LIFTING LOG FINAL REPORT

Joe Melito & Neil Kalanish

8 December 2022

1. Introduction

1.1. Purpose and Scope

Today many people heavily rely on their mobile devices to hold and store their most personal data. Whether it is their photos, credit cards, daily journal, etc. People are constantly using their phones to do the most basic tasks. However, there are still many people within the weight lifting community who opt out of using their phones to track their progress and would much rather just use pen and paper. The reasoning behind this lies behind the most common complaint found this community. Workout applications are just bad. They add way too many unnecessary features, have clunky UI/UX designs, and are too time consuming for users when they are at the gym. Our objective here is to design a mobile iOS application that would allow our users to login, track their current workout session, view their previous sessions, and add any additional comments or notes. Our application will separate us from our competition because we are focusing only on the weight lifting community.

Our software will allow users to input their workout sessions data. Which will then be stored in a database. Users can then view their previous workout sessions within our app. They can also rely on our application to handling any calculations for their workouts such as Wilks and 1RM.

The Following Test Cases Will Be Within Scope:

- Account Creation & Deletion
- Login Credentials
- Navigation Between All Views Within Our UI
- Accurately Displaying Requested Data (Viewing Previous Workouts)
- Accurate Submission of User Data to Our Database.

- Accurate Calculations (1RM & Wilks Calculators)
- Users Ability To Enter In The Expected Data (User Enters in String When Integer is Expected)

The Following Test Cases Will Not Be Within Scope:

- Unit Testing of Data Transactions & Calculations
- User Gets Phone Call or Text During Usage (I.E. User Leaves Our Application Mid Flow)
- User's Phone Battery Dies
- User's Phone Crashes Due to OS Related Issues
- User Upgrades or Downgrades OS on Device
- User is using an Unreleased or Beta Version of OS
- User Loses Internet Connectivity

1.2. Product Overview (including capabilities, scenarios for using the product, etc.)

See Subsections Below For Further Details.

1.2.1 Project Goals & Objectives - User

- To allow our users to signup and create and account using firebase authentication.

- Seem-less UI design that eliminates unnecessary bloat that other workout apps have. We want our users to be able to quickly and easily track and record their weight lifting workout sessions.

- Using Google Firebase to security store all our user's private information.

- Allow cross platform availably with other Apple Products. Such as, MacBooks, iPads, and the Apple Watch.

1.2.2 Project Goals & Objectives - Technical

We want to keep our frontend interface separate from our backend. This will be accomplished by creating micro services that handle all of our backend data. This will be accomplished by the following:

- Store user created data within our database. This includes user account information (first & last name, date of birth, gender, username, email, & password) and created workouts data (workout type, sets & reps, weight lifted, & any additional comments or notes the user adds)

- Query our database for any user requested data.

- Received a JSON output back from our database. Which we will then deserialize.

- Format the data within our created micro services.

- Display the data within our frontend user interface.

With keeping our backend and frontend separate, it allows us too quickly and sufficiently make any necessary changes and / or future enhancements. An example of this would be switching from Google Fireback to Apple CloudKit.

1.2.1 Project Context

Our project was designed to allow us to grasp a better understanding of Apple's development ecosystem. We wanted to learn about Swift, Xcode, and all other aspects of iOS development while creating a full fledge application. We also wanted to obtain a better understanding of Google's Firebase Database engine.

Our application's context is within the health and fitness category. We wanted to solely focus on weight lifters who are interested in tracking, planning, and viewing their weightlifting workout sessions. We aimed to help eliminate the dependency weight lifters have to recording workouts on pen and paper.

