

FitHub

Design Doc

Team 5: Andy Plank, Jonathan Huber, Ray Truong, Collin Li, Trevor Hill, Brian Long

Index

● Purpose	2
○ Functional Requirements	
○ Non-Functional Requirement	
● Design Outline	5
○ Components	
○ High-Level Overview	
● Design Issues	7
○ Functional Issues	
○ Non-Functional Issues	
● Design Details	10
○ Class Diagram	
○ Descriptions of Classes and Interaction between Classes	
○ Sequence of Events	
○ API Routes	
○ UI Mockup	

Purpose

Being fit is already hard enough, people should not have to create new workouts, remember them, and track their progress. FitHub allows users to log their progress, gives them suggestions for new exercises, and connects them with others so they can review their workouts. Many workout apps have features such as logging, workout suggestion, and social networking, but none combine all of them into one, easy to use, mobile app.

Our app seeks to satisfy this absence in the marketplace, providing a workout companion that provides all of the commonly desired features amongst users of other products into one elegant UI.

Functional Requirements

1. Profile and Account Management

- 1.1. As a user, I would like to register for an account using Google.
- 1.2. As a user, I would like to log into my account.
- 1.3. As a developer, I would like to insert custom workouts into a profile.
- 1.4. As a developer, I would like to insert custom exercises into a profile.
- 1.5. As a user, I would like to view my profile.
- 1.6. As a user, I would like to edit my profile.
- 1.7. As a user, I would like to view personal record statistics.
- 1.8. As a user, I would like to view total volume lifted statistics.
- 1.9. As a user, I would like to view the number of days of activity with a days-worked log in the style of GitHub's commit history.

2. Interacting with Workouts

- 2.1. As a user, I would like to view a calendar of my previous workouts.
- 2.2. As a user, I would like to be able to view my workouts from any device that has FitHub installed.
- 2.3. As a user, I would like to select a day from the calendar to view the workouts for that day.
- 2.4. As a user, I would like to select a workout from a day and see more details about it.
- 2.5. As a user, I would like to be able to view all exercises within a previous workout.
- 2.6. As a user, I would like to be able to view all weight and rep counts within a previous workout.

3. Logging

- 3.1. As a user, I would like to log a standard workout on a day.
- 3.2. As a user I would like to log a custom workout on a day.
- 3.3. As a user, I would like to log sets, reps, and weights associated with a workout.
- 3.4. As a user, I would like to schedule a standard workout for a day in the future.
- 3.5. As a user, I would like to schedule a custom workout for a day in the future.
- 3.6. As a user, I would like to re-log a previous workout from a past day.
- 3.7. As a user, I would like to receive a notification when I have “missed” my workout.

4. Creating Workouts

- 4.1. As a user, I would like to create a custom workout.
- 4.2. As a user, I would like to add standard exercises to a workout.
- 4.3. As a user, I would like to create a custom exercise.
- 4.4. As a user, I would like to add custom exercises to a workout.
- 4.5. As a user, I would like to have workouts suggested to me based on muscle group.
- 4.6. As a user, I would like to have workouts suggested to me based on completion time (if time allows).

5. Social Platform

- 5.1. As a user, I would like to search another user’s public profile.
- 5.2. As a user, I would like to view another user’s public profile.
- 5.3. As a user, I would like to publicly post one of my workouts.
- 5.4. As a user, I would like to view the profile of someone who publicly posted a workout.
- 5.5. As a user, I would like to have a publicly viewable page that displays my workout activities.
- 5.6. As a user, I would like to have the means to access my personal records, statistics, and activity log from my profile page.
- 5.7. As a user, I would like to view publicly posted workouts as an Instagram style list.
- 5.8. As a user, I would like to filter the publicly posted workouts by muscle group.
- 5.9. As a user, I would like to sort the publicly posted workouts by time.
- 5.10. As a user, I would like to rate a publicly posted workout.

- 5.11. As a user, I would like to comment on a publicly posted workout (if time allows).
- 5.12. As a user, I would like to save a publicly posted workout to my collection (if time allows).

6. Nutritional Information

- 6.1. As a user, I would like to log my caloric intake.
- 6.2. As a user, I would like to log my nutritional intake (if time allows).
- 6.3. As a user, I would like to view my previous caloric intake as a graph/chart (if time allows).
- 6.4. As a user, I would like to see a graph/chart of my caloric intake (if time allows).

Non-Functional Requirements

1. Usability

One of the biggest complaints about workout apps is a lack of user-friendliness. Since our application is very feature-dense, we need to make general navigation simple and self-explanatory. Users should be able to navigate without a tutorial or guide.

