1.1 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.

2 User Requirements Definition

2.1 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.

2.2 Product Functions

2.2.1 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.

2.2.2 Registered users will be able to sign in and post a workout routine to the application for others to try out.

2.2.3 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.

2.2.4 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.

2.2.5 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.

2.2.6 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.

2.2.7 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.

2.3 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.

3 System Architecture

3.1 Operating Environment

3.1.1 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 though 4. See architecture diagram under its heading below.

3.1.2 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.

3.1.3 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.

3.2 General Constraints

3.2.1 The software requires an Apple device running iOS version 8 or later with a touch screen and simulated keyboard.

3.2.2 The ability of the software to interface with an online database must be implemented.

3.2.3 The software must be able to access the devices GPS to offer location based functionality.

3.3 Assumptions & Dependencies

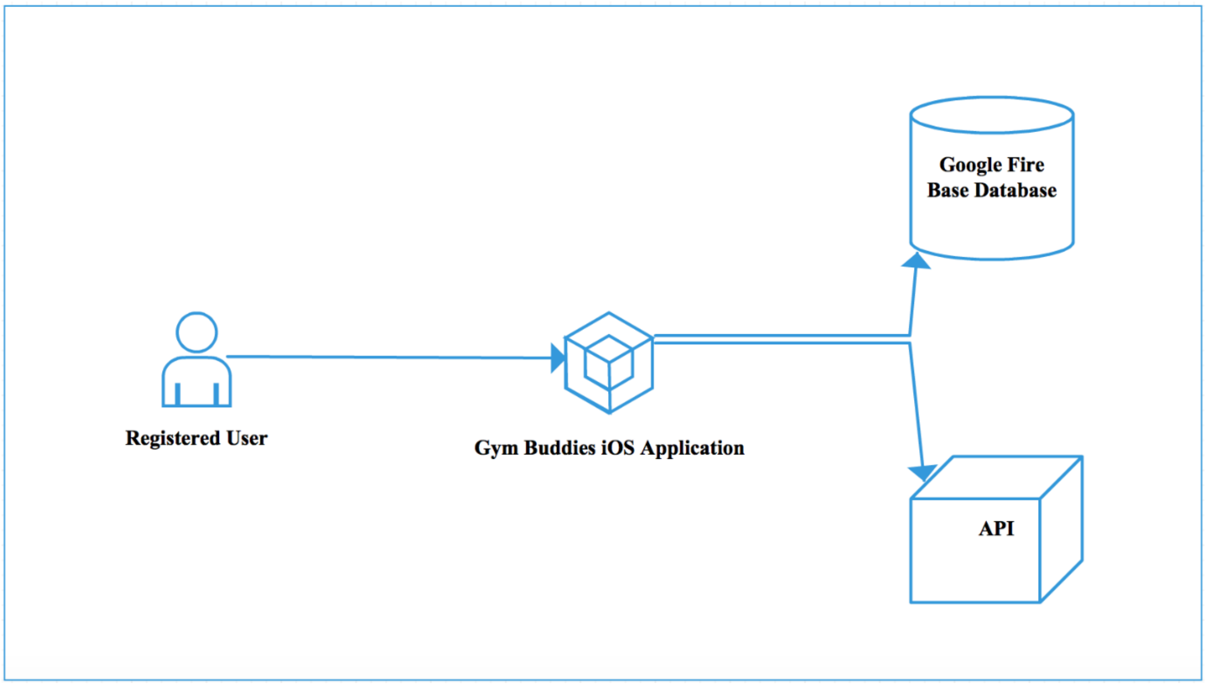
3.3.1 If no exercise routines are created by users, there will be no routines listed in the application for users.

3.3.2 If no internet connection is available, users will be unable to log in or access any functionality of the software.

3.3.3 Unregistered users will have no access to the software.

3.3.4 Users will only be permitted to register and gain access to the software once they agree to the terms of service agreement.

3.4 Architecture Diagram



4 System Requirements Specification

4.1 External Interface Requirements

User Interface

4.1.1 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.

4.1.2 The application should have its logo present on each screen once a logo has been designed.

4.1.3 A user friendly color scheme should be chosen, UI design should be carried out with visually impaired and color blind users in mind.

4.1.4 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.

4.1.5 The GUI should have continuity, all screens should have the same design and layouts should be consistent.

Hardware Interfaces

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

API Interfaces

4.1.7 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.

4.1.8 The software must show mapping information around the user's current location using the Google maps API.

5 Data Requirements

5.1 User Data

5.1.1 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.

5.1.2 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.

5.2 Workout Program Data

5.2.1 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.

5.2.2 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.

6 Functional Requirements

6.1 User

6.1.1 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.

6.1.1.1 User registration and login shall be mandatory.

6.2 Create an Account

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

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

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

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

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

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

6.2.6.1 The system should explain how the submitted password is unsecure.

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

6.3 Login

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

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

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

6.4 Submit Exercise Program

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

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

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

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

6.5 View Existing Programs

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

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

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

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

6.6 Rate a Program

6.6.1 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.

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

6.7 Save a Program to Personal List

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

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

6.8 Edit Personal Program

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

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

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

6.9 Send Message

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

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

6.10 Find Gym Nearby

6.10.1 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.

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

6.10.2.1 Such information could be opening hours, price etc.

7 Non Functional Requirements

7.1 Portability

7.1.1 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.

7.2 Reliability

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

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

7.3 Ease of Use

7.3.1 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.

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

7.4 Speed

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

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

7.4.3 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.