1.3. Structure of the Document

As you continue on with the rest of our applications documentation you will find the following categories:

- Project Management Plan
- Requirements
- Architecture
- Design
- Test Management

- Defects / Known Bugs

1.4. Terms, Acronyms, and Abbreviations

- Apple's Ecosystem: Apple's proprietary software & hardware. This includes all professional and consumer based software and hardware.
- Swift: Apple's programming language used to develop apps for their hardware.
- Xcode: Apple's IDE for Swift development.
- iOS, MacOS, iPadOS, WatchOS: Apple's software that runs on their various products.
- Google Firebase: Google's backend database engine.
- View: UI layout and/or window that is used within Swift development.
- Structures: Programming objects within Swift.
- Services: A set of software functionalities found within our softwares architecture.
- Extensions: Custom add on to an already existing class built into Swift.
- One Repetition (1RM): The maximum amount of weight a weightlifter can lift in one repetition.
- Wilks: Used to measure relative strengths of powerlifters within different weight classes.

2. Project Management Plan

Our project management plan followed the Waterfall mythology. We felt that it best fit our schedules as we are both working full time in the software engineering profession. We also felt that with a 16 week course Agile would be harder to implement. Please see further details below in the project management subsections.

2.1. Project Organization

As mentioned above, our project followed the waterfall mythology. With a team of only two for developers we split up the work by having one developer work full time on front end development while the other focused strictly on the backend. We incorporated a Kanban board to help keep track of our tasks and met regularly to make sure we were on schedule.

2.2. Lifecycle Model Used

The lifecycle model used for our project was the standard software development lifecycle within the waterfall mythology. This following stages we followed are:

- Requirement Analysis
- Defining Requirements
- Software Design
- Implementing Code
- Testing
- Deployment (Dev & QA Environments)
- Maintenance (Future Enhancements Planned)

2.3. Risk Analysis

The only risk involved with this project was that our team had to quickly learn how to use Apple's development tools, Swift & Xcode. Alongside learning Google's backend database engine, Firebase. All other forms of risk are negated as this project is not planned for a production release.

2.4. Hardware and Software Resource Requirements

Since this application was development using Apple's software development tools, our application can only run on Apple devices such as the iPhone or iPad and these devices must be running the latest version of iOS. As of this publication iOS is on version 16.1.

2.5. Deliverables and schedule

Our deliverables and schedule are found below. As mentioned, one developer focused on the front end while the other was on the backend. (Note: The few requirements that contain both front and backend development were coordinated between both developers.)

Project Timeline / Schedule

Requirement	Department	Specification	Due Date
Login UI	Front End	Design Login UI	10/01/2022
Create User UI	Front End	Design Account Creation UI	10/01/2022
Login Authentication	Backend	Allow Users To Authentication Login Using Google Firebase Services.	10/15/2022
New User Authentication	Backend	Allow Users To Create a New Account Using Google Firebase Authentication Services.	10/20/2022

Main Dashboard UI	Front End	Design Main Dashboard UI	10/01/2022
Main Dashboard Functionality	Front End	Allow Users to Navigate the Application Using the Main Dashboard.	10/01/2022
Start Workout UI	Front End	Design Start Workout UI	10/01/2022
Start Workout Submission	Backend	Allow Users to Submit Inputted Workout Data.	11/15/2022
View Previous Workouts	Backend / Front End	Allow Users to View Previously Submitted Workout Data.	11/22/2022
1RM Calculator	Backend / Front End	Design 1RM Calculator With Functionality Based On User's Input	10/31/2022
Wilks Calculator	Backend / Front End	Design Wilks Calculator With Functionality Based On User's Input	10/31/2022
Settings	Backend / Front End	Allow for Users to Logout or Delete Account	10/31/2022

3. Requirement Specifications

3.1. Stakeholders for the system

Our only current stakeholders are the two developers working on the project, Joe Melito & Neil Kalanish. However, as our application grows we aim to bring on newer stake holders and cater solely to the health & fitness community.

