
19	System Evolution Moving Forward.....	66
19.1	With Further Development & Research .....	66
20	Bibliography .....	67
21	Appendix.....	68
22	Project Proposal .....	68
23	Project Objectives .....	68
24	Project Background.....	68
25	Project Functionality.....	69
25.1	User Sessions & Authentication .....	69

25.2	Submit & View User Submitted Programs .....	69
25.3	My Programs.....	70
25.4	Community Discussion & Private Messaging .....	70
25.5	Gyms Nearby .....	71
26	Target Audience.....	71
27	Technical Approach.....	71
28	Special Resources Required.....	72
29	Project Plan .....	73
30	Technical Details .....	74
31	Project Evaluation.....	74
32	Reflective Journal: September .....	75
32.1	September 19 <sup>th</sup> .....	75
32.2	September 20 <sup>th</sup> .....	75
32.3	September 23 <sup>rd</sup> .....	76
32.4	September 26 <sup>th</sup> .....	76
32.5	September 30 <sup>th</sup> .....	76
32.6	October 1 <sup>st</sup> .....	76
32.7	October 2 <sup>nd</sup> .....	77
33	Reflective Journal: October .....	77
33.1	October 5 <sup>th</sup> .....	77
33.2	October 6 <sup>th</sup> .....	77
33.3	October 10 <sup>th</sup> .....	77
33.4	October 12 <sup>th</sup> .....	77
33.5	October 14 <sup>th</sup> .....	78
33.6	October 15 <sup>th</sup> .....	78
33.7	October 16 <sup>th</sup> .....	78
33.8	October 18 <sup>th</sup> .....	78
33.9	October 19 <sup>th</sup> .....	78
33.10	October 20 <sup>th</sup> .....	78
33.11	October 21 <sup>st</sup> .....	78
33.12	October 24 <sup>th</sup> .....	79
33.13	October 28 <sup>th</sup> .....	79
33.14	October 30 <sup>th</sup> .....	79
34	Reflective Journal: November .....	79
34.1	This Month.....	79
34.2	Next Month .....	79
34.3	Reflection.....	79
35	Reflective Journal: December.....	80
35.1	This Month.....	80
35.2	Next Month .....	80
35.3	Reflection.....	80
36	Reflective Journal: January.....	80
36.1	This Month.....	80
36.2	Next Month .....	81
36.3	Reflection.....	81
37	Reflective Journal: February.....	81
37.1	This Month.....	81
37.2	Next Month .....	82
37.3	Reflection.....	82
38	Reflective Journal: March.....	82

38.1	This Month.....	82
38.2	Next Month.....	82
38.3	Reflection.....	83

# 1 Executive Summary

Gym Buddies is a community based health and fitness mobile application aiming to curb the rising trend in obesity levels and increase physical activity by tapping into the huge market penetration of smart phone devices. The application offers an endless supply of workout programs and health and fitness advice through user generated content as well as a community where users can advise and aid each other with their health and fitness goals.

Studies by the World Health Organization show that Ireland is set to become the most obese nation in Europe by 2030. The purpose of Gym Buddies is to help people avoid becoming a statistic and increase their quality of life. According to a survey carried out by Eir (Formerly Eircom), 70% of the Irish population now owns a smart phone, this affords a huge opportunity to change people's behaviors through a device most consider essential to their daily lifestyle.

Gym Buddies aims to be a one stop workout companion for a user looking to exercise and improve their personal health. A survey carried out during the gathering of requirements showed a very positive response to the premise of using a mobile application for the purposes outlined above and an overwhelming majority confirmed they already use online sources to put together their workout programs. Respondents also agreed they were more likely to stick with exercising as part of a group or in collaboration with other people. These findings show that by combining all of the things people find useful into one application, Gym Buddies has the potential to fill a gap in the market and help people get active and improve their health.

## 2 Project Introduction & Background/Research

### 2.1 Introduction & Aims

Gym Buddies is a community based health and fitness mobile application. It aims to offer a social environment for people interested in health and fitness. The application will be fueled by user generated content, users will be able to submit and view exercise programs within the application. All programs in the application will be submitted by users of the application, so if a user finds a certain program works really well for them or finds a useful workout on the internet, they can submit it to the application for other users to try out. Users will be able to rate programs submitted to the application, this will help ensure high quality content within the application.

Users can browse and try out programs as provided in the app, or they can save a program to their own personal list and make tweaks to them in order to better suit their needs. A user could have an injury or other health related reason not to do a certain exercise so they can then switch out certain ones as they wish from the program they are using.

The main focus of the application is to create a positive community for like-minded people to have a place to get together and help each other in achieving a common goal. The application will provide a discussion section where people can discuss the programs in the application and any other health and fitness topic they wish. Additionally, a one to one messaging system will be attempted, this serves many purposes but the main one is to help users pose any questions they may have about a program to its author.

This idea was chosen as there is huge interest in the health and fitness industry and activity in this area continues to rise. While the health and fitness industry continues to grow at a rapid pace, unfortunately, so too do obesity levels. Ireland is on track to become the most obese nation in Europe by 2030 so another tool to help tackle this, as well as encouraging people to be more active, can only be a good thing. The application aims to give people a tool to help them be their best self and turn the tide on obesity. To this end, the application will also have a “gyms nearby” feature. As the name suggests, this feature will show users the closest gyms to their current location plotted on a map.

## 2.2 Project Background: Customer Survey

As part of background research into the viability of the application a customer survey was carried out at a local Gym in north County Dublin, there were 36 respondents. The aim of the survey was to establish how open gym users are to the main ideas and functionality of the application. The survey consisted of five questions and was left at the sign in desk for anyone interested in participating. The questions were as follows:

- 2.2.1 Q1: Would you use a workout companion to aid you with your workouts? Y/N
- 2.2.2 Q2: Would you find an application that enables you to access and manage your workouts and programs at any time useful? Y/N
- 2.2.3 Q3: Do you often use the internet or other public sources to find workouts, programs or fitness advice to aid in your own training? Y/N
- 2.2.4 Q4: Would you feel more encouraged to exercise as part of a group or in collaboration with others? Y/N
- 2.2.5 Q5: Do you feel the ability to customize a workout or program to suit your personal needs is important? Y/N

### **Feedback:**

Q1: 77.8% of people responded positively to question 1.

Q2: 72.2% of people responded positively to question 2.

Q3: 83.3% of people responded positively to question 3.

Q4: 61.1% of people responded positively to question 4.

Q5: 88.8% of people responded positively to question 5.

Feedback to the questions above was generally positive and reinforced the idea that people are far more willing to embrace technology to aid them with their health and fitness than ever before. Most people were willing to use an application to manage their workouts and many of them often used the internet to source information and find workouts. Over 60% of people surveyed indicated they felt more encouraged to exercise as part of a group or in collaboration with others. An overwhelming majority (88.8%) of people indicated it was very important that they had the ability to customize their workouts to better suit their needs.

## 2.3 Project Background: Competition Research

As part of the research into the viability of the application, research was carried out on three applications currently in existence that serve a similar purpose to the proposed application. Details of the applications and the comparisons made are listed below:

- 2.3.1 **Gym Workout Tracker & Trainer:** This application allows users to browse a list of premade programs and edit them if they wish. However, the list of programs on selection are static and won't change dynamically. The application also does not offer any social or community based functionality. Gym Buddies supports dynamic user generated content and has a virtually endless source of workout programs. Gym Buddies also offers social functionality to foster a sense of community and offer a positive environment to encourage exercising.
- 2.3.2 **Daily Workouts:** This application supplies the user with premade, static workouts. The workouts provided are heavily focused around the idea of working out at home. Unlike Gym Buddies, users of this application will not enjoy dynamic user generated programs, users are unable to edit programs, there are no social features and a lot of the applications features are only available after purchases are made.
- 2.3.3 **Fitness & Bodybuilding:** This application allows users to browse premade programs and to create their own. Users are not able to edit existing programs or submit their own for other users. The application also has no community functionality.

## 2.4 Research Analysis & Conclusions

Based on feedback given in the questionnaire, most people are happy to use a mobile application to help them access and manage their exercise programs. Most people surveyed often use the internet as a source of information to help them workout and a majority of the people who participated said they find the ability to customize their programs to suit their needs very important. A lot of people also indicated they feel more encouraged to exercise as part of a social setting.

When evaluating applications currently on the market, it was established that the applications researched did not allow user submitted content, comprehensive editing of workouts and did not offer any social features.

Gym Buddies aims to plug the gap in these areas and offer a one stop place for people to develop their health and fitness goals and to socialize with like-minded people who share similar goals.

## 2.5 Technologies/Resources

Technology/Resource	Description
Apple Mac Device	Developing iOS applications requires the XCode Integrated Development Environment. This IDE is not available on Windows so an Apple Mac is required.
XCode	XCode is the Integrated Development Environment used to create applications for iOS and other Apple Devices.
iPhone Simulator	The iPhone Simulator is an emulator used to test iOS applications outside of their native environment. It is used to run iOS applications created in XCode on your Mac device without the need for a physical iPhone device.
iPhone Device	An eventual goal is to run the application on the device it was designed to run on. While the simulator will serve all required purposes during early development, full testing and demonstrations will require the application to run on a physical device.
Apple ID	iOS development requires an Apple ID, this is then used to create a Developer account which is used to allow testing of your code on physical devices that have your Apple ID signed in.
Google Firebase	A No SQL, real-time, online database used to permanently store application data.

## 3 User Requirements Definition

### 3.1 Product Perspective

The software will be written in Swift 3 using Xcode 8. The software will run on iOS, Apple's mobile platform for the iPhone and iPad. It will provide users with the ability to find or submit workout routines inside the application. Users will be able to register an account and log in using their details to take part in an online community and gain access to user submitted content. Users can take part in public group discussions or one on one private conversations.

### 3.2 Product Functions

- 3.2.1 Users will be able to register to the application with their personal details. Initially this will be via email and password but future changes could allow for Google and Facebook log in functionality.
- 3.2.2 Registered users will be able to sign in and post a workout routine to the application for others to try out.
- 3.2.3 Registered users will be able to sign in and browse a list of user submitted exercise programs and choose one they like to try out for themselves.

- 3.2.4 Registered users will be able to sign in and browse a list of user submitted exercise programs and save them to their own personal list. This serves the purpose of allowing users to edit a routine they find to better suit their needs. Some users may have a medical reason preventing them from doing certain exercises or they may just want to switch it up a bit to better suit their goals.
- 3.2.5 Registered users of the application will be able to participate in group discussions within the application or opt to communicate one on one with another single user. This allows users to discuss things as part of a group and meet new people and also allows them to discuss things privately or ask the author of a program a question should they need clarification on any aspect of their program.
- 3.2.6 Registered users will be able to apply positive or negative ratings to programs hosted within the application. This will help ensure the quality of the applications content and programs listed will be displayed by most positive votes, to most negative votes.
- 3.2.7 Registered users will be able to use the application to find a Gym/Sports/Leisure Center nearby. This will be useful for people who are new to exercising as they may need to find a gym near their place of employment or their home.

### 3.3 User Characteristics

#### **Unregistered User**

Unregistered users will not be able to gain access to the application or use any of its features. The whole focus of the application is on user submitted content and social interactions, none of this is possible without an account alias to tie a user to. Additionally, the services offered by the application will be behind a ToS (Terms of Service) agreement, this model was chosen as users will be following programs submitted by other users completely at their own risk. No liability is accepted by the developer of the application or any parties in connection with the developer.

#### **Registered User**

Registered users will gain full, non admin access to the application. Once a user registers their details and agrees to the ToS they will be able to post workout routines, view workout routines, rate workout routines, save routines to their personal list for edits, chat with other users and use the gyms nearby feature.

#### **Administrator**

The administrator will maintain the data in the application behind the scenes. The administrator will deal with managing user accounts if any bans are levied against a user. They will also manage workout program data based on user feedback. If a program receives enough negative feedback it will need to be removed from the application by the administrator. The administrator will also carry out day to day housekeeping within the application and ensure the quality of the application content.

## 4 System Architecture

### 4.1 Operating Environment

- 4.1.1 The application is designed to work on all devices that are running iOS version 8 or later. Currently this includes the iPhone family, models 5 through 7, and the iPad, generations 2 through 4. See architecture diagram under its heading below.
- 4.1.2 The application will store all data in a Google Firebase database. The application will require an online connection at all times to allow for user sign on, user communication, mapping and location functionality and access to online user submitted content.
- 4.1.3 Development of the software will be carried out in Xcode 8 using Swift 3. The software will run natively on Apple devices capable of running iOS version 8 or later.

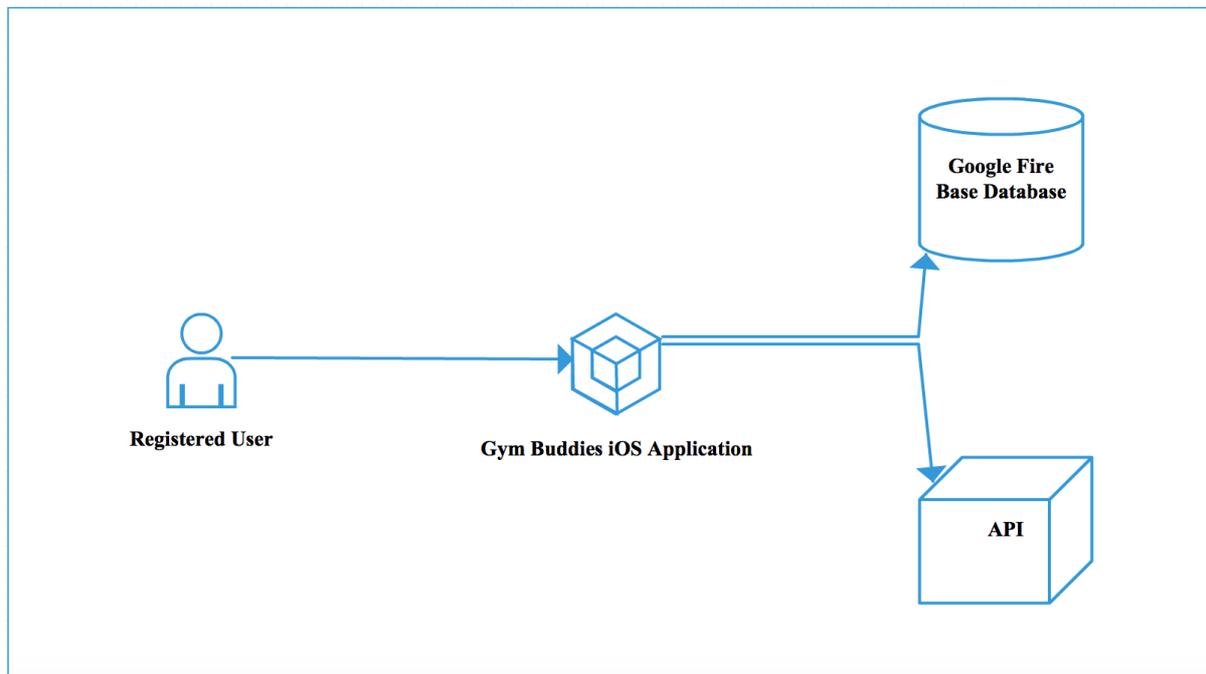
### 4.2 General Constraints

- 4.2.1 The software requires an Apple device running iOS version 8 or later with a touch screen and simulated keyboard.
- 4.2.2 The ability of the software to interface with an online database must be implemented.
- 4.2.3 The software must be able to access the devices GPS to offer location based functionality.

### 4.3 Assumptions & Dependencies

- 4.3.1 If no exercise routines are created by users, there will be no routines listed in the application for users.
- 4.3.2 If no internet connection is available, users will be unable to log in or access any functionality of the software.
- 4.3.3 Unregistered users will have no access to the software.
- 4.3.4 Users will only be permitted to register and gain access to the software once they agree to the terms of service agreement.

## 4.4 Architecture Diagram



## 5 System Requirements Specification

### 5.1 External Interface Requirements

#### User Interface

- 5.1.1 The user interface shall offer the user a logical representation of what the software is asking the user to do. Dropdown menus and buttons should be used where possible to aid the user. Input hints shall be used to aid the user when entering data.
- 5.1.2 The application should have its logo present on each screen once a logo has been designed.
- 5.1.3 A user friendly color scheme should be chosen, UI design should be carried out with visually impaired and color blind users in mind.
- 5.1.4 The UI should have well defined constraints to ensure that the software displays correctly on the screens of all compatible devices. The UI should display in both portrait and landscape.
- 5.1.5 The GUI should have continuity, all screens should have the same design and layouts should be consistent.

## Hardware Interfaces

- 5.1.6 The system shall be operated with a compatible Apple device using the devices touch screen, virtual keyboard and GPS location hardware.

## API Interfaces

- 5.1.7 The software must store user information and user submitted content in a Google Firebase database using Cocoa Pod files to achieve communications between the database and the application.
- 5.1.8 The software must show mapping information around the user's current location using the Google maps API.

# 6 Data Requirements

## 6.1 User Data

- 6.1.1 A user's email address will need to be stored in order to give each user an alias to operate the application under. The email address will be used for account validation and to tie a user to their content.
- 6.1.2 A user will need to create a password in order to verify themselves when accessing the system. The password will need to be stored in the system and tied to a user's email address.

## 6.2 Workout Program Data

- 6.2.1 The application will need to manage the input and display of public workout programs. All public programs will be added to the application by the user and will be available for use by other users. Workout program data will include, but not be limited to, Workout Name, Workout Type, Workout Duration, Exercise Name, Exercise Sets, Exercise Reps and Exercise Instructions.
- 6.2.2 The application will need to manage the input and display of personal workout programs. All personal programs will be saved from the public workouts section and edited by the user pulling down that data. Personal workouts will be stored separately to public workouts and are tied to an individual user and is only visible to them. Workout program data will include, but not be limited to, Workout Name, Workout Type, Workout Duration, Exercise Name, Exercise Sets, Exercise Reps and Exercise Instructions.

## 7 Functional Requirements

### 7.1 User

7.1.1 All users of the software shall have the ability to create an account which is used to store user data and tie user actions to a user alias.

*7.1.1.1 User registration and login shall be mandatory.*

### 7.2 Create an Account

7.2.1 The system should provide the user with an easy to use GUI to facilitate their creating an account.

7.2.2 The system shall ask for an email address and password.

7.2.3 The system shall notify the user if incorrect characters are used in the email or password fields.

7.2.4 The system should notify the user if their email has already been used.

7.2.5 The system should notify the user if any required fields are left empty.

7.2.6 The system should not allow the user to create weak or unsecure passwords.

*7.2.6.1 The system should explain how the submitted password is unsecure.*

7.2.7 The system should prevent the user from completing registration if the terms of service has not been agreed to.

### 7.3 Login

7.3.1 The system should provide a user friendly GUI to allow the user to login when the application launches.

7.3.2 The system should prompt the user for their email address and password.

*7.3.2.1 The system should notify the user if submitted information is incorrect.*

### 7.4 Submit Exercise Program

7.4.1 The system should provide an intuitive UI for logged in users to allow them to submit their exercise routine to the application.

*7.4.1.1 The system should prevent the user submitting a blank or empty routine.*

7.4.2 The system shall add successfully submitted routines to the Google Firebase Database.

7.4.3 The system should display user submitted routines from the Google Firebase database in the appropriate section of the application.

## 7.5 View Existing Programs

7.5.1 The system should provide intuitive and user friendly navigation to allow users to locate the current list of user submitted exercise routines.

7.5.2 Once selected, the system shall retrieve all user submitted programs from the Google Firebase database and display them to the user.

7.5.3 The system shall display full details of an exercise routine once one is selected by the user.

7.5.4 The system should allow quick and easy navigation between different routines in the list.

## 7.6 Rate a Program

7.6.1 The system should provide a button to rate a program. Programs can be rated up or down based on the level of success the user has with them.

*7.6.1.1 Programs will be displayed based on ratings. Lower rated programs will be pushed to the bottom of the list before being removed.*

## 7.7 Save a Program to Personal List

7.7.1 The system should provide a button to save a program from the public list of user submitted programs to their own personal list.

*7.7.1.1 Programs saved to a personal list are not visible to other users.*

## 7.8 Edit Personal Program

7.8.1 The system should provide an intuitive and user friendly UI to allow the user to view and manage their personal list of workout programs.

7.8.2 The system should allow editing of programs saved to a user's personal list.

*7.8.2.1 Multiple edits can be made and all changes must be saved in real time*

## 7.9 Send Message

7.9.1 The system shall provide an interface for sending messages between users.

7.9.2 Messages should be sent in real time and have no delays.

## 7.10 Find Gym Nearby

7.10.1 When selected, the system should display a Google maps page displaying the user's current location and all Gyms/Sports/Leisure Centers within a 4km radius.

7.10.2 The system should provide relevant information to the user acquired from Google maps.

*7.10.2.1 Such information could be opening hours, price etc.*

## 8 Non Functional Requirements

### 8.1 Portability

8.1.1 The application will be written in Swift 3 using the Xcode 8 IDE, the application will run natively on any Apple device running iOS 8 or later. This includes devices like the iPhone, iPad, iPad Air and iPod. Additionally, Windows Bridge can be used to migrate the application to UWP and run on Windows 10 devices once development has finished.

### 8.2 Reliability

8.2.1 The system should be extremely reliable and have an approximate up time of 99.999%.

8.2.2 In the event of a crash or any other error, the System should inform the user of any problems and gracefully terminate.

### 8.3 Ease of Use

8.3.1 The application should be user friendly and intuitive to use. GUIs should make their functions clear and navigation around the application should be straight forward.

*8.3.1.1 Users should be comfortable using the application after 20 minutes of use.*

### 8.4 Speed

8.4.1 The application should open and be ready to use within 10 seconds of being selected.

8.4.2 The UI should be quick and smooth with no delays between button presses and screen reaction.

8.4.3 All database reads/writes should take no longer than 5 seconds. If the database encounters any errors, a user friendly warning should be displayed to the user.

### 8.5 Size

8.5.1 The size of the software in relation to storage media should be no larger than 250MB.

## 8.6 Privacy

- 8.6.1 All data retained by the system will be stored in accordance with the Data Protection Act 1988 and the Data Protection (Amendment) Act 2003.

