PUMP PATH FINAL REPORT

Joe Melito & Neil Kalanish

Spring 2024

1. Introduction

1.1. Purpose and Scope

Previously, our team started working on the basic functionality of Pump Path during our semester in EEC 521. The premise of the application was to provide everyday fitness enthusiasts and power lifters, with a simple straight forward app to track their workouts. Currently, the apps on the market are packed full of extra features that the average fitness enthusiast would not use. Our app's premise was to provide very basic functionality such as: the ability to login and save basic workout information with a minimal UI. At the end of the Fall 2022 semester we demoed our application to friends, family, and power lifters and to our surprise we were met with praise and excitement. We also received feedback on features that our users would love to see added or improved. For this class we intend on implementing those features with the hopes of publishing our app to the Apple App Store.

Our software will allow users to create a personal profile, input their workout sessions, view their previous workouts, calculate 1RM & Wilks, and create 531 workout plans.

The Following Test Cases Will Be Within Scope:

- Account / Profile Creation & Deletion
- Login Credentials (Firebase or Apple ID Auth)
- 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, & 531 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 or their Apple IDs.

- Seem-less Modern 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 Apple CloudKit's security store all our user's private information.
- Allow cross platform availably with other Apple Products.

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.
- 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 our backend transition from Google Fireback to Apple CloudKit. Our original design allows this to happen seamlessly.

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, CloudKit, 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.
- Apple CloudKit: Apple's backend database engine.
- Google Firebase: Google's backend database engine.
- TestFlight: Apple's beta testing program.
- 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.
- 531 Workout Plan: A strength training program that focuses on increased weight loads of lifts such as, squats, bench press, deadlifts, & presses, over a four week period.
- 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 also have bi-weekly meetings with our project sponsor to make sure we were on track for deadlines and coding to the correct user specifications. We incorporated a Kanban board to help keep track of all our tasks.

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

Our risks for our project include a total rewrite of code base. This was due to use switching away from Google Firebase as our backend and migrating over to Apple's CloudKit. During this transition we needed to verify that all our code worked properly and that our user data was being encrypted and stored privately. We also updated our UI, moving away from Storyboard, which allowed us to have a more modern look for our end users.

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 17.0.

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	2/9/2024
Create User UI	Front End	Design Account Creation UI	2/9/2024
Login Authentication	Backend	Allow Users To Authentication Login Using Google Firebase Services.	2/16/2024
New User Authentication	Backend	Allow Users To Create a New Account Using Google Firebase Authentication Services.	3/1/2024
User Profile	Backend / Front End	Allow the user to setup a new profile which would record information used for our calculations.	2/9/2024
Main Dashboard UI	Front End	Design Main Dashboard UI	2/9/2024
Main Dashboard Functionality	Front End	Allow Users to Navigate the Application Using the Main Dashboard.	2/9/2024
Start Workout UI	Front End	Design Start Workout UI	2/9/2024
Start Workout Submission	Backend	Allow Users to Submit Inputted Workout Data.	3/29/2024
View Previous Workouts	Backend / Front End	Allow Users to View Previously Submitted Workout Data.	3/29/2024
1RM Calculator	Backend / Front End	Design 1RM Calculator With Functionality Based On User's Input	3/15/2024
Wilks Calculator	Backend / Front End	Design Wilks Calculator With Functionality Based On User's Input	3/15/2024
531 Workout Plan	Backend / Front End	Workout plan based on calculations taken from the user's profile.	3/15/2024

3. Requirement Specifications

3.1. Stakeholders for the system

Our current stakeholder and sponsor for this project is Matt Murphy. Matt is a personal trainer for power lifters and contacted us about our project after we originally created it