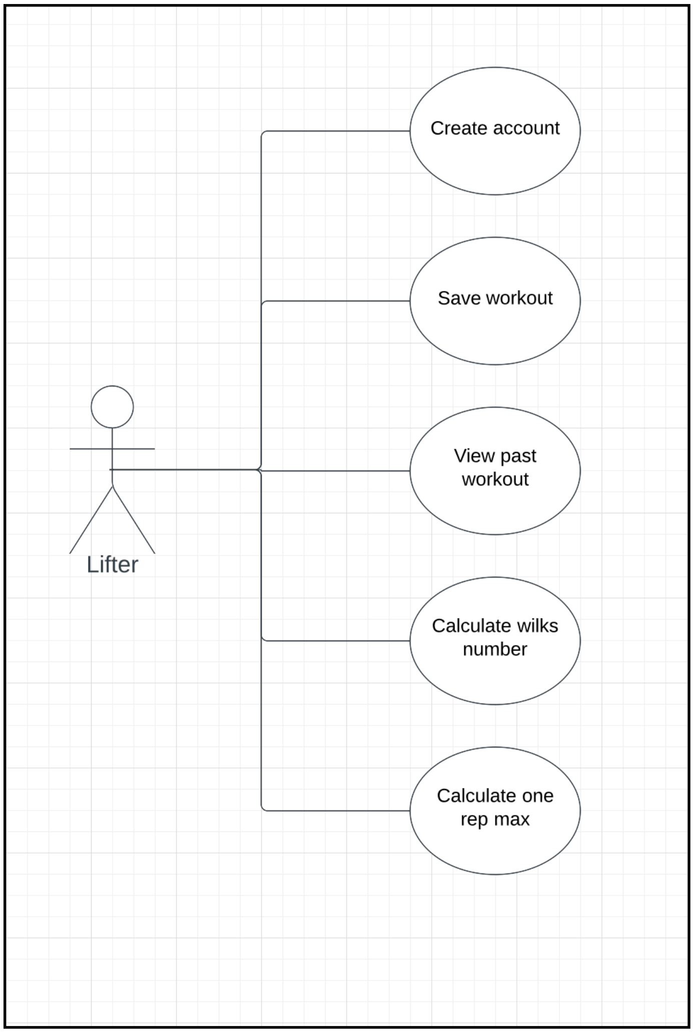
3.2. Use cases

Our application has 5 different use cases

- Create Account

- Save Workout
- View Past Workouts
- Calculate Wilks Number
- Calculate One Rep Max

3.2.1. Graphic use case model



3.2.2. Textual Description for each use case

Use case: Create Account	
Actors	Lifter (User)
Description	Allow a new user to create an account to start using the app
Data	Email address and password
Stimulus	User fills out information
Response	Either confirmation of account creation or error that user account couldn't be created
Comments	A brand new user of the app can create and app to track all of their workouts and app use

Use case: Save workout	
Actors	Lifter (User)
Description	Allow a user to save their workout
Data	The users workout information, workout, weight, reps, sets and any comments about the workout
Stimulus	User fills out information

Response	The user will get a notification that the information was saved or an error that it wasn’t
Comments	This allows the user to log all of the information about their workout. The comments section isn’t required

Use case: View past workouts	
Actors	Lifter (User)
Description	Allow a user to view past workout
Data	No data is required to view the past workouts but there must be information in the database to display
Stimulus	User selects past workout
Response	A view of past workouts
Comments	This will allow the user to see all of their past workouts

Use case: Calculate Wilks Number	
Actors	Lifter (User)
Description	This will allow the user to calculate their wilks number
Data	Users bodyweight, gender, max bench, max squat and max deadlift, all in pounds.
Stimulus	User inputting data
Response	The results of the calculations
Comments	This will give the user their wilks number

Use case: Calculate One Rep Max	
Actors	Lifter (User)
Description	This will allow the user to calculate a one rep max
Data	Weight lifted and how many reps
Stimulus	User inputting data
Response	The results of the calculations
Comments	This will give an estimate of what a users one rep max might be based on what they have lifted.

3.3. Rationale for your use case model

Besides the creating an account and calculators we wanted to mimic what a lifter would get with pen and paper. A straight forward user experience where it is easy to write down, save, your workout and quickly look back and view past workouts. The calculators a two added small features that were quick and easy to implement and helpful for lifters to know.