2. Scalability

Our app will store a lot of information in the user database. The database needs to be able to organize all of this information easily and traversed efficiently no matter how much data there is. We also need to be able to scale the database storage size up and down to handle a user base of one to a user base of one thousand.

3. Security

Sensitive user data, such as email and password, will be stored in a database. This database needs to be protected from SQL injections and passwords need to be encrypted in case of a breach.

4. Performance

The front-end will have to request data from the back-end which will be communicating with the database. This amount of network communication will dramatically slow down the app which will provide a bad user experience. Since the user experience is important to us, we need to ensure that our app can display all content within one second of a user trying to access it.

Design Outline

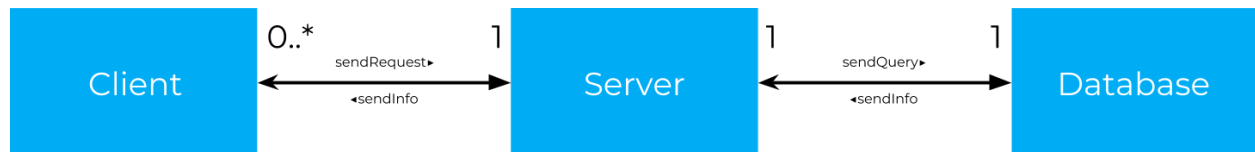
Our project is a mobile application that allows users to create, log, and discover various workouts. Our implementation will be a simple Client-Server-Database model. Multiple iOS/Android React Native clients will request the server, the server will handle the request and query the database, and the database will hold all of the information.

Components

1. iOS/Android React Native Client (Client)
 - a. The Client, running on multiple phones, will be the interface to our project.
 - b. The Client will send data to, and receive data from, our Server by HTTP requests in JSON format.
 - c. Data will be received from the Server, and then used in various parts of our application such as saved workouts.
2. Node.js Web Server with Express API (Server)
 - a. The Server will allow users to send and receive information with requests to simple URLs.
 - b. The Server will accept HTTP requests using REST URI conventions.
 - c. The Server will act as a middleman for the Client and the Database by receiving a request, querying the Database, and then returning the data in JSON format.
3. MongoDB Non-relational Database (Database)
 - a. The Database will hold all of the data used in our application.
 - b. Typical data stored in the Database will be links between users, saved workouts, and statistics about the user.
 - c. If scaling is needed, other parts of the application can be modified without losing data.

High-Level Overview

From left to right, the Client forms a many-to-one relationship with our Server in order to allow for multiple mobile clients contacting one web server API. When the Client sends a request to the Server, the Server will handle the routing, process the request, and then call a specific function in order to query the Database. When the Database receives a query, it handles the query within our Mongo database, and then returns the data to the Server. From there, the Server then handles the data and then returns it to the Client. Finally, the Client displays the information.



Design Issues

Functional Issues

1. Do users need to login to use the app, and if so, how?
 - Option 1: Allow users to create a unique username and password
 - Option 2: Allow creation of a login id via Google+, Facebook, etc
 - Option 3: No account creation needed
 - Decision: We chose to allow login through Google because we feel it would be easier and in the end, more secure. Because none of us have experience in creating a secure login system, it would probably be in the best interest of the users to have a prebuilt login system from a large company.
2. When should we update user stats on the profile page? (e.g. period of time they haven't exercised)
 - Option 1: Update values everytime they change
 - Option 2: Update values in set intervals
 - Decision: We chose to update values in set intervals rather everytime they change. By doing so, we lessen the amount of spam. This makes the UI look cleaner overall.
3. How should exercise activity be displayed on the profile page?
 - Option 1: Display in a series of colored blocks similar to GitHub activity
 - Option 2: Have a calendar and mark off each day the user worked out
 - Decision: We chose to display exercise activity similar to that of GitHub. This would allow us to display a more broad overview of the user's workout commitment. For example, GitHub has blocks covering a whole year of commits, each with a different shade green. For our app, we will use shades of blue to describe the intensity of the workout that day.
4. What can the user search in the feed?
 - Option 1: Users can search other users
 - Option 2: Users can search for workouts
 - Option 3: Users can search for both workouts and other users
 - Decision: We chose to only display other users when they search and let workouts have their own dedicated page. This would clean up any cluttering in the search results and would allow us to categorize workouts on their own page.

5. How should an exercise's number of sets (with corresponding weight and repetitions) be structured within the UI?

Option 1: Users can specify number of sets ahead of time, and then fill in weights and repetitions as they are completed into a provided chart-like structure.

Option 2: Users can add new sets to a workout as they are performed, with each new set appearing below the previous.

Decision: We chose to allow users to add sets as they are completed. This allows for less wasted space within the UI, and prevents the user from having to predetermine the number of sets they plan on performing ahead of time.