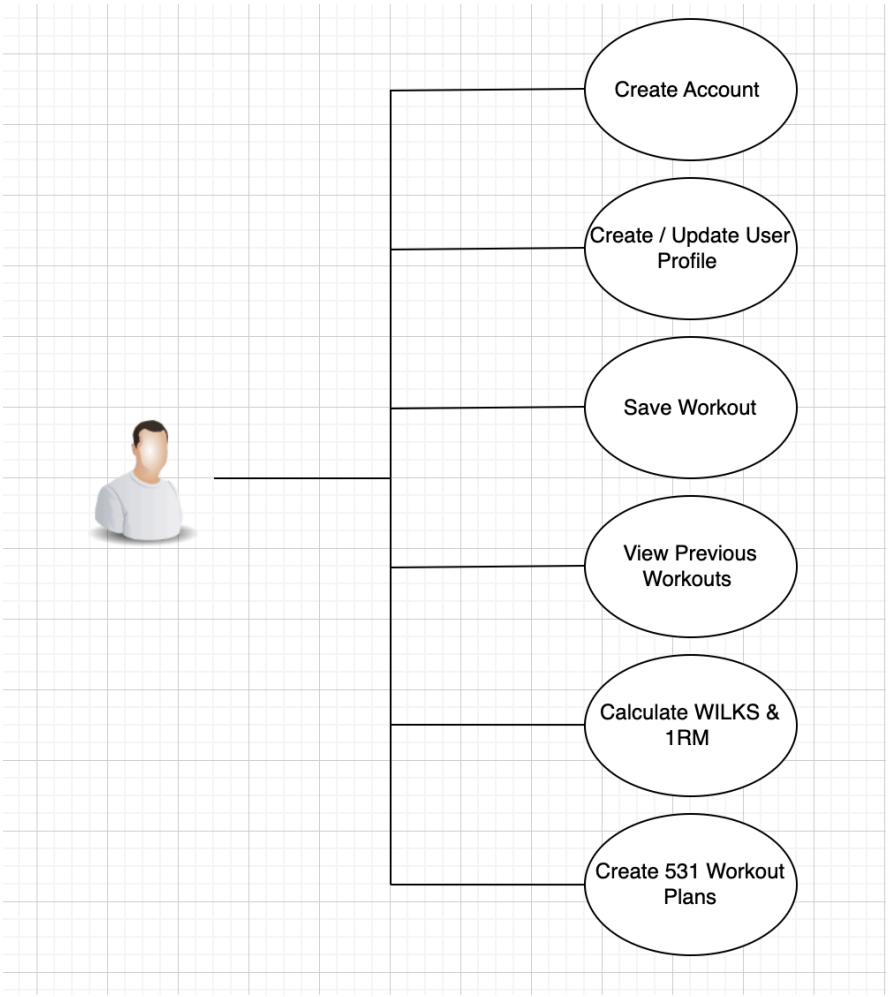
back in EEC 521. He expressed interest in it as he, along with his customers, in our application after seeing how easy and seamless it was to use. Our goal is for the application to assist not only Matt but the entire health and fitness community.

3.2. Use cases

Our application has 7 different use cases

- Create Account
- Create User Profile
- Save Workout
- View Past Workouts
- Calculate Wilks Number
- Calculate One Rep Max
- Created 531 Workout Plans

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: Create & Update Profile	
Actors	Lifter (User)
Description	Allow a new user to create and update their user profile
Data	Gender, Weight, Max Bench Press, Max Squat, Max Deadlift, Max Overhead Press
Stimulus	User fills out information
Response	Either confirmation of profile creation / Update or error that user profile couldn't be updated
Comments	A brand new user of the app can create their initial profile or update it based on their workout progress.

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.

Use case: 531 Workout Plan	
Actors	Lifter (User)
Description	Allow a new user to generate a 531 workout plan based off their user profile inputs.
Data	Gender, Weight, Max Bench Press, Max Squat, Max Deadlift, Max Overhead Press
Stimulus	User fills out information
Response	531 Plan will automatically calculate and populate the view for the user. If this does not occur an error would be displayed when entering data into the user profile during creation.
Comments	A new wants to create a workout plan based off the 531 regimen.

3.3. Rationale for your use case model

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 While also providing 531 workout plans, WILKS and 1RM calculations.