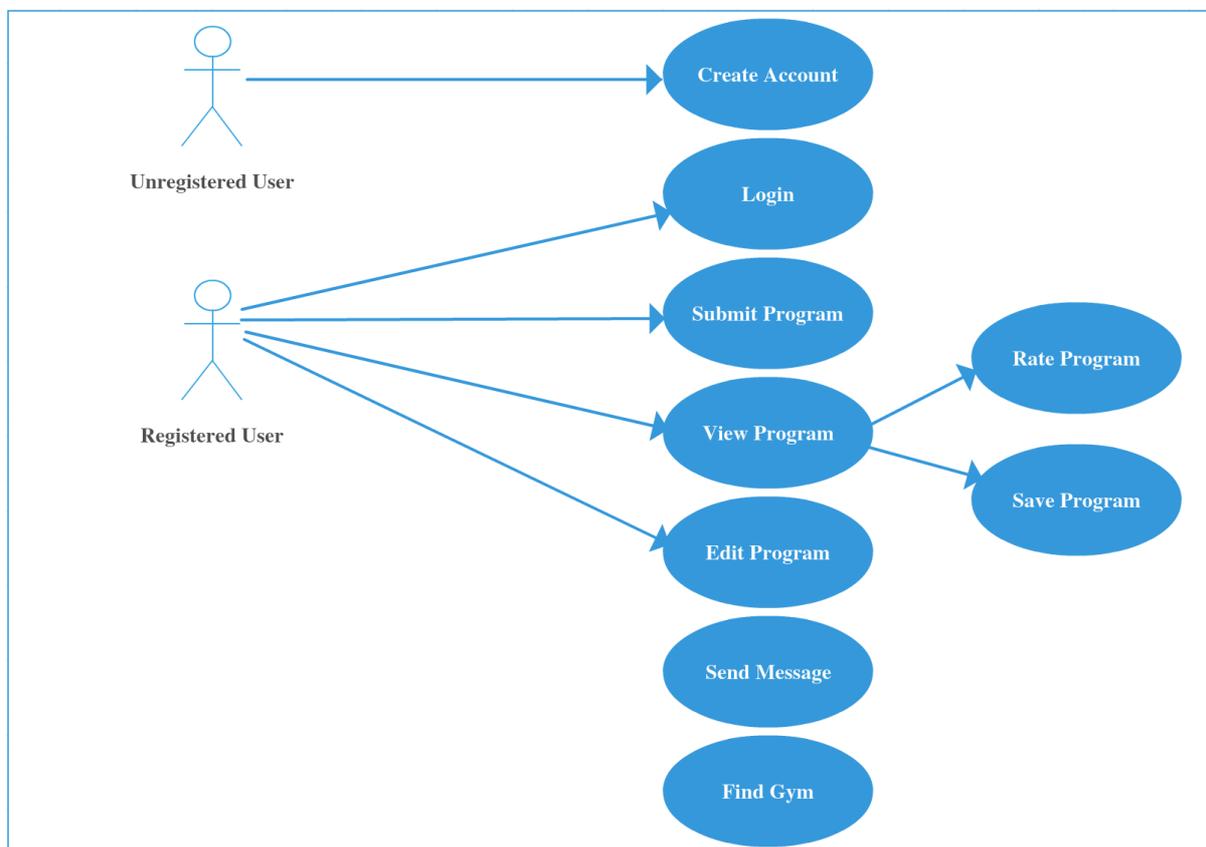
## 8.7 System Stress Management

- 8.7.1 The system will initially handle up to 100 simultaneous users. This is due to the Firebase database setup as it can only handle 100 simultaneous reads/writes at any one time on the basic plan.

## 9 Use Cases

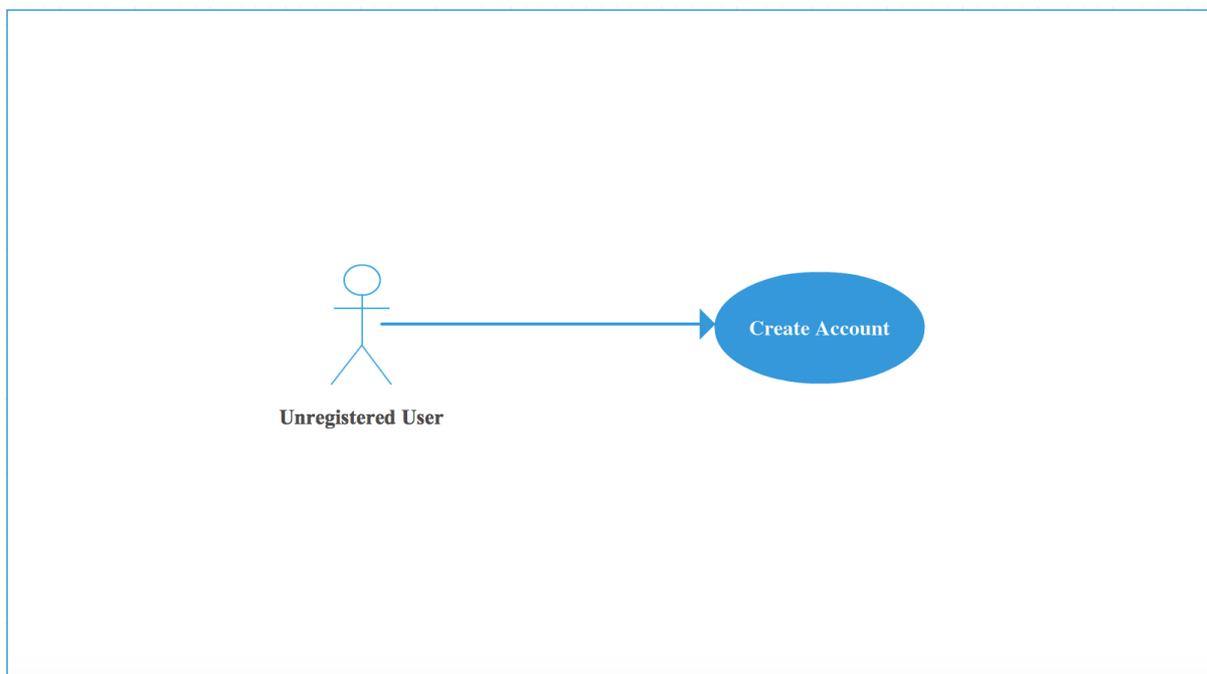
### 9.1 Use Case Index

Use Case ID	Use Case Name	Primary Actor	Scope	Complexity	Priority
1	Create Account	Unregistered User	In	Medium	High
2	Login	Registered User	In	Medium	High
3	Submit Program	Registered User	In	Medium	High
4	View Program	Registered User	In	Medium	High
5	Rate Program	Registered User	In	Low	Medium
6	Save Program	Registered User	In	High	Medium
7	Edit Program	Registered User	In	High	Medium
8	Send Message	Registered User	In	High	Medium
9	Find Gym	Registered User	In	Low	Medium

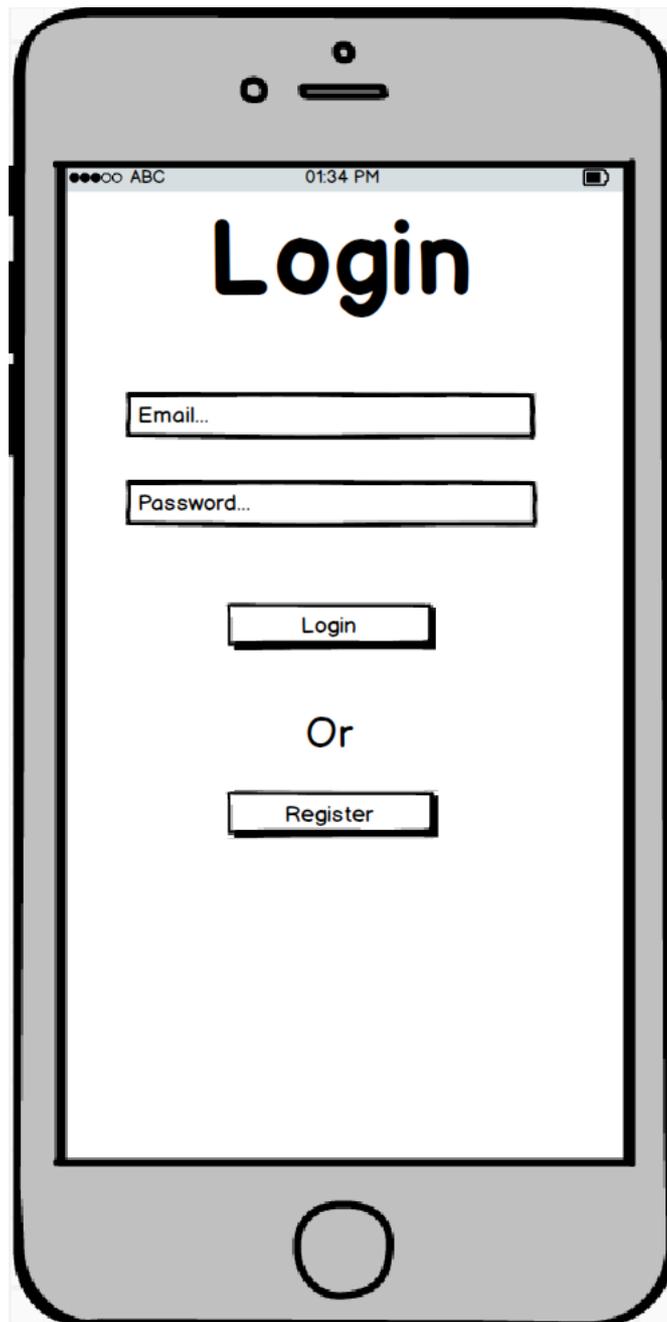


## 9.2 Use Case #1 – Create Account

Use Case Element	Description
Use Case Number	1
Application	Gym Buddies
Use Case Name	Create Account
Use Case Description	A user starts the application for the first time and is prompted to create an account.
Primary Actor	Unregistered User
Precondition	The application is started by a new user who does not have an account.
Trigger	The unregistered user taps the register button
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case starts when a new user launches the application for the first time and opts to create an account. [AF1]</li> <li>2. The user inputs their email and password into the allotted text fields. [AF2]</li> <li>3. The application notifies the user that the account has been created and grants them access to the application.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The user does not want to create an account and chooses to exit the application at that point.</li> <li>2. The user enters invalid characters into the text field or leaves them blank. The system notifies the user of their error.</li> </ol>
Termination	The flow is terminated once the user is successfully registered and their details stored in the database.
Post Condition	The use is returned to the Login page and the system enters a wait state.

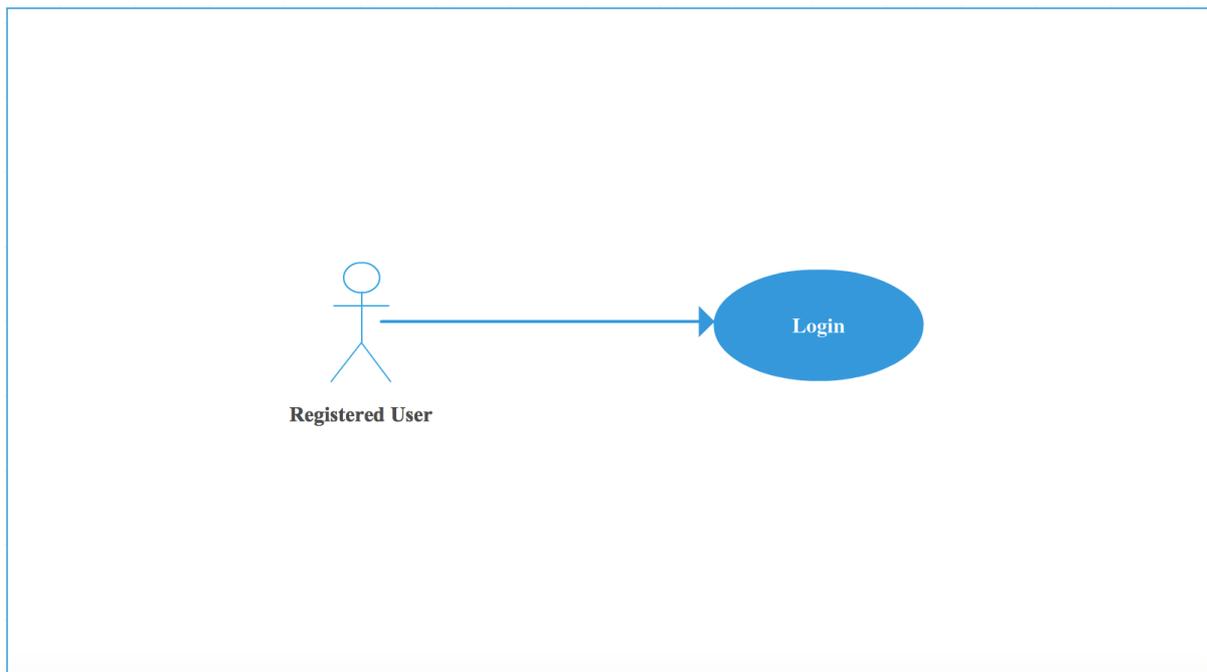


Use case #1 – Create Account Mockup

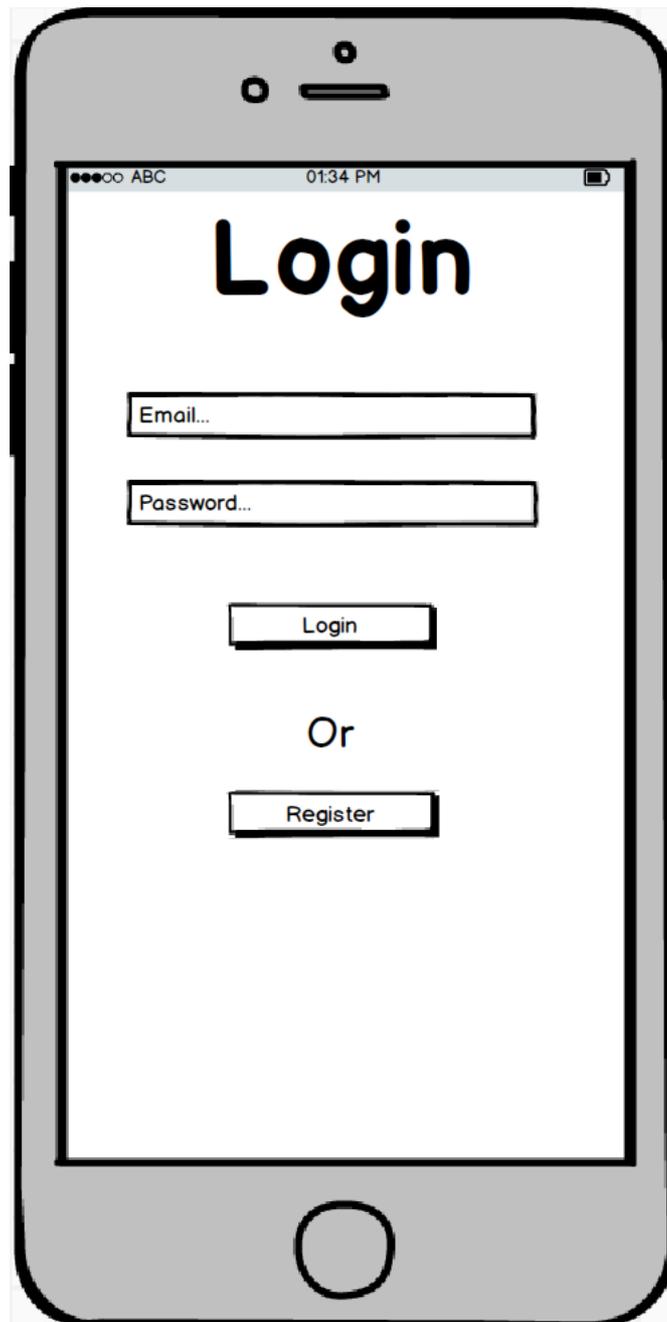


### 9.3 Use Case #2 – Login

Use Case Element	Description
Use Case Number	2
Application	Gym Buddies
Use Case Name	Login
Use Case Description	A registered user starts the application and is prompted to login using their personal details.
Primary Actor	Registered User
Precondition	The application is started by a registered user.
Trigger	A registered user starts the application
Basic Flow	<ol style="list-style-type: none"><li>1. This use case starts when a registered user runs the application and is presented with a login GUI page.</li><li>2. The user enters their username and password and is granted access to the system.[AF1]</li></ol>
Alternate Flows	<ol style="list-style-type: none"><li>1. The user inputs an incorrect username or password and is notified by the system.</li></ol>
Termination	The flow is terminated when the user is validated by the system and logged in to the application.
Post Condition	The user is brought to the application Home page and the system enters a wait state.

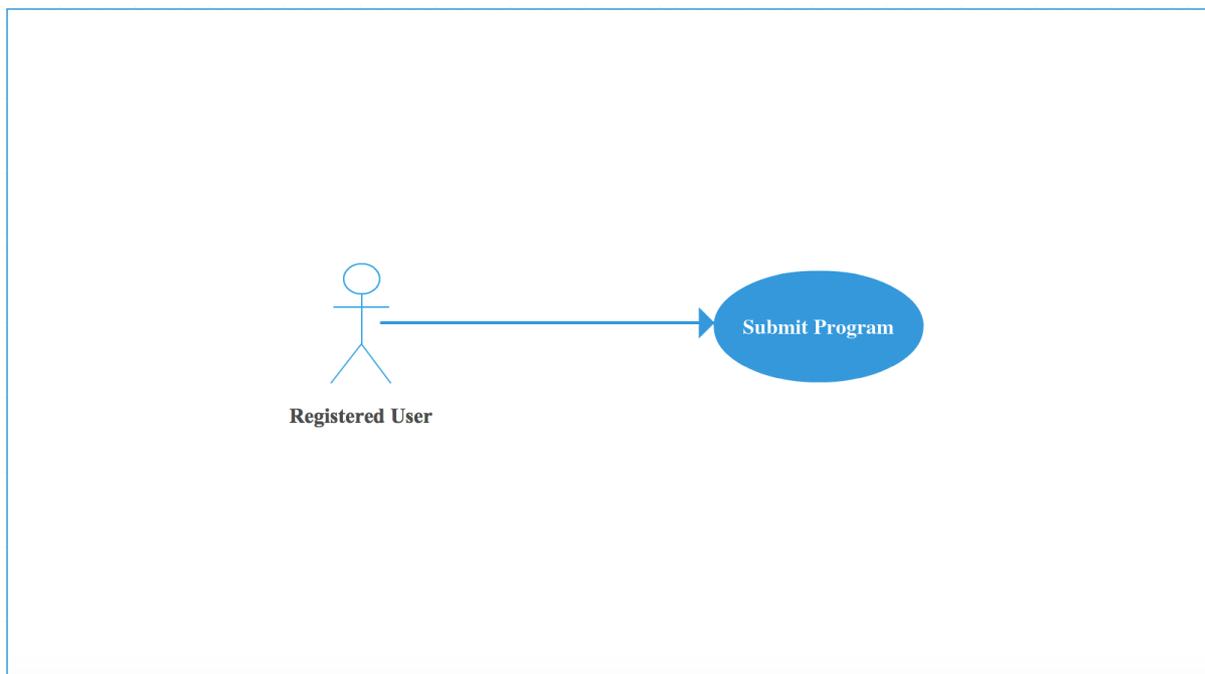


## Use Case #2 – Login Mockup

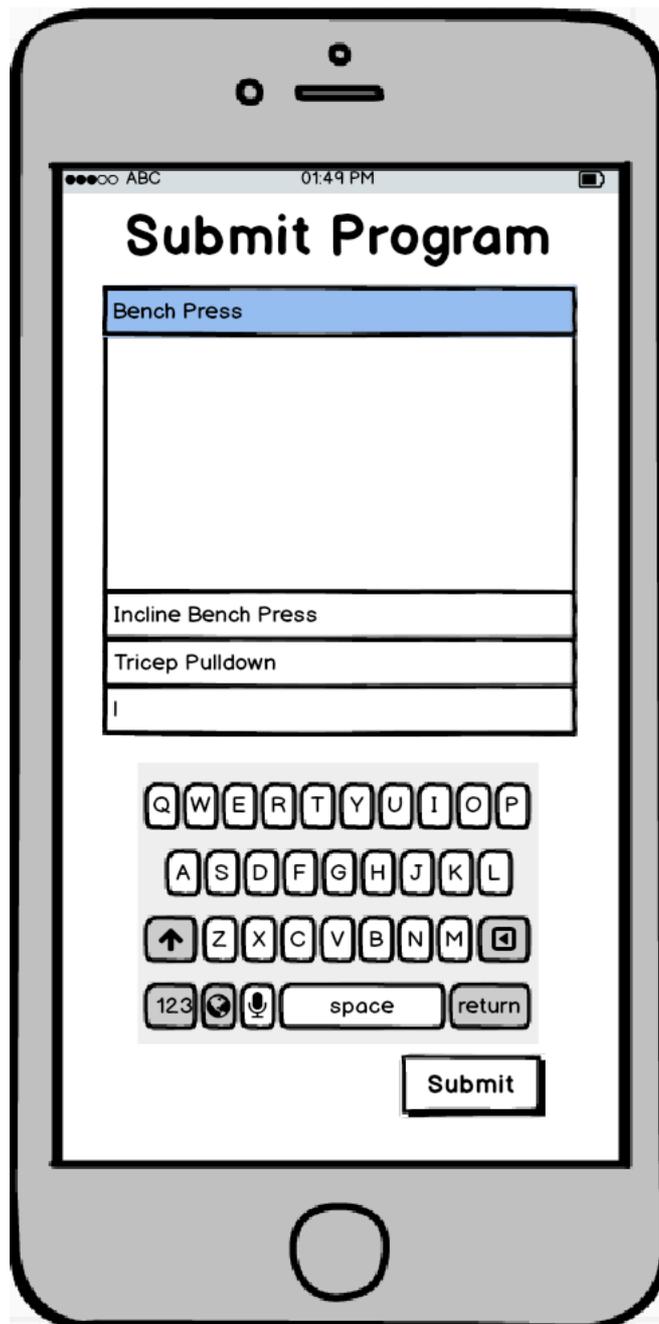


## 9.4 Use Case #3 – Submit Program

Use Case Element	Description
Use Case Number	3
Application	Gym Buddies
Use Case Name	Submit Program
Use Case Description	A registered user creates and submits an exercise program to the application.
Primary Actor	Registered User
Precondition	A registered user must be logged in.
Trigger	A registered user logs in and taps the Submit Program button.
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case begins when a logged in user clicks the Submit Program Button.</li> <li>2. The user proceeds to enter the program details into the supplied fields. [AF1]</li> <li>3. Once all details have been added the user then clicks the Submit button to add their program to the application.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The user enters invalid information or leaves some or all fields empty and the system informs the user of their error.</li> </ol>
Termination	The flow is terminated when the user successfully submits their program to the application and it is saved to the database.
Post Condition	The user is returned to the All Programs page and the system enters a wait state.