Non-Functional Issues

1. What type of database is appropriate to store our data?

Option 1: A relational database (e.g. mySQL, Microsoft SQL, etc)

Option 2: A NoSQL database (e.g. MongoDB)

Decision: We decided on a NoSQL database, more specifically MongoDB. We chose this because a few of our members have a little bit of experience creating and hosting MongoDB databases on the Google Cloud Platform. We also feel that with our structure of creating workouts and saving other people's workouts, a non-relational database would be most efficient.

2. How are we going to host our backend services?

Option 1: Google Cloud Platform

Option 2: Amazon Web Services

Decision: We chose to host our backend services using the Google Cloud Platform for a couple of reasons. First, a couple of our members have experience using the services that Google provides. Second, one member has credit to use on the platform, so we will be able to host everything for free, no matter the size of the application.

3. Which language is best suited to implement our backend services?

Option 1: JavaScript

Option 2: Java

Option 3: Python

Decision: We chose to implement our backend using JavaScript since a majority of our group has experience implementing backend services in JavaScript. We plan on implementing our backend service using the Node.js environment, which has thousands modules that can be helpful.

4. How can we update the feed to reflect current activity as the number of users increase?

Option 1: Let the user manually refresh the page (e.g. pull down and let go)

Option 2: Automatically refresh the feed after a set time interval

Decision: We decided to let each user manually refresh the page, as that is the implementation that most modern social medias use. We also feel that it would be easier to implement.