3.4. Non-functional requirements

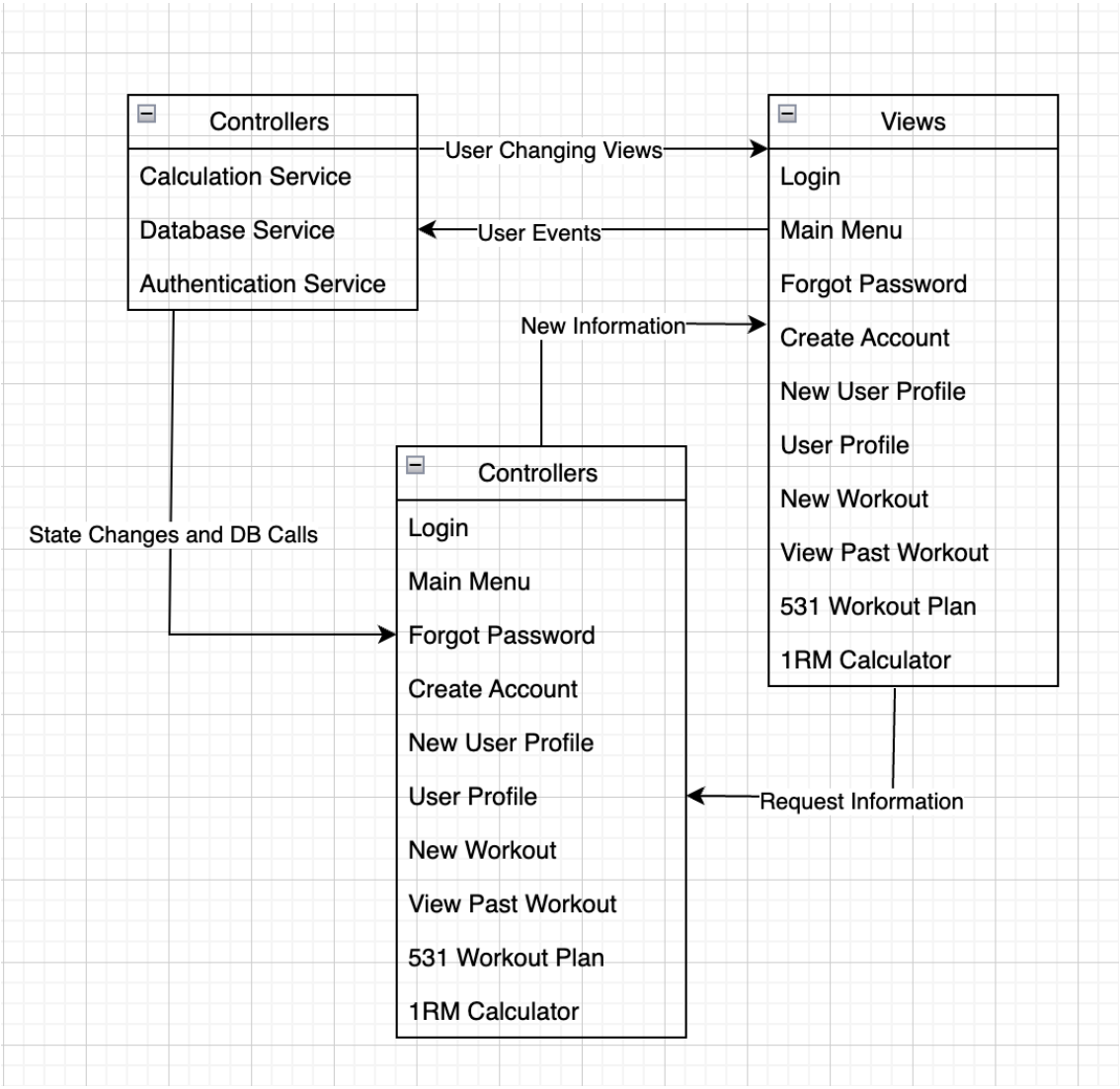
The only non-functional requirement we have is for our application to run on other Apple Devices. To achieve this we are creating the app using Apple's swift language so it will scale and change based on the device with minimal configuration from us.

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 5.10
- IDE: Xcode
- Source Control: Github (https://github.com/Neilkal867/LiftingLog)
- Database: CloudKit
- Authentication: Firebase Authentication
- Hardware: Macs & iPhone

4.4. Rationale for your architectural style and model

After extensive research we found the most common and streamlined implementation methods for simple mobile applications. 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. This allows our architecture to be flexible when upgrading ‘pieces’ of our application.