7.5 Size

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

7.6 Privacy

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

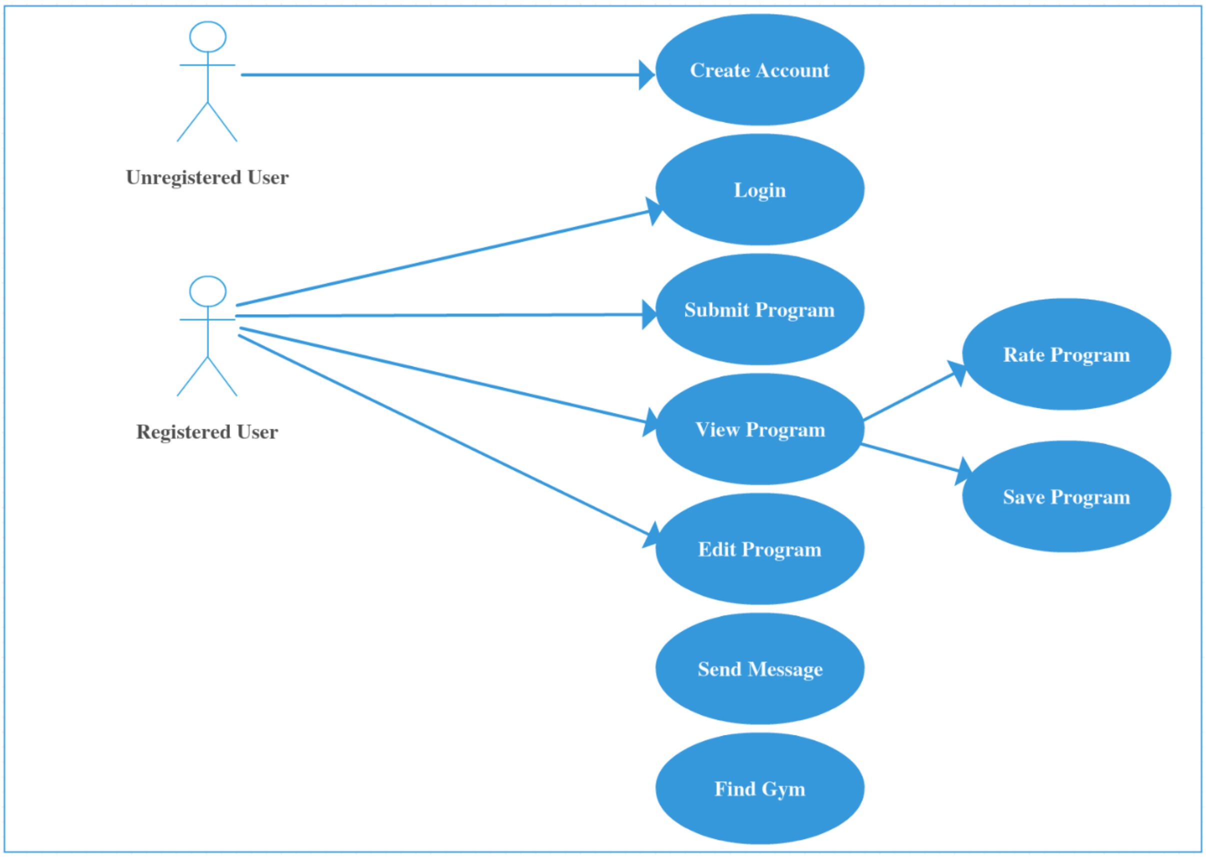
7.7 System Stress Management

7.7.1 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.