5. What framework are we going to use to implement our application across multiple platforms?

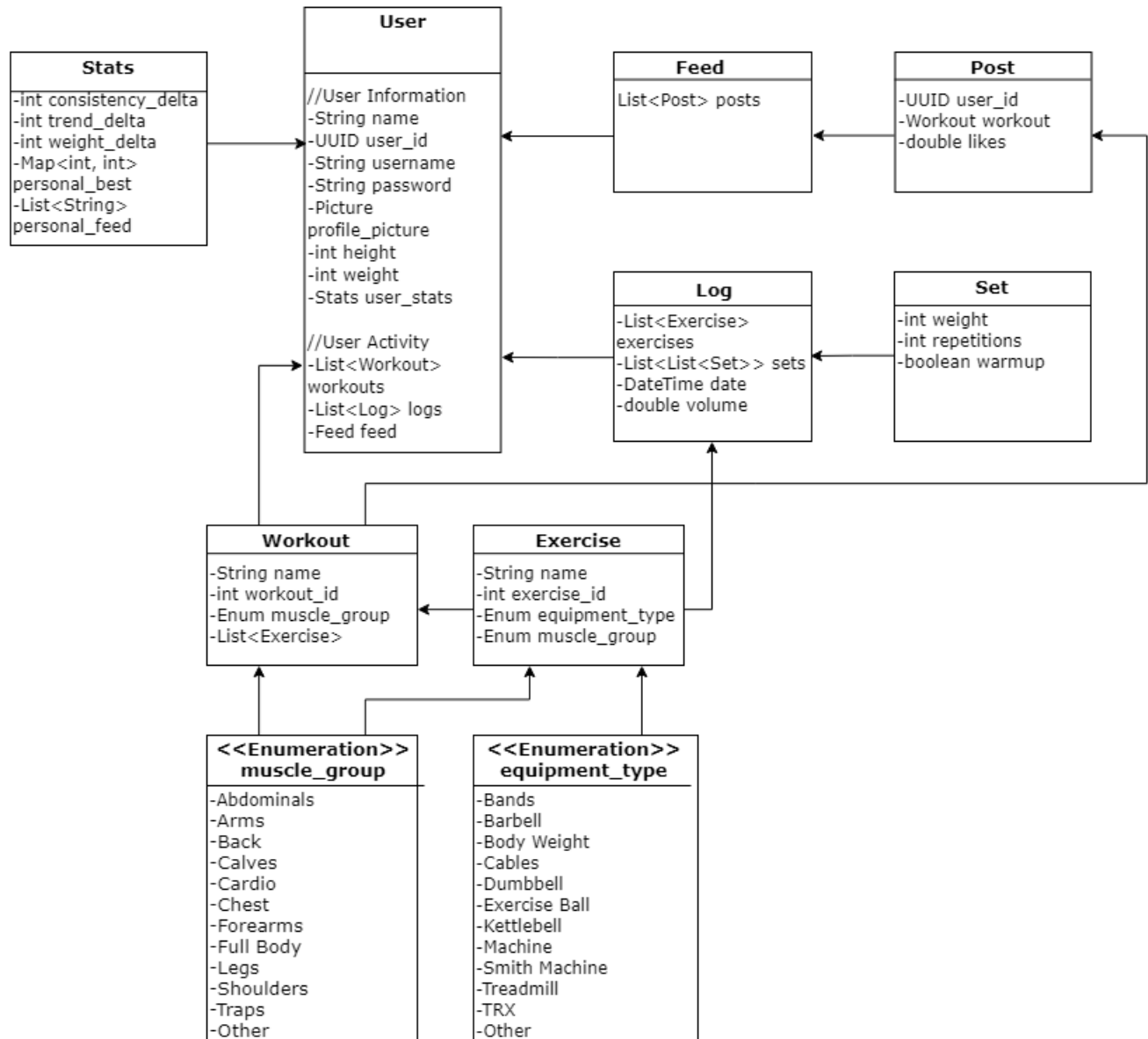
Option 1: React Native

Option 2: Flutter

Decision: We decided to go with React Native, as one of our members has experience with both React and React Native. Also Flutter requires you to write in Dart, which would require us all to spend a significant amount of time learning it.

Design Details

Class Diagram



Descriptions of Classes and Interaction between Classes

The classes represent the essentials of what a user needs to use our app. Using these classes, and their respective get/set methods, should make the interaction between the database and the user quick and easy.

User:

- Represents a unique user.
- Contains personal info about the user such as their name, weight, height, etc.
- Used as the main class to store stats and workouts of the user.
- Created when a person creates an account.
- Stats is a property in the user information section of the User class, and is updated whenever the user completes a workout.
- Workout is a property in the user activity section of the User class, containing all privately created workouts, as well as workouts saved from the public feed.
- Log is a property in the user activity section of the User class, and tracks the exercises the user did along with sets and weights.
- Feed is a property in the user activity section of the User class, containing workouts and/or users explicitly searched for by the user.

Exercise:

- Represents a specific exercise.
- Contains two identifiers, `equipment_type` and `muscle_group`, to classify exercise.
- Created when a user adds the exercise to their workout.

Workout:

- Represents a list of exercises.
- Contains a name, `workout_id`, and `muscle_group` to categorize the workout.

Set:

- Represents the amount of weight and number of repetitions performed within an exercise.
- Contains a boolean, initialized to false, if the user wants to classify the exercise as a warmup.

Log

- Represents what the user did for the day.
- Contains list of workouts, a list containing lists of sets, a `DateTime` object, and volume.

Post

- Represents a workout shared on the global feed.
- Contains a user id, specific workout, and amount of likes.

Feed

- Represents a list of posts.
- Elements within the list will be displayed on the global feed.

Stats

- Represent the stats of a user.
- Contains information about the user's improvement trends.
- Contains a mapping of the user's personal bests and a personal feed.