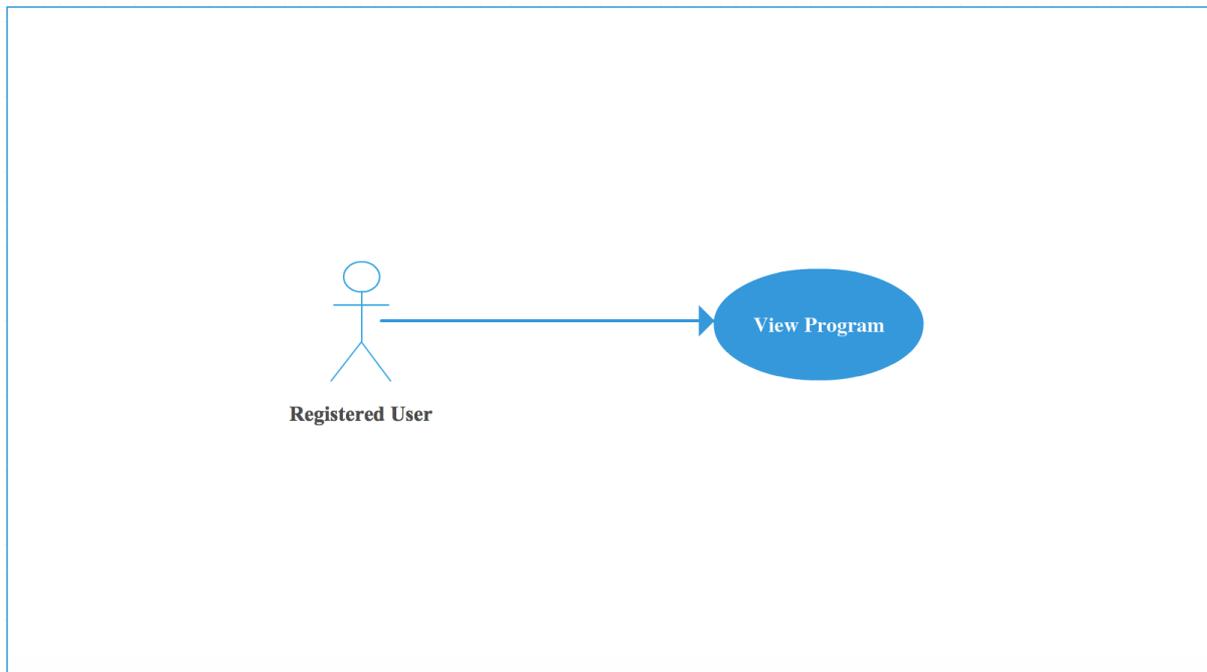


### Use Case #3 – Submit Program Mockup

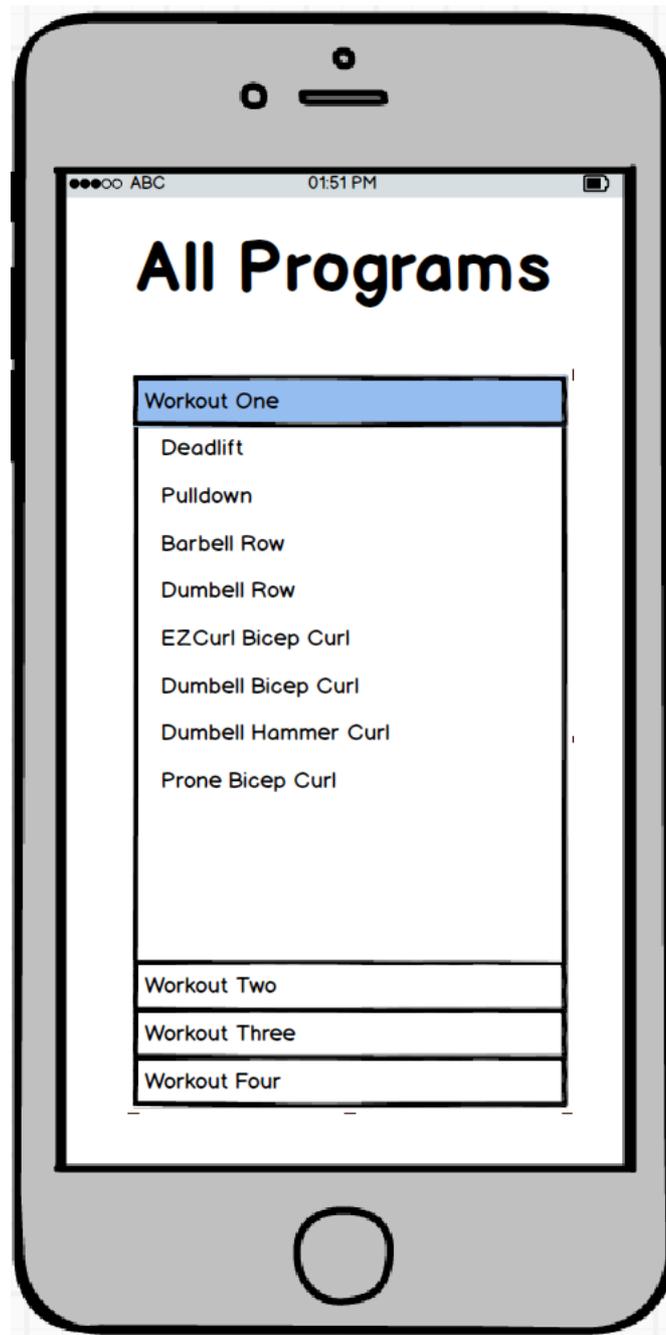


## 9.5 Use Case #4 – View Program

Use Case Element	Description
Use Case Number	4
Application	Gym Buddies
Use Case Name	View Program
Use Case Description	A registered user logs in and navigates to the All Programs section of the application.
Primary Actor	Registered User
Precondition	A registered user must be logged in. Programs must exist in the application.
Trigger	A registered user logs in and taps the All Programs button.
Basic Flow	<ol style="list-style-type: none"><li>1. This use case begins when a logged in user clicks the All Programs button.</li><li>2. The user is then presented with a list of all the programs hosted by the application. [AF1]</li></ol>
Alternate Flows	<ol style="list-style-type: none"><li>1. The application has no user submitted programs and has nothing to display to the user. The system notifies the user of this and returns them to the home page.</li></ol>
Termination	The flow is terminated when the user is presented with a list of all programs or is informed that there are no programs.
Post Condition	Depending on whether there are programs to show, the user is either brought to the All Programs page or the Home page.

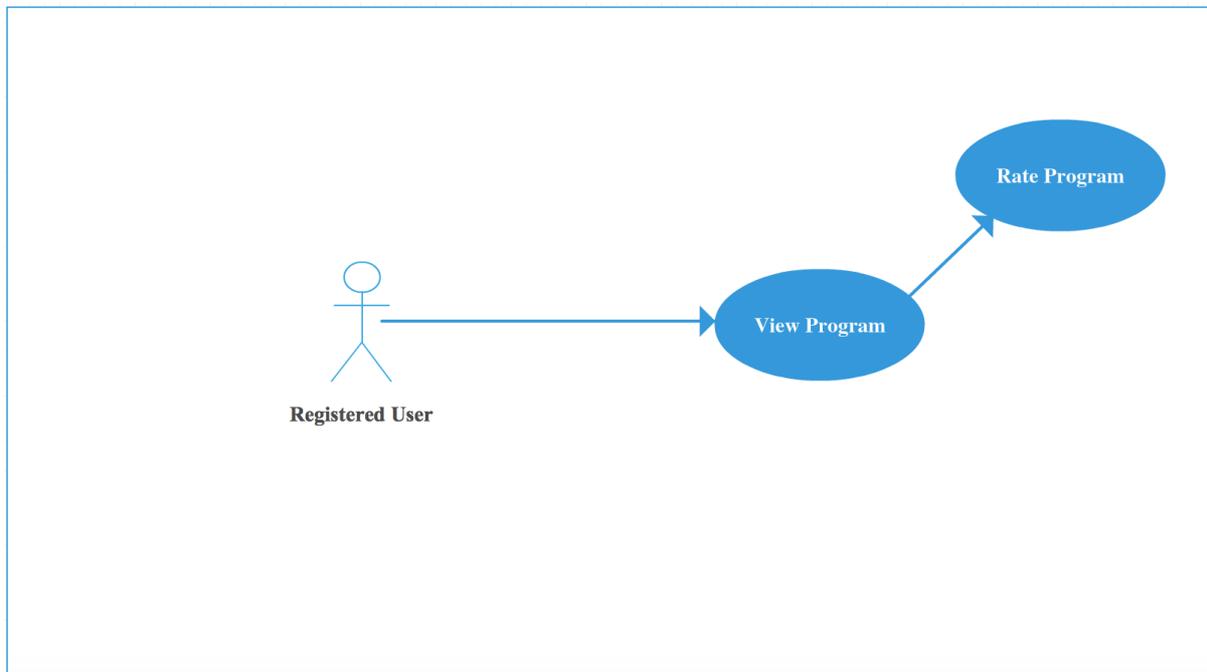


## Use Case #4 – View Program Mockup

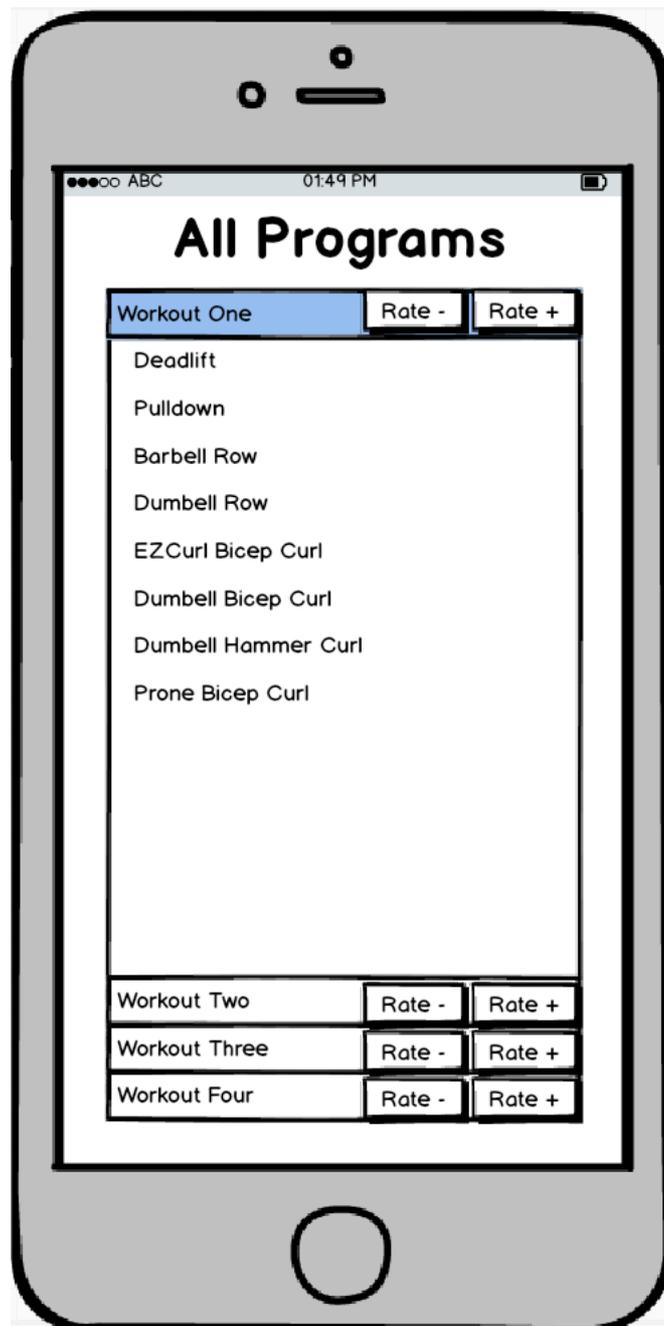


## 9.6 Use Case #5 – Rate Program

Use Case Element	Description
Use Case Number	5
Application	Gym Buddies
Use Case Name	Rate Program
Use Case Description	A registered user logs in and navigates to the All Programs section of the application and applies a rating to a listed program.
Primary Actor	Registered User
Precondition	A registered user must be logged in and on the All Programs page. Programs must exist in the application.
Trigger	A registered user logs in and taps the All Programs button then the Rate button.
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case begins when a logged in user clicks the All Programs button and then the Rate Button.</li> <li>2. The user is then presented with a list of all the programs hosted by the application. [AF1]</li> <li>3. The user then clicks the Rate button beside one of the programs to apply their positive or negative rating.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The application has no user submitted programs and has nothing to display to the user. The system notifies the user of this and returns them to the home page.</li> </ol>
Termination	The flow is terminated when the users rating is applied to the program and is saved to the database.
Post Condition	The user remains on the All Programs page and the system enters a wait state.

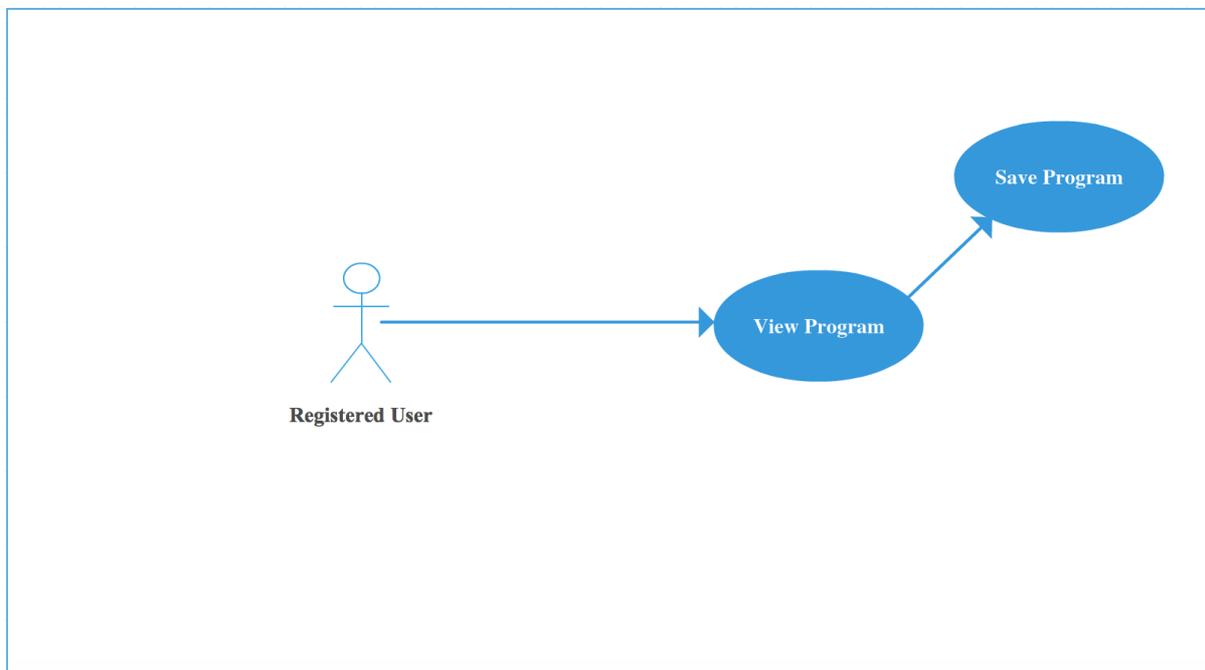


## Use Case #5 – Rate Program Mockup

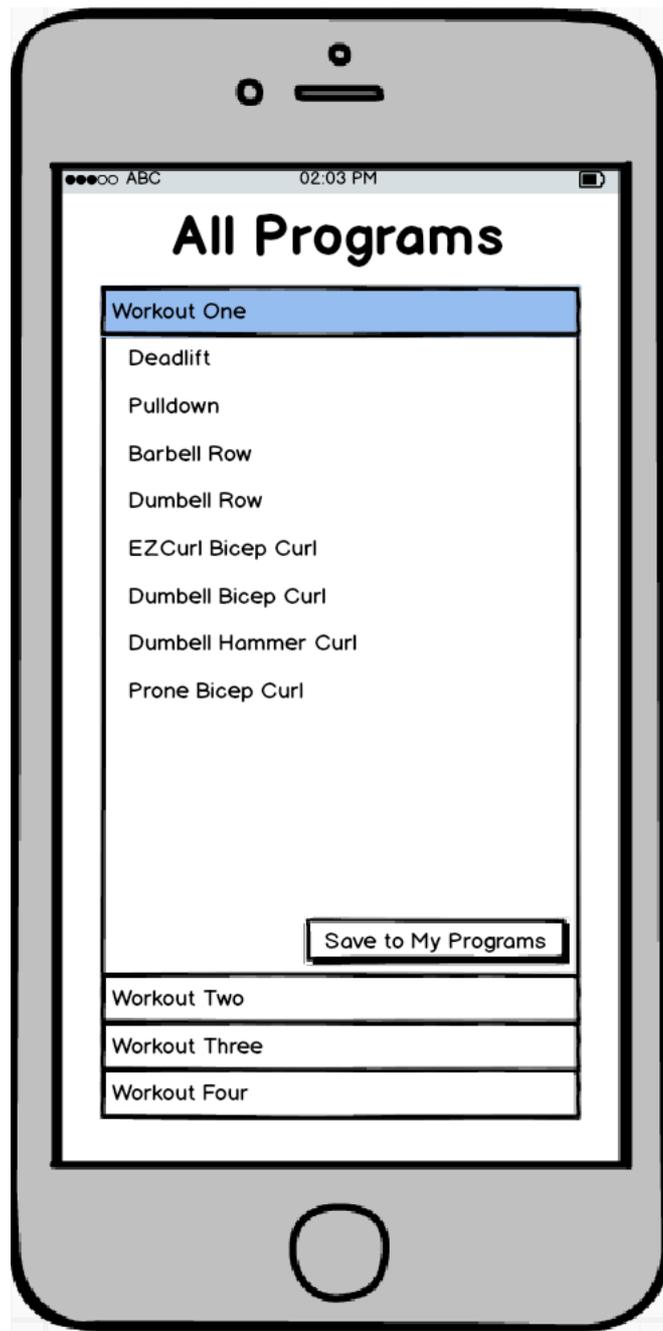


## 9.7 Use Case #6 – Save Program

Use Case Element	Description
Use Case Number	6
Application	Gym Buddies
Use Case Name	Save Program
Use Case Description	A registered user logs in and navigates to the All Programs section of the application and taps the Save Program button.
Primary Actor	Registered User
Precondition	A registered user must be logged in and on the All Programs page. Programs must exist in the application.
Trigger	A registered user logs in and taps the All Programs button then the Save Program button.
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case begins when a logged in user clicks the All Programs button and then the Save Program button. [AF1]</li> <li>2. When the user taps the Save Program button beside one of the programs it is saved to their personal list.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The application has no user submitted programs and has nothing to display to the user. The system notifies the user of this and returns them to the home page.</li> </ol>
Termination	The flow is terminated when the program is successfully saved to the user's personal list and the database is updated accordingly.
Post Condition	The user is returned to the My Programs page and the system enters a wait state.

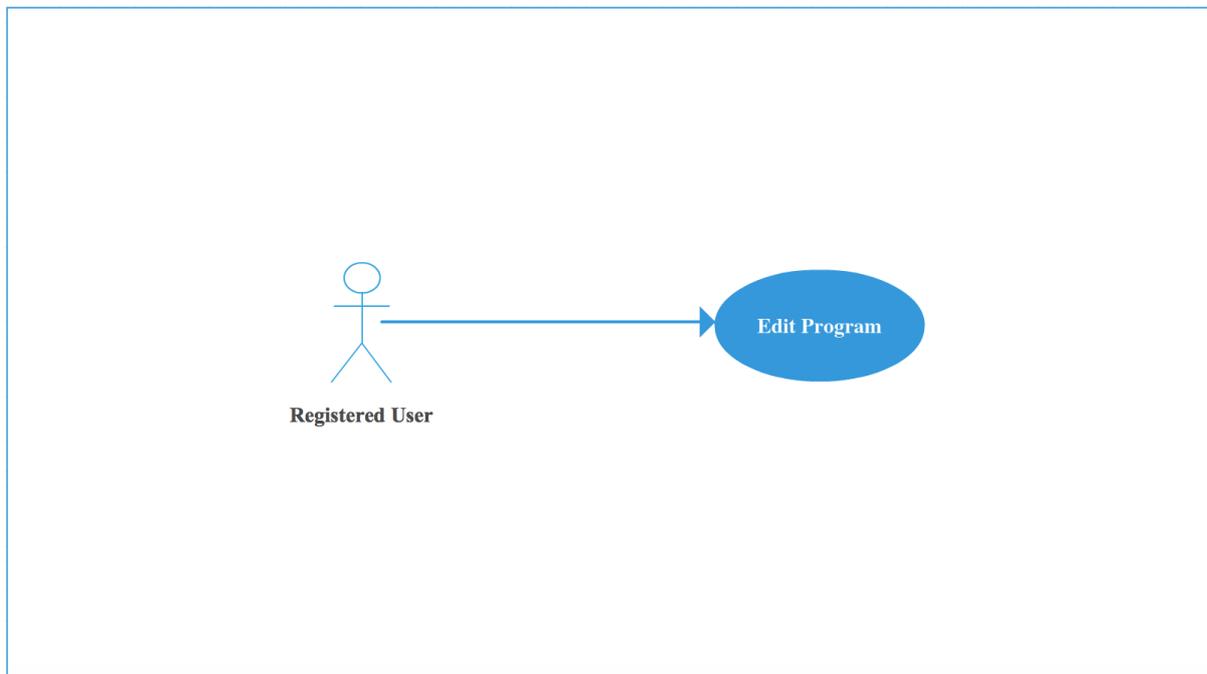


## Use Case #6 – Save Program Mockup

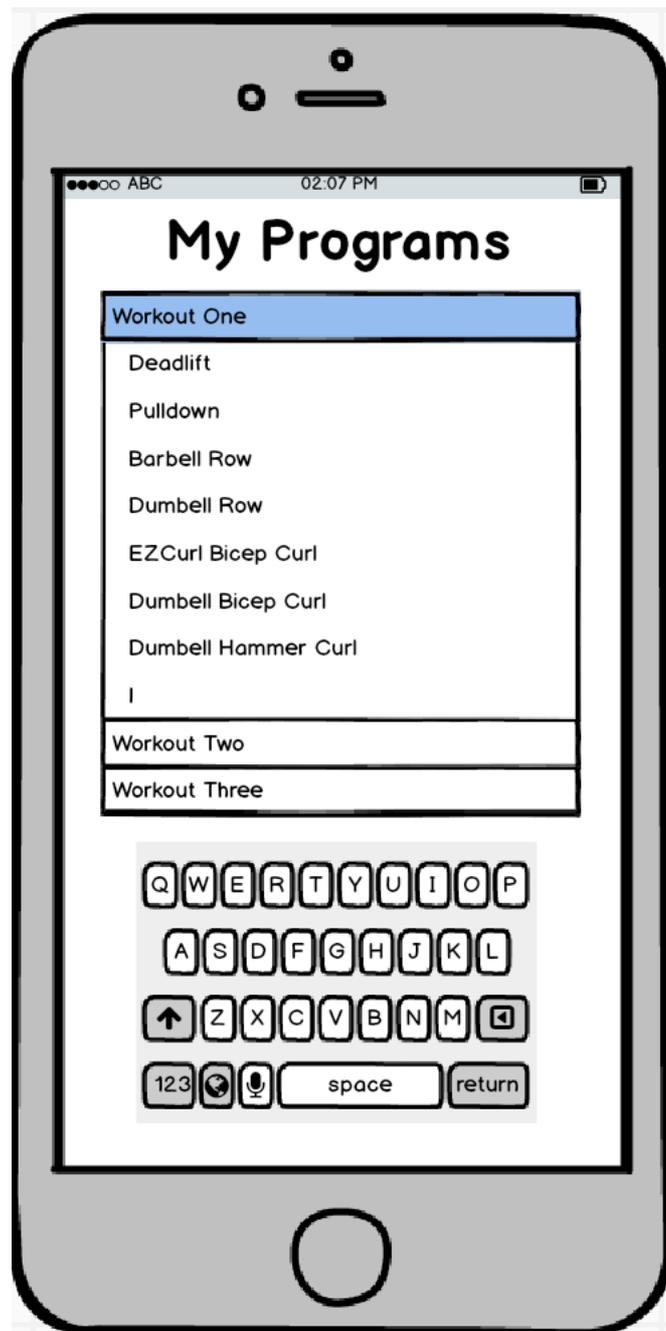


## 9.8 Use Case #7 – Edit Program

Use Case Element	Description
Use Case Number	7
Application	Gym Buddies
Use Case Name	Edit Program
Use Case Description	A registered user logs in and navigates to the My Programs section of the application and taps the Edit Program button to make changes to their program.
Primary Actor	Registered User
Precondition	A registered user must be logged in and on the My Programs page. The user must have at least one program saved to their list.
Trigger	A registered user logs in and taps the My Programs button then the Edit Program button.
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case begins when a logged in user taps the My Programs button and then the Edit Program button. [AF1]</li> <li>2. When the user taps the Edit Program button beside one of the programs they are presented with a GUI with edit fields allowing them to edit the program.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The user has no programs saved and the system has nothing to display to the user. The system notifies the user of this and returns them to the home page.</li> </ol>
Termination	The flow is terminated when the user's program is successfully edited and the database is updated accordingly.
Post Condition	The user is returned to the My Programs page and the system enters a wait state.