8 Use Cases

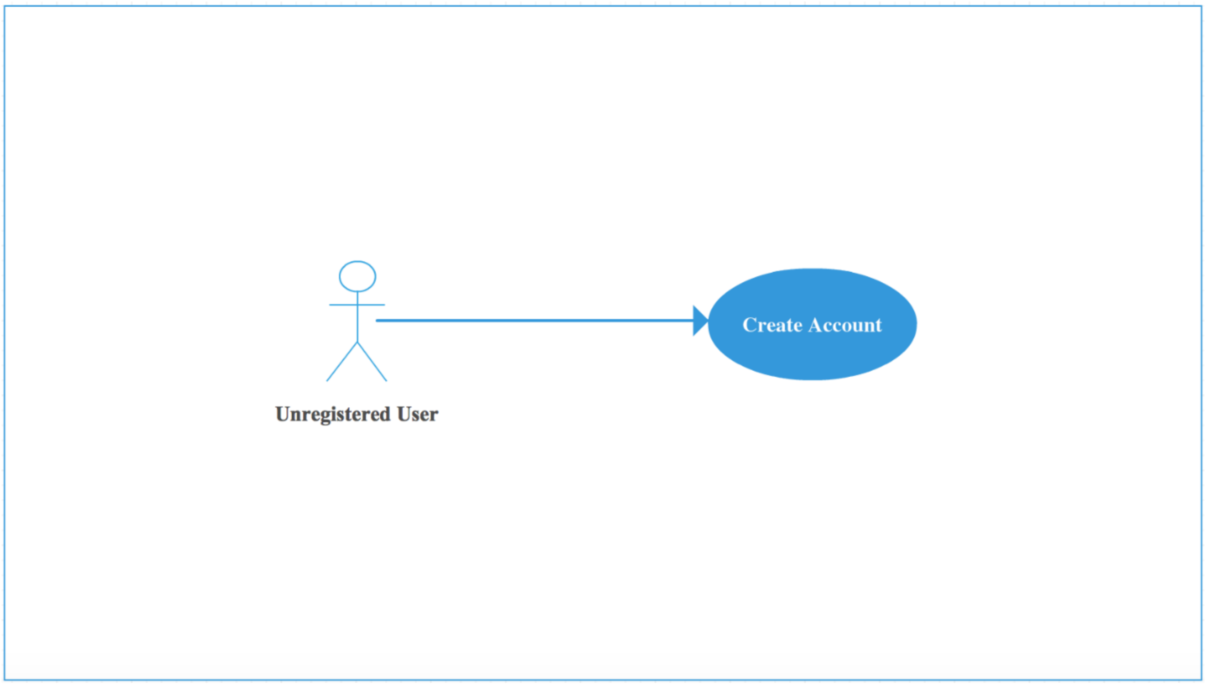
8.1 Use Case Index

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

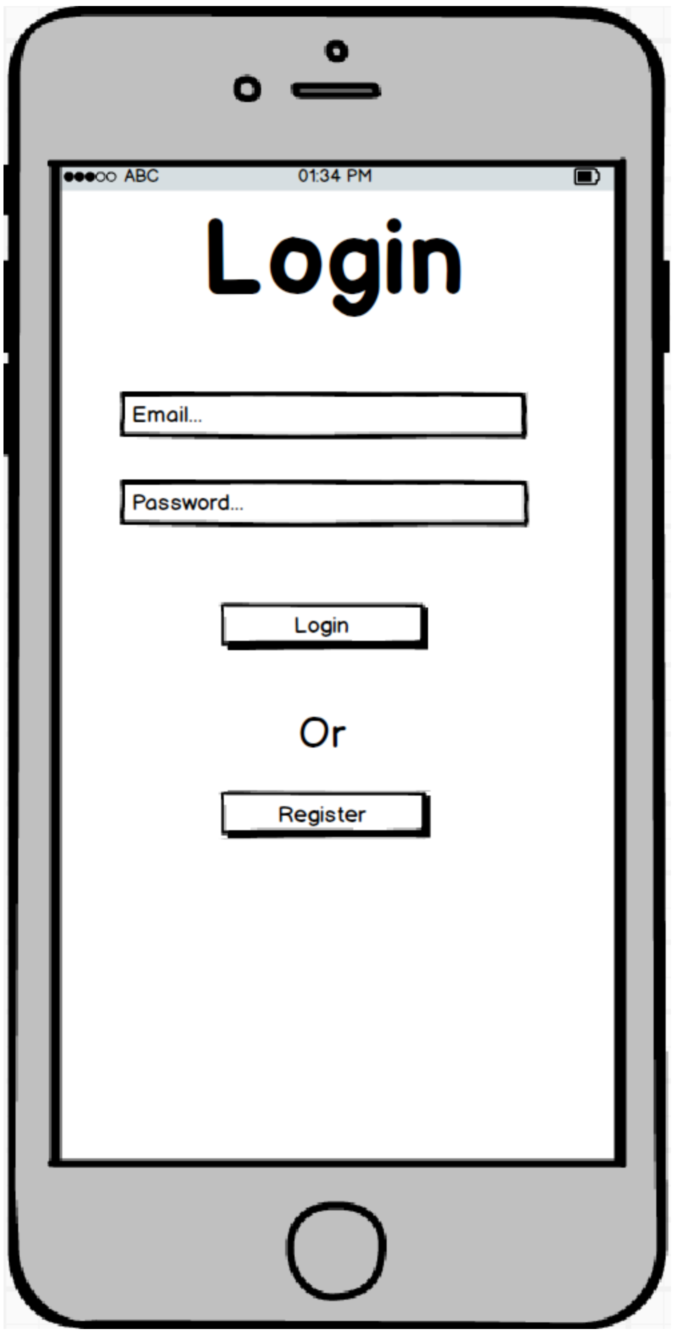


8.2 Use Case #1 – Create Account

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	1. This use case starts when a new user launches the application for the first time and opts to create an account. [AF1] 2. The user inputs their email and password into the allotted text fields. [AF2] 3. The application notifies the user that the account has been created and grants them access to the application.
Alternate Flows	1. The user does not want to create an account and chooses to exit the application at that point. 2. The user enters invalid characters into the text field or leaves them blank. The system notifies the user of their error.
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.

