

Requirements Document

Brandon Lee, Rutger Farry, Michael Lee

CS 461
Fall 2016
4 November 2016

Abstract

C7FIT is a mobile application for iPhone developed as a senior software engineering project under the supervision of eBay Inc. This app is essentially a health and fitness application that will be utilized by members of Club Seven Fitness in Portland. Currently Club Seven gym clients have difficulty tracking their workouts, goals, and schedules in the digital world. C7FIT aims to integrate existing technologies from Club Sevens website as well as various iOS frameworks to provide users an accessible interface for interacting with their health and fitness goals on a mobile platform. The following document exercises the requirements of this project and formulates the plan of approach in developing this application in the next coming months. Such areas focused on include the project purpose, scope, perspective, interfaces, functions, constraints, and attributes. This document will additionally have elements of the higher level overview of the projects technical components.

Contents

1	Introduction	2
1.1	Purpose	2
1.2	Scope	2
1.3	Definitions	2
1.4	Overview	2
2	Overall Description	2
2.1	Product Perspective	3
2.1.1	System Interfaces	3
2.1.2	User Interfaces	3
2.1.3	Hardware Interfaces	3
2.1.4	Software Interfaces	3
2.1.5	Communication Interfaces	3
2.1.6	Memory Constraints	4
2.2	Design Constraints	4
2.2.1	Operations	4
2.2.2	Site Adaption Requirements	4
2.3	Product Functions	4
2.4	User Characteristics	4
2.5	Constraints, Assumptions, and Dependencies	4
3	Specific Requirements	5
3.1	External Interface Requirements	5
3.1.1	User Interfaces	5
3.1.2	Hardware Interfaces	8
3.1.3	Software Interfaces	8
3.1.4	Communication Interfaces	8
3.2	Classes and Objects/Functional Requirements	8
3.2.1	DataStore	8
3.2.2	Responding to State Changes	9
3.3	Performance Requirements	9
3.4	Logical Database Requirements	9
3.5	Software System Attributes	9
3.5.1	Reliability	9
3.5.2	Availability	10
3.5.3	Security	10
3.5.4	Maintainability	10
3.5.5	Portability	10
3.6	Gantt Chart	11
4	Signed Participants	12

1 Introduction

1.1 Purpose

The purpose of this document is to deliver a clear specification of the requirements for the C7FIT app. The document will define a clear objective for the development of C7FIT including software features, product interfaces, design constraints, product functions, user characteristics, assumptions, and dependencies.

1.2 Scope

C7FIT is an iOS mobile application designed for Club Seven Fitness in Portland. The app allows for gym members to engage with Club Sevens content on the mobile platform. In addition to interacting with user inputted data from Firebase, our app will utilize iOSs HealthKit, which brings a plethora of user health data. Such application includes displaying the users daily level of activity. Furthermore, an activity tracking feature will allow for users to record running workouts and store such data locally. Finally, this software will extend to incorporate features from the traditional fitness scope including timers/stopwatches, contacting the local gym (Club Seven) through an embedded messaging service, daily workouts and video tutorials, and health statistics and records (eg. mile time, max bench press). Links to the eBay store for fitness equipment will also be natively built. The above is a listing of the essential features required for base viability. Additional features may be incorporated in future development. Stretch goals will be evaluated as development progresses.

1.3 Definitions

Term	Definition
User	An individual who interacts with the mobile application
C7FIT	Our iOS app
Firebase	Google's Backend as a Service
Club Seven Fitness	Our client's client, a gym business based in Portland
HealthKit	iOS Health Framework
MapKit	iOS Map Framework
API	Interface to a piece of software exposed to the developer

1.4 Overview

The following document includes two main sections. The first section illustrates an overall description of the product. Additionally, this part will delve into user characteristics and constraints.

The second half of this document goes into the specific requirements of the products including external interface requirements, functional requirements, performance requirements, logical database requirements, and system attributes. Appendices at the end will provide structure for the results and release plans.

2 Overall Description

This section will provide an overview of the application and how it interacts with its related components. This section will also detail the general functionality and how the it will be used by customers. Finally it will discuss the constraints and assumptions in the design of the application.

2.1 Product Perspective

The product will consist of the mobile iOS application for Club Seven. The application will be used to interact with the Firebase API, the eBay store, and draw data from the native iOS frameworks: HealthKit and MapKit.