3.4. Non-functional requirements

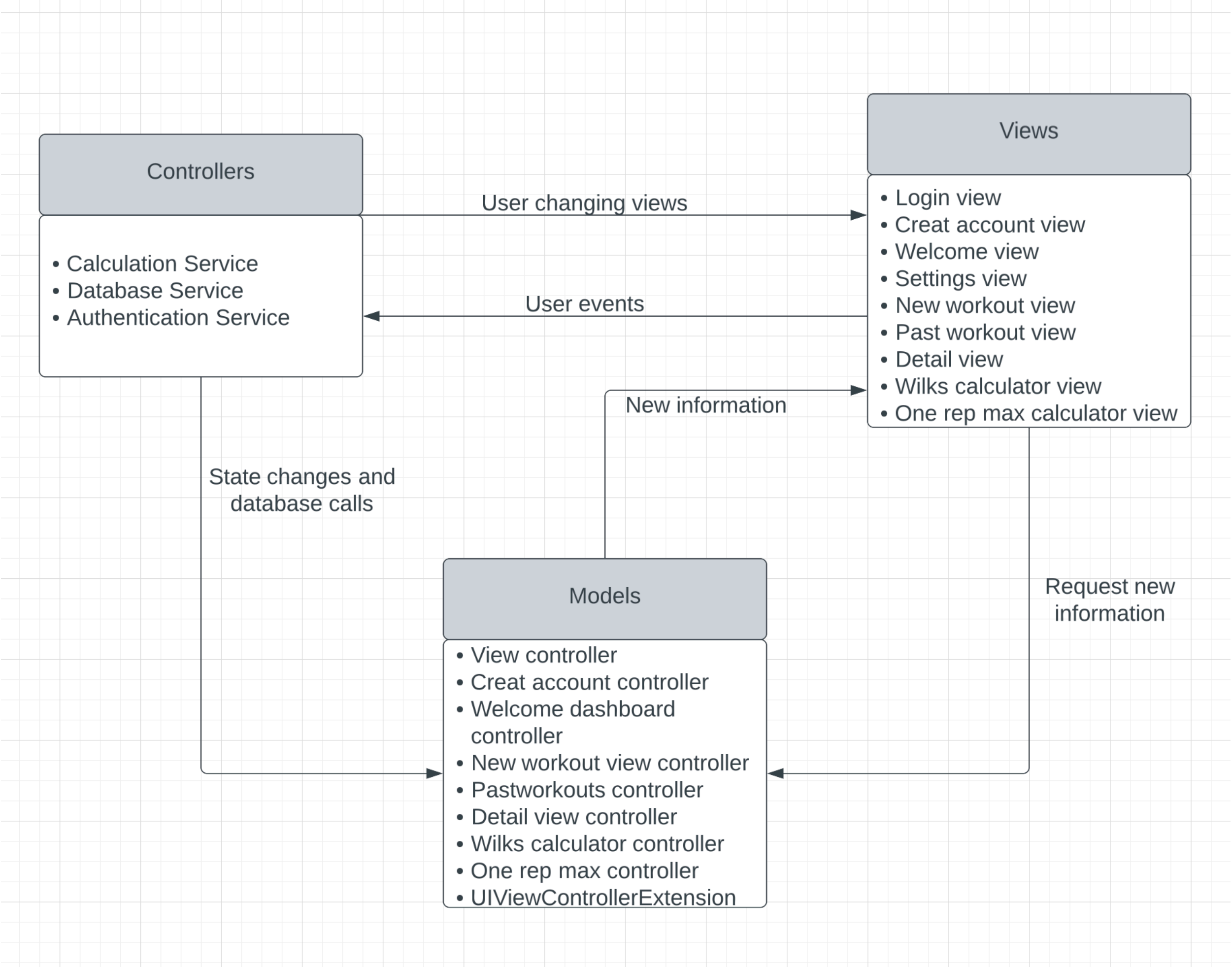
The only non-functional requirement we have is that the application can run on iPhone, iPad and MacBook. This is a requirement because we want the user to be able to view their workouts from any where they might be. To achieve this we are creating the app using Apple's swift language so it will scale and change based on the device with minimal configuration from us.