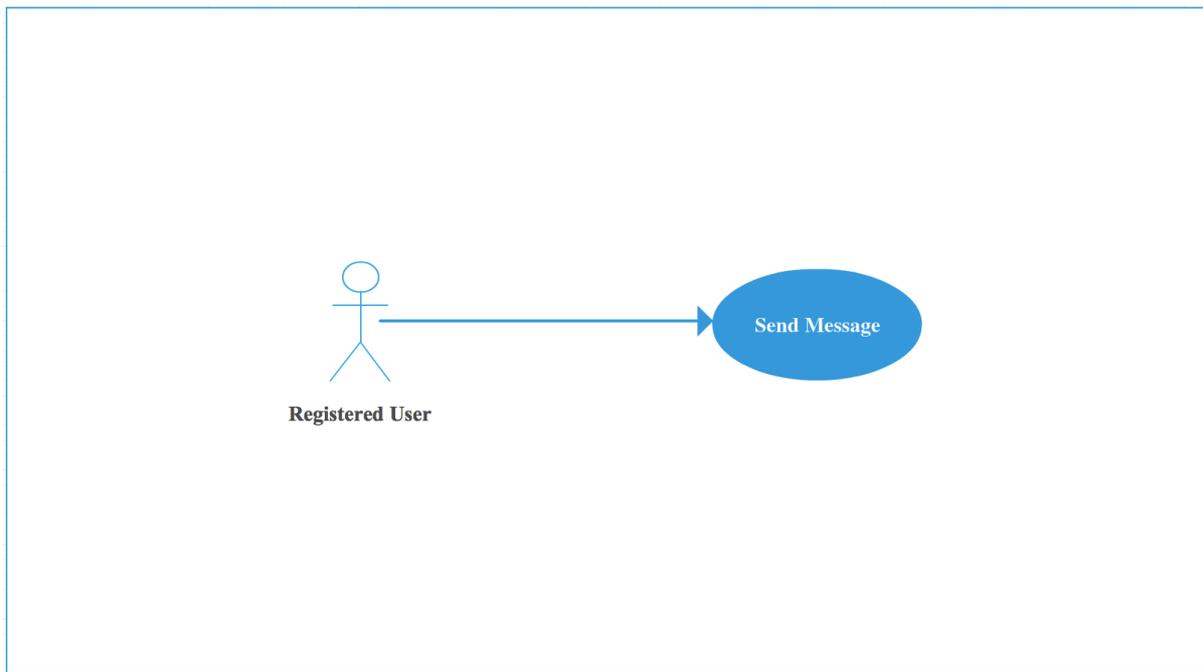


## Use Case #7 – Edit Program Mockup

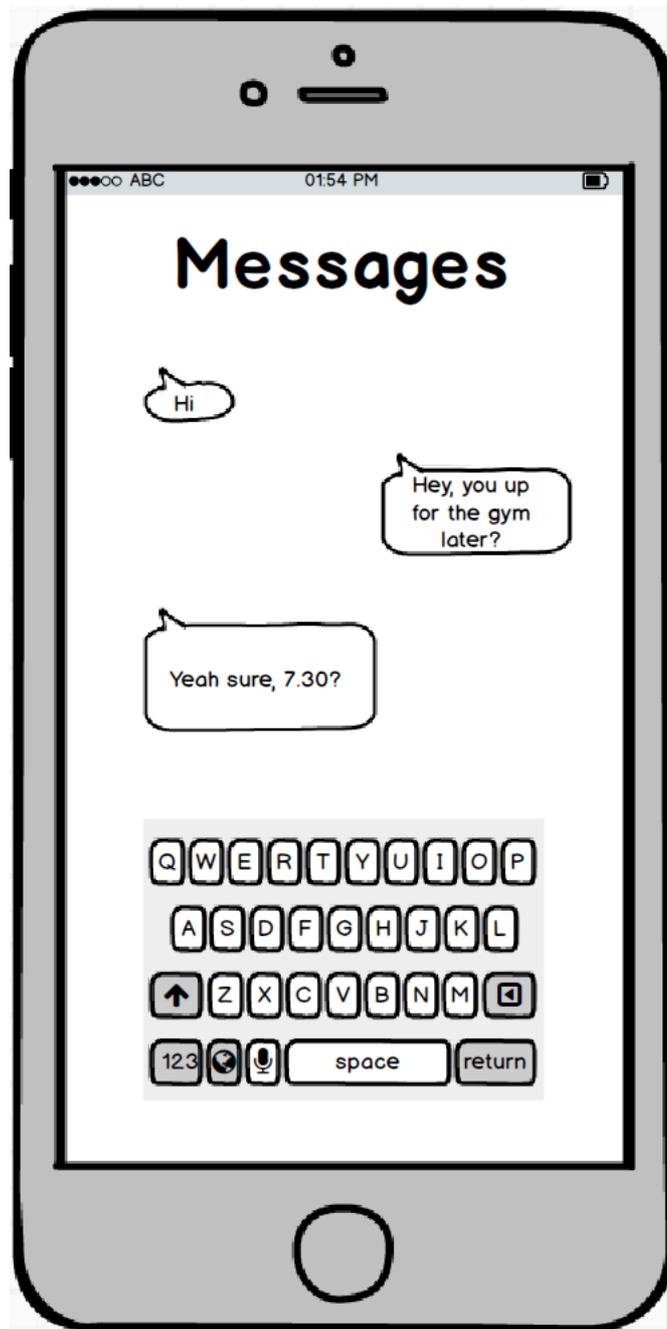


## 9.9 Use Case #8 – Send Message

Use Case Element	Description
Use Case Number	8
Application	Gym Buddies
Use Case Name	Send Message
Use Case Description	A registered user logs in and navigates to the Messages section of the application and taps the Send Message button and selects a user to send a message to.
Primary Actor	Registered User
Precondition	A registered user must be logged in and on the Messages page. The user must know the alias of the other user they wish to message.
Trigger	A registered user logs in and taps the Messages button and then sends a message to another user.
Basic Flow	<ol style="list-style-type: none"> <li>1. This use case begins when a logged in user taps the Messages button and selects a user to message. [AF1]</li> <li>2. Once a user is selected the user is presented with a UI enabling them to type their message.</li> <li>3. The message is sent when the user taps Send.</li> </ol>
Alternate Flows	<ol style="list-style-type: none"> <li>1. The user enters an incorrect user alias and the system notifies the user of this and returns them to the messages page.</li> </ol>
Termination	The flow is terminated when the user's message is sent to the recipient and the database is updated accordingly.
Post Condition	The user is returned to the Messages page and the system enters a wait state.

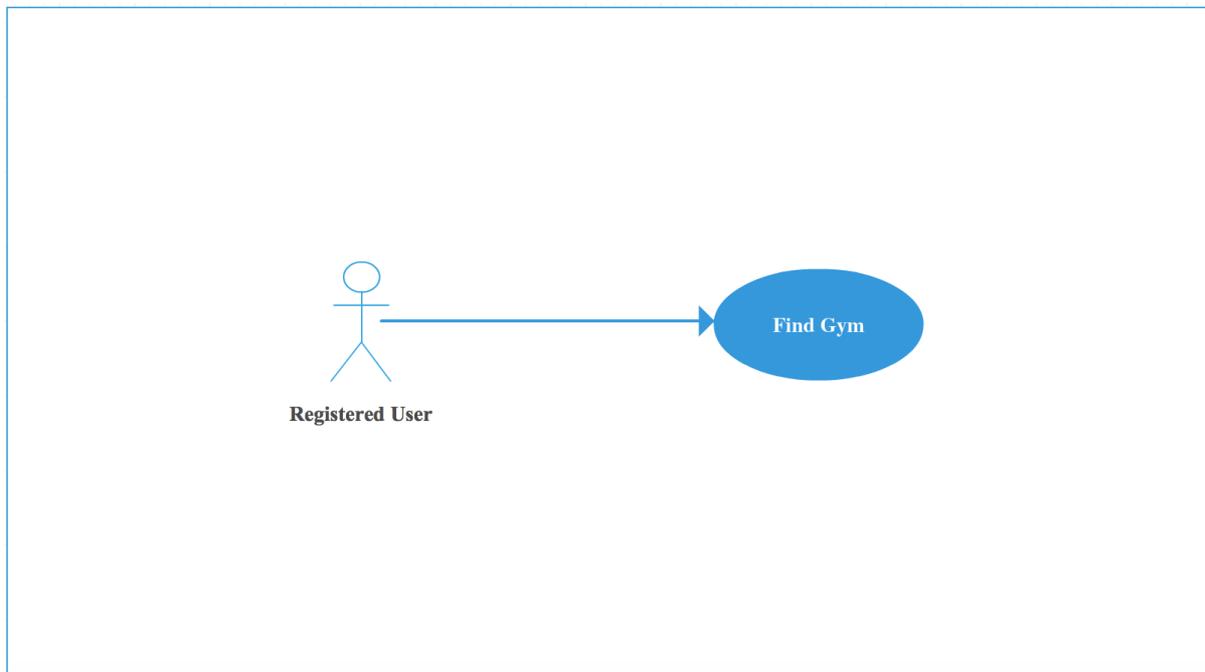


## Use Case #8 – Send Message Mockup

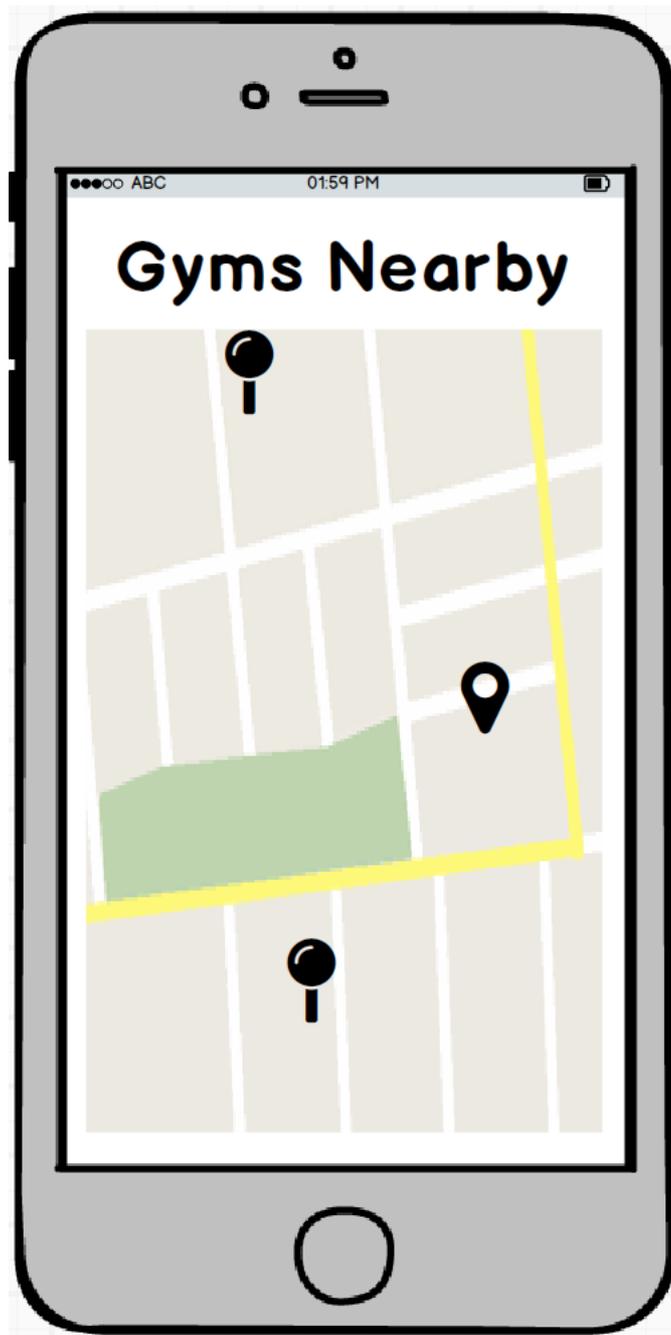


## 9.10 Use Case #9 – Find Gym

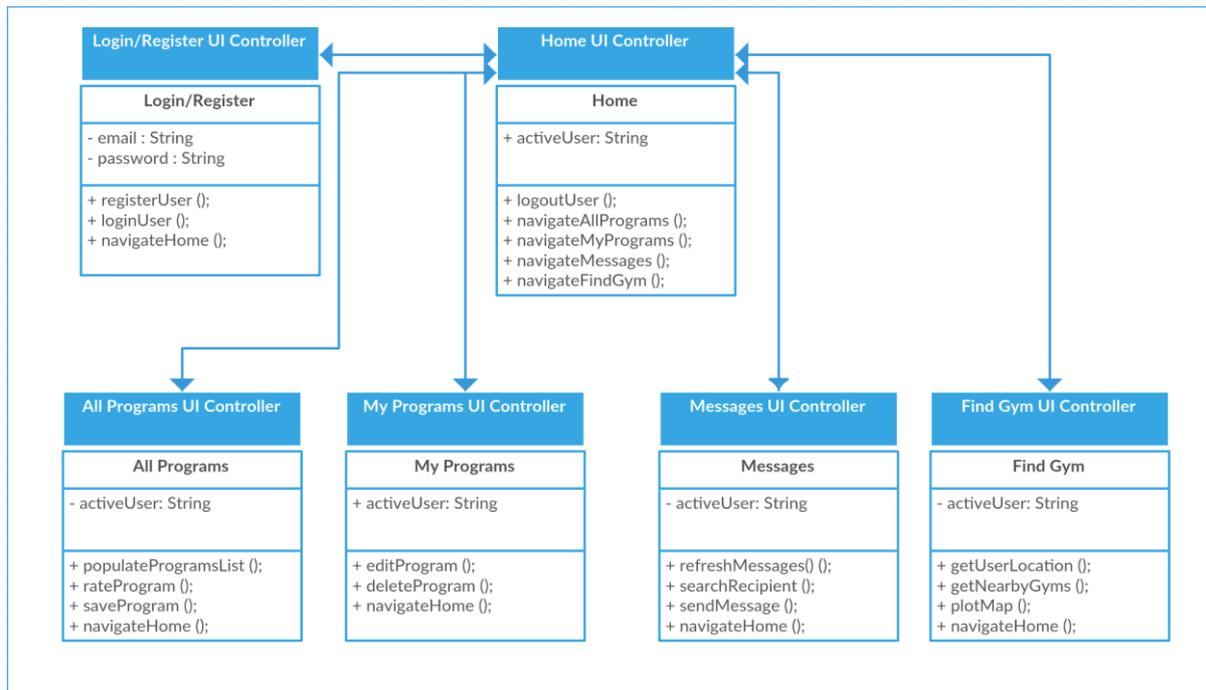
Use Case Element	Description
Use Case Number	9
Application	Gym Buddies
Use Case Name	Find Gym
Use Case Description	A registered user logs in and navigates to the Find Gym section of the application and taps the Find a Gym button.
Primary Actor	Registered User
Precondition	A registered user must be logged in and on the Find Gym page.
Trigger	A registered user logs in and taps the Find a Gym button in the Find Gym Section.
Basic Flow	<ol style="list-style-type: none"><li>1. This use case begins when a logged in user navigates to the Find Gym section.</li><li>2. The Application will then request access to the location functionality of their device. [AF1]</li><li>3. The system will then present the user with a Google Maps page with all Gyms within a 4km radius plotted on the map.</li></ol>
Alternate Flows	<ol style="list-style-type: none"><li>1. The user declines access to their location and the app returns them to the Home page.</li></ol>
Termination	The flow is terminated when the user is presented with the map.
Post Condition	The user remains on the map page and the system enters a wait state.



Use Case #9 – Find Gym Mockup



## 10 Storyboard/Class Diagram



## 11 Analysis & Design Specification

## 12 Analysis & Design Introduction

### 12.1 Purpose

The purpose of the product analysis and design section is to detail and track the necessary information required to effectively define the system architecture and system design so as to provide the developer with guidance on the architecture of the system being developed.

The intended audience of this section consists of the Project Supervisor, Project Developer, National College of Ireland faculty and National College of Ireland external markers. Subsections of this document such as User Interface details, may on occasion be made available to product users/clients and other stakeholders whose approval of the UI is required.

## 13 General Overview & Design Guidelines

This section describes guidelines around assumptions, constraints and standards when designing and implementing the Gym Buddies system.

## 13.1 Assumptions, Constraints & Standards

13.1.1 The system will require an internet connection for authentication and initial downloading of application data.

*13.1.1.1 Offline caching of data can be implemented but periodic internet access is required to update the cache with new data and for initial data access.*

13.1.2 The system will run natively on devices running iOS8 or later. This means the system will need to be developed in the Swift 3 programming language using the XCode 8 IDE.

13.1.3 The system will need to interface with Google's Firebase Database technology for data storage and file storage.

*13.1.3.1 This will require the inclusion of Firebase pod files within the Gym Buddies project. These can be installed using Cocoa Pods via the Terminal command line.*

13.1.4 The system will handle up to 100 simultaneous connections initially. This is due to limitations in the Firebase Service on the basic plan.

13.1.5 The system will need to interface with the Google Maps framework.

13.1.6 The system will need to make use of a messaging/chat framework.

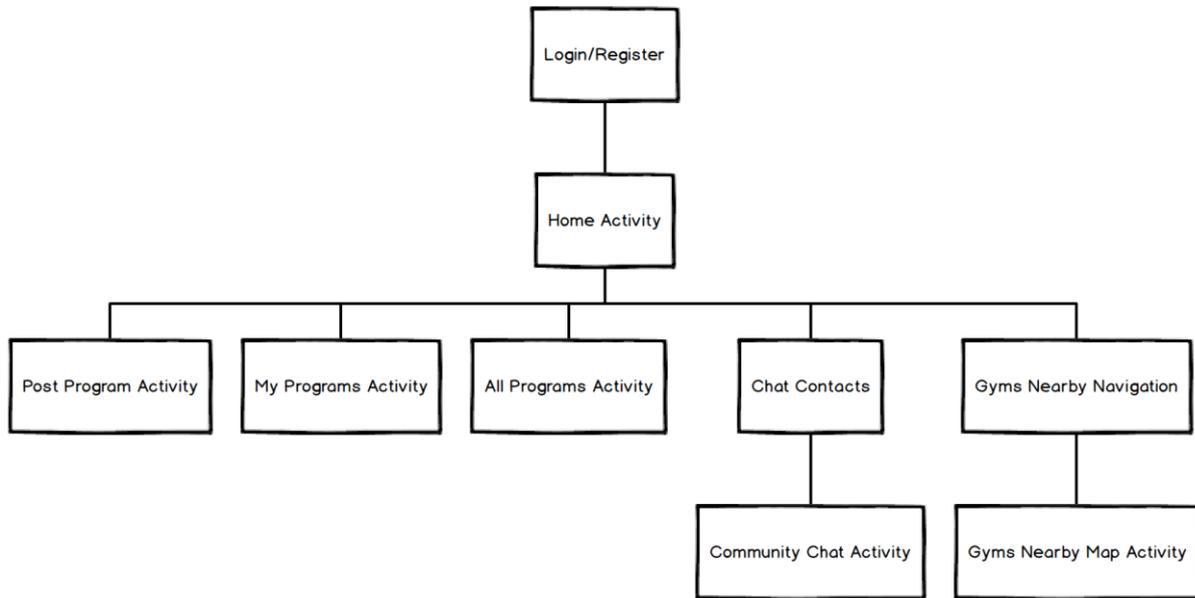
## 14 Architecture Design

This section outlines the system and hardware architecture design of the system being developed.

### 14.1 Logical View

The logical view of the Gym Buddies application will focus on a central point of navigation. All sections of the application will be reachable from a central home activity. Backward navigation will be possible from all sections of the application to return the user to the activity they were on previously. This will help ensure the application will follow all accepted usability and design standards, i.e. all navigation and layouts will be consistent, accessible and understandable. A hierarchal diagram of the proposed UI layout is included below.

## Hierarchal Diagram



## 14.2 Hardware Architecture

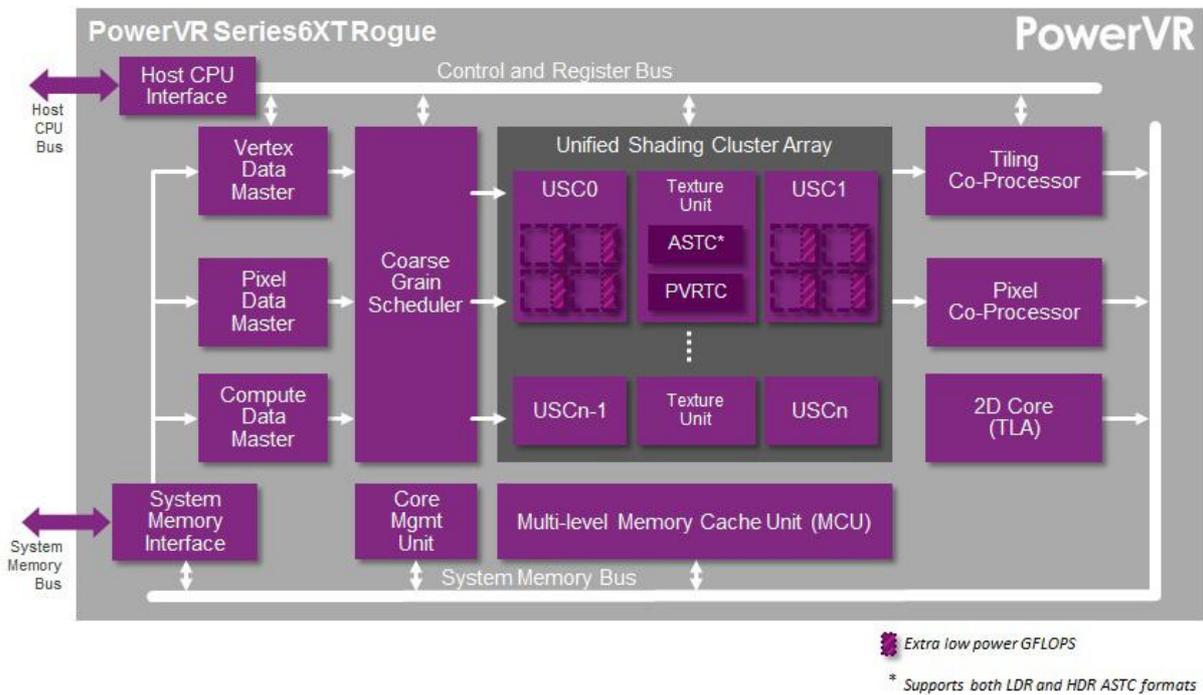
This section will detail the hardware components that make up Apple’s iPhone device and how those components interoperate.

### Relevant Hardware Components

<b>Display</b>	4.7” or 5.5” depending on model.
<b>Storage</b>	32GB, 64GB or 128GB internal flash storage depending on model.
<b>CPU</b>	Apple A9 chip with 64-bit architecture embedded with M9 motion coprocessor.
<b>Camera</b>	12 Megapixel main camera and 5 megapixel front facing camera.
<b>Location &amp; Network</b>	Assisted GPS and GLONASS, Digital Compass, Wi-Fi, Mobile Data and iBeacon micro-location.
<b>Operating System</b>	iOS 8, iOS 9 or iOS10 depending on model.

Below is an example in diagram form of the iPhone hardware architecture:

## Comprehensive Hardware Architecture Diagram



\*Diagram Provided by ExtremeTech.com

### 14.3 Software Architecture

In this section, the software architecture of the Gym Buddies system will be detailed. Based on the main requirements of the system, i.e. a social health and fitness mobile application, the iOS platform was chosen. To develop for this platform, the XCode IDE is also required and this IDE only operates on the Mac OS platform.

The Gym Buddies system will run natively on an iPhone device running iOS 8 or later. Data persistence will be achieved using Google’s Firebase database technology. Firebase is a NoSQL, JSON based storage option that stores and retrieves information based on key-value pairs. Interfacing with Firebase will be carried out using Firebase APIs which are included in the XCode project via Cocoa Pods pod files.

Gym Buddies will also need to interface with Google Maps via the Google Maps APIs. These APIs are also included in the XCode project via Cocoa Pods pod files.

