SOFTWARE DESIGN SPECIFICATION

Lifting Log

Joe Melito (2571661) & Neil Kalanish (2651086)

1.0 Introduction

Lifting Log will be developed using Apple's programming language, Swift, in addition to being developed within their IDE, Xcode. We also are implementing a backend data base using Google Firebase. This will allow for data to be stored and passed security for our users. We will be following the object oriented approach for our application and also creating micro services to decouple our backend from our frontend.

1.1 Goals and objectives

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 Statement of scope

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

1.3 Software context

Our software's product placement is within the workout & health community. The main feature is to allow users to track, plan, and view their weight lifting workout sessions. We will solely be focusing on weight lifting as we want to remove any unnecessary 'workout bloat' from our application and eliminate the dependency weight lifters have to recording workouts on pen and paper.

1.4 Major constraints

Since our software is being developed for a 16 week course the major constraint is time. The team consists of two developers who are working full time within the software development profession. So learning Apple’s programming language, Swift, and Google’s Firebase database engine will be a challenge.

2.0 Data design

The data structures we will be using include a user, workout, and workout calculation. All of this data will be stored within our database. As mentioned previously, our database of choice is Google Firebase.

2.1 Data structures

Data Structures within our application include:

- User Structure:
    - First Name
    - Last Name
    - Date of Birth
    - Gender
    - Username / Email Address
    - Password
    - Workout Data
- Workout Structure:
    - Workout Type
    - Amount of Sets & Reps
    - Weight Lifted
    - Additional Comments / Details (Custom User Input)
- Workout Calculations Structure:
    - Max Dead Lift
    - Max Bench
    - Max Squats

2.2 Database description

Google Firebase will be used to store all the corresponding data structures mentioned above. This enables our users to query the database and view all workout or personal information that is stored in our backend. We will also be implementing Google Firebase authentication to allow our users too quickly authentic themselves using preexisting google or social media accounts.

3.0 Architectural and component-level design

The architecture for our app is going to follow a “Smart & Dumb” component design. We will define a smart component as a component that can handle logic like performing calculations, fetching and saving data. A dumb component will be a component that is only going to display information and not perform any sort of calculations or saving of data. Following this pattern all of our front end components (views) won’t do any sort of saving or perform any logic, they will hand off any work that needs to be done to a smart service. Services are going to perform all work required by the app then hand the information back to the view for the user to see.

3.1 Architecture diagrams

See attached software diagram.

3.2 Description for Components

Our application is going to consist of four main components the front-end, a database service, authentication service and calculation service.

3.2.1 Front-end

3.2.1.1 Interface description

The frontend is going to be our “dumb” component and will just pass information from the use to any of the required services and display the services response. This will allow the front end to change as we need it and grow as the app grows all without having to change any calls or calculations happening. This will also provide the inverse, we can update the database, change where the information is store and make calculations more accurate without having to worry about the UI changing and confusing the user.

3.2.1.2 Static models

N/A

3.2.1.3 Dynamic models

N/A

3.2.2 Database Service

3.2.2.1 Interface description

The database service is going to handle all of the saving and downloading of data. It will be responsible for making all calls to the DB and modeling the data as required then sending the data to the front end to display.

3.2.2.2 Static models

Database Service

```
UserName: String
Date: String
WorkoutName: String
Struct Workout{
    Lift: String
    Reps: Int
    Sets: Int
    Comments: String
}
```

SaveWorkout(Date, Workout)
- Return with True or False for successful Save

LoadWorkout(Username)
- Return all workouts from user

ConvertJsonToWorkout(JsonFile)
- Return Workout
- This will convert Json returned from Firebase and transform it to our workout struct and hand it back to LoadWorkout()

3.2.2.3 Dynamic models

N/A

3.2.3 Authentication Service

3.2.3.1 Interface description

We are using the firebase Authentication service for this app so we will just need to build out a service that takes in the users name, email and password then passes that firebase for verification. If the username and password are correct we will tell the viewer to show the homepage and if it is invalid we will tell the viewer to display an error.

3.2.3.2 Static models

Authentication Service

UserName: String Username: String Password: String
CorrectUsernameAndPassword(Username, Password) - Return with True or False for valid username and password

3.2.3.3 Dynamic models

N/A

3.2.4 Calculation Service

3.2.4.1 Interface description

The calculation service will be able to perform all of the calculations that the app will offer. Currently we are going to offer a wills number calculator and a one rep max calculator, but as the app goes our offerings will grow and all the calculations will reside in this service.

3.2.4.2 Static models

Calculation Service

Bodyweight: Int OneRepMax: Int MaxBench: Int MaxSquat:Int MaxDeadlift:Int Weight: Ints Reps: Ints
CalculateWilksNumber(Bodyweight,MaxBench,MaxSquat,MaxDeadeadlift) - Return your wilks number CalculateOneRepMax(Weight, Reps) - Return a potential one rep max

3.2.4.3 Dynamic models

N/A

3.3 External Interface Description

Lifting Log is being created within the Apple ecosystem of applications. This would allow this application to run on over Apple devices such as: iPads, MacBooks, Apple Watch, and all current supported iPhone devices. The only external software that we will be interfacing with is our Google Firebase database.