5. Design

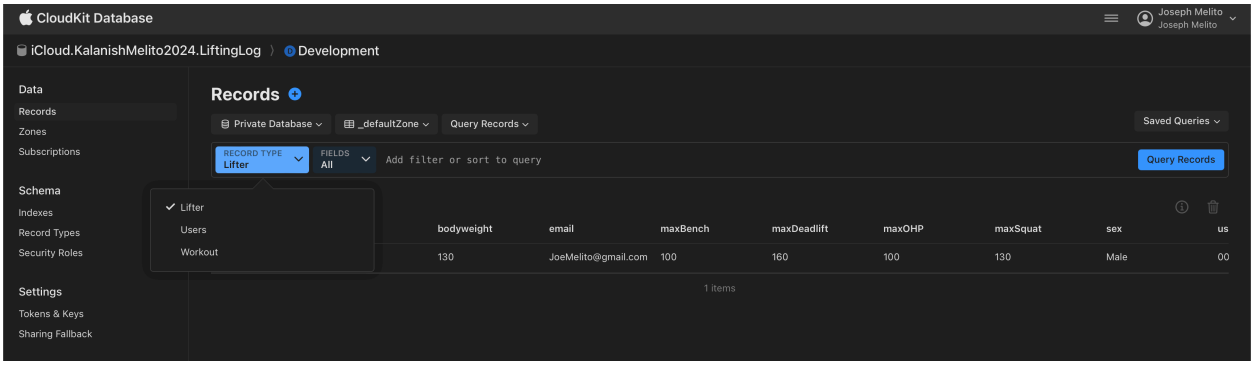
5.1. User Interface design

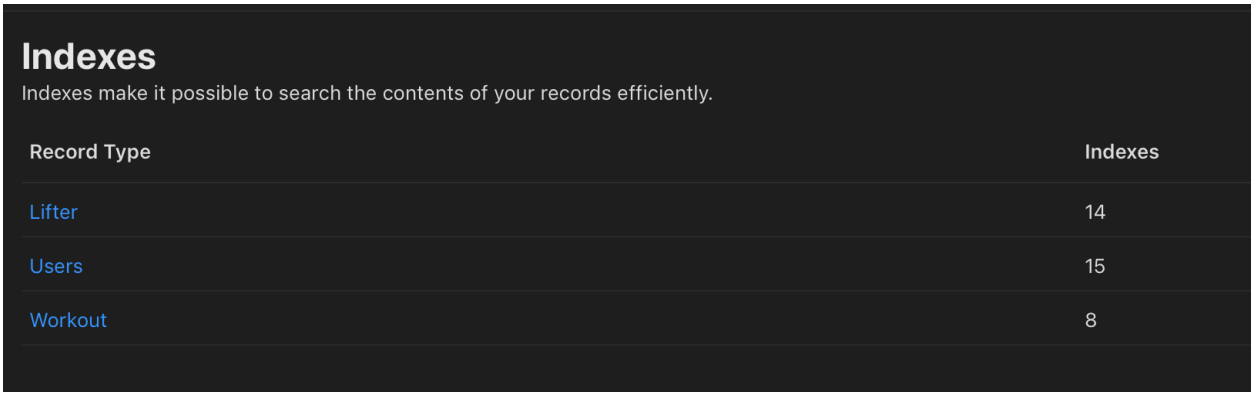
We offer a basic and straight forward user interface. Our application isn’t geared towards having a lot of functionality it is designed to be simple and straight forward. We are trying to mimic ‘pen and paper’ is why we keep it simple. As of right now it only has ten different screens and but the bulk of the apps functionality lays within four of those.

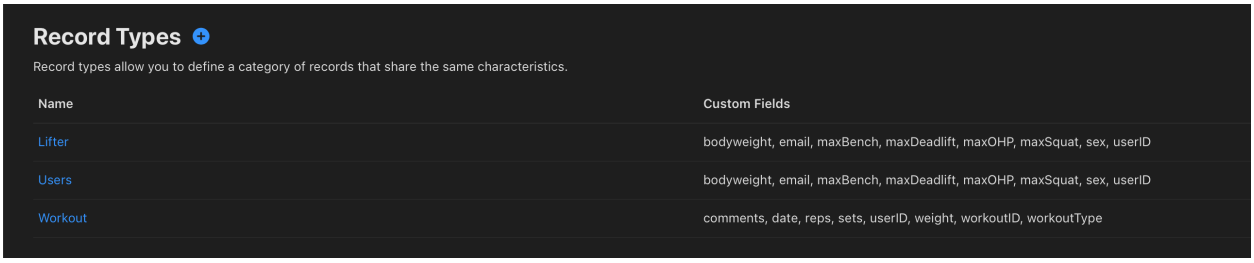
5.2. Components design (static and dynamic models of each component)

All of the components except a few are based off of static models. Due to the simplicity of the application the login, create account, and create workout screens are all static displays and won't change. The view workout,531 workout plan, and user profile components are the dynamic ones that will change based on the users input.