4. Architecture

4.1. Architectural style(s) used

From what we learned in class we are using a traditional MVC (model view controller) architecture. There are different views that the user see's and each view has a model or controller that listens to changes and services or controllers that do work when stuff changes.

4.2. Architectural model (includes components and their interactions)



4.3. Technology, software, and hardware used

- Language: Swift version 5.7
- IDE: Xcode
- Source Control: Github (https://github.com/Neilkal867/LiftingLog)
- Database: Firebase Firestore Database
- Authentication: Firebase Authentication
- Hardware: Macbook air

4.4. Rationale for your architectural style and model

From reading online this was one of the most common implementation methods for simple mobile applications that required calls to a database. We have services to make all of the calls to the DB and do calculations, we have views that display the information and when a user interacts with the view a controller handles all of the events.

5. Design

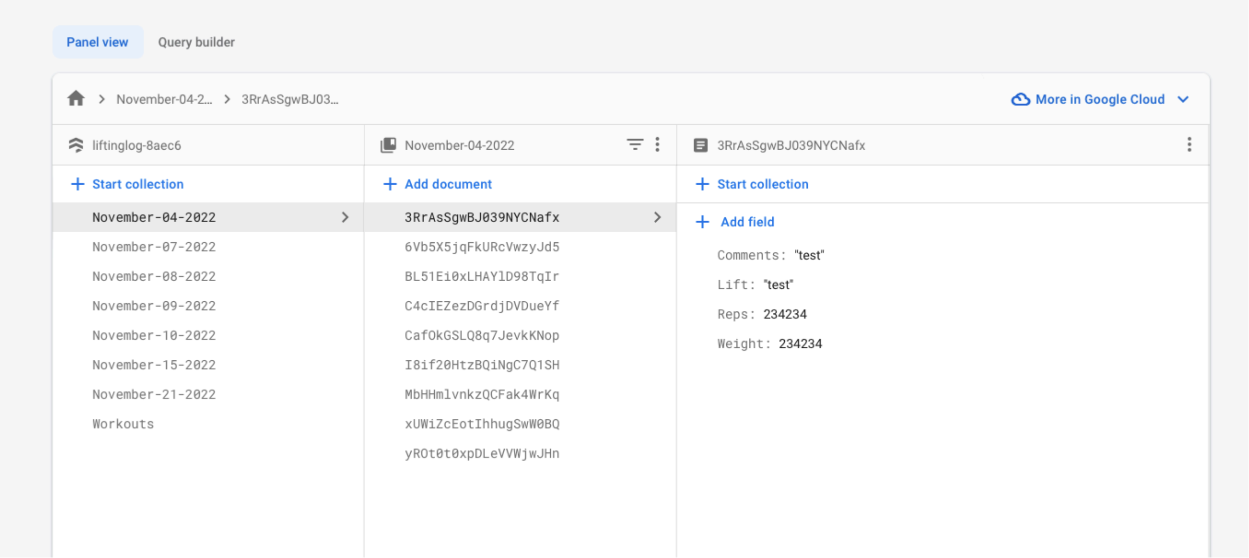
5.1. User Interface design

We offer a basic and straight forward user interface. Our application isn't geared towards having a lot of functionality it is designed to be simple and straight forward. We are trying to mimic what a lifter would use pen and paper for so that is why we keep it simple. As of right now it only has nine different screen and but the bulk of the apps functionality lays within three of those.

5.2. Components design (static and dynamic models of each component)

All of the components except a few are based off of static models. Due to the simplicity of the application the login, create account, settings and create workout screens are all static displays and won't change. The view workout components are the dynamic ones that will change based on the users input.