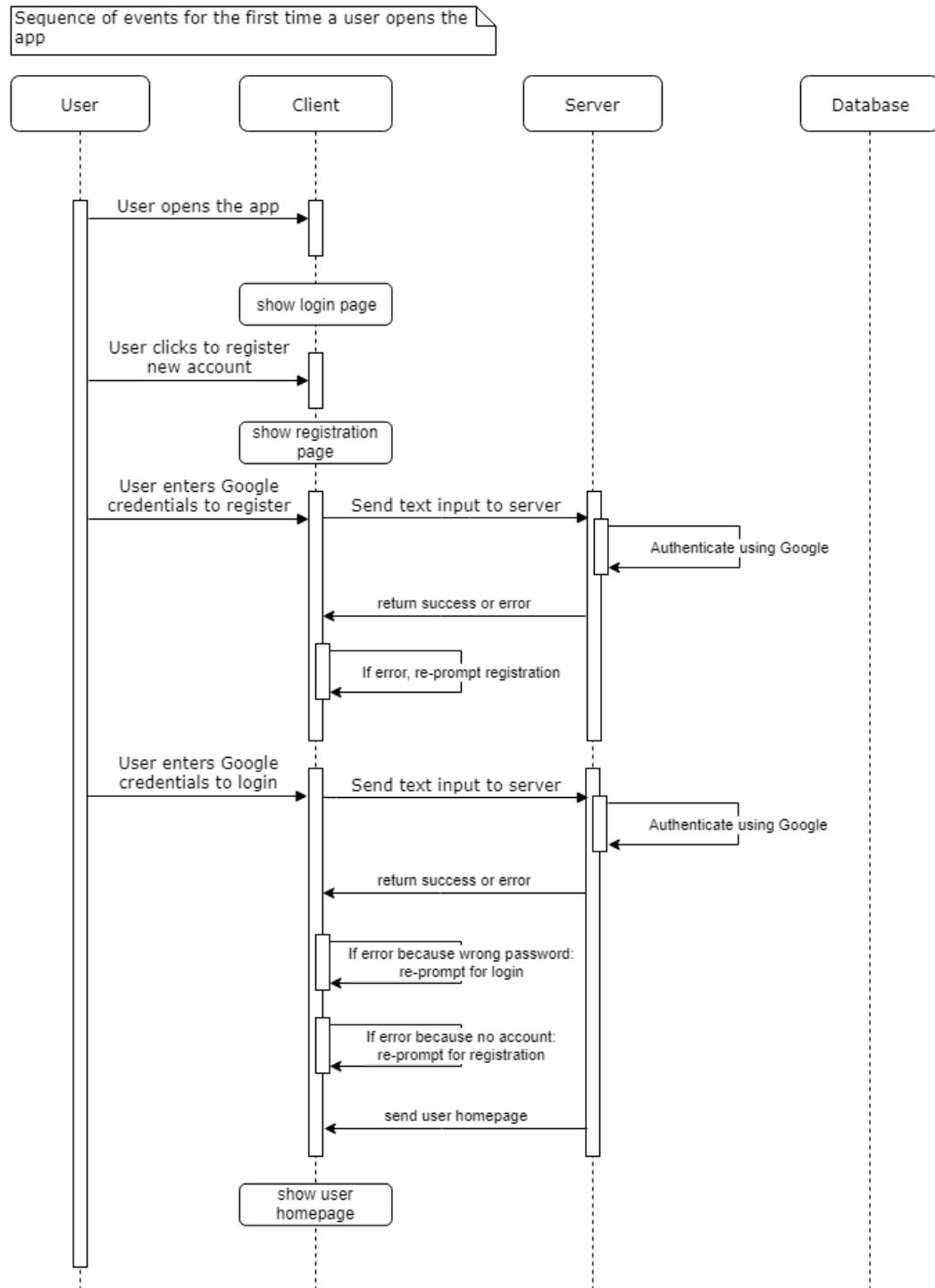
muscle_group

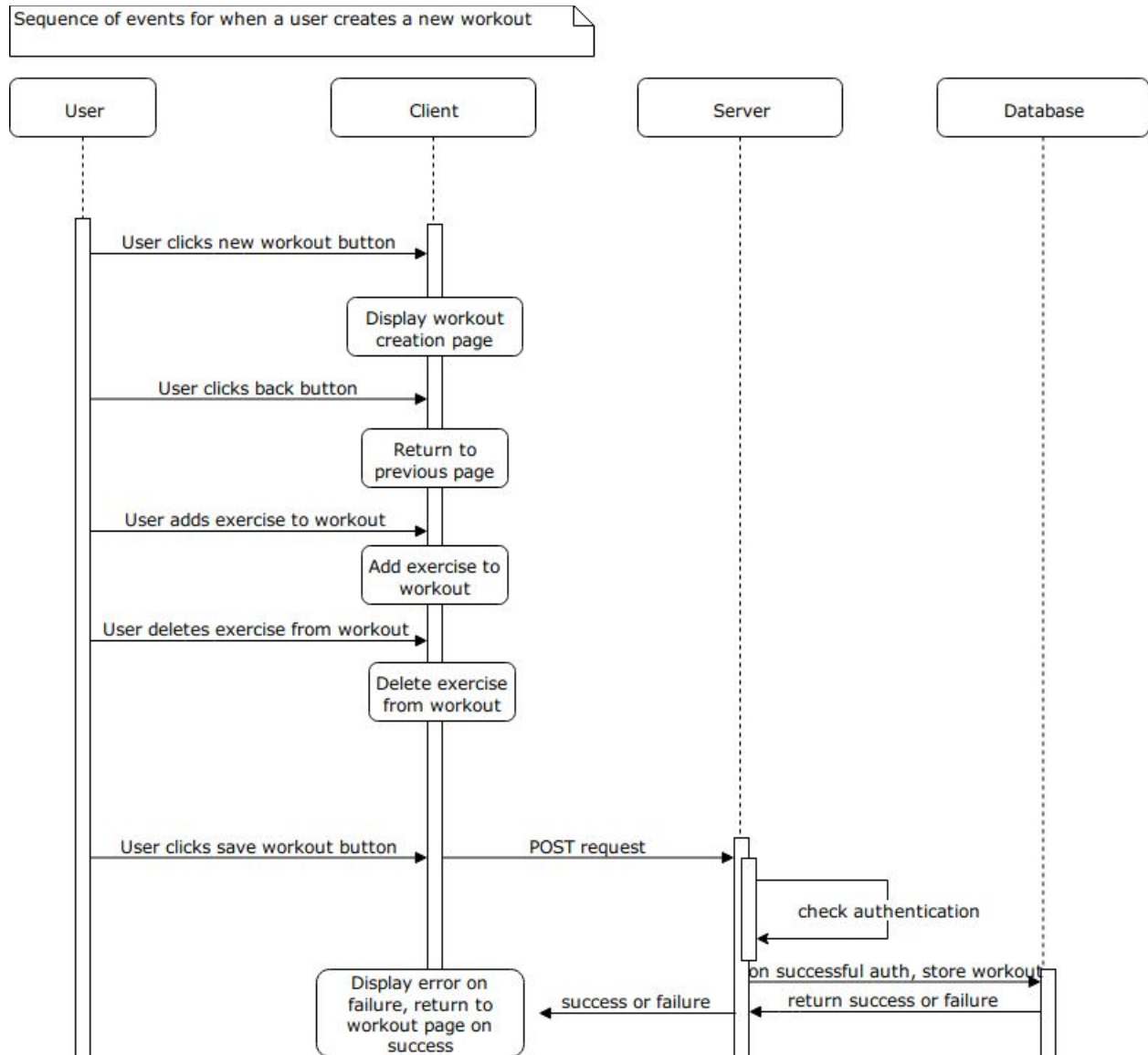
- Used to identify what muscle group the exercise or workout is training.
- Defined whenever an exercise or workout object is created by the user.