As a result, the Gym Buddies system’s software architecture will comprise of a 3 tiered architecture. The 3 tiers consist of:

Tier	Description
Presentation Tier	User Interface, translates tasks and results to be user readable and understandable.
Logic Tier	Performs evaluations and calculations as well as moving processed data between the surrounding tiers.
Data Tier	Stores and retrieves information from database.

## 14.4 Security Architecture

### Blanket Database Security

Firebase offers a blanket security measure when accessing data in the database. Application profiles need to be manually added by the Database Administrator in order for any piece of software to have access to the database. If an application's profile is not listed under approved applications in database settings, no access to the database is granted.

### Real-time Database Rules

Real-time Database rules add an additional layer of security for data access within a Firebase Database. Specific rules can be added to individual actions such as read and write etc. This makes it possible to apply permissions to an individual user/user group.

Example:

```
{
  "rules": {
    "users": {
      "$uid": {
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

### Data Security During Transmission

Eavesdropping or interception during transmission is tackled by Firebase via 2048bit SSL encryption of data during transmission. This ensures that data read from or written to the database is kept secure during transmission and is protected from interception, theft and modification.

### User Authentication

As an application level security measure, users will need to authenticate themselves in order to access application content. Authentication is done via email and password and authentication is also done via Firebase which affords this process the 2048bit SSL encryption used for all transmission of data. Passwords are encrypted on the server and are not readable even to the Database Administrator. Accounts can be managed and moderated by the Database Administrator if any system abuse takes place.

## 14.5 Communication Architecture

The Gym Buddies application will reside on the iPhone device of the user. The application will need to be side loaded on to the device via the XCode IDE. The application will run locally on the host device.

The application will need to interface with the Firebase Database backend and perform CRUD functionality over a network connection. User authentication is required so a network connection is a mandatory requirement in order for the application to function.

A network connection is also required for messaging and map functionality. Messages will be stored in and retrieved from the Firebase backend and the Google Maps functionality will require an internet connection in order to fully function.

A network connection can be made available to the application through both Wi-Fi and network data plans, the application is compatible with both.

## 14.6 Performance

System performance will be measured in time, space (storage) and bandwidth.

The system should be able to carry out operations and respond to user interactions in a timely manner. Navigation should occur instantly with no delays between button taps and screen responses. Database reads/writes should take no longer than 5 seconds and any errors should be swift and user-friendly.

The system should require no more than 250MB of storage space on the user's device. While most iPhones have ample storage, some users may have lower hardware specifications or have high storage requirements. Gym Buddies should not have a huge storage footprint. The database should have the required storage for several hundred users and their associated data.

The Firebase backend will support the storage and retrieval of media files. Supported media files include images and videos of the more popular file extensions, i.e. PNG, MP4 etc. The only performance variable for this functionality should be the user's connection.

The database should be able to handle up to 100 simultaneous connections without any degradation of performance. This includes user authentication and any CRUD operations. Full system bandwidth will result in a user-friendly notification to the accessing user.

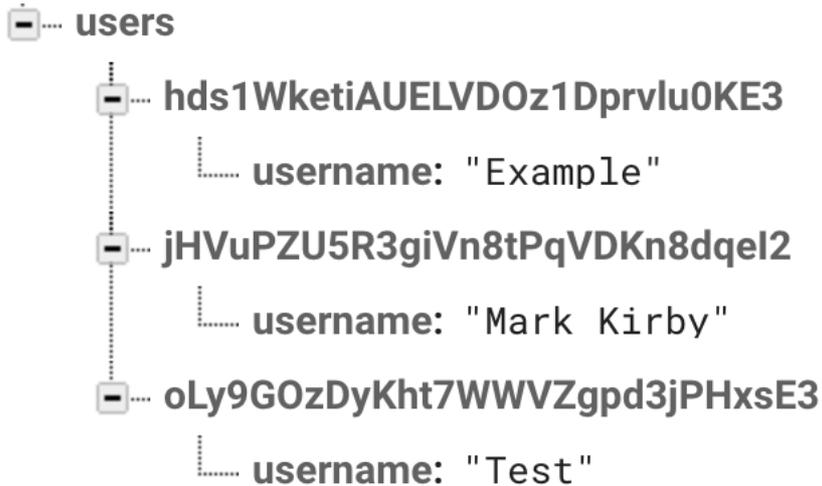
The Firebase administration panel will allow developers and Database Administrators to perform maintenance on the database and purge any redundant data or files. Inactive users can also be removed from the system.

## 15 System Design

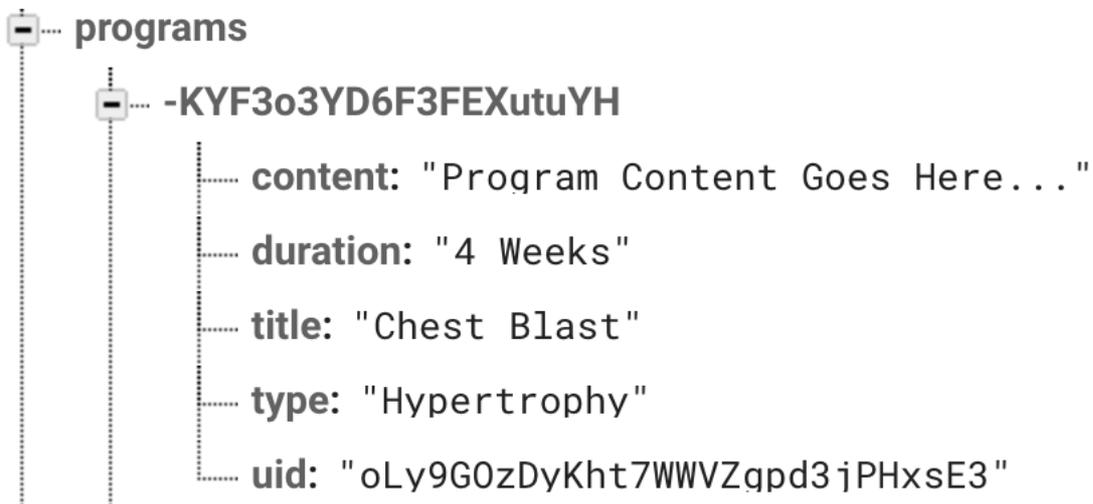
### 15.1 Database Design & Data Conversions

Firebase is a NoSQL, JSON based database. As such, data is not stored in traditional database tables, they are stored as JSON objects that are stored and accessed using key-value pairs. The data is stored under the following structure:

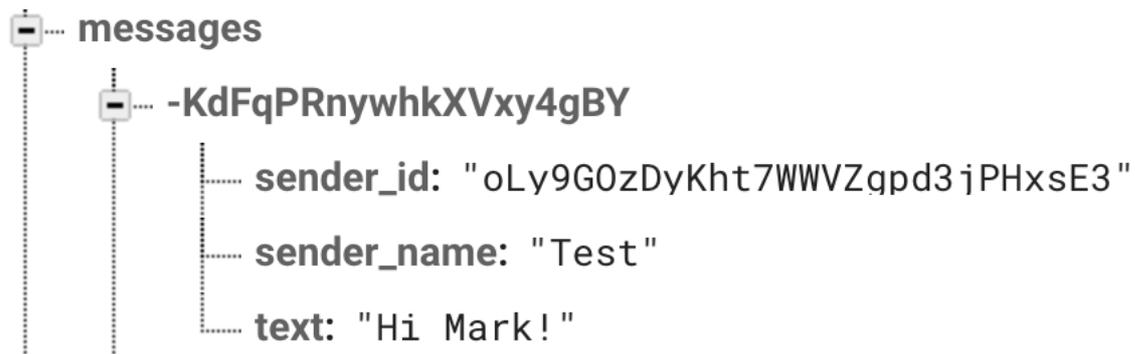
#### Users



#### Programs



## Messages



## Data Conversions

Data will need to be converted from generic String type to JSON and from JSON back to generic String type. This conversion will be performed when saving user input from the user interface (UI) into the Firebase database backend and when data is being retrieved from the Firebase Database backend and presented to the user in the UI. Currently no other data conversions need to be performed.

## 15.2 Application Program Interfaces

### Firestore

1.1.1 Data persistence will be carried out using Google’s Firestore Database backend technology for data storage, user storage/authentication and file storage. An API will be required in order for the system to be able to interface with Firestore. The required API files can be included in the project via pod files installed using Cocoa Pods. Required pod files:

```
pod 'Firestore'
pod 'Firestore/Core'
pod 'Firestore/Auth'
pod 'Firestore/Database'
pod 'Firestore/Storage'
```

### Google Maps

1.1.2 Google Maps functionality will be required for the “Gyms Nearby” functionality. The Google Map API will be required in order for the system to be able to interface with Google Maps. As with Firestore, the required files can be included via Cocoa pods. Required pod files:

```
pod 'GoogleMaps'
pod 'GooglePlaces'
```

## JSQMessages

- 1.1.3 The system will need to provide users with a way to send messages to each other. To accomplish this a messaging framework will need to be implemented. JSQMessagesViewController is a messages UI library for iOS developed by Jesse Squires. It makes methods for messaging and messaging UI elements available to an application. Similar to Firebase and Google Maps, the required files can be included via Cocoa Pods. Required pod files:

```
pod 'JSQMessagesViewController'
```

## 16 Implementation

This section will provide descriptions and code snippets for the main functional requirements of the Gym Buddies application. Some code snippets shown may be incomplete and are subject to change before the project presentation. Code snippets may be cut down to demonstrate key points in a concise manner.

### 16.1 Register New User

The code snippet below shows the logic involved for registering a new user to the application and saving their credentials to the online Firebase database.

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

    let email = emailTextField.text
    let password = passwordTextField.text
    let username = usernameTextField.text

    FIRAuth.auth()?.createUser(withEmail: email!, password: password!, completion: {
        (user, error) in
        if error != nil{
            //Error creating account
            let errorMessage = error?.localizedDescription
            let alert = UIAlertController(title: "Error", message: errorMessage, preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
            self.present(alert, animated: true, completion: nil)
        }
        else{
            //Success

            if let uid = FIRAuth.auth()?.currentUser?.uid{

                let userRef = FIRDatabase.database().reference().child("users").child(uid)

                let object = ["username":username]

                userRef.setValue(object)
            }

            //Alert user of account creation

            let alert = UIAlertController(title: "Success", message: "Account Created Successfully", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: { (action) in
                //Will send user to sign in screen when OK is tapped
                let vc = self.storyboard?.instantiateViewController(withIdentifier: "HomeVC")
                self.present(vc!, animated: true, completion: nil)
            }))
            self.present(alert, animated: true, completion: nil)

            /*let alert = UIAlertController(title: "Success", message: "Account Created Successfully", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
            self.present(alert, animated: true, completion: nil)*/

        }
    })
}
```

## 16.2 Sign in Registered User

The code snippet below shows the logic involved for signing in a user that is already registered with the system. Credentials supplied by the user are then compared with credentials stored in the online Firebase Database.

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

    let email = emailTextField.text
    let password = passwordTextField.text

    FIRAuth.auth()?.signIn(withEmail: email!, password: password!, completion: { (user, error) in
        if error != nil{
            //error logging in user
            let alert = UIAlertController(title: "Error", message: "Incorrect Username/Password", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
            self.present(alert, animated: true, completion: nil)
        }else{
            //success
            let vc = self.storyboard?.instantiateViewController(withIdentifier: "HomeVC")
            self.present(vc!, animated: true, completion: nil)

            /*let alert = UIAlertController(title: "Success", message: "Successfully Signed In", preferredStyle: .alert)
            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: nil))
            self.present(alert, animated: true, completion: nil)*/
        }
    })
}
```

## 16.3 Post New Exercise Program

The below code snippets shows the logic involved when a user posts a new program to the public list of programs for other users to consume.

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

    //Establish currently logged in user
    if let uid = FIRAuth.auth()?.currentUser?.uid{
        //Grab values from fields
        if let title = titleTextField.text{
            if let type = typeTextField.text{
                if let duration = durationTextField.text{
                    if let content = contentTextView.text{
                        //Establish Logged in User's Username
                        let userRef = FIRDatabase.database().reference().child("users").child(uid)
                        userRef.observeSingleEvent(of: .value, with: { snapshot in
                            let userDict = snapshot.value as! [String:AnyObject]
                            let username = userDict["username"] as! String

                            //Set up data for database
                            let postObject: Dictionary<String, Any> = [
                                "author" : username,
                                "title" : title,
                                "type" : type,
                                "duration" : duration,
                                "content" : content
                            ]

                            //Post Values to Database
                            FIRDatabase.database().reference().child("programs").childByAutoId().setValue(postObject)

                            let alert = UIAlertController(title: "Success", message: "Program Posted Successfully",
                                preferredStyle: .alert)
                            alert.addAction(UIAlertAction(title: "OK", style: .default, handler: { (action) in
                                //Will send user to Home screen when OK is tapped
                                let vc = self.storyboard?.instantiateViewController(withIdentifier: "HomeVC")
                                self.present(vc!, animated: true, completion: nil)
                            }))
                            self.present(alert, animated: true, completion: nil)
                        })
                    }
                }
            }
        }
    }
}
```

## 16.4 Loading Exercise Programs from Database into Table View

The two code snippets below show the logic involved for loading exercise programs from the online Firebase database into a table view to be displayed in the application. Similar logic is used to grab and display data for a user's private list of programs with the exception that some extra parameters are evaluated to ensure that the programs loaded belong to the logged in user. Programs are loaded from the database and added to an array, this array is then iterated and each index is added to the table view.

```
//Method to load data from Firebase
func loadData(){

FIRDatabase.database().reference().child("programs").observeSingleEvent(of: .value, with: { (snapshot) in
    //Snapshot holds value and it is casted to NS Dictionary
    if let programsDictionary = snapshot.value as? [String: AnyObject]{
        for program in programsDictionary{
            self.programs.add(program.value)
        }
        self.programsTableView.reloadData()
    }
})
}
```

```
func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withIdentifier: "Cell", for: indexPath) as! AllProgramsTableViewCell

    // Configure the cell...
    let program = self.programs[indexPath.row] as! [String: AnyObject]

    //Populate row
    //Grab title and add it to cell
    cell.titleLabel.text = program["title"] as? String
    //Grab type and add it to cell
    cell.typeLabel.text = program["type"] as? String
    //Grab duration and add it to cell
    cell.durationLabel.text = program["duration"] as? String
    //Grab content and add it to cell
    cell.contentTextView.text = program["content"] as? String
    //Grab author and add it to cell
    cell.authorLabel.text = program["author"] as? String

    /*let authorUid = program["uid"] as? String

    let userRef = FIRDatabase.database().reference().child("users").child(authorUid!)
    userRef.observeSingleEvent(of: .value, with: { snapshot in
        let userDict = snapshot.value as! [String:AnyObject]
        let username = userDict["username"] as! String
        cell.authorLabel.text = username
    }*/)

    return cell
}
```

## 16.5 Save Public Program to Personal List

The below code snippet shows the logic involved when a user presses the save button for a particular public program in order to save it to their own personal list. The currently logged in user is established and this information is used as an identifier so that the data that is saved is saved under their unique user ID and is available only to them.

```
@IBAction func saveButtonTapped(_ sender: Any) {  
  
    //Establish currently logged in user  
    if let uid = FIRAuth.auth()?.currentUser?.uid{  
        //Grab values from fields  
        if let title = titleLabel.text{  
            if let type = typeLabel.text{  
                if let duration = durationLabel.text{  
                    if let content = contentTextView.text{  
                        if let author = authorLabel.text{  
                            //Set up data for database  
                            let postObject: Dictionary<String, Any> = [  
                                "uid" : uid,  
                                "title" : title,  
                                "type" : type,  
                                "duration" : duration,  
                                "content" : content,  
                                "author" : author  
                            ]  
  
                            FirebaseDatabase.database().reference().child("my_programs").childByAutoId().setValue(postObject)  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```

## 16.6 Message Handling & Retrieval

Both text and media based messages are stored in and read from the online Firebase Database. When a message is retrieved from the database and displayed in the table view it is handled in much the same way as the exercise programs.

A big difference is how these messages are handled in terms of listening for new messages. This required a listener for both text and media messages to be constantly observing the database and waiting for any new messages to be added. When a new message is added, it is then pushed to the table view in the application dynamically. The code snippets below will show the logic involved for sending and listening for new messages.

### Sending Text Based Messages

```
//Logic for sending text based messages and saving them to Firebase  
func sendMessage(senderID: String, senderName: String, text: String){  
  
    let data: Dictionary<String, Any> = [Constants.SENDER_ID: senderID, Constants.SENDER_NAME: senderName, Constants.TEXT:  
        text]  
  
    DBProvider.Instance.messagesRef.childByAutoId().setValue(data)  
}
```

## Sending Media Messages

```
//Logic for sending media files
func sendMedia(image: Data?, video: URL?, senderID: String, senderName: String){

    //Image Logic
    if image != nil{

        DBProvider.Instance.imageStorageRef.child(senderID + "\(NSUUID().uuidString).jpg").put(image!, metadata: nil){
            (metadata: FIRStorageMetadata?, err: Error?) in

                if err != nil{
                    print("ERROR SAVING MEDIA")
                }
                else{
                    self.sendMediaMessage(senderID: senderID , senderName: senderName, url: String(describing: metadata!.
                        downloadURL(!)))
                }

            }

        }

    }

    //Video Logic
    else{

        DBProvider.Instance.videoStorageRef.child(senderID + "\(NSUUID().uuidString)").putFile(video!, metadata: nil){
            (metadata: FIRStorageMetadata?, err: Error?) in

                if err != nil{
                    print("ERROR SAVING MEDIA")
                }
                else{
                    self.sendMediaMessage(senderID: senderID, senderName: senderName, url: String(describing: metadata!.
                        downloadURL(!)))
                }

            }

        }

    }

}

}
```

## Selecting File Type for Media Message

```
//Attachment Button Logic
override func didPressAccessoryButton(_ sender: UIButton!) {

    //Set up User Alert Requesting Media Type
    let alert = UIAlertController(title: "Media Messages", message: "Please Select Media Type", preferredStyle: .actionSheet)

    //Cancel Option
    let cancel = UIAlertAction(title: "Cancel", style: .cancel, handler: nil)

    //Photos Option
    let photos = UIAlertAction(title: "Photos", style: .default, handler: {(alert: UIAlertAction) in
        self.chooseMedia(type: KUTTypeImage)
    })

    //Video Option
    let videos = UIAlertAction(title: "Videos", style: .default, handler: {(alert: UIAlertAction) in
        self.chooseMedia(type: KUTTypeMovie)
    })

    //Add Options to Alert
    alert.addAction(photos)
    alert.addAction(videos)
    alert.addAction(cancel)

    //Present Alert
    present(alert, animated: true, completion: nil)

}
```

## Listeners for New Text & Media Messages

```
//Listener for new text messages in Firebase
func observeMessages(){

    DBProvider.Instance.messagesRef.observe(FIRDataEventType.childAdded){ (snapshot: FIRDataSnapshot) in

        if let data = snapshot.value as? NSDictionary{
            if let senderId = data[Constants.SENDER_ID] as? String{
                if let senderName = data[Constants.SENDER_NAME] as? String{
                    if let text = data[Constants.TEXT] as? String{
                        self.delegate?.messageReceived(senderId: senderId, senderName: senderName, text: text)
                    }
                }
            }
        }
    }
}

//Listener for new media messages in Firebase
func observeMediaMessages(){

    DBProvider.Instance.mediaMessagesRef.observe(FIRDataEventType.childAdded){ (snapshot: FIRDataSnapshot) in

        if let data = snapshot.value as? NSDictionary{
            if let id = data[Constants.SENDER_ID] as? String{
                if let name = data[Constants.SENDER_NAME] as? String{
                    if let fileURL = data[Constants.URL] as? String{
                        self.delegate?.mediaReceived(senderId: id, senderName: name, url: fileURL)
                    }
                }
            }
        }
    }
}
}
```

## 16.7 Centralized Navigation and Signing Out

Gym Buddies features centralized navigation. This means from the home page, the user can navigate to any other section of the application, and all sections of the application allow the user to return to their home screen. The home screen also allows the user to sign out and returns them to the Sign in screen. If a user does not sign out, they remain signed in to the application even if they exit the program. This saves the user having to sign in every time they launch the application.