5.3. Database design



liftinglog-8aec6

November-04-2022
November-07-2022
November-08-2022
November-09-2022
November-10-2022
November-15-2022
November-21-2022
Workouts

November-04-2022

3RrAsSgwBJ039NYCNafx
6Vb5X5jqFkURcVwzyJd5
BL51Ei0xLHAYlD98TqIr
C4cIEZezDGrdjDVDueYf
CafOkGSLQ8q7JevkKNop
I8if20HtzBQiNgC7Q1SH
MbHHmlvnkzQCFak4WrKq
xUWiZcEotIhhugSwW0BQ
yROt0t0xpDLeVVWjwJHn

3RrAsSgwBJ039NYCNafx

Comments: "test"
Lift: "test"
Reps: 234234
Weight: 234234

The database we are using is a Firebase Firestore Database. Firestore is a NoSql DB that is created and supported by Google. The Firestore Database is setup that "collections" contain the date of the workout. Each collection has a "document" and the documents contain each workout in a day and inside each "document" contains fields which are the works comments, lift, reps and weight. Firebase doesn't offer a way for you to get all of the collections back so we put all of the dates in a collection then pull back all the documents from that collection which mirrors all the collections. Then when a user selects a collection we can retrieve all of the documents and fields to display the workouts to the users.

5.4. Rationale for your detailed design models

This was our first mobile application that either of us made so we are following best practices and standards we found online.

5.5. Traceability from requirements to detailed design models

We don't have any documented traceability for the design model. As we were writing the application we would research what we needed for that individual use case and implement that.

6. Test Management

6.1. A complete list of system test cases

ID	TC1
Test Task	User Login: No Account
Test Input	Username: TEST1234@Test.com Password: TEST123!
Expected Output	User will be prompted with a dialog box stating that their account cannot be found.
Description	User enters in 'account information' without registering an account. User should be notified that signup is required.

ID	TC2
Test Task	User Login: All Inputs Are Responsive
Test Input	Tester may enter any input values.

Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC3
Test Task	User Login: Wrong Username & Password
Test Input	Username: dev Password: TEST123!
Expected Output	User will be prompted with a dialog box stating that their login credentials are incorrect.
Description	User enters wrong ‘account information’ User should be notified that their password and / or username is incorrect. User must be asked to try again.

ID	TC4
Test Task	Create Account: Account Creation Successful
Test Input	Tester may enter any values for account creation.
Expected Output	Account should be created and allow the user to login.
Description	Users must be able to create an account / sign up for our application.

ID	TC5
Test Task	Create Account: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC6
Test Task	Create Account: Account Creation Cancellation
Test Input	Tester may enter any input values.
Expected Output	When a user cancels account creation; no account is created and they are redirected to the login screen.
Description	Users must be able to cancel account creation if they so wish.

ID	TC7
Test Task	Main Dash: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC8
Test Task	1RM Calculator: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC9
Test Task	1RM Calculator: Output Calculated Correctly
Test Input	Tester may enter any input values.
Expected Output	An integer output value.
Description	Users must be able to enter in the corresponding data for 1RM and receive and accurate output value.

ID	TC13
Test Task	Create Workout: Data Properly Saved For Future Use
Test Input	Tester may enter any input values.
Expected Output	Inputted values will be saved for future use
Description	Users must be able to create a workout, input data, save data, and come back to it at a later time for modification or submission.

ID	TC14
Test Task	New Workout: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC15
Test Task	New Workout: Data Submission To Database
Test Input	Tester may enter any input values.
Expected Output	All inputted data must be shown within our database
Description	Users must be able to input and submit workout data to our database.

ID	TC16
Test Task	Past Workout: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC17
Test Task	Past Workout: Data Retrieved is Accurate
Test Input	Tester may select a previous workout

Expected Output	Data retrieved must match user’s previous workout data submissions.
Description	Users must be able to view previous submitted workout data.