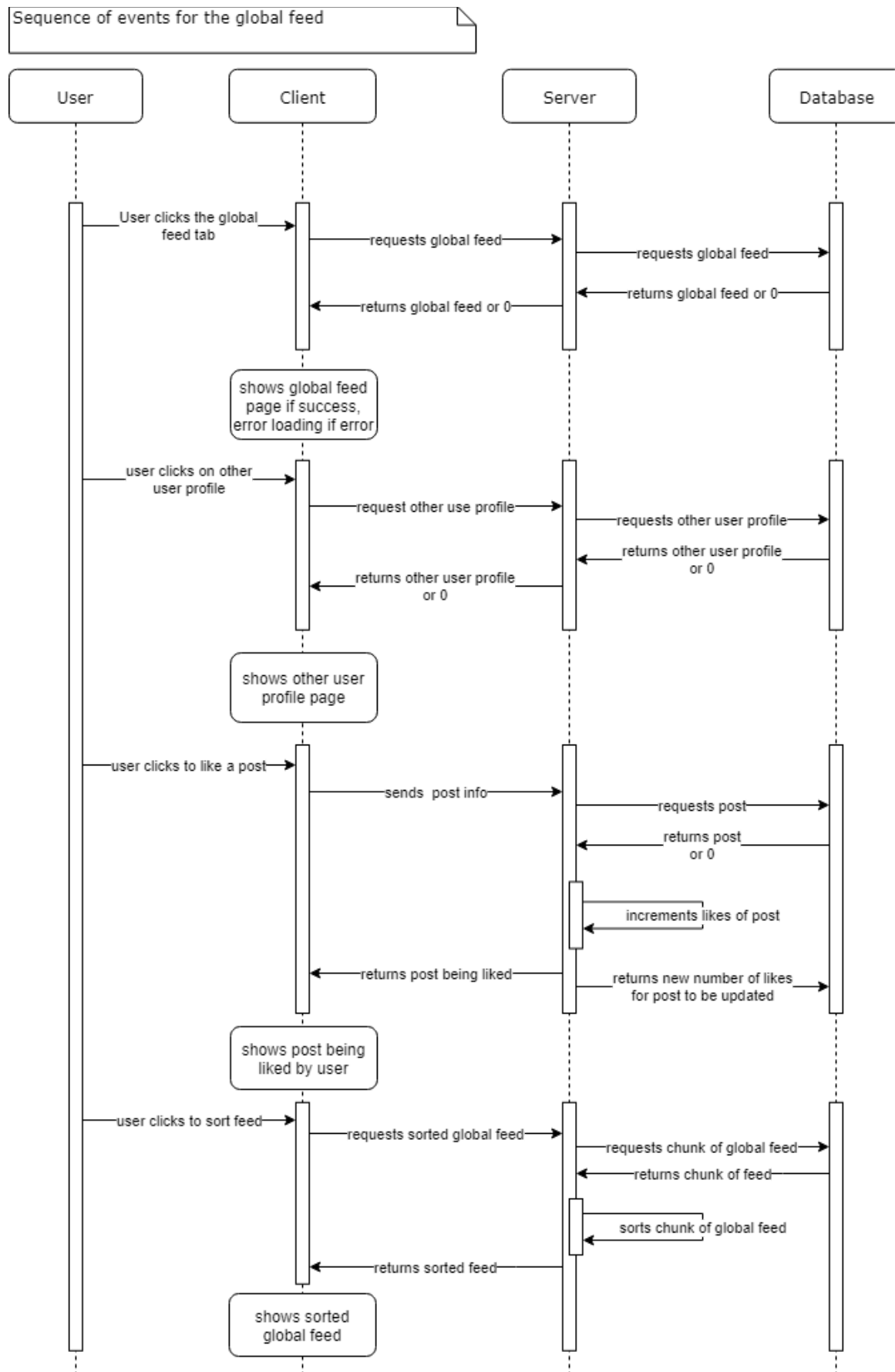
equipment_type

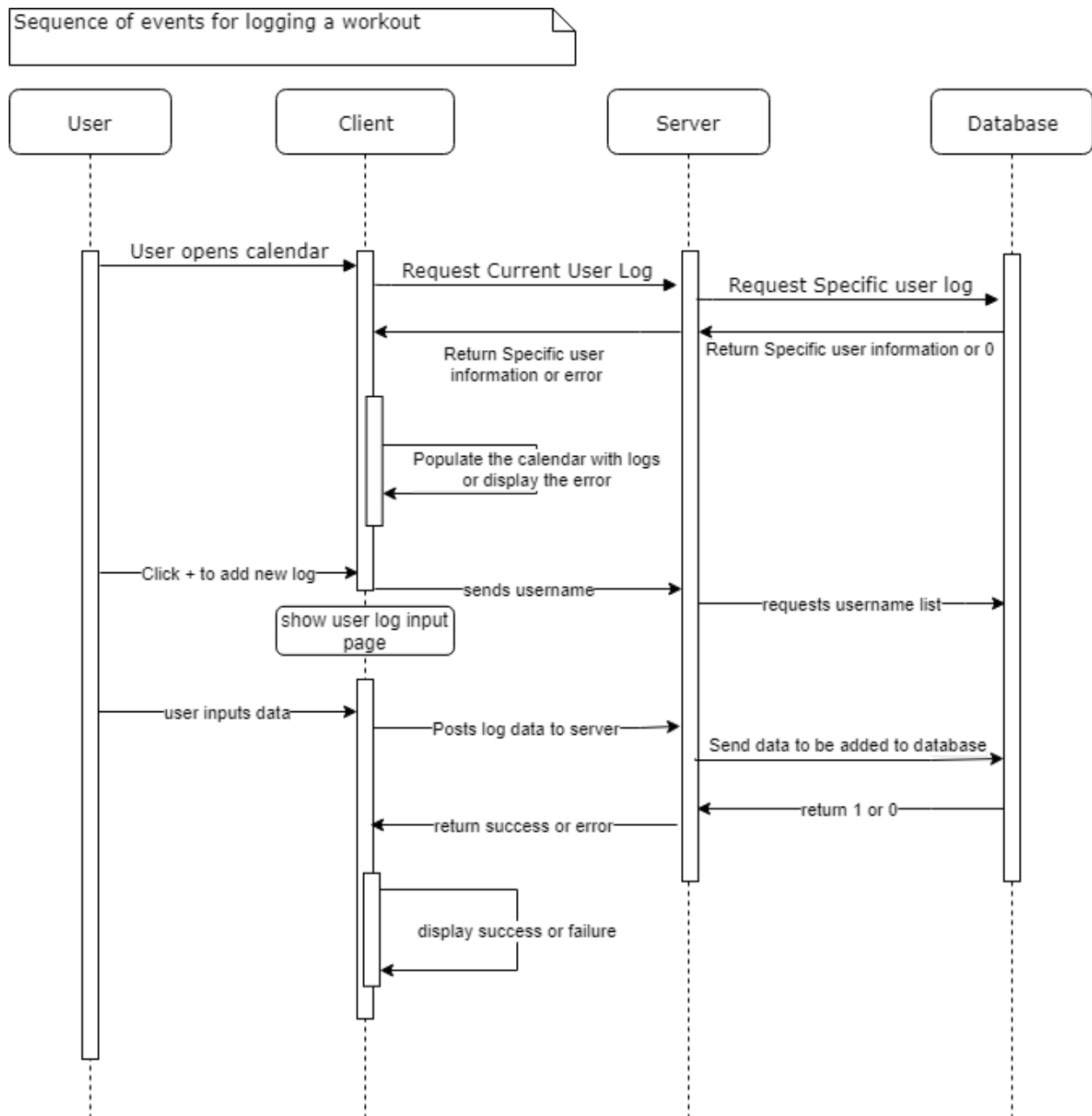
- Used to identify what equipment the exercise requires.
- Defined whenever a workout object is created.