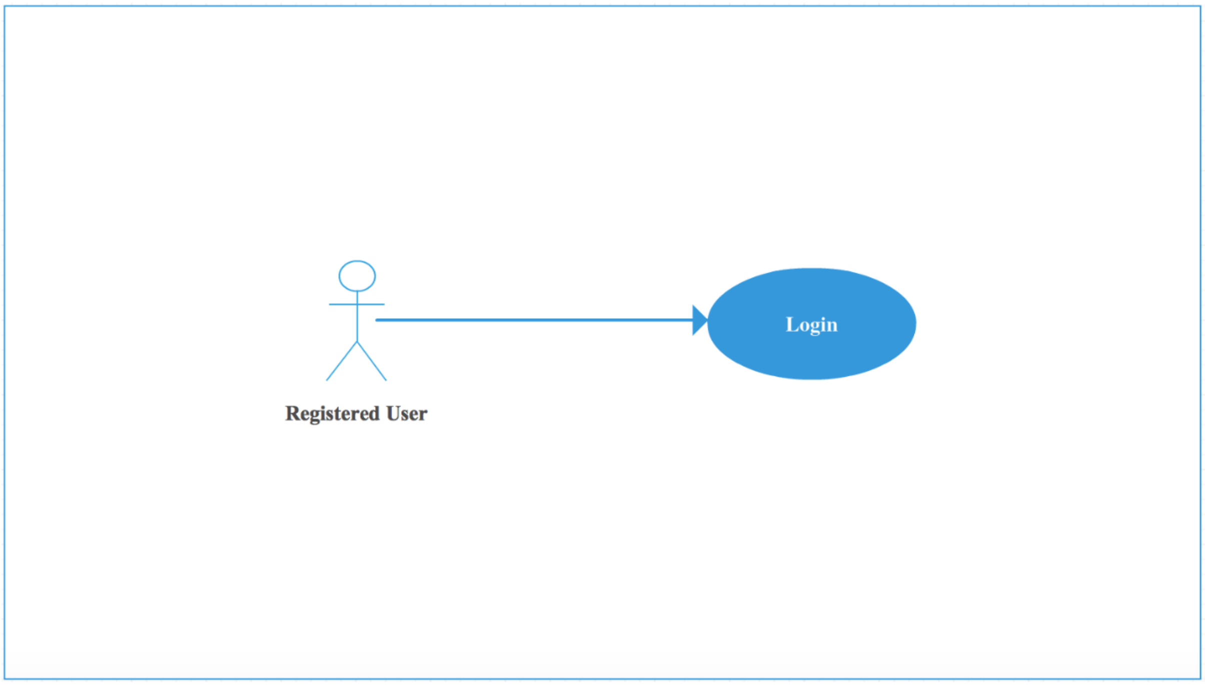


Use case #1 – Create Account Mockup

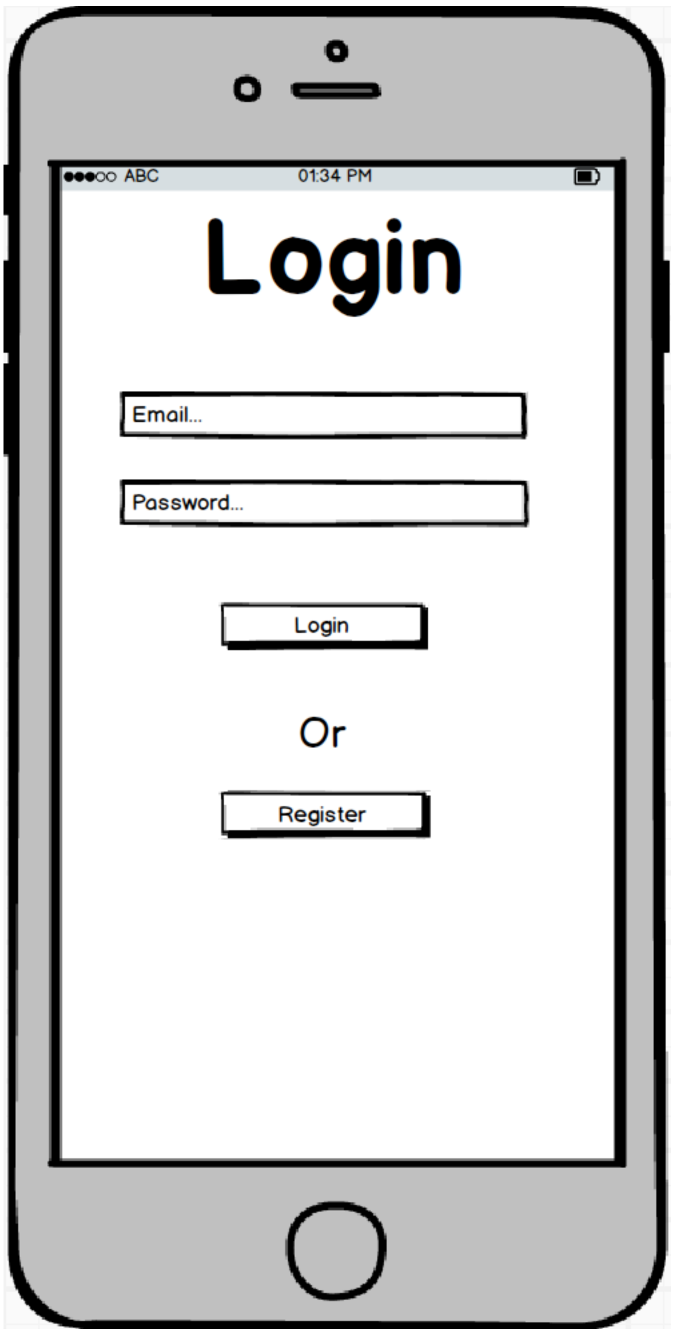


8.3 Use Case #2 – Login

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	1. This use case starts when a registered user runs the application and is presented with a login GUI page. 2. The user enters their username and password and is granted access to the system.[AF1]
Alternate Flows	1. The user inputs an incorrect username or password and is notified by the system.
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.