```
//Sign User Out
@IBAction func signOutButtonTapped(_ sender: Any) {
    do{
        try FIRAuth.auth()?.signOut()

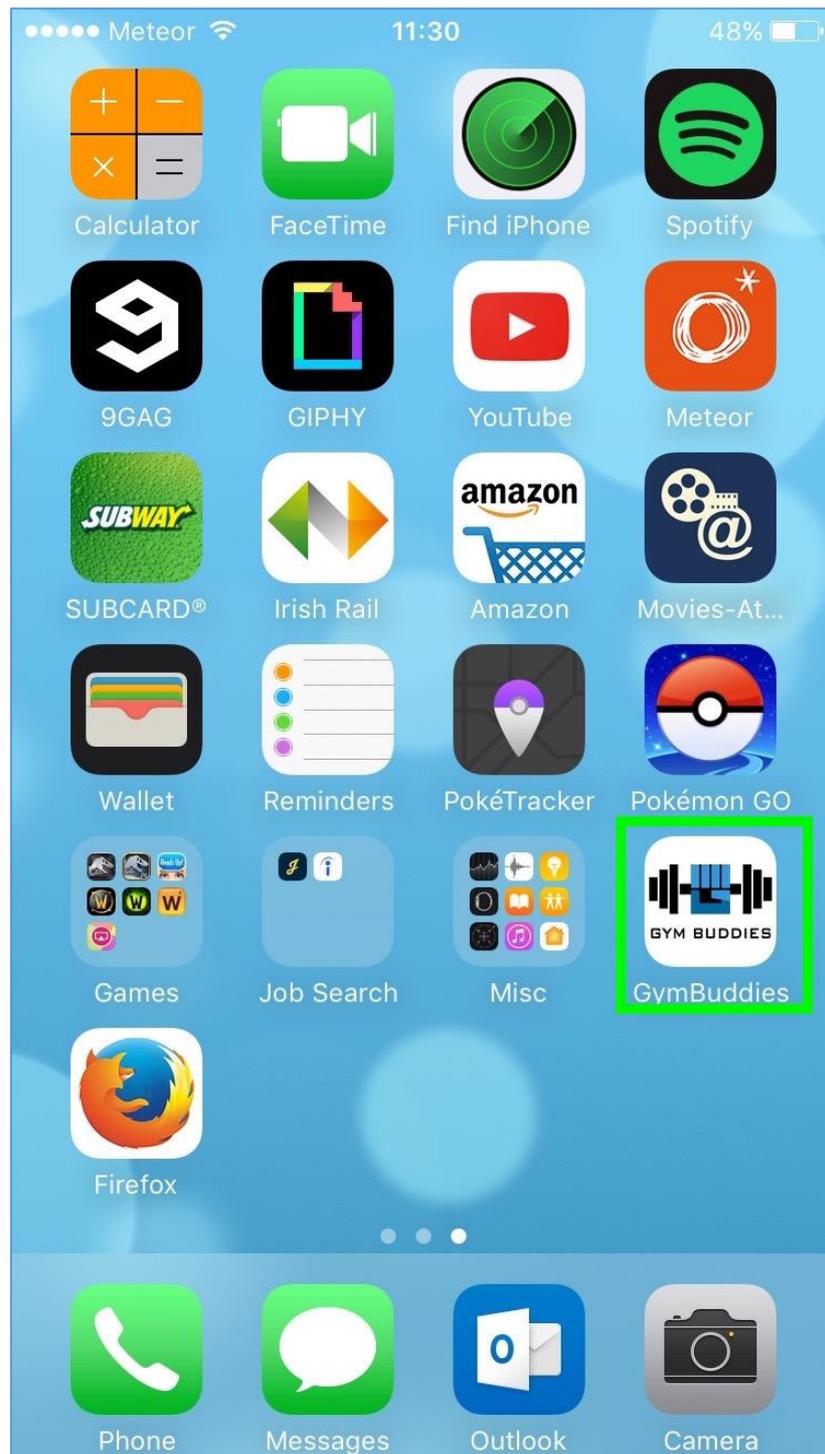
        let vc = self.storyboard?.instantiateViewController(withIdentifier: "SignInVC")