5.3. Database design







The database we are using is a Apple's CloudKit Database. CloudKit is a Sql DB that is created and supported by Apple. The CloudKit Database is setup with "records" that contain custom fields to hold data. Each custom fields can be a different type (String, Int, Float, etc). In our database schema we have records for lifter, user, and workout. User allows us to bridge our DB with Google's Firebase authentication system and link it to Lifter. Which then holds all of the basic account information; such as, body weight, email, max lifts, gender, and user ID. As for our workout record, that stores data associated with submitting workouts; such as, workout type, comments, date, reps, sets, and weight.

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	My Profile: 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	TC18
Test Task	My Profile
Test Input	Tester may select the ‘Update Profile’ button in the “My Profile Tab”
Expected Output	User will be prompted with a “Profile Successfully Updated” Dialog box.
Description	Users have the ability to update their gender, body weight, and max lifts within their profile.

ID	TC19
Test Task	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	TC20
Test Task	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.

ID	TC21
Test Task	531 Workout Plan
Test Input	Tester may enter any input values from my profile to see if 531 Workout Plan is displaying correct values.
Expected Output	An integer output value.
Description	Users must be able to view their 531 workout plan which corresponds to values entered within the user profile section.

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
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

Associated Test Case IDs	Requirement	Department	Specification	Design	Testing
TC13, TC14, & TC 15	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.
TC 16 & TC17	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.

Associated Test Case IDs	Requirement	Department	Specification	Design	Testing
TC8 & TC9	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
TC19 & TC20	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
TC17 & TC18	My Profile (Update)	Front End	The Page will allow the user to see their personal details and update their max lifts	There are several input fields that the user can interact with. There are also text fields that display the total weight lifts along with a WILKS calculation	- Verify that values have been updated and saved - Verify that user cannot enter anything but integer or decimal values.

Associated Test Case IDs	Requirement	Department	Specification	Design	Testing
TC21	531 Workout Plan	Front End	The page allows the user to access their 531 workout plan through a tabbed view. Each tab will showcase a 4 week plan of their workout.	There are no input fields. All fields are static and are not interactable to the user. They should only display information. However at the top of the screen there are a series of tabs that the user can navigate through	- Verify that the user can navigate through each tab. - Verify that all tabs are displaying the correct data.
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 authenticatio n service will be a class that contains all of the variables and functions required to communicate with the Firebase Authenticatio n 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
All TCs are relevant	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
All TCs are relevant	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 and utilize Apples TestFlight program 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

Our Team was successful in creating a mobile application that can run on Apple Devices to help track a users workout and create 531 workout plans. We set out to create an effective yet simple way to assist lifters who still track workouts with pen and paper. Additionally, we wanted to give the user an app that avoids tracking a million different things or is loaded with ads and unnecessary features. We are currently working with Apple to get our application through their TestFlight program in hopes to eventually upload it to the Apple App Store.

Throughout this project we enhanced our skillset in swift, mobile app development, and working with authentication services & a database. Additionally we met regularly to ensure our application met the requirements of our sponsor. This gave us insight to how software engineering is more than just programming. In order to create a successful product you must also understand how to market and communicate with your users.

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

Overall the project was successful for a college project. Our team had minimal experience with the tech stack we used. We also took it a step further and worked not only with our sponsor to ensure requirements and deadlines were met, but we also worked with Apple engineers to get into the TestFlight program with hopes of a launch on the Apple App Store.

7.2. Future development

There is a lot for future development. There were some new concepts we learned late in the project we would like to possible implement such as generative AI. This could give our users the ability to create detailed workout plans, diet regimens, or even and in-depth analysis of their workout data. We would also like to note that there is also improvement within our codebase through refactoring.

Another big upgrade would be continuously working with our sponsor and future clients to ensure an even more streamlined application. We do not want to limit ourselves to the existing features we currently have today. We want to work with our users to understand their wants & needs so that we can implemented them into our application.

Lastly one of the improvements we could make would be a completing Apple's TestFlight program and launching our application onto the App Store. We also want to expand our application to the Apple Watch and possibly even onto Apple's new VR/AR headset. Our goal is to give our users the ultimate workout experience wherever they are.