2.1.1 System Interfaces

The system will be contained and limited to iPhone, running iOS 10, as the product is purely an iOS application. The Firebase API will provide the users external account data. It will provide users the ability to create a profile and input relevant health data. The application will also provide location information using MapKit, so the user knows their relative distance for classes and trainers.

The application will also use the MapKit and HealthKit independently of the Firebase API. HealthKit will be used to track the users general fitness levels, such as: steps per day, sleep, calories burned, etc. MapKit will be used to chart the GPS location of the user during their exercise activities.

As a stretch goal, the eBay API will be included as a way to monetize the application. Users will be able to see and purchase relevant products.

2.1.2 User Interfaces

The application will consist of 5 main topical screens: Home, Schedule, Store, Activity, and Profile. Each screen will maintain a consistent layout shown in the wireframes drawn up in our second meeting. In general, the screens will have a navigation bar along the bottom of the screen divided up into the 5 topical screens. Certain screens that require information from the Firebase API will show different information depending on whether the user is signed in or not. More specifically, each screen will have table rows that either contain information or bring the user to other more specific parts of the app. Once the rows have been tapped, the user will be directed to a new screen and have the option to go back with a `<` button on the top bar.

2.1.3 Hardware Interfaces

An iPhone capable of running iOS 10, the iOS version were targeting, will be required to use this application. In addition, the phone will need to have the available memory space to download the app.

2.1.4 Software Interfaces

The Firebase API and eBay API will require developer accounts and accounts in order to function in our application. Their purpose has been discussed above: the Firebase API is key to the health data storage of the application, and the eBay API will be used for monetization. The application will be primarily in portrait mode.

2.1.5 Communication Interfaces

Wifi and cellular data will provide internet access to get the data from the Firebase API.

2.1.6 Memory Constraints

The specific memory constraints of the application will depend on the device running the app. We are currently targeting a multitude of devices, iPhones (4s, 5, 6, 7) so the specific constraints will vary depending on the RAM and space available to each of these devices. However, the application will need to store some data locally like the activity map and personal statistics.

2.2 Design Constraints

2.2.1 Operations

The user will be required to sign in to Firebase to access the full capabilities of this application.

2.2.2 Site Adaption Requirements

The User Interface of this application will be in English. We will not provide other languages (as of initial release). The application will fit and work on current iPhones running iOS 10: including iPhone 4S to 7 and sizes 4,5,6, and 6 plus.

2.3 Product Functions

This mobile application will be a personalized app that connects the functionality of the eBay API to the C7FIT gyms customers. The functionality again can be split up into the 5 main portions Home, Schedule, Store, Activity, and Profile, with Home having the most varied functionality. Home will be the starting point for the user and provide general overview on their daily options. This includes a daily workout video from YouTube, and a daily motivational quote. The schedule screen links to the online schedule and contact information for the gym. The Store screen will display shopping information for curated fitness products from the eBay API. The Activity screen will display the users current daily activity, and their PR in various fitness tests. Additionally, this screen will have stopwatches and maps to assist their workouts. Finally, the Profile screen will contain their profile and personal information.

2.4 User Characteristics

The target audience for the product are the members of C7FIT gym. These gym members will only be able to access the services that are provided by the gym.

2.5 Constraints, Assumptions, and Dependencies

This application depends on having a Firebase account for the majority of the data. Additionally, it requires the user have internet connection as most of the data is pulled through external APIs. The key aspects of the application is constrained by the limitation of this API.

Additional caching of the Firebase data will not be supported outside of the suite of tools Firebase already provides, and as such the application will not display information when the user does not have access to the API be it through internet or lack of account. The daily videos and motivational quotes come from Firebase and will cycle through a static set of quotes and video links.

We assume that the mobile device has the capability to run our application at an acceptable level. If the phone does not have enough resources, then the application may fail to run as intended or at all.

The application is dependent on the Firebase API, native HealthKit and MapKit, for most of its functions.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

C7FIT is a user-facing application, so naturally most of the design emphasis will be on the user interface and presenting information from external sources (such as the ebay API, GPS, external files etc) in a pleasing and easily digestible manner on the user interface. The main objective of the user interface is to help the user accomplish their most common tasks in as little time and effort as possible. Secondary goals are to represent Club Seven Fitness brand, while not being distracting, and to make tasteful use of animations to keep the user engaged, especially during operations that may require some time to complete such as API calls. (Designers often refer to these animations as microinteractions.)