        self.present(vc!, animated: true, completion: nil)
    }
    catch{
        print("Error signing out user")
    }
}
```

## 17 GUI Design & Layout

### 17.1 App Icon

An application icon was created for the Gym Buddies application and is shown below. The app icon allows the user to launch the application from their devices home screen.



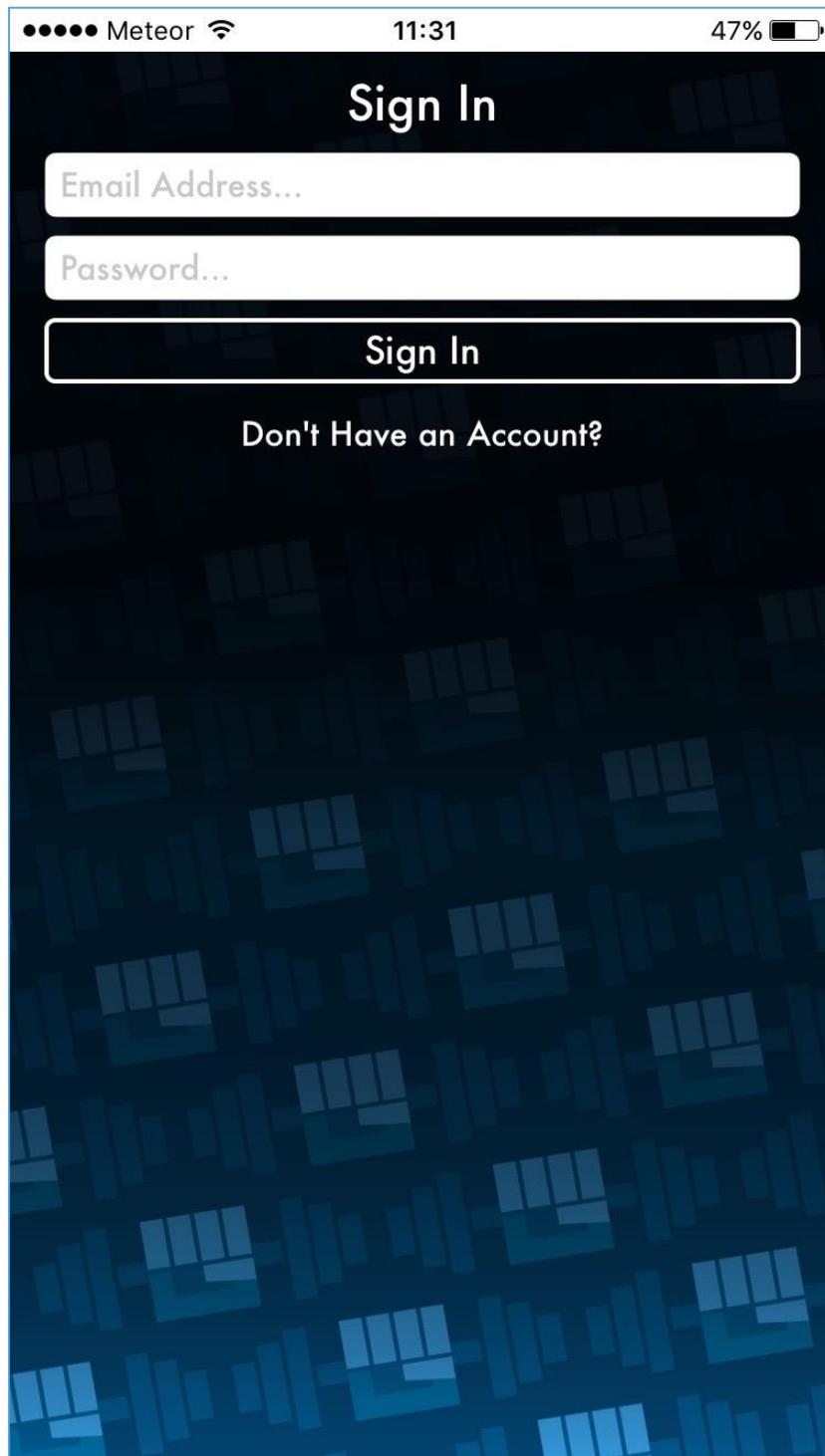
## 17.2 App Launch Screen

An app launch screen was created for the Gym Buddies application and is shown below. The launch screen is displayed to the user while the application loads in the background.



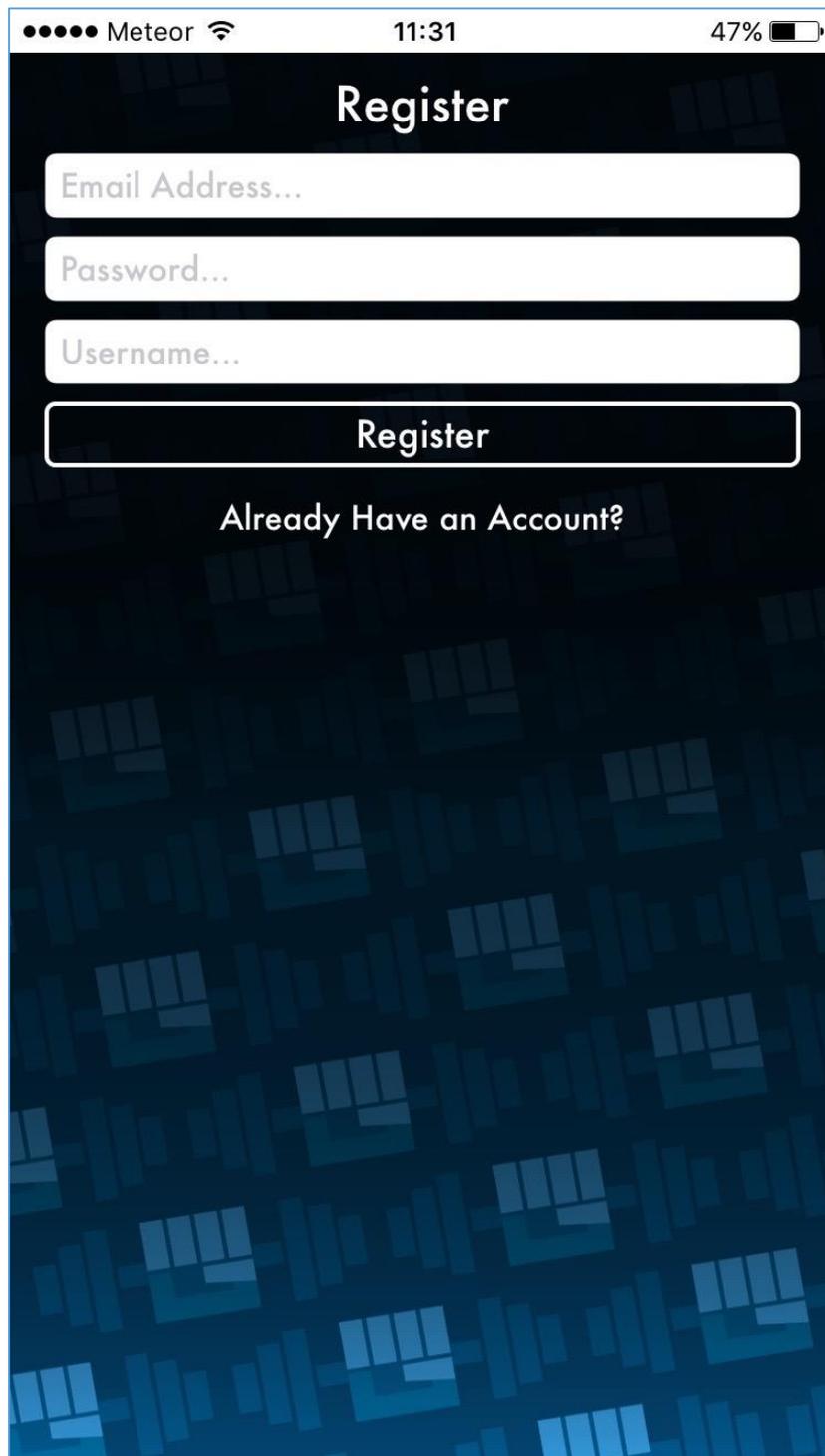
### 17.3 Sign in Screen

The Sign in screen allows the user to sign in to the application with an existing account.



## 17.4 Register Screen

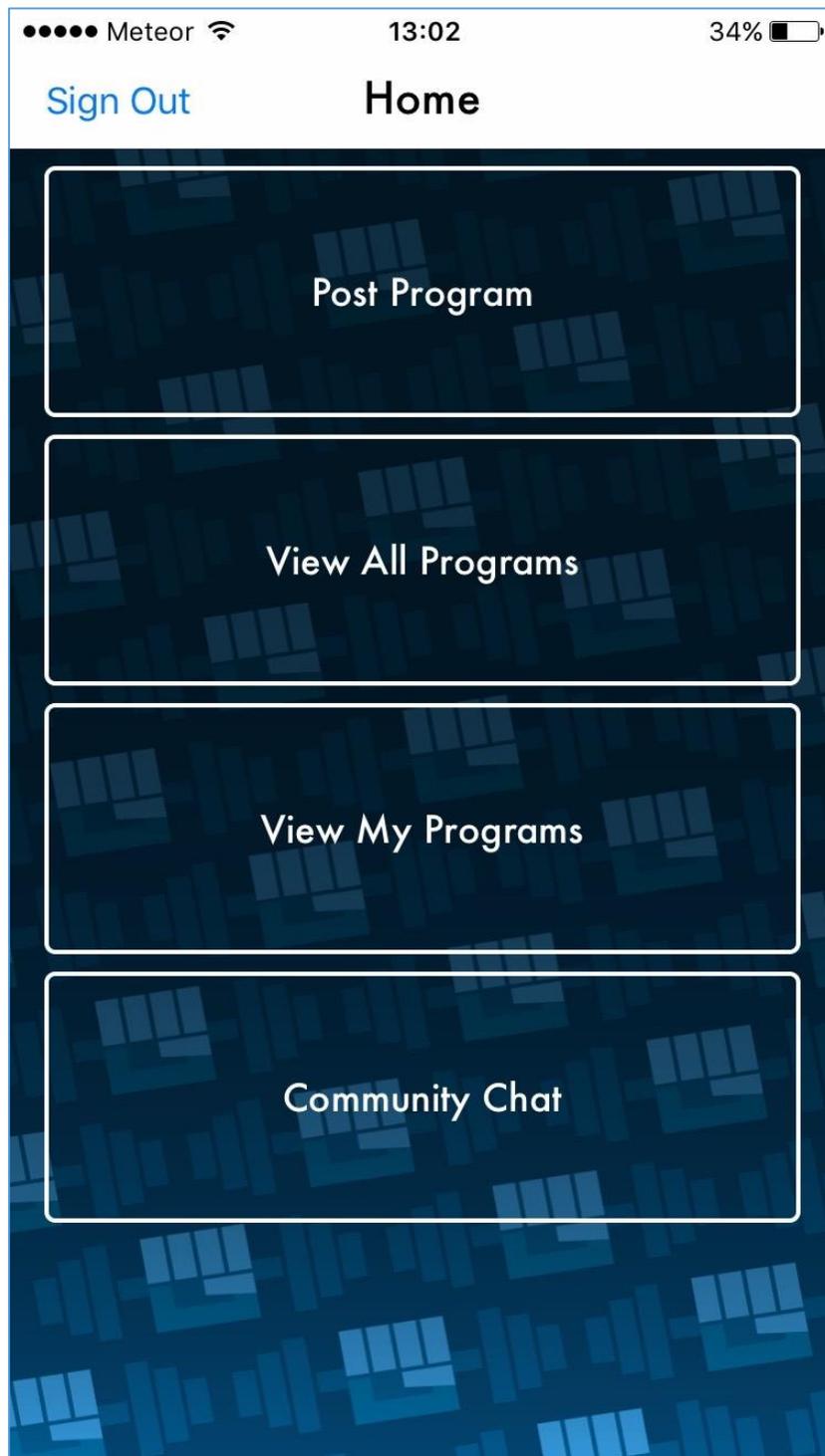
The Register screen allows the user to create an account using an email address and password and choose their username.



The screenshot shows a mobile application interface for a registration screen. At the top, the status bar displays 'Meteor' with a Wi-Fi icon, the time '11:31', and a battery level of '47%'. The main title 'Register' is centered at the top of the screen. Below the title are three input fields: 'Email Address...', 'Password...', and 'Username...'. A 'Register' button is positioned below these fields. At the bottom of the screen, there is a link that says 'Already Have an Account?'. The background of the screen features a dark blue pattern of abstract, overlapping rectangular shapes.

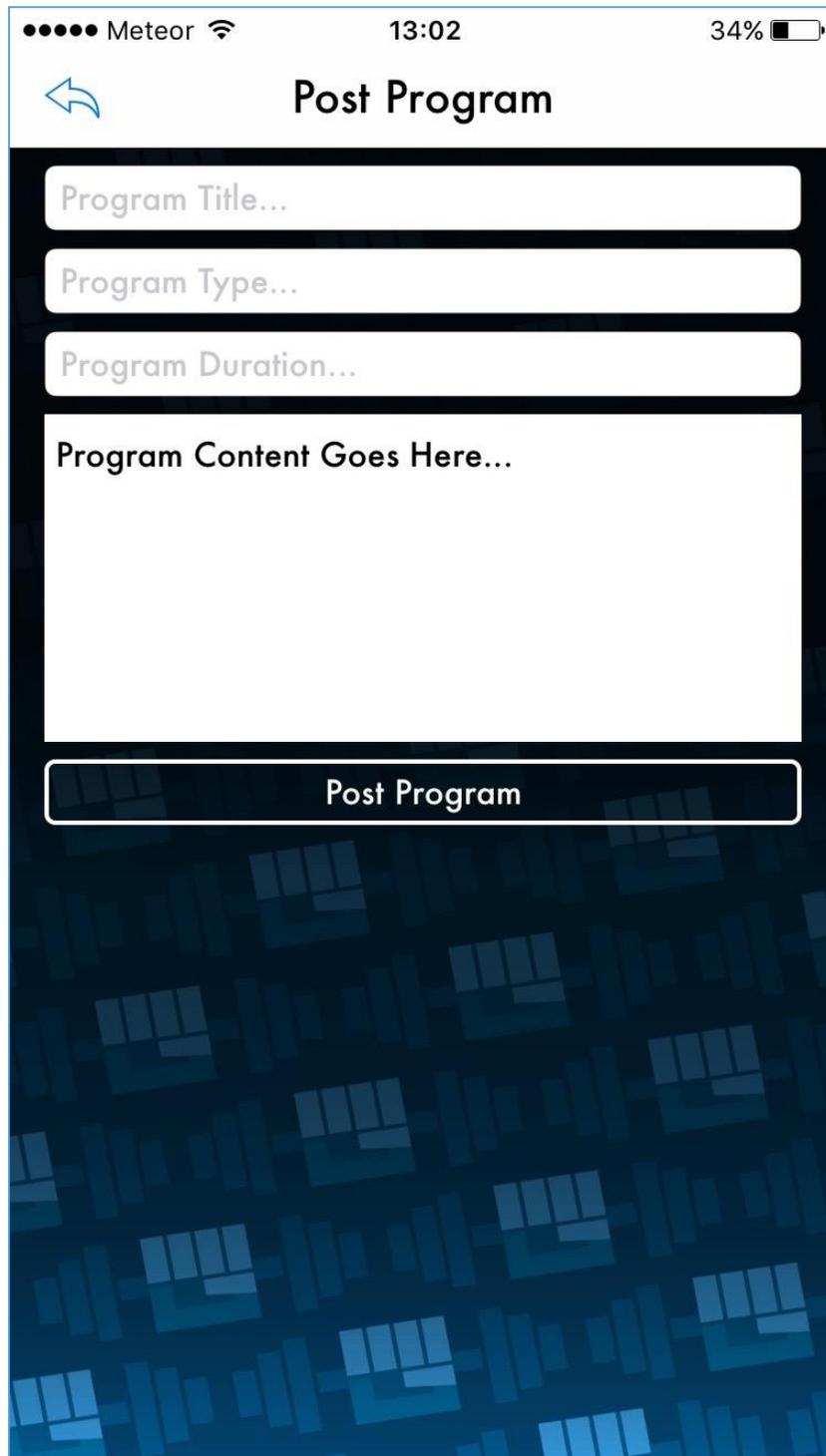
## 17.5 Home Screen

The Home screen provides the user with a central point of navigation to reach all other sections of the application. The user can also sign out here.



## 17.6 Post Program Screen

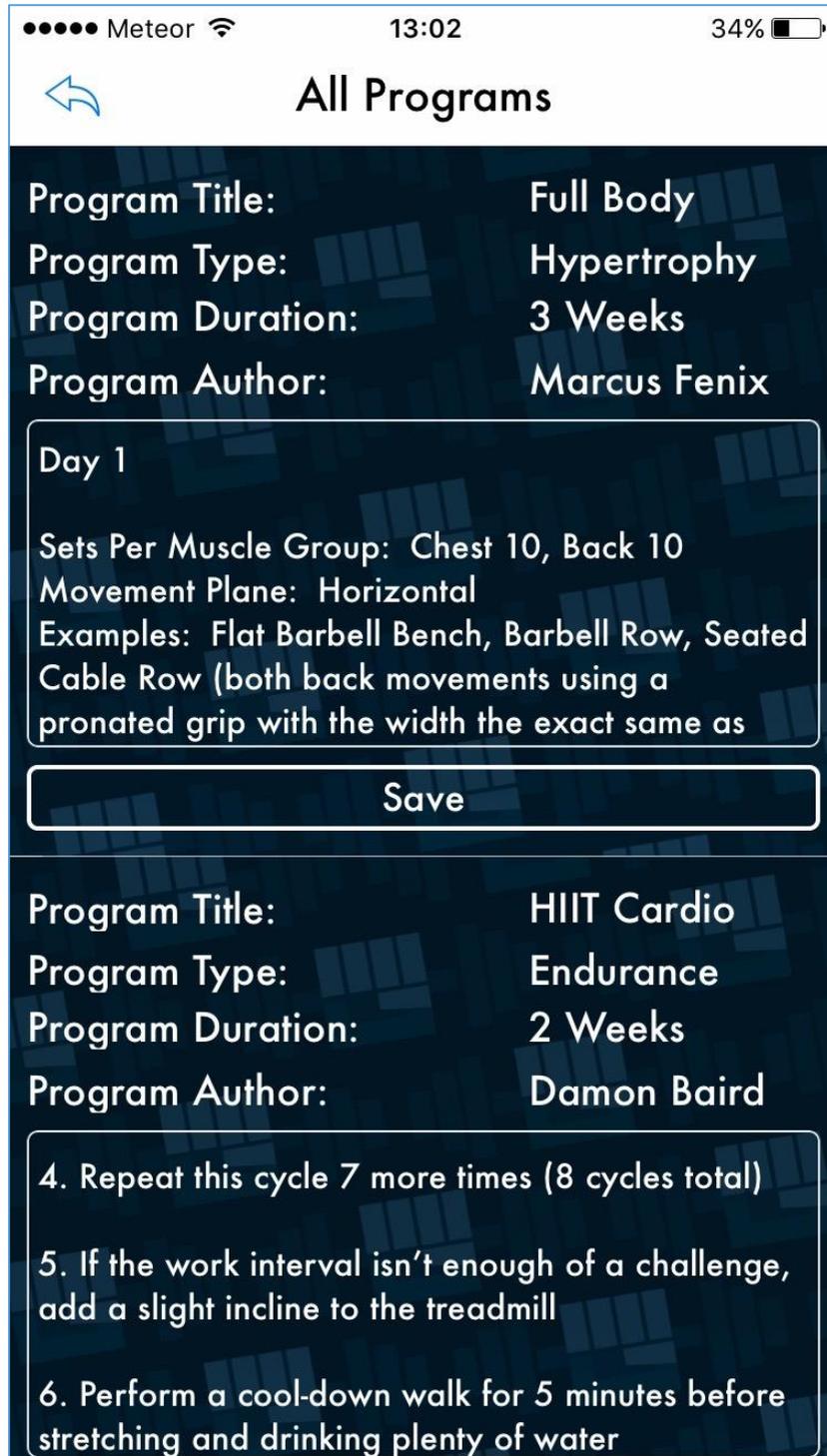
The Post Program screen allows the user to post an exercise program to the public list of exercise programs for consumption by other users of the application. Users can also use the back button to return to the Home screen from here.



The screenshot shows a mobile application interface for posting a program. At the top, the status bar displays 'Meteor' with signal strength, Wi-Fi, the time '13:02', and a 34% battery level. Below the status bar is a navigation bar with a blue back arrow on the left and the title 'Post Program' in the center. The main content area is a white form with a black border, containing four input fields: 'Program Title...', 'Program Type...', 'Program Duration...', and a larger text area labeled 'Program Content Goes Here...'. At the bottom of the form is a white button with a black border labeled 'Post Program'. The background of the screen is a dark blue pattern of light blue squares.

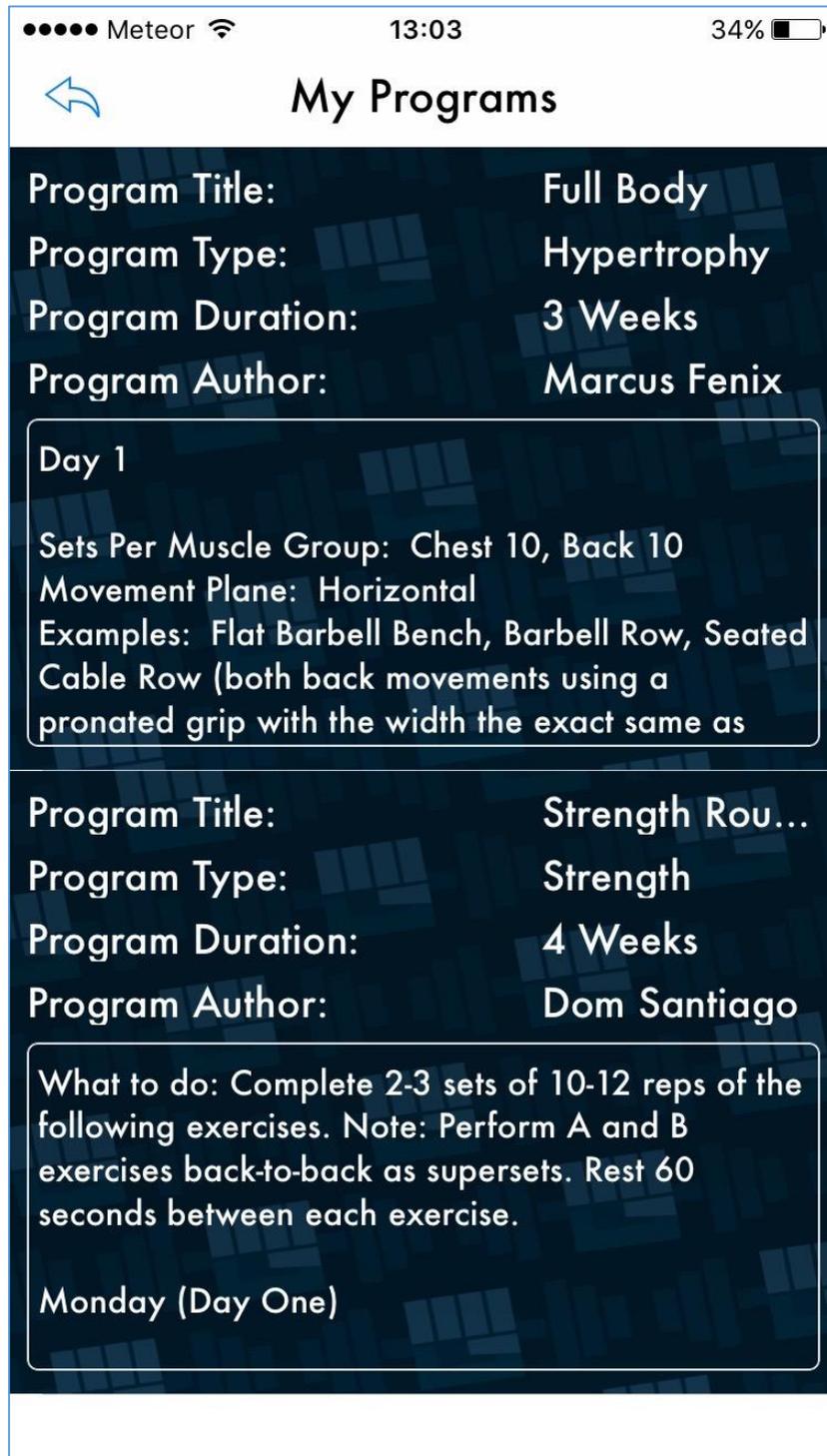
## 17.7 All Programs Screen

The All programs screen allows all users to browse the list of user submitted public programs. Users can choose to save any of these programs to their own private list or return to the Home screen from here.



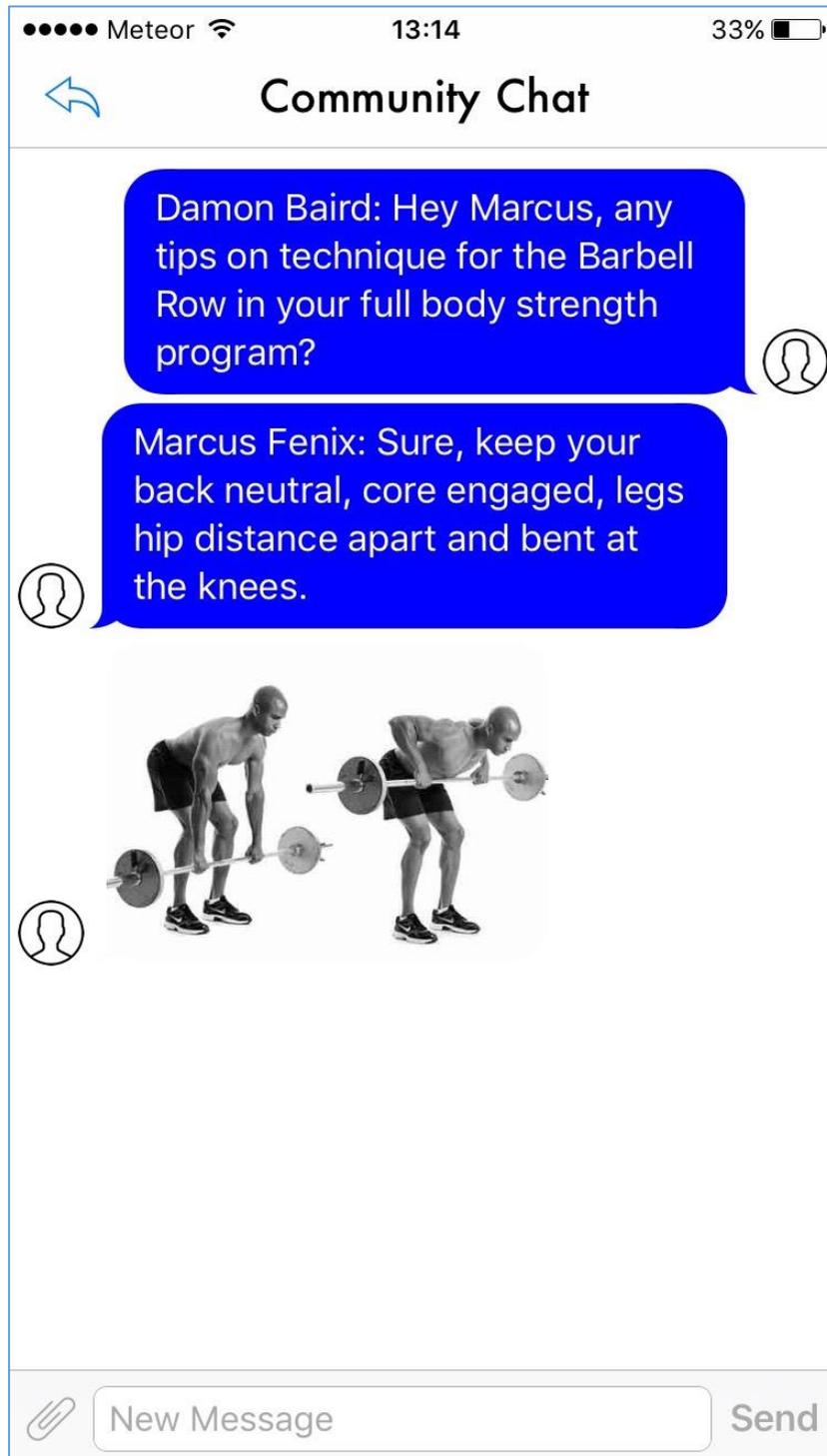
## 17.8 My Programs Screen

The My Programs screen allows the user to browse their list of private programs that they saved from the public list. Users can return to the Home screen from here using the back button.



## 17.9 Community Chat Screen

The Community Chat screen allows all users to communicate and have discussions. Users can ask for help or advice about programs listed in the application or health and fitness in general.



## 18 Testing & Usability

“The Tester” refers to the individual being interviewed and monitored to gain their feedback while they carry out test cases on the Gym Buddies application. Volunteer testers were invited individually into a room and provided with a brief outlining what would be required from them while carrying out testing on the application. The testers were made aware that notes would be taken by the developer based on the sessions and that the developer would be reviewing their responses for the purpose of developing the Gym Buddies application.

It was made clear to testers that there were no correct or incorrect answers and their honest feedback was crucial to the development of the application. It was emphasized that they have to think aloud as they carried out their tasks and they were also encouraged to do so if they encountered any difficulties. Testers were asked to keep any questions they might have until the end of the test.

### 18.1 5 Second Test

The goal of this test was to establish if the application’s design and key visual properties convey to the user the purpose of the application. From the 5 second test the user should be able to answer three crucial questions:

1. Who are you?
2. What product or service do you provide?
3. Why should I care (What is in it for me)?

To establish whether the Gym Buddies application makes this information clear or not, several users took part in a 5 second test and were asked three questions once the time limit expired.

1. What do you think this application does?
2. What do you like most about this application?
3. What do you like least about this application?

#### 5 Second Rest: Results

What do you think this application does?	What do you like most about this application?	What do you like least about this application?
“Gym helper”	“Clear labeling”	“Have to log in”
“Workout assistant”	“Simple design”	“Nothing I can think of”
“I assume it is an exercise program manager”	“Divided into clear sections”	“Blue color scheme”
“Application for finding a gym buddy”	“Useful features”	“ Not really a topic that interests me as I don’t workout ”
“Allows you to make exercise programs”	“Good graphics”	“Can’t think of anything”

## 18.2 Think Aloud test

The think aloud tests provides a wealth of valuable information as it demonstrates a user's thought process when interacting with an application. The think aloud test makes it possible to identify issues before a product goes live. A list of the most common functionality in the application was compiled and added to the think aloud test. Using this list of functionality, questions were drawn up to evaluate the design and layout of the application.

### Think Aloud Tasks

- Create an account
- Login
- Create a workout program
- Browse public workout programs
- Save a public program to your private programs
- Browse community chat

### Think Aloud Test: Results

Task	Result
Create an Account	<ul style="list-style-type: none"><li>• Without hesitation the testers successfully created an account.</li><li>• Required steps were taken quickly and efficiently.</li></ul>
Login	<ul style="list-style-type: none"><li>• Without hesitation testers successfully logged in quickly with no issues.</li></ul>
Create a workout program	<ul style="list-style-type: none"><li>• Testers completed task successfully on first attempt.</li><li>• Some users were not sure of what to type in to some fields as they do not participate in exercise but they understood the principle.</li></ul>
Browse public workout programs	<ul style="list-style-type: none"><li>• Testers completed task quickly without issues.</li><li>• Testers commented on the navigation being easy due to the home page.</li></ul>
Save a public program to your private programs	<ul style="list-style-type: none"><li>• Testers navigated to the public programs section and quickly identified the save button.</li><li>• Testers successfully completed this task with little to no instructions.</li></ul>
Browse community chat	<ul style="list-style-type: none"><li>• Some testers exhibited some slight hesitation regarding contacts list with the feature being a community chat.</li><li>• Once it was explained that the list was to show active users, testers understood.</li></ul>

## 18.3 Trunk Test

The main purpose of the Trunk Test is to examine how quickly users can gain their bearings inside an application when they are blindly placed in a particular section of the application. During this test, all participants were placed in the "All Programs" sections and where asked to carry out the following tasks:

- Identify what section of the application they are currently located in
- Identify section purpose
- Identify any possible options available to the user in the section

### **Trunk Test: Results**

- The “All programs” section was clearly identified by all testers due to its clear title labeling in the navigation controller bar at the top of the section.
- The programs list was identified immediately and all users browsed the section with no issues.
- Users clearly identified they had the option to save programs in the list using the “Save” button. Users also identified the back button in the navigation bar which they knew would return them to a previous section.
- Testers had no hesitation identifying that the presented application was a workout assistant app.

Overall, Gym Buddies passed the trunk test successfully. When observing testers, no hesitation was displayed when identifying the type of application, they were presented with. All testers successfully identified the name of the application section they were in, what purpose it serves and what options they have in the given section of the application.

### **18.4 Usability Testing: Heuristic Evaluation**

Heuristic Evaluation (Nielsen & Molich method) was performed as part of the suite of testing carried out on the Gym Buddies Application. The purpose of this test is to identify potential usability violations early and carry out any required redesigns and improvements as early as possible.

All participating testers were asked to evaluate the application based on the following areas:

- Visibility of System Status
- Match Between the System and the Real World
- User Control and Freedom
- Consistency and Standards
- Error Prevention
- Recognition Rather than Recall
- Flexibility and Ease of Use
- Aesthetic and Minimalist Design
- Recognize, Diagnose and Recover from Errors
- Help and Documentation

*\*Prior to testing, all testers had these headings explained to them*

## **Summary of Heuristic Evaluation Results:**

### 18.4.1 Visibility of System Status

The Gym Buddies application exhibited no major issues regarding visibility. All sections of the application were deemed consistent and presented well. Multiple testers communicated navigation in particular as being a strong point.

### 18.4.2 Match Between the System and the Real World

The Gym Buddies application relies heavily on text based features. Text is used primarily for labeling, user authentication and user supplied content. All static text used in the application is straight forward and easy to understand. Testers did not communicate any confusion or difficulties.

### 18.4.3 User Control and Freedom

Most feedback regarding navigation was positive and product functionality was deemed easily accessible. Users were particularly happy with how consistent and straight forward navigation was and liked the home screen and navigation bar.

### 18.4.4 Consistency and Standards

Testers unanimously stated that the design of the application was consistent and the color scheme and design patten was also consistent throughout.

### 18.4.5 Error Prevention

During testing very few errors occurred. Some users mistyped their credentials when logging in and others typed unsecure passwords when registering an account. All of these cases were caught by the applications error handling and an appropriate help message was displayed to the user. Individual testers did not encounter the same error twice during testing.

### 18.4.6 Recognition Rather than Recall

Testers were satisfied that meaningful titles and labels were used throughout the application. Testers never appeared confused about a section they were in or what its purpose was.

### 18.4.7 Flexibility and Ease of Use

Testers praised the application's consistent design and its clearly defined sections. Testers seemed to enjoy the functionality on offer and communicated little to no difficulties during use.

### 18.4.8 Aesthetics and Minimalist Design

A majority of testers expressed they liked the GUI design and the background imagery used. Testers also communicated their appreciation of the centralized navigation on offer as well as clearly defined sections.

#### 18.4.9 Recognize, Diagnose and Recover from Errors

Assessment of this area was difficult as minimal errors occurred during testing. Errors that did occur were clearly defined and seemed to help testers recover from their mistakes.

#### 18.4.10 Help and Documentation

Gym Buddies currently has no Help or FAQ section and some users did note its absence. This feedback has been noted and the feature will be included as a priority for Phase 2.

### 18.5 Testing: Conclusions

All testers completed their tasks successfully in a short period of time compared to time allotted. This indicated the application has a high level of effectiveness. Testers communicated very little effort was involved to complete their objectives which shows high performance under efficiency. Some users offered their opinion about their preferences for navigation and some small missing features but overall the application performed well and received mostly positive feedback. Tester feedback was crucial and any and all issues raised will help shape the application during the development lifecycle.

## 19 System Evolution Moving Forward

### 19.1 With Further Development & Research

- 19.1.1 Social Media Login functionality could be added at a later date. Some users are less likely to manually sign up for services with their email address and password. Having Social Media Login integration allows them to sign in to the application without having to register an additional account.
- 19.1.2 The application will be developed on iOS but could be ported to Android once development is complete.
- 19.1.3 While the application will be developed for iOS, it is possible to migrate the software to run on Microsoft's UWP using the Windows Bridge migration tool once development is complete.
- 19.1.4 As a way of generating revenue in the future, in application advertisements could be implemented. Arrangements with Gyms and Health supplement companies could be reached to advertise their products and services in the application.

## 20 Bibliography

Apple Inc, 2016. *Create a Table View*. [Online]

Available at:

<https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/CreateATableView.html> [Accessed 10 October 2016].

Apple Inc, 2016. *Start Developing iOS Apps (Swift)*. [Online]

Available at:

<https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/> [Accessed 10 October 2016].

Google, 2016. *Add Firebase to your iOS Project*. [Online]

Available at: <https://firebase.google.com/docs/ios/setup> [Accessed 12 October 2016].

Google, 2016. *Get Started with Firebase Authentication on iOS*. [Online]

Available at: <https://firebase.google.com/docs/auth/ios/start> [Accessed 15 October 2016].

Google, 2016. *Read and Write Data on iOS*. [Online]

Available at: <https://firebase.google.com/docs/database/ios/read-and-write> [Accessed 15 November 2016].

TheSwiftUniverse, 2016. *iOS Swift Tutorial: App Like Instagram*. [Online]

Available at: <https://www.youtube.com/watch?v=AdTvmvPSnlQ> [Accessed 5 December 2016].

Stack Overflow, 2016. *Change the Color of the Navigation Bar*. [Online]

Available at: <http://stackoverflow.com/questions/39931463/swift-ios-change-the-color-of-a-navigation-bar> [Accessed 15 November 2016].

Awesome Tuts, 2016. *Installing JSQMessagesViewController*. [Online]

Available at:

[https://www.youtube.com/watch?v=uSIlpfNMgVs&list=PLZhNP5qJ2IA0ZamF\\_MDzvmb3bNMv-mLt5&index=9](https://www.youtube.com/watch?v=uSIlpfNMgVs&list=PLZhNP5qJ2IA0ZamF_MDzvmb3bNMv-mLt5&index=9) [Accessed 5 February 2017].

Jesse Squires, 2016. *JSQMessagesViewController*. [Online]

Available at: <https://github.com/jessesquires/JSQMessagesViewController> [Accessed 15 February 2017].

## 21 Appendix

## 22 Project Proposal

## 23 Project Objectives

The general goal of the software project is to develop a non-trivial IT system, evaluate and present it at two different opportunities where it will be assessed by a critical audience. Completion of the project will require a number of skills such as independent learning, problem solving, coding, debugging, testing, time management and presentation. The aim is to develop a piece of software in an approved area of our choosing. The software must offer useful and innovative functionality, filling a gap in a current technology area or branching into a new one.

## 24 Project Background

Gym Buddies is a community based health and fitness mobile application. It will offer a social environment for people interested in health and fitness. The application will be fueled by user generated content, users will be able to submit and view exercise programs within the application.

All programs in the application will be submitted by users of the application, so if a user finds a certain program works really well for them or finds a useful workout on the internet, they can submit it to the application for other users to try out. Users will be able to rate programs submitted to the application, this will help ensure high quality content within the application.

Users can browse and try out programs as provided in the app, or they can save the program to their own personal list and make tweaks to them in order to better suit their needs. A user could have an injury or other health related reason not to do a certain exercise so they can then switch out certain ones as they wish from the program they are using.

The main focus of the application is to create a positive community for like-minded people to have a place to get together and help each other in achieving a common goal. Studies have shown that people maintain their motivation much better when exercising as part of a group and sharing experiences together. To this end, the application aims to provide a discussion section where people can discuss the programs in the application and any other health and fitness topic they wish. Additionally, a one to one messaging system will be attempted, this serves many purposes but the main one is to help users pose any questions they may have about a program to its author.

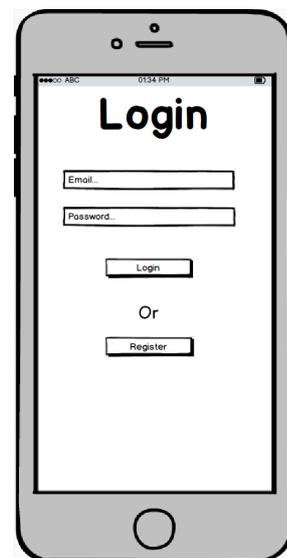
This idea was chosen as there is huge interest in the health and fitness industry and activity in this area continues to rise. While the health and fitness industry continues to grow at a rapid pace, unfortunately, so too do obesity levels. Ireland is on track to become the most obese nation in Europe within the next 10 years so another tool to help tackle this, as well as encouraging people to be more active, can only be a good thing. The app aims to give people a tool to help them be their best self and turn the tide on obesity. To this end, the application will also have a “gyms nearby” feature. As the name suggests, this feature will show users the closest gyms to their current location plotted on a map. All the proposed features of the application will be broken down in more detail under the Project Functionality section.

## 25 Project Functionality

### 25.1 User Sessions & Authentication

As a community based application it is a necessity for users to be able to sign up and log in with a unique user ID and password. In order to offer this functionality, users will need to be able to register and then log in to the application. The application will require an online database to store users and offer authentication.

For this application, Google’s Firebase database has been chosen. Firebase can be configured to offer a lot of security benefits such as minimum password length, letters and numbers and password encryption. Not even the database administrator can view a password belonging to a user of the application.

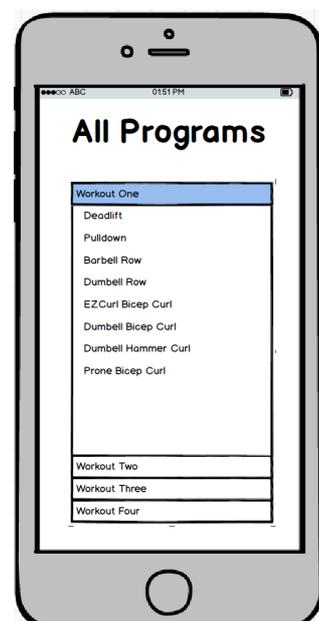


### 25.2 Submit & View User Submitted Programs

Once registered as a user of the application, users will be able to browse through exercise programs within the application to find one that suits their needs and matches their goals.

Users will also be able to submit their own exercise programs to the application for other users to view and try out. If a user has a program that works well for them or they find something great online and want to share it with the community, they can easily do so.

A ratings system will also be implemented so users can rate a program they try out up or down. If a program gets too many down votes it will first be put to the bottom of the list before being removed by administrators/moderators. More up votes mean more prominent placement in the list.

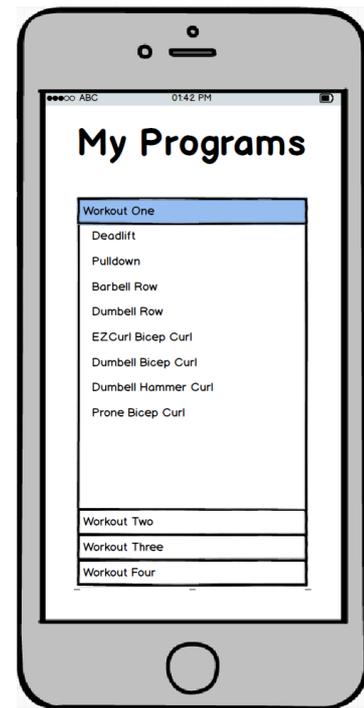


### 25.3 My Programs

An important feature, and one that was suggested during the project pitch, is one that was actually being considered over the past few weeks, “My Programs”.

The application will offer the ability to save a workout program from the list of user submitted content to your own personal list of workouts. This is an extremely useful feature as a user may find a routine that nearly perfectly meets their needs but there could be one or two aspects of the program they either can’t do, or don’t want to do.

There are always certain lifts or cardio routines that people hate doing, there are also medical reasons as to why people may not be able to do certain things. Someone with a lower back injury would probably be advised not to do deadlifts for example. This feature lets users easily pull down and make changes to a routine they want to try out and lets them keep track of it in their own private section.



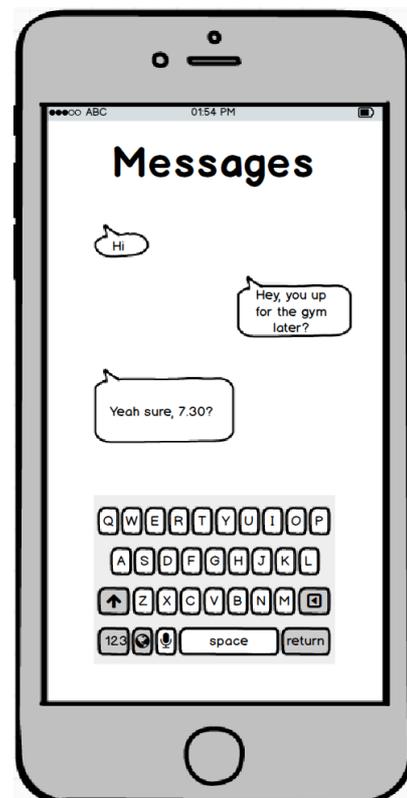
### 25.4 Community Discussion & Private Messaging

Part of the core concept for this application is the sense of community on offer to the users. Working out and getting healthy is a tough journey, especially for beginners.

In order to create a great community of people that help each other, it is vital that users are able to communicate in a practical way that best nurtures a sense of community.

To this end, the application aims to offer a place for public discussion, topics can vary from talking about the different programs posted in the application, to nutrition advice, to gym recommendations. It is important for people to be able to get the information they need and to potentially make friends.

And the other side of the communication coin is one to one messaging. If a user meets someone they befriend and wants to discuss things one to one, this feature will suit them. Or if someone has a question about a program posted in the application, why not message the author?



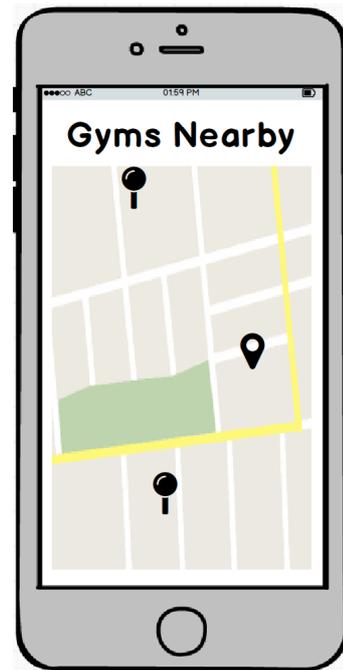
## 25.5 Gyms Nearby

Another useful feature the application will offer is the ability to find a gym or activity center nearby. If a user of the app is new to working out and has never been a member of a gym this is especially useful.

This feature will show the user in their current location on a map and within a certain proximity will pin point a gym or leisure center.

This will also help fuel discussions in the social section of the application as people will be able to discuss gyms they are shown nearby and ask if anyone else has used them before and if so, are they any good.

A future feature could expand on this to also display things like membership prices and opening hours as this information could potentially be available through Google Maps.



## 26 Target Audience

Gym Buddies is aimed at users of any experience level. Whether you are a complete beginner to exercising, or a more experienced athlete looking to mix things up, Gym Buddies has something for everyone. Beginners will find a host of user submitted content that will start them on the road to being fitter and healthier. Experienced users will have a new tool to look through and find new routines they may not have tried before.

The most important thing is that beginners will have other beginners to talk to and motivate each other. They will also have more experienced people to provide them with workouts that suit their needs and offer help and advice. Experienced users will have other experienced users to discuss goals with and help each other progress.

The only restriction on the applications target audience is that users will need to be 18 years of age or older. As all the exercise programs submitted will be user created, users will need to agree to the Terms of Service when registering to use the application as they will be following routines at their own risk.

## 27 Technical Approach

An Agile approach will be used when developing this project. Development will be an iterative process and will require a high level of adaptability as design and logic changes are inevitable. Activities will be as follows:

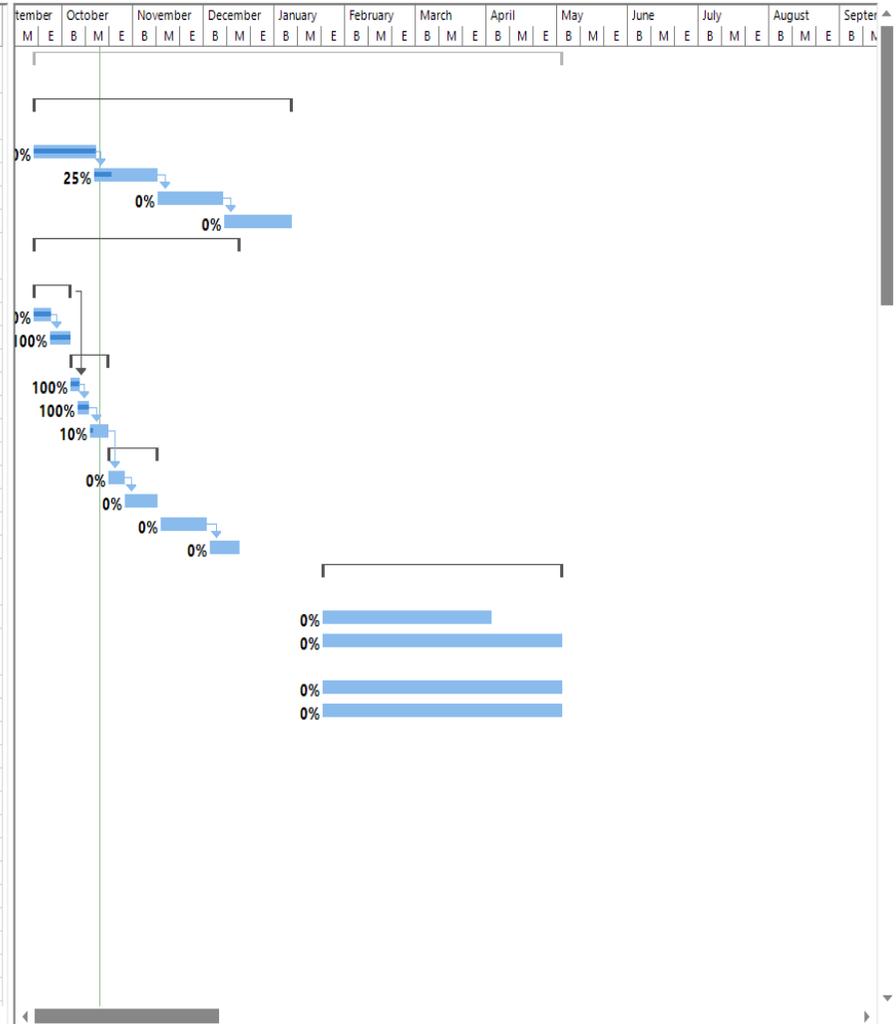
- Topic research and self-learning over the course of the whole project
- Requirements gathering based on proposed functionality and supervisor feedback
- Iterative implementation
- Testing and debugging in cycles during development

## 28 Special Resources Required

Resource	Description
Apple Mac Device	Developing iOS applications requires the XCode Integrated Development Environment. This IDE is not available on Windows so an Apple Mac is required.
XCode	XCode is the Integrated Development Environment used to create applications for iOS and other Apple Devices.
iPhone Simulator	The iPhone Simulator is an emulator used to test iOS applications outside of their native environment. It is used to run iOS applications created in XCode on your Mac device without the need for a physical iPhone device.
iPhone Device	An eventual goal is to run the application on the device it was designed to run on. While the simulator will serve all required purposes during early development, full testing and demonstrations will require the application to run on a physical device.
Apple ID	iOS development requires an Apple ID, this is then used to create a Developer account which is used to allow testing of your code on physical devices that have your Apple ID signed in.
Google Firebase	A No SQL, real-time, online database used to permanently store application data.

# 29 Project Plan

ID	Task Name	Duration	Work	Start	Finish	Predecessors	Resource Names	% Complete
0	Software Project Plan	224 days	245 hrs	Mon 19-9-16	Mon 1-5-17			10%
1	Reflective Journals	109 days	20 hrs	Mon 19-9-16	Fri 6-1-17			30%
2	RJ #01	26 days	5 hrs	Mon 19-9-16	Sat 15-10-16			100%
3	RJ #02	26 days	5 hrs	Sat 15-10-16	Thu 10-11-16	2		25%
4	RJ #03	27 days	5 hrs	Fri 11-11-16	Thu 8-12-16	3		0%
5	RJ #04	28 days	5 hrs	Fri 9-12-16	Fri 6-1-17	4		0%
6	Project Deliverables Semester 1	87 days	100 hrs	Mon 19-9-16	Thu 15-12-16			29%
7	Project Pitch	15 days	8 hrs	Mon 19-9-16	Tue 4-10-16			100%
8	Prepare Notes	7 days	4 hrs	Mon 19-9-16	Mon 26-9-16			100%
9	Prepare Slides	8 days	4 hrs	Mon 26-9-16	Tue 4-10-16	8		100%
10	Project Proposal	15 days	20 hrs	Wed 5-10-16	Thu 20-10-16			55%
11	Research	3 days	4 hrs	Wed 5-10-16	Sat 8-10-16	7		100%
12	Project Plan & Gantt Chart	4 days	6 hrs	Sat 8-10-16	Wed 12-10-16	11		100%
13	Compile Document	7 days	10 hrs	Thu 13-10-16	Thu 20-10-16	12		10%
14	Requirements Spec	20 days	20 hrs	Fri 21-10-16	Thu 10-11-16			0%
15	Research	6 days	5 hrs	Fri 21-10-16	Thu 27-10-16	13		0%
16	Compile Document	13 days	15 hrs	Fri 28-10-16	Thu 10-11-16	15		0%
17	Project Prototype	19 days	40 hrs	Sat 12-11-16	Thu 1-12-16			0%
18	Mid-Point Presentation Prep	12 days	12 hrs	Sat 3-12-16	Thu 15-12-16	17		0%
19	Project Deliverables Semester 2	101 days	125 hrs	Fri 20-1-17	Mon 1-5-17			0%
20	Showcase Materials	71 days	20 hrs	Fri 20-1-17	Sat 1-4-17			0%
21	Final Project Hard Copies Documentation	101 days	50 hrs	Fri 20-1-17	Mon 1-5-17			0%
22	Software & Doc Upload	101 days	5 hrs	Fri 20-1-17	Mon 1-5-17			0%
23	Project Presentations	101 days	50 hrs	Fri 20-1-17	Mon 1-5-17			0%



## 30 Technical Details

- iOS mobile application developed in Swift 3 using the Xcode IDE
- Google Firebase database and Cocoa Pods
- Google Maps API
- iPhone Hardware API's

## 31 Project Evaluation

The project will be evaluated using unit tests, system tests and automation where possible. Primarily, testers will be sought out to participate in carrying out Quality Assurance (QA) Engineering on the project. Thanks to experience gained in carrying out QA on a regular basis during the work placement module, effective QA test plans will be drawn up and provided to participants in order for them to carry out thorough manual testing of the application.

As part of carrying out their QA duties, testers will document the result of each activity they carry out, afterwards documented results will be compared with expected results. Highlighting tasks without telling a tester how to do them is a great way to uncover bugs and errors that a regular user would encounter but a developer of the system could easily miss.

## 32 Reflective Journal: September

### 32.1 September 19<sup>th</sup>

I started my first week back in college on Monday September 19<sup>th</sup>, our first class was a Software project class and during this class a lot of the expectations for the module were outlined by Eamon Nolan. I wasn't too surprised to find out that there were deliverables in relation to the Software Project due within 2 weeks of starting back, the only thing now was to find a way of fitting these deliverables in around my five other modules and the labs and CAs they would have.

The deliverables included coming up with an idea for the Software project and fleshing it out enough to make a small set of basic slides for upload to Moodle. These slides would then be viewed by three members of college staff who would then be present for a presentation designed for you to pitch your idea without the aid of any props. The lecturers will then vote for whether or not they approve your idea. If approved, you can carry on with your idea and construct a formal proposal and Software Requirements Specification. If your idea is declined, you have to select a pre-approved project idea from the college, this would not be ideal.

### 32.2 September 20<sup>th</sup>

I had several ideas when arriving back to college, but one was always more fleshed out than the others. I went with my idea for a health and fitness mobile application, I chose the mobile stream so it made more sense to go with a mobile application for my final year Software Project.

My idea basically involves a social mobile application that will allow users to consume workout programs submitted by other users and also to be allowed to rate these programs to ensure the quality of the content in the application. I have multiple stretch goals I want to implement but I have never done anything resembling them before so they will be a massive part of my self-learning this year.

These ideas involve adding in a discussion forum inside the application to allow users to discuss content within the application as well as offer help and advice to each other on health and fitness in general. I also would like to implement a private message feature to allow people to message each other one on one, this would be great for any users who want to talk to the author of a program they're using and ask any questions they may have. Lastly, I want to implement a mapping feature to help users find Gyms near where they're currently located.

I am really setting the bar high this year, I had the luxury in the past of having a lot more time to compile my ideas and this afforded me an opportunity to pre-research my functionality and get a better idea of what was involved. This time, I have to come up with an idea to match the standards of a final year project and I have to outline ideas before doing any of the functionality on a platform I have no experience developing on. This year will be interesting.

### 32.3 September 23<sup>rd</sup>

My project pitch slides are due on Monday October 3<sup>rd</sup> and my pitch will take place on Wednesday October 5<sup>th</sup>, this gives me a little time to do some research and practice in developing on a mobile platform. I dabbled in Android for a week or 2 during the summer holidays between second and third year, I didn't really like the IDE (Integrated Development Environment) Android Studio, while it was possible to use other tools, a lot of the documentation online covered Android Studio.

In perhaps another risky move, I decided to try out developing iOS applications using the Swift programming language and the XCode 8 IDE. I looked at a lot of Apple's documentation on Swift today and looked at a few online tutorials. I think I picked the worst time to learn Swift as Apple have just released Swift 3. This means that a lot of the content online is now outdated Swift 2 code and for whatever reason there are quite large syntax differences when coding in Swift 3 versus Swift 2. I will need to think on this a bit it seems.

### 32.4 September 26<sup>th</sup>

I played around with some more Swift 3 tutorials today and I actually found a small amount of material that helped me learn some of the basics, I'll now have to look into integrating an online database with Swift as this is pretty central to most of my applications functionality. After some research it appears Swift won't connect to MySQL directly for security reasons, it needs a PHP bridge as a middleman. The problem is that my efforts to find working examples of this in Swift 3 has not been successful, I can only find older Swift versions and these are now outdated.

After researching further, I found a lot of people online recommending using Google's Firebase database technology. Firebase is a real-time database which provides an API that allows developers to store and sync data across multiple clients. I will look into this further.

### 32.5 September 30<sup>th</sup>

After some more research I had some luck integrating Firebase into an iOS Swift 3 project. I found out it would have been easier to implement it on Android which comes as no surprise since Google owns both Android and Firebase, but I learned a lot implementing it on iOS. I have experience now installing Cocoa Pods through the terminal on MacOS and including firebase authentication files in a Swift project. I will now start working on getting some log in functionality working and look into the rest of my project functionality.

### 32.6 October 1<sup>st</sup>

I have the weekend now to compile my idea into several slides for the judges to look at before my pitch. I have now made 3 slides giving an extremely brief outline of my project idea and the functionality I hope to implement. Now to hope it is enough to get approval!

### 32.7 October 2<sup>nd</sup>

With my slides finished I now have to go over my pitch on paper. I need to outline how I will go about explaining my project to the judges and also anticipate what questions they will have. My Supervisor gave 5 questions that we should answer during our pitch so I plan on sticking as close to them as possible when presenting. I am learning how to develop on a brand new platform and offering functionality and features I haven't done before so hopefully my idea will be enough and they can appreciate the challenge I have set myself.

## 33 Reflective Journal: October

### 33.1 October 5<sup>th</sup>

I have my project pitch today. I have gone over my notes a lot last night and this morning and I think I'm ready for the pitch this afternoon.

### 33.2 October 6<sup>th</sup>

I completed my project pitch yesterday afternoon and I think my project was accepted. While I never approach anything with a high level of optimism, I certainly was not expecting such an indifferent reception. It nearly felt like the three people in front of me were being asked to provide their own money to fund my project.

While my project is well above the level of all my projects in previous years it definitely felt like I had my work cut out for me pitching it to the lecturers. They said my idea has a lot of potential but I may need to build on one of my features to make it a bit more useful. I had actually been thinking about a way of building on the main feature of the application anyway so this was actually reasonable enough. I will speak to Eamon next week and find out for sure if my project was accepted.

### 33.3 October 10<sup>th</sup>

I spoke to Eamon this morning and my project was indeed accepted with the revision of the features I discussed with the lecturers I pitched to. So instead of only having user submitted exercise programs as is, I need to make it possible for users to save a program from the ones on offer and have it in their own personal list and be able to edit it and make changes to suit their needs. As I mentioned before, this was an idea I was actually playing around with in my head over the last few weeks so that's fine.

### 33.4 October 12<sup>th</sup>

The next item on the agenda is the formal fleshed out project proposal. I have done many of these over the years but I imagine this proposal will be a lot more detailed and require a lot more justification of features and decisions. I will speak to Eamon about the template for the proposal and then try to figure out what headings best suit my project.

### 33.5 October 14<sup>th</sup>

The proposal is due on the 21<sup>st</sup> of October so I have a week to get it finished and uploaded to Moodle. I emailed Eamon to ask about the proposal template provided on Moodle and we agreed that using a mixture of that proposal template and the proposal I did for my third year software project would be best. So now just to actually do it!

### 33.6 October 15<sup>th</sup>

I made a start on the project proposal today. I have my headings finalized and have made a good start on the content. I'll need to work on my project plan and some mockups as these are required for the proposal as well.

### 33.7 October 16<sup>th</sup>

I finished off the content of the proposal today. All my headings now have their content complete and I now just have to do my mockups and my project plan Gantt chart. Fitting in all this documentation around my other modules is proving quite difficult at the moment, sometimes feels like I'm drowning in work. Will start on the Gantt chart and mockups soon.

### 33.8 October 18<sup>th</sup>

I drew up my project plan/Gantt chart for my proposal today. I ended up having to use a free online alternative to Microsoft Project as I was working on this at home. It was actually a half decent version if not a bit buggy. I did up my plan in detail for semester one and a little less detailed for semester two. Just the mockups left to do now before Friday.

### 33.9 October 19<sup>th</sup>

Completed my mockups for the proposal today. I had initially done really basic looking ones with an online tool but when I added them to the document they just looked terrible. I downloaded a 30-day trial of Balsamiq and I mocked up all the GUIs for the features proposed on my document. Balsamiq was a great tool and very intuitive to use. Was happy with my mockups and they are now added to the final proposal.

### 33.10 October 20<sup>th</sup>

Since my proposal is finished and I have a day before it has to be uploaded I decided to send a copy to Eamon to see if he had any pointers or could find any issues. Hopefully he has time to get back to me before it is due for upload.

### 33.11 October 21<sup>st</sup>

I heard back from Eamon with feedback on my project proposal. He was happy with my document and he said it was good for upload. I have now uploaded my proposal and its officially off my plate for now!

### 33.12 October 24<sup>th</sup>

Now that our project proposals have been submitted we are now formally assigned our project supervisors. I had contacted my supervisor during the summer and asked him to be my supervisor so he has been assigned to me and we can now proceed with meetings and email contact in relation to moving forward with the project.

### 33.13 October 28<sup>th</sup>

My supervisor has requested that all his assigned students meet with him on the Monday following reading week, so 7<sup>th</sup> of November. I will definitely be aiming to make it for the meeting but it will be a tight fit around studying for a CA on the Tuesday and all the other assignments I'm working on. Security paper, SRS for the Project and Web API CA. I'll have to start working on the SRS soon.

### 33.14 October 30<sup>th</sup>

I spent some time looking at the SRS template on Moodle and looking over some of my SRS documents from third year. I'll have to mull over some ideas and start properly on the document over the reading week. I'm taking Halloween off though!

## 34 Reflective Journal: November

### 34.1 This Month

This month I managed completing my SRS document including all diagrams. I forwarded it on to my project supervisor for feedback and was advised it was completed to his satisfaction.

I also have mostly completed my project technical report. I have added in all required details from my SRS and my proposal and also completed all extra required headings for the document. I have forwarded this on to my supervisor and it has been approved.

I also spent some time sourcing some good iOS development material to look over for developing the prototype for the midpoint presentation.

### 34.2 Next Month

With most of the documentation for semester one completed for the software project module, I will focus on compiling my presentation for the midpoint assessment and also work on developing my prototype. I'm currently thinking just get one or two smaller features working and have navigation working to show the main concepts of the application minus their functionality.

### 34.3 Reflection

While I got a good amount of work done this month, I am a bit frustrated at the workload for this semester in general. I would have liked to have already made a start on the prototype as leaving it until the last couple of weeks is just adding stress.

## 35 Reflective Journal: December

### 35.1 This Month

This month I completed my mid-point project technical report. All content and diagrams have been completed and appended to the document. I also compiled a set of Power Point presentation slides for my mid-point presentation and received approval for them from my supervisor.

I coded up a prototype of my project for demonstration at the mid-point presentation. The prototype demonstrates 4 of the final application's main requirements: user registration, user log in, submit a program and view all programs.

I completed the mid-point presentation at the end of December and it went very well. The presentation and prototype demo were well received and I was happy with the feedback given.

### 35.2 Next Month

With semester one now completed and the mid-point presentations over, I will now have to begin development of my project in earnest. Once my examinations are finished I will begin to build on the prototype and carry out more research and self-directed learning on how to complete the rest of the project functionality.

### 35.3 Reflection

I am happy with the work I completed this month. It was a lot to fit into one month but I managed it in the end. I have a lot of revision to do for four exams in January so the project will take a back seat for a while but I will resume work in late January/early February.

## 36 Reflective Journal: January

### 36.1 This Month

This month I did some polishing on the prototype I used for the mid-point presentation in order for me to be able to build on it and use it as my main project. I nested an extra query in the method that loads data for the table view that adds the username of the author of a program to each program listed. Using MySQL this would be the easiest thing in the world but with Firebase it is quite difficult.

I did some usability work on the application sections built so far so as when the user taps off the virtual keyboard or text field that the keyboard auto-dismisses. This means the keyboard no longer stays present until the user navigates away from that page.

I did some work on the UI constraints and added some new fields to the existing sections. The UI now has the required constraints to appear in the correct place across portrait and landscape orientations and also on different screen sizes and devices.

Lastly, I spent time researching and reviewing documentation and online resources around iOS development using Swift 3 and looked into ways of implementing some of the features I wish to include in the application. I managed finding a small amount of useful material considering the amount of time I spent looking but hopefully it will be enough to build on.

### 36.2 Next Month

Now that the prototype is in a condition that I can build upon it for my final project and I have some material to review from my research, I will hopefully be able to begin implementing more functionality into the application over the course of February. I hope to have some cloud messaging functionality implemented or at least a good idea of how to go about it. I also want to have made some progress on the analysis and design documentation.

### 36.3 Reflection

I am happy I got some of the polishing of the application done and that I now have the author of each program correctly displaying in the table view. I am slightly frustrated as already the other course work I have is eating into the time I wanted to spend working on the project.

Every module has a sizeable project again this semester and the deliverables fall at very awkward times so it means less time working on the project. It is frustrating because I know if I had more time to focus on the project I would be able to produce a much more polished and complete application.

## 37 Reflective Journal: February

### 37.1 This Month

This month I focused more on coding up project features. I started working on the messaging side of things and found a useful framework that can be implemented to enable messaging in an iOS application. The JSQMessages framework makes several methods available that allows the developer to build a messaging service.

I managed getting text based messages full implemented. Users of the application can navigate to the messaging section and from there they can see a list of users that are signed up to the application. This way they know if the person they want to message is available to see their message. Messages are being written to and read from the Firebase Database.

The decision was made to implement a community chat feature and push 1 on 1 private chat out to phase 2 after initial development is completed and the application is launched. Due to time constraints, it would not be possible to implement both a 1 on 1 chat feature and a community chat feature and the community aspect of the applications means the community chat was given priority.

While users can currently send text based messages to each other via community chat, I am currently working on making it possible for users to add images and videos to the chat. This would be a great addition to the baseline functionality as users will be able to take photos and record videos and then share them with people who ask for help or advice. It also makes it possible for users to share their progress with others.

Lastly, I have nearly completed my showcase poster. Should be finished entirely by the end of next month.

### 37.2 Next Month

Next month I want to have the cloud messaging functionality completed. I also want to complete the Analysis and Design document as it is weighing on my mind that I still have not made much progression with it due to the amount of work on my plate with other modules.

### 37.3 Reflection

I made some good progress this month with coding. I still have the rate program and my programs use cases left to do and if I have time the gyms nearby use case. There is still a good bit of work left to do both in terms of coding and documentation so I just have to keep at it.

## 38 Reflective Journal: March

### 38.1 This Month

This month I finished off my showcase poster and had it approved by my supervisor.

I finished off the cloud messaging functionality of my project and have it working with images and video.

I made good progress with the Analysis & Design document and have most of it finished, just the testing and screenshot sections remain.

I implemented functionality allowing users to save programs from the public list to their personal list for easy access.

Lastly I made some good progress with the UI. I have styled some good background images to be used throughout the application and aim to make more progress with the UI next month also.

### 38.2 Next Month

Next month I hope to complete the testing, UI and implementation section of the project documentation. I also want to complete work on the UI of the app and have a launch icon, launch screen and other UI elements finished. I will aim to polish up some functionality and carry out some bug squashing also. The semester is nearly over now and I just have to manage my time around studying for my exam and finishing off the project.

### 38.3 Reflection

I am relatively happy with my progress with coding up the app and writing up the documentation. It was a battle managing my time working on the project against completing all of my other college work but I feel I managed it well. Once my exam is out of the way I will have time to fully focus on finishing up work on the project.