ID	TC18
Test Task	Settings: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	TC19
Test Task	Settings: Logout Feature
Test Input	Tester may select the ‘Logout’ button.
Expected Output	User will be logged out of application
Description	Users must be able to securely log out of our application.

ID	TC20
Test Task	Settings: Account Deletion Feature
Test Input	Tester may select the ‘Delete Account’ button.
Expected Output	User will be prompted with a dialog box warning them; if they hit ‘cancel’ they remain logged in; if they hit ‘ok’ their account information will be removed from the database
Description	Users must be able to delete their accounts when they no longer wish to use lifting log.

ID	Wilks Calculator: All Inputs Are Responsive
Test Input	Tester may enter any input values.
Expected Output	All input fields should be responsive.
Description	Users must be able to interact with this part of the application.

ID	Wilks Calculator: Output Calculated Correctly
Test Input	Tester may enter any input values.
Expected Output	An integer output value.
Description	Users must be able to enter in the corresponding data for Wilks and receive and accurate output value.

6.2. Traceability of test cases to use cases

Test Case Traceability Matrix

Associated Test Case IDs	Requirement	Department	Specification	Design	Testing
TC1 & TC3	User Login	Backend	The user login screen will allow our user to enter in their information (username & password) or create a new account.	The page will have two textfields; for username and password. It will also contain two buttons; a login button which will take the user to the main dashboard and an account creation button which will take the user to the account creation page.	- Verify that the user cannot login with the wrong. credentials - Verify that the user cannot login without filling in both textfields. (username & password) - Verify the user is able to transition to the account creation page.
TC5 & TC6	User Account Creation	Front End	The user creation screen will take in the users Name, Username and password for account creation. If the username already exists, prompt the user to pick a new one	The page will have three input text boxes to take in the user information. There will be a submit button to create the account and a cancel button to go back.	-Verify each input box can be typed in -Verify you cannot click submit with an empty input box

Associated Test Case IDs	**Requirement**	**Department**	**Specification**	**Design**	**Testing**
TC7	User Main Dashboard	Front End	The main dashboard will contain all of the options for the app. The user will be able to start a workout, create a workout, load a workout and access the settings or calculators	The design of the main screen will be a list of buttons labeled with each function	-Verify each button takes you to the appropriate screen for example, the 'Start workout" button takes you to the start workout screen
	User Settings	Front End	The settings page will contain various fields which the user will be able to interact with to change dynamics within the application.	The design is still being work on as our settings requirements are not fully spec-ed out.	- Verify that the settings toggle appropriately based on user selection.
	Workout Creation	Front End	The user workout creation page will contain various textfields which will allow the user to enter in corresponding data pertaining to their workout. It will also contain two buttons which will allow the user to submit or cancel their data submission.	The workout creation page will contain 5 textfields which will allow the user to input the following: - Workout Type - Amount of Sets - Amount of Reps - Weight Lifted - Custom Notes This UI will also contain two buttons which will submit the data the user inputted and another button to cancel the workout creation.	- Verify the ability to allow user input. - Verify that users can cancel current workout creation.

Associated Test Case IDs	**Requirement**	**Department**	**Specification**	**Design**	**Testing**
	Previous Workouts	Front End	The previous workout page will allow the user to view requested data from previous workouts. The data will be presented to the user within non interactive textfields. The user will also be able to navigate out of the previous workout to either the main dashboard or to select another previous workout to view.	This page will contain various output text fields that reflect user requested data. These fields will be the same as the workout creation fields. We will also give the user the ability to cancel or view another workout through two different button options	- Verify corrected requested data is being outputted to the user. - Verify that the user cannot make changes to previous submitted data.
	1RM Calculator	Front End	This page will allow the user to enter in the necessary data to calculate 1RM. Users should also be allow to cancel their calculation.	This page will contain two input fields: Weight and Amount of Reps. It will also have a calculation and cancel button. The output will display in a non interactive textfield	- Verify the output calculation is correct. - Verify user cannot change output textfield