Apples UIKit API provides minimally styled basic design elements, such as buttons, labels, sliders, text fields, and more. We intend to use these as the building blocks for our application wherever possible. In the MVP phase, unless a special element is absolutely necessary, we will just use the basic elements given to us by UIKit in their standard form.

Following completion of the MVP, we intend to add styling and animations to the most used elements, keeping minimalism, consistency, and reusability as a goal. Near the beginning of the project, we will design each screen using Sketch (a vector design program ideally suited for UIs). We will then implement this design down to the pixel in our app after the MVP stage.

The UI will have five tabs: Home, Store, Schedule, Activity, and Profile. The tabs will be displayed using UIKit's UITabBar, and each underlying view will be managed by a single UITabBarController. Each tab is explained below:

1. Home Tab
2. Home will be a tableview with three cells. Some of these will be tappable, allowing users to view more detail than the snippet shown in the cell:
3. Todays workout video
4. This will display a Youtube video
5. The video URL will be retrieved from a static file either in Firebase or a GitHub repo
6. The video will be different every day
7. Selecting the video will open it in a landscape, full-screen view
8. Todays Motivation
9. This is a simple cell that just contains a motivational phrase
10. The phrase will be fetched from a static file in a GitHub repo

11. Store
12. The Store view will be a tableview of items available for sale from the Club Seven eBay store
13. Each cell will contain a picture of the item, its price, and name. Other metadata from eBay may be included
14. The store will use the eBay API to list products sold by C7Fit
15. Tapping a cell will show more information about the item, with the option to purchase it on eBay
16. Schedule Tab
17. The schedule will be a WKWebView (basically an embedded web browser) that opens the schedule webpage from Club Seven Fitness's website
18. As a stretch goal, we may hook into a Google Calendar or parse the HTML to display a calendar using native elements
19. Activity Tab
20. The activity tab screen will be a collection of tools, user activity, and health results.
21. Today's Activity
22. The top section of the Activity Tab shall have a title Today Activity.
23. The content in the today activity section will include steps and active time from the Health Kit SDK.
24. If the user does not have a profile on Health Kit SDK then a message will be displayed to the user . To get activity go to health app on your phone
25. Workout Tools
26. The section session shall have a title Workout Tools
27. There will be three tools in this section.
28. The first tool will be stop watch.
29. The stopwatch cell shall have text saying Stop Watch and shall have a right caret.
30. If the user taps the stopwatch cell a new screen shall be started.
31. The stopwatch screen shall have a title Stop Watch and a left caret.
32. If the left caret is selected the user will be returned to the activity tab.
33. The stopwatch screen shall have a timer.
34. The stopwatch screen shall have a Start button that starts the timer.
35. The stopwatch screen shall have a Stop button that stops the timer.
36. The time shall remain on the screen until the user taps the start button again.
37. The second tool will be countdown time.
38. The countdown cell shall have text saying Countdown Timer and shall have a right caret.

39. If the user taps the countdown cell a new screen shall be started.
40. The countdown screen shall have a title Countdown Timer and a left caret.
41. If the left caret is selected the user will be returned to the activity tab.
42. The countdown screen shall have a timer.
43. The countdown screen shall have an option to select a timer duration.
44. The countdown screen shall have a Start button that starts the timer.
45. The countdown screen shall have a Stop button that stops the timer.
46. The time shall remain on the screen until the user taps the start button again.
47. Map
48. The map screen shall have a title saying Map and a right caret.
49. The map shall use the Mapkit SDK.
50. The map shall show map with current location.
51. The map shall a Start Activity - have a start activity option.
52. The map shall a Track Activity - track activity to record the track and draw it on the map.
53. The map shall a Finish Activity - stop the activity.
54. The map shall a Activity Details - Time and Distance for activity.
55. The map shall a Save Activity - Save locally on device in core data with time as title.
56. The map shall a View an Activity - Stretch goal to remap an activity.
57. Test Results
58. The test results section shall list all current recorded results.
59. The test results section will have an edit option.
60. The edit option will launch a new screen that list the test results and allows the user to enter the correct data.
61. Test results will include: Mile Time, Number of Push Up in a Minute, Number of Situps in a Minute, Leg Press, Bench Press, Lat Pull.
62. Profile Tab
63. The more tab shall display the signed in users profile.
64. The elements that will be displayed include picture, name, and info from the HealthKit service.

3.1.2 Hardware Interfaces

C7FIT is an iOS 10 application, meaning it will run on any Apple iOS device capable of running iOS 10, or the next approximately 5 iOS operating systems produced in the future. We will ensure to not use any deprecated APIs, and since Apple usually does not deprecate APIs without several years advance notice, we can expect C7FIT to be compatible with future Apple devices for at least the next three years.