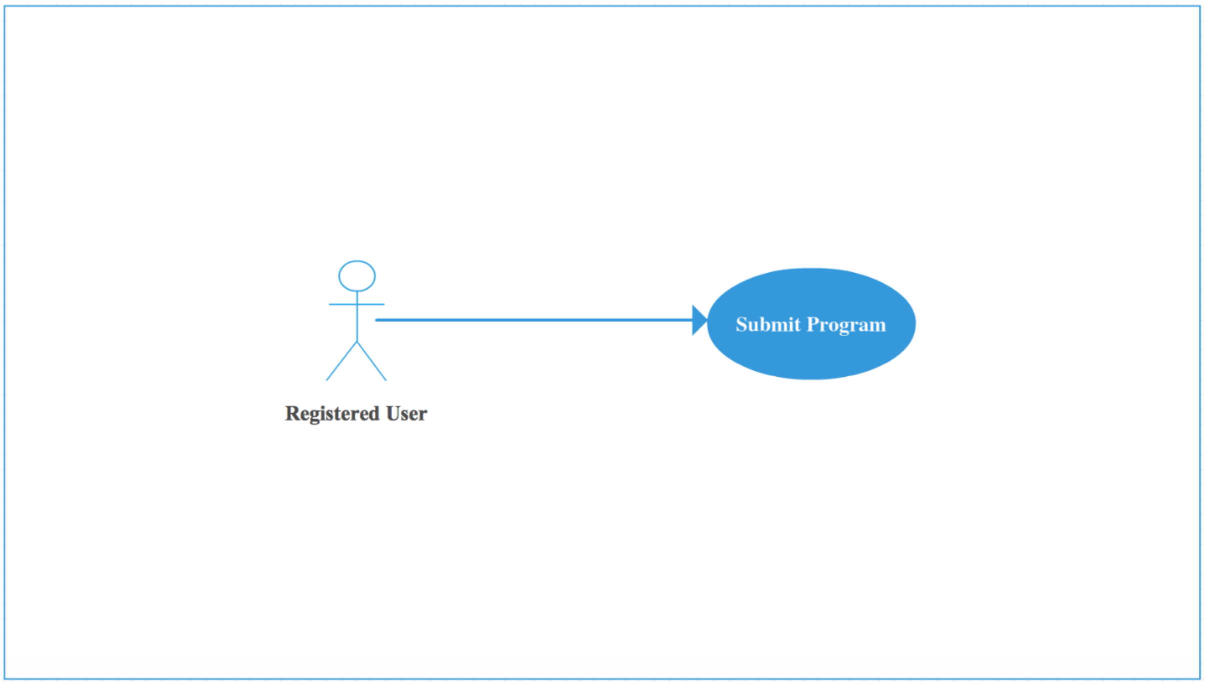


Use Case #2 – Login Mockup

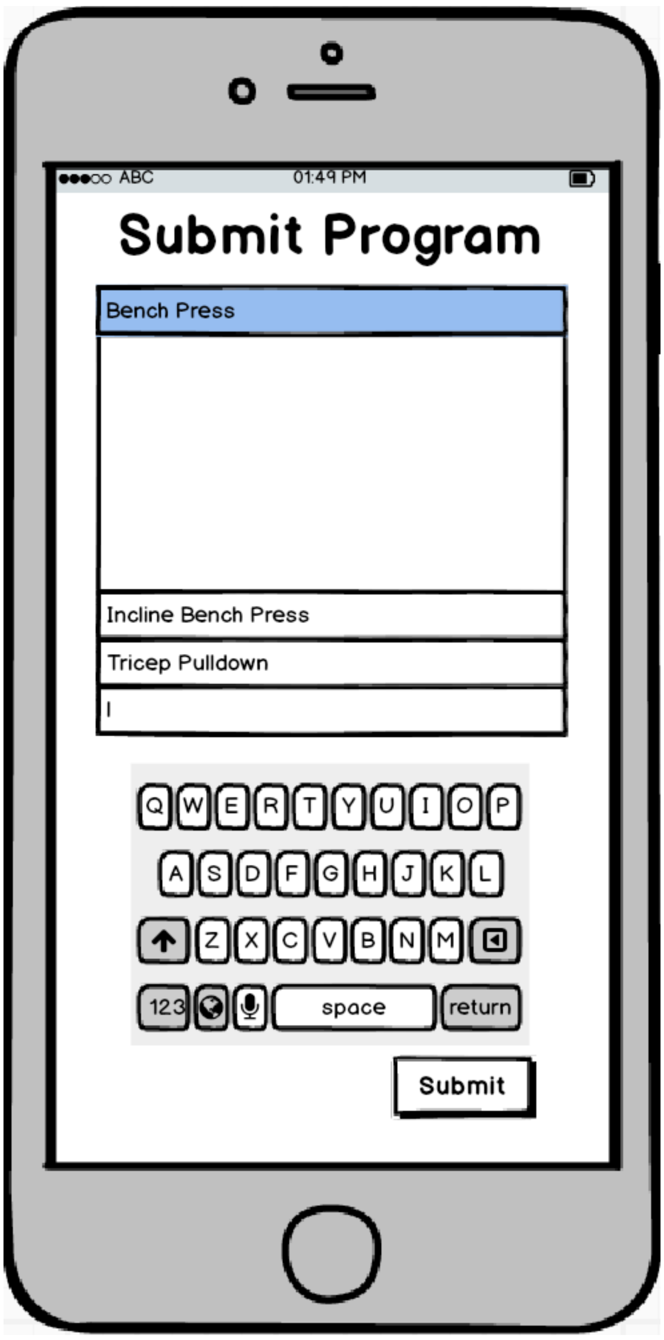


8.4 Use Case #3 – Submit Program

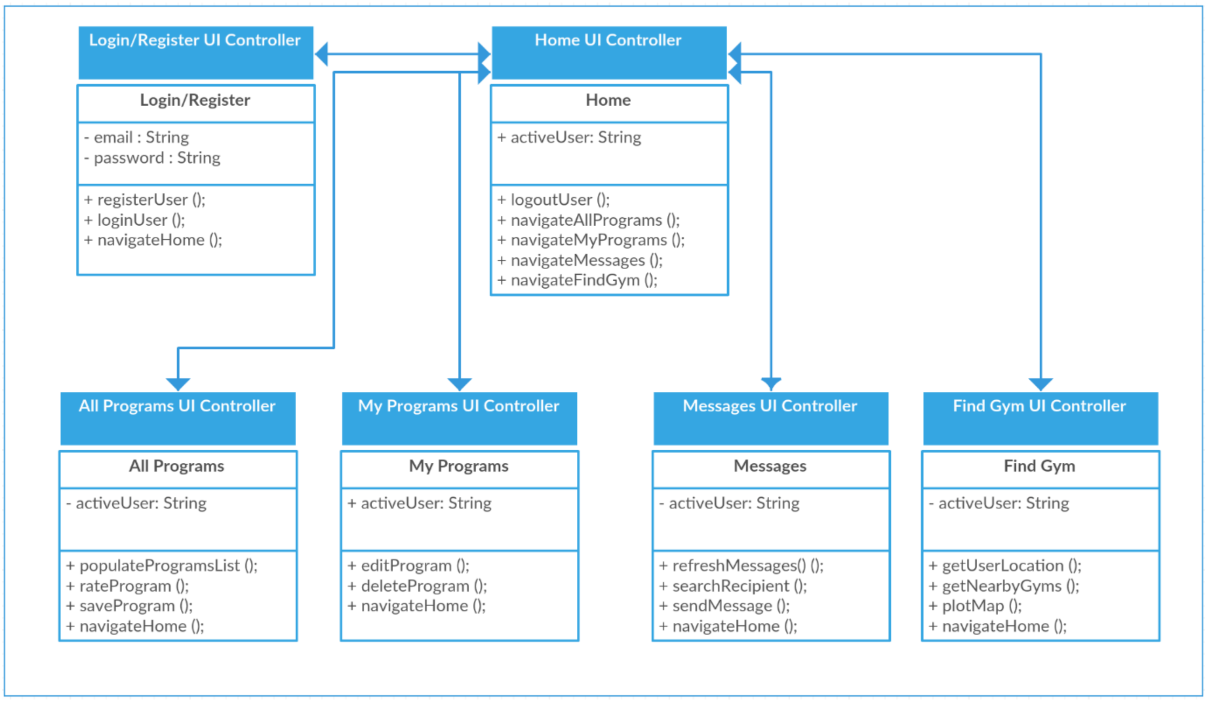
Use Case Element	**Description**
Use Case Number	3
Application	Gym Buddies
Use Case Name	Submit Program
Use Case Description	A registered user creates and submits an exercise program to the application.
Primary Actor	Registered User
Precondition	A registered user must be logged in.
Trigger	A registered user logs in and taps the Submit Program button.
Basic Flow	1. This use case begins when a logged in user clicks the Submit Program Button. 2. The user proceeds to enter the program details into the supplied fields. [AF1] 3. Once all details have been added the user then clicks the Submit button to add their program to the application.
Alternate Flows	1. The user enters invalid information or leaves some or all fields empty and the system informs the user of their error.
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 Storyboard/Class Diagram



10 Analysis & Design Specification

11 Analysis & Design Introduction

11.1 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.

12 General Overview & Design Guidelines

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

12.1 Assumptions, Constraints & Standards

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

12.1.1.1 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.

12.1.2 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.

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

12.1.3.1 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.