4.0 User interface design

The benefit of developing an application within Apple's ecosystem is Xcode and Swift include a user interface builder; Swift Storyboard. We will be using Swift Storyboard to build out our entire user interface for our application. All of our screens will be wrapped in a Navigation Controller, which allows users to seamlessly navigate between and all screens listed below.

4.1 Description of the user interface

Our user interface will include the following:

- Login Screen:
    - This will allow the user to type in their username / email and password to log into our application.
    - If a user does not have a login already they have access to create a new account.
- Account Creation Screen:
    - This will give the user the ability to create a new account by entering the following data:
        - First & Last name
        - Email / Username
        - Password
        - Date of Birth
    - If however, the user no longer wanted to create a new account they will have the ability to cancel the creation.
- Lifting Log Main Dashboard / Home Screen:
    - This is the main screen the user will be presented with upon logging into our application. From here the user can navigate to the following screens:
        - Start or Create a Workout
        - View Previous Workouts
        - Settings
        - Wilks Calculator
        - 1RM Calculator

- Starting or Create a Workout Screen:
    - These screens will allow our user to start entering data in for their corresponding workout. The date fields associated with these screens include:
        - Workout Type
        - Amount of Sets & Reps
        - Weight Lifted
        - Additional Comments
- View Previous Workouts Screen:
    - This screen will allow our user to select a previous workout based on a date picker. Once the user selected the date the application will query our database and present the corresponding data which will be represented in the following fields:
        - Workout Date
        - Workout Type
        - Amount of Sets & Reps
        - Weight Lifted
        - Additional Comments
- Settings Screen:
    - By selecting this option the user will be presented with different settings within our application. Some settings include:
        - Light or Dark Mode
        - Time Zone
        - Date format
    - Please note our settings options and/or screen are subject to change base on the development process as some settings may not be necessary to our user base.
- Wilks Calculator Screen:
    - Our Wilks calculation screen will have the following user input fields:
        - Gender
        - User's Weight
        - Max Dead Lift
        - Max Bench
        - Max Squat
    - Once our users enter in all the information above they will initiate the calculation via a button and a Wilks calculation output will appear at the bottom of the screen.

- 1RM Calculator Screen:
    - Our 1RM calculation screen will have the following user input fields:
        - User’s Weight
        - Amount of Reps
    - Once our users enter in all the information above they will initiate the calculation via a button and a 1RM calculation output will appear at the bottom of the screen.

4.2 Interface design rules

Swift Storyboard has a vast library of UI components such as buttons, textfields, and views. We will be implementing the default designs for the majority of our UI components as we want our application to be as minimal as possible. This gives users the ability to quickly log into our application and enter or view previous data without any unnecessary bloat. As a future enhancement we will look to implement custom UI component designs if necessary.

5.0 Restrictions, limitations, and constraints

As part time students with full time jobs working within the software development profession; our limitations include learning iOS development and Google’s Firebase database engine. We will be developing this project completely from scratch while also teaching ourselves Swift, Xcode, & Firebase. Due to our lack of knowledge, implementation of our database and data structures may not be as efficient as a full fledge production application. Our primary goal is to learn about new technologies and build an iOS application. As our application development and class advances we will be able to make future enhancements to our project allowing our database and data structures to be architected and implemented effectively

6.0 Appendices

Presents information that supplements the design specification.

6.1 Requirements traceability matrix

Please see attached.

6.2 Implementation issues

Implementation issues will be addressed as they are presented. We do not anticipate or foresee any issues at the moment. However, if an issue does occur we will reference our project timeline and adjust development or features accordingly.

Requirements Traceability Matrix

Requirement	Justification	Department	Specification	Design	Testing
Design User Login	Users will need a way to log into our application to use it.	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.
User Account Creation	A new user needs a way to create a new account.	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
User Main Dashboard	A user needs a way to navigate the app and access each screen.	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

Requirement	Justification	Department	Specification	Design	Testing
User Settings	A user needs a way to change settings within our application.	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.
User Workout Creation	A user needs a way to create a new workout.	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.

Requirement	Justification	Department	Specification	Design	Testing
User View Previous Workouts	A user needs a way to view previous workout submissions.	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.
User View 1RM Calculation	A user needs to view their 1RM calculation	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
User View WILKS Calculation	A user needs to view their WILKS calculation	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

Requirement	Justification	Department	Specification	Design	Testing
Authentication Service	The app needs a way to authenticate each user	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
Calculation Service	The app needs a service to perform all calculations	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

Requirement	**Justification**	**Department**	**Specification**	**Design**	**Testing**
Database Service	The app needs a service to perform all calls to the database	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