The app will be specifically designed with the iPhone 5s and up in mind, but will be compatible with all iPhones down to the iPhone 4s. We intend to use an adaptive design to stretch to make maximum use of the variety of displays on these devices.

3.1.3 Software Interfaces

The primary software the app will interface with is the iOS operating system, as well as the UI framework provided by iOS called UIKit, and a large collection of helper libraries contained in Apples Foundation framework. These libraries provide for everything from app instantiation to making network calls, to saving to disk, to drawing UI elements such as buttons and sliders.

We may also elect to bring in a few external 3rd party libraries to use instead of the Foundation / UIKit framework in areas that Apples API is known to be a little rough. We intend to minimize the use of third party frameworks / libraries as little as possible however. We will only use the leader in that field to reduce the likelihood of it being deprecated, and will wrap it in our own API to enable easy swapping of underlying APIs and prevent building dependencies in our code on potentially fickle open-source projects.

3.1.4 Communication Interfaces

C7FIT will make extensive use of Internet REST APIs, especially the Firebase API to provide functionality to the user. It will also make use of the eBay API if we meet our stretch goals to allow users to buy gym equipment online. Finally, the app will make use of the iPhones GPS unit to communicate with satellites and determine the users precise location during a workout. Luckily, this functionality is largely abstracted by the iOS Location API, and just involves a few lines of code.

The most challenging external communications will be those with the eBay API, as we will have to build a communication library with them based on their JSON REST APIs.

3.2 Classes and Objects/Functional Requirements

The main objective behind our program design is to reduce state to as few places as necessary. We plan on implementing our architecture with the principles of functional reactive programming in mind, so that developers can focus on building functionality of the app instead of worrying about bugs in their data flow.

3.2.1 DataStore

We will use the single-source-of-truth principle, along with dependency injection to reduce confusion about where data is coming from. All of the applications state will be kept in a central DataStore.

This will have fields containing data structures capable of holding all the data needed for all the apps views, organized hierarchically roughly around the purpose / source of the data. The DataStore will be initialized upon app instantiation and a reference to it will be dependency-injected into each View Controller. This way, every View Controller will share the same source of truth, ensuring consistency between views. This will also ease caching and local storage.

Information will be requested from the DataStore using functions exposed by the stores interface. Upon fetching the data through whatever means necessary (network call or GPS call, retrieval from disk, etc) the view controller will be notified with the requested data, allowing it to update its UI. The View Controller will not perform any modifications to the data, besides what is necessary to display it nicely. No methods that modify the display of the information should require asynchronous operations. The method for notifying the ViewController of state changes will be discussed in the next section.

3.2.2 Responding to State Changes

There are several ways of responding to state changes, from callbacks to promises. We intend to use a functional reactive approach, in which the state of views is never explicitly changed, only described. Apple does not provide a functional reactive library, but there are two leading 3rd party libraries widely in use today: RxSwift and ReactiveCocoa. Both are similar, but ReactiveCocoa has deeper integration with Apples UIKit and Cocoa Touch frameworks. Receiving data from the API, the user changing screens, inputting text, pushing buttons, etc are state changes the app will respond to.

3.3 Performance Requirements

Modern iOS devices are relatively overpowered for this sort of application, so computational performance shouldnt be a worry. A much bigger priority than performant code should be readable code.

Places we should focus on performance are asynchronous calls, especially network calls. After the MVP, we should put priority on caching and prefetching data. If no cached data exists, we should display a useful animation that indicates progress.

3.4 Logical Database Requirements

We will be using Firebase for all database needs. Apple does provide a few set of APIs for caching on-device, but our source of truth will be coming from the Firebase API.

3.5 Software System Attributes

3.5.1 Reliability

We will implement measures to ensure that the app functions well in all edge situations. Our programming methodologies described in the Classes and Objects section will ensure that developers are unable to compile the application without first accounting for all possible data states, including lack of GPS hardware, lack of network capability, disk failure, and more. Developers will be able to display helpful, user readable messages in all of these situations.

3.5.2 Availability

Most of the apps features will require internet access. However we dont intend to restrict access to the applications other features that dont require connectivity in the event that a network connection is lost. Features such as using the GPS or timer workout tracker, should still be functional. Additionally, the UI should display informative, but not disruptive information in disabled interface elements.