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

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

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

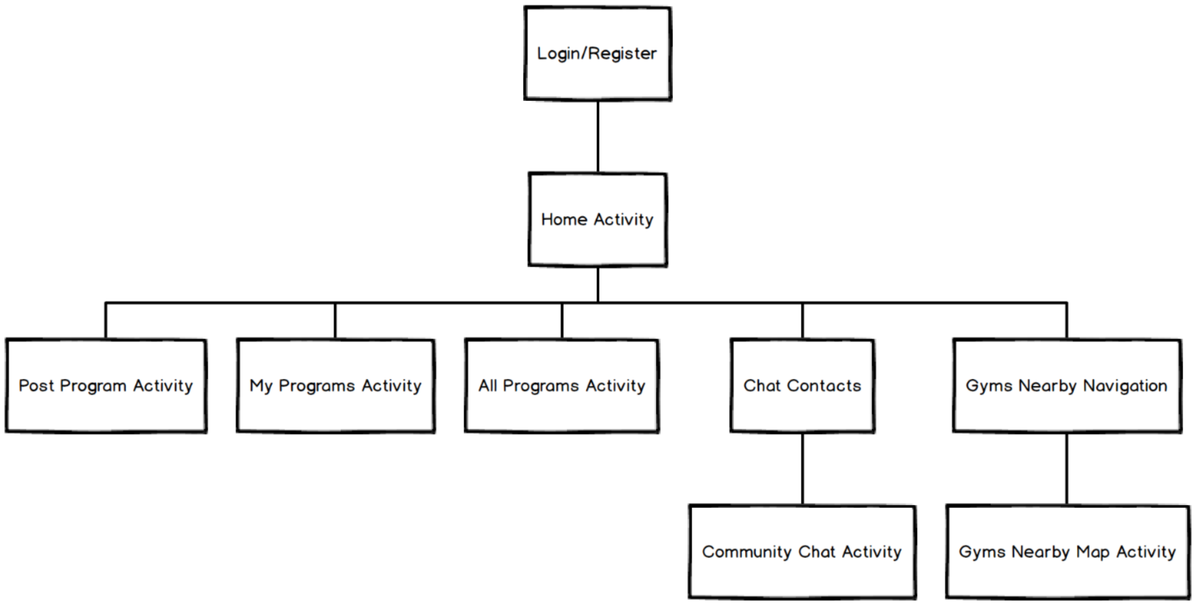
13 Architecture Design

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

13.1 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



13.2 Hardware Architecture

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

Relevant Hardware Components

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

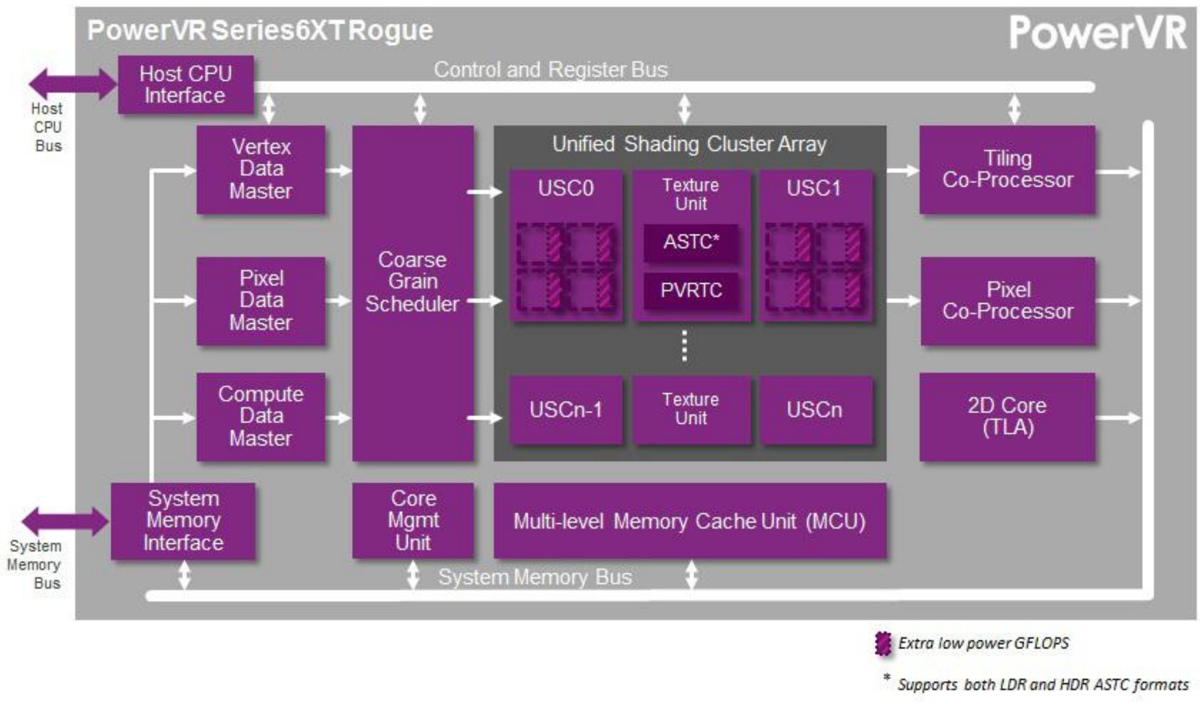


*Diagram Provided by ExtremeTech.com

13.3 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.

13.4 Security Architecture

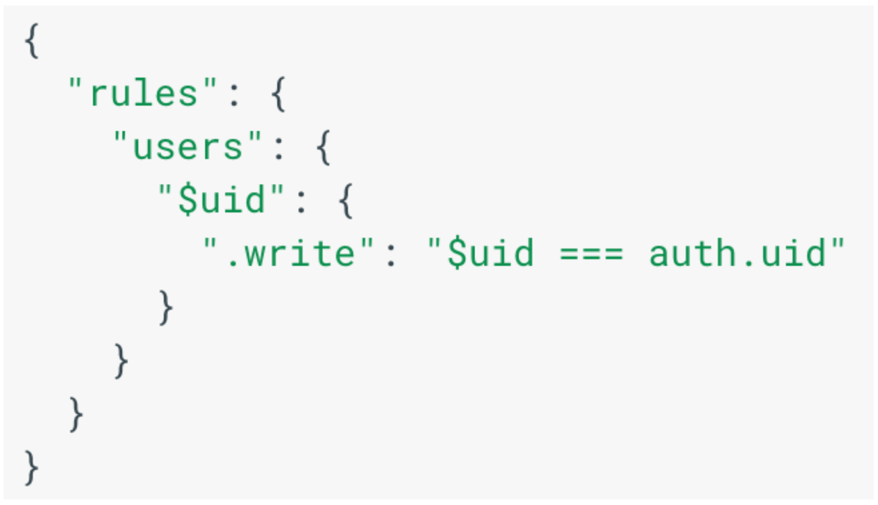
Blanket Database Security

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.

13.5 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.

13.6 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.

14 System Design

14.1 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:

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.

14.2 Application Program Interfaces

Firebase

1.1.1 Data persistence will be carried out using Google's Firebase 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 Firebase. The required API files can be included in the project via pod files installed using Cocoa Pods. Required pod files:

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 Firebase, the required files can be included via Cocoa pods. Required pod files:

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:

15 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.

15.1 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.

15.2 Sign in Registered User

The code snippet below shows the logic involved for singing 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.

15.3 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.

15.4 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.

15.5 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.

15.6 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.