Associated Test Case IDs	**Requirement**	**Department**	**Specification**	**Design**	**Testing**
	Wilks Calculator	Front End	This page will allow the user to enter in the necessary data to calculate 1RM. Users should also be allow to cancel their calculation.	This page will contain five input fields: Gender, User's Weight, Max Deadlift, Max Bench, and Max Squat. It will also have a calculation and cancel button. The output will display in a non interactive textfield	- Verify the output calculation is correct. - Verify user cannot change output text field
TC2 & TC4	Authentication Service	Backend	This is a service that will handle all calls to the firebase authentication service. It will get the users name, username and password then pass them to firebase for authentication. If the username and password are valid it let the UI know to go to the main screen and if the information isn't valid it will alert the user of an error.	The authentication service will be a class that contains all of the variables and functions required to communicate with the Firebase Authentication service	- Verify that the service can create an account with a unique username and password - Verify that the service returns "True" with a valid user name and password - Verify that the service returns "False" with an invalid user name and password

Associated Test Case IDs	**Requirement**	**Department**	**Specification**	**Design**	**Testing**
	Calculator Services	Backend	This service will perform all calculations that the app will offer. The initial app will have calculators to calculate one rep maxes and a users wilks number.	The calculation service will be a class that contains all of the variables and functions required to perform the one rep max and wilks number calculations	- Verify that the calculation functions return the correct values
	Database Services	Backend	We want one class that will handle all database calls. This class will save workouts, load workouts and transform data to something useable by the front end.	The database service will be a class that contains all of the variables and functions required to perform all CRUD operations with the firebase DB. When calling into the DB firebase will return a json string so this service needs to mold the information from a json format to a useable struct for the frontend	- Verify the service can retrieve information from the DB - Verify the service can transform json to a usable struct - Verify the service can save information in the DB - Verify that the service throws an error anytime it cannot communicate with the DB

6.3. Techniques used for test case generation

Due to our limited amount of time to write this application we primarily used manual testing to verify the success of all our test cases. As a future enhancement we are looking to incorporate unit tests to help streamline and add an extra layer of validation to our application.

6.4. Test results and assessments (how good are your test cases? How good is your software?)

All our current test cases are associated with user stories that were expected to be deliverables by the end of our semester. We have completed all user stories with successful test results. Therefore, our test cases are complete and successful while our software has met all current deliverables.

6.5. Defects reports

7. Conclusions

Joe and I were successful in creating a mobile application that can run on iPad or iPhone and track a users workout. We set out to create an effective yet simple way to replace lifters who still track workouts with pen and paper because they don't want an app that tracks a million different things or is loaded with ads and unnecessary features. With a few improvements on how we save and load workouts, we could probably publish the app to the app store but neither of us want to pay $100 a year to maintain it on there.

Throughout this project we both agree that we learned a lot about coding in swift, mobile app development and working with a database. Neither of us have worked with any of this stuff so it was informative. The content went along well with what we learned in class on picking an architecture style to coming up with use cases and presenting our ideas.

7.1. Outcomes of the project (are all goals achieved?)

Overall the project was successful for a college project. Neither of us have worked with any of this technology before (Swift, firebase or Xcode) so the fact that we were able to meet our original design idea is impressive. There isn't much in terms of design but overall the functionality that we originally setup to create is there.

7.2. Future development

There is a lot for future development. There were some new concepts we learned late in the project so for the app to grow there would need to be some heavy refactoring done. The big thing we learned is how to handle escaping and completing closures in Swift. To grow the app a lot of the database calls we need to be converted over to escaping closures.

Another big upgrade would be the UI, right now it is very basic. It would be nice to add some design and color to it instead of all of the default buttons and views. The app icon is also just a picture of Ronnie Coleman so it would be nice to swap that out for a nice custom logo.

Lastly one of the improvements we could make would be adding support for more logging in options. Currently it is only email but the Authentication service has support for Facebook, instagram, Apple and gmail. We would also need a way for the use to reset the password or change their password