While we could cache network requests to the Firebase API while the device is offline, we will opt to just make the API attempt fail quickly and show an error to the user. Weve found that this helps prevent users from making unintentional actions to their account.

3.5.3 Security

The C7FIT app will contain some potentially sensitive health information, but not to the point where most users would want to add additional security measures such as TouchID or a passcode (common security measures on iOS).

Purchasing from eBay will be done in the eBay app/mobile website, which will handle all payment information and verification. C7FIT will just deep-link to the eBay app (or website if the app is not installed on the users device)

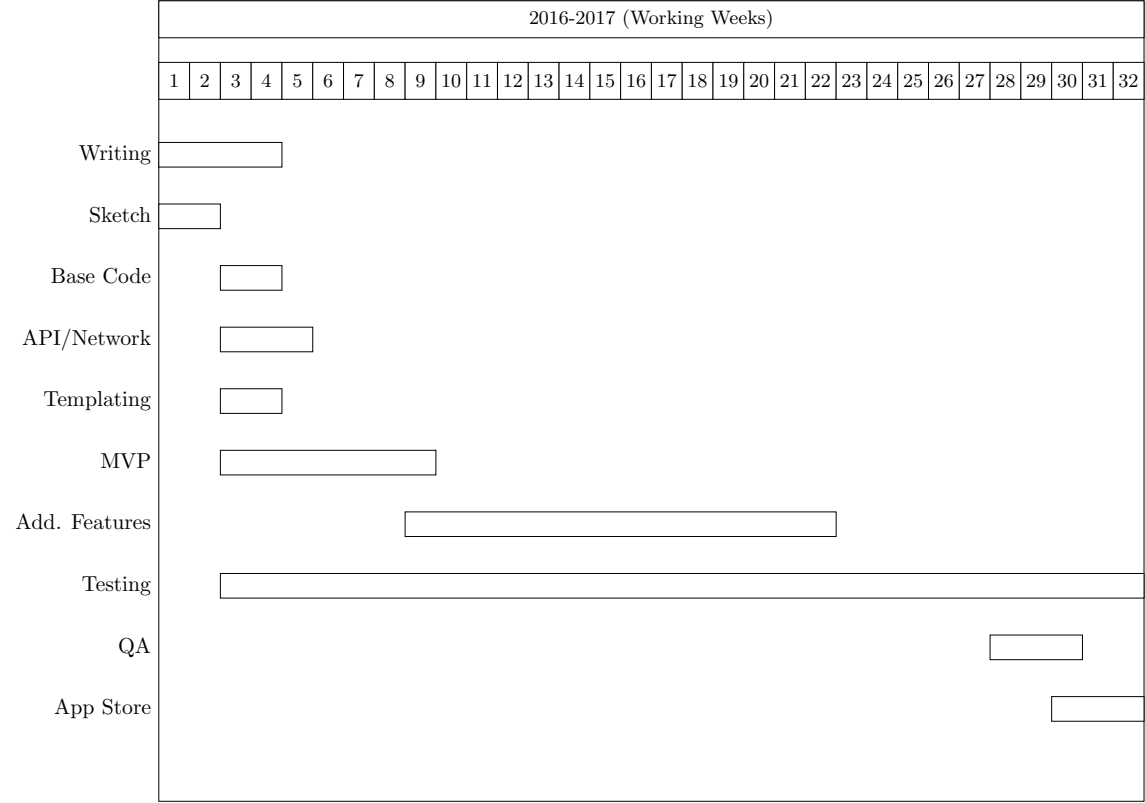
3.5.4 Maintainability

As the application will be used by Club Seven gym goers long after we complete development, there will need to be some form of maintenance in order to ensure C7FIT retains its quality and user base. In order to attain this, the app will be handed off to the eBay mobile team in Portland after publishing on the App Store. A small team of developers, whom we are currently working with will ensure the maintenance. Additionally, eBay Inc. will have the documentation we write this term for reference in the situation the app requires to be maintained long term.

3.5.5 Portability

The app is an iOS iPhone application. Because of the architecture of Apples development ecosystem, it should be relatively easy to port C7FIT to other Apple devices such as iPad. However for an Android application or any non-iOS device, there would be a substantial amount of more work to do before being able to publish. In terms of translation portability, we can use NSLocalizedString to load localized strings from a centralized source of truth upon our client's request.

3.6 Gantt Chart



4 Signed Participants

Students

Brandon Lee

Rutger Farry

Michael Lee

Client