Selecting File Type for Media Message

Listeners for New Text & Media Messages

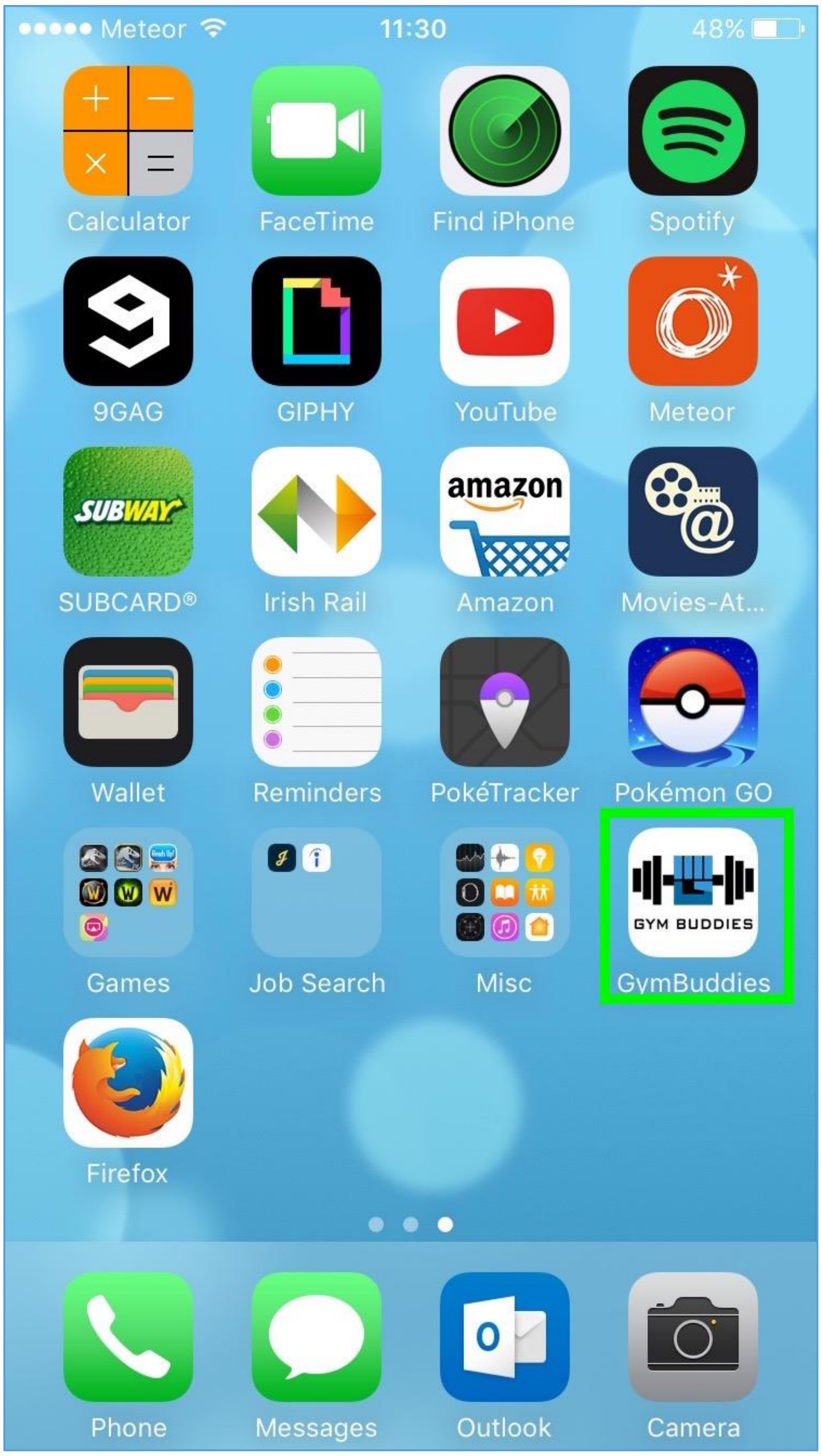
15.7 Centralized Navigation and Signing Out

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 them 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.

16 GUI Design & Layout

16.1 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.

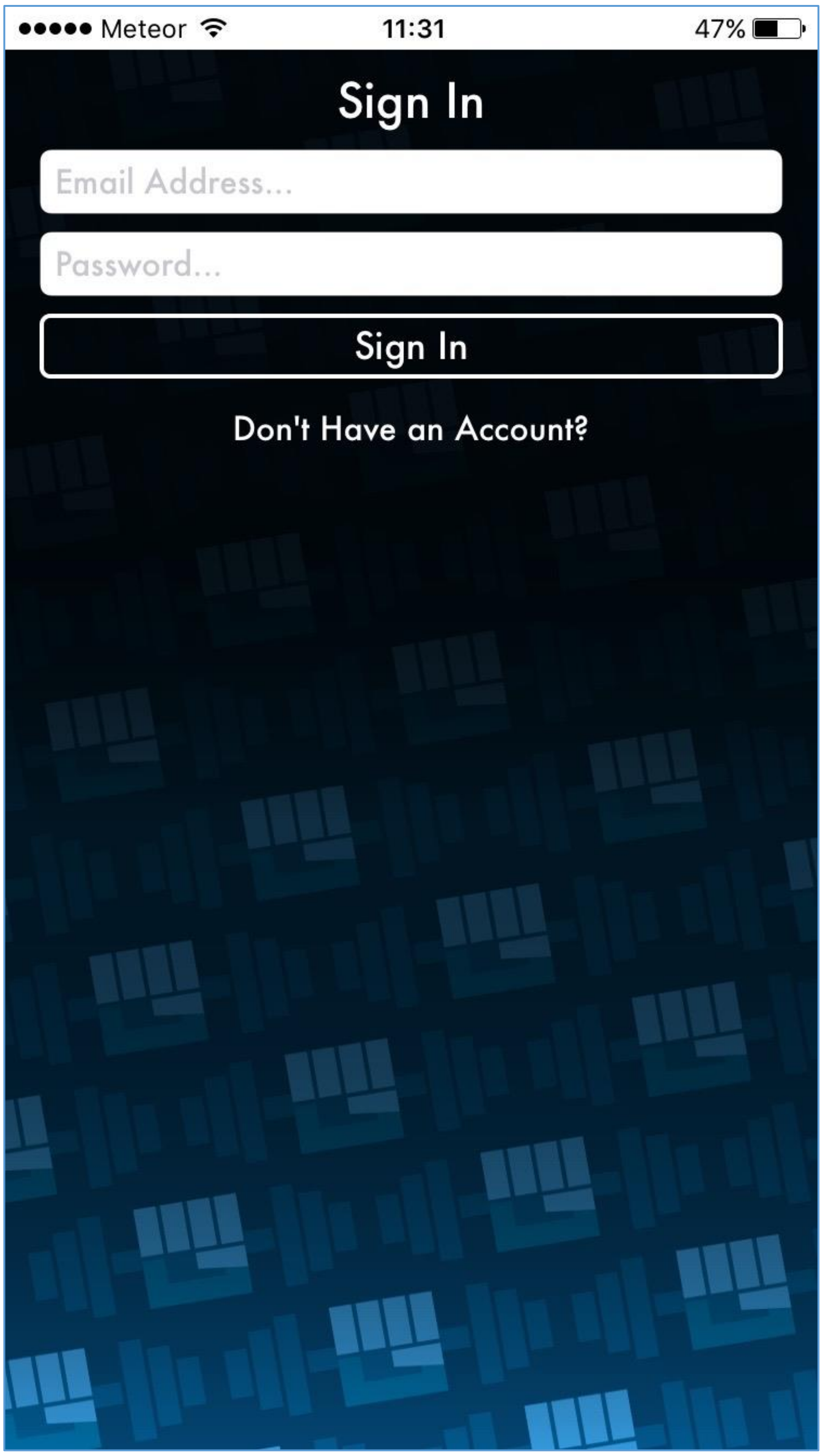


16.2 App Launch Screen

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.

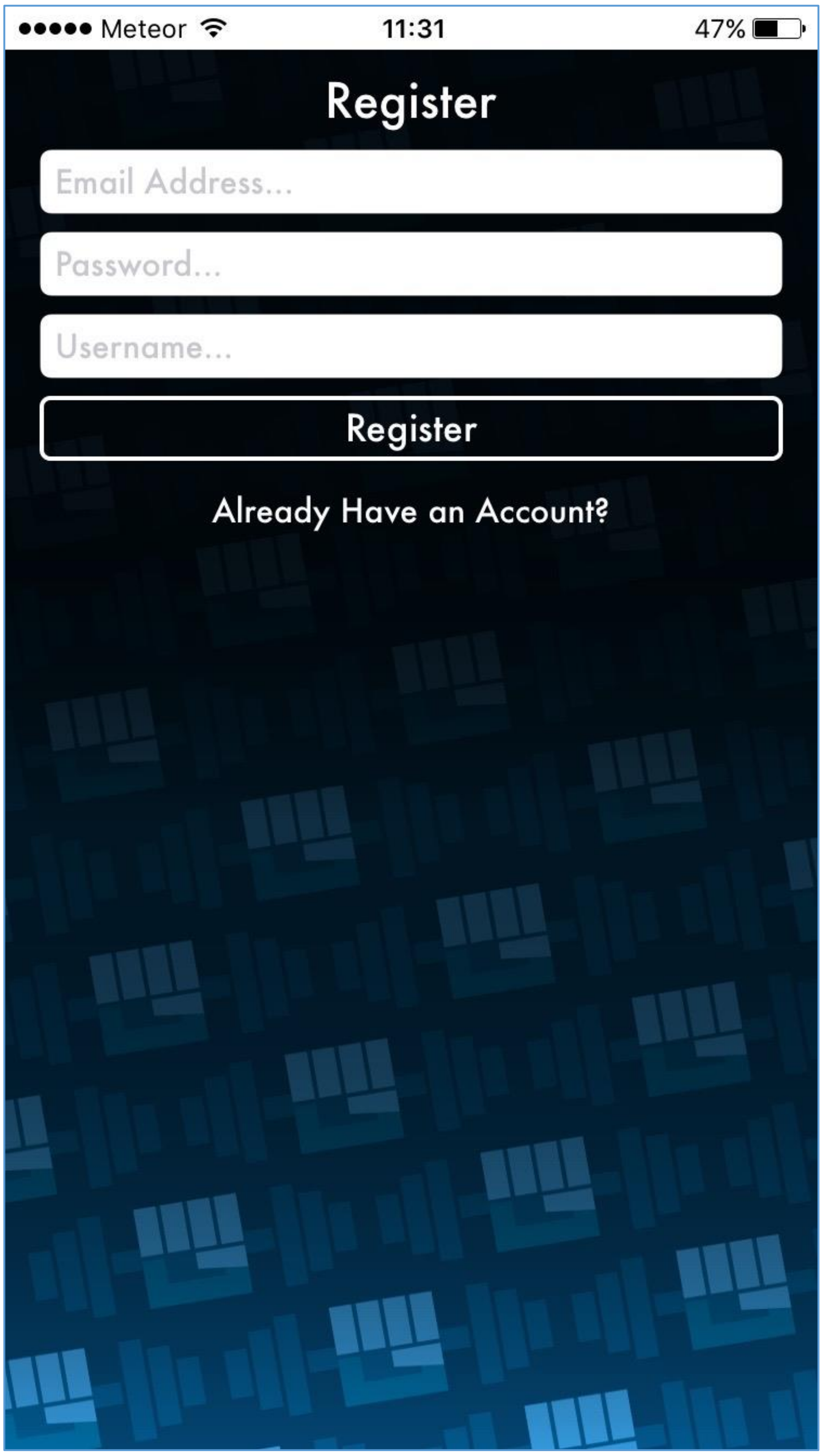
16.3 Sign in Screen

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



16.4 Register Screen

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



16.5 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 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.

17.1 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"

17.2 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	• Without hesitation the testers successfully created an account. • Required steps were taken quickly and efficiently.
Login	• Without hesitation testers successfully logged in quickly with no issues.
Create a workout program	• Testers completed task successfully on first attempt. • 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.
Browse public workout programs	• Testers completed task quickly without issues. • Testers commented on the navigation being easy due to the home page.
Save a public program to your private programs	• Testers navigated to the public programs section and quickly identified the save button. • Testers successfully completed this task with little to no instructions.
Browse community chat	• Some testers exhibited some slight hesitation regarding contacts list with the feature being a community chat.

	• Once it was explained that the list was to show active users, testers understood.

17.3 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.

17.4 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:

17.4.1 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.

17.4.2 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.

17.4.3 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.

17.4.4 Consistency and Standards

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

17.4.5 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.

17.4.6 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.

17.4.7 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.

17.4.8 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.

17.4.9 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.

17.4.10 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.

17.5 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.

18 System Evolution Moving Forward

18.1 With Further Development & Research

18.1.1 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.

18.1.2 The application will be developed on iOS but could be ported to Android once development is complete.

18.1.3 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.

18.1.4 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.

19 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 [Accessed 5 February 2017].

Jesse Squires, 2016. *JSQMessagesViewController*. [Online]
Available at: https://github.com/jessesquires/JSQMessagesViewController [Accessed 15 February 2017].