Sequence of Events









API Routes


Purpose	API Route	Supported Requests
Logs in the user	/login	POST
Creates a user	/users/new	POST
Gets the workouts for a user	/users/:user/workouts	GET
Gets or sets the stats for a user	/users/:user/stats	GET, POST
Gets or sets the personal information for a user	/users/:user/details	GET, POST
Searches for a user	/users/search/:query	GET
Adds a workout for a user	/workouts/add	POST
Gets, edits, or deletes a workout	/workouts/:workout	GET, POST, DELETE
Gets the global feed	/workouts/global	GET
Likes a workout	/workouts/:workout/like	POST


UI Mockup

LOGIN


Welcome to FitHub


Login








Login



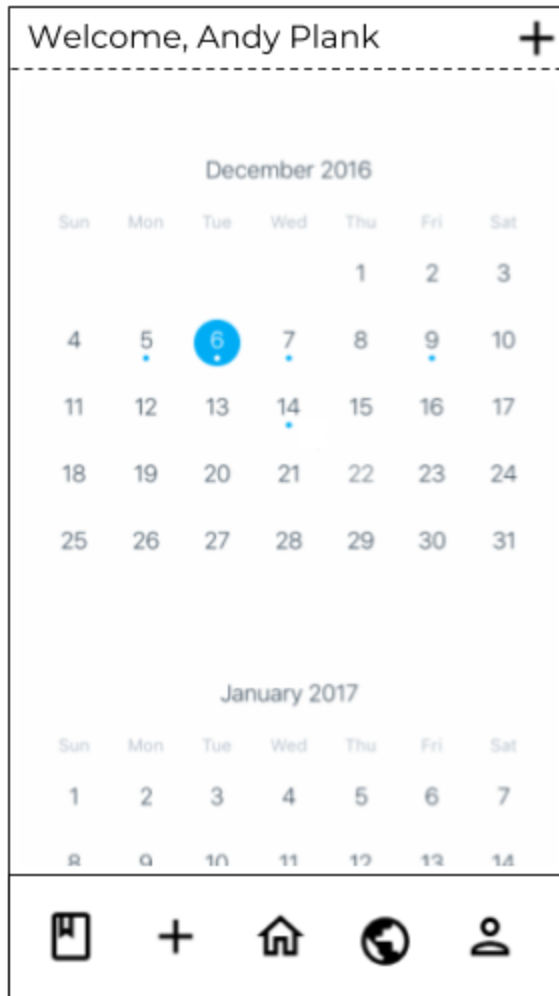




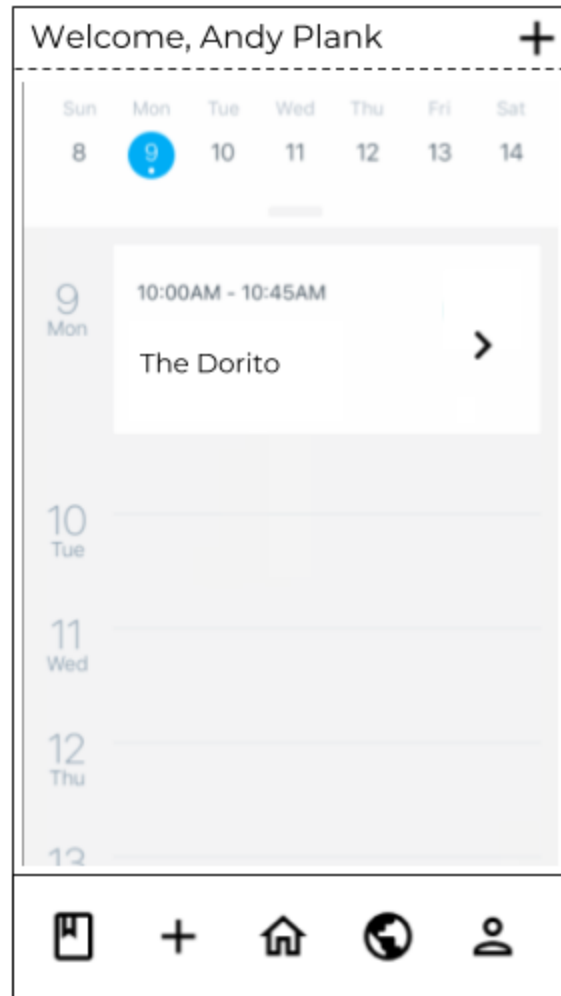









HOME PAGE (CALENDAR)



HOME PAGE (AGENDA)



WORKOUTS PAGE

Saved Workouts		+
Muscle Group		
Lower Body	6 Exercises	>
Upper Body	6 Exercises	>
Back		
Lats	3 Exercises	>
Rear Delts	3 Exercises	>
Traps	2 Exercises	>
Chest		
Chest	4 Exercises	>
Upper Chest	3 Exercises	>
Lower Chest	2 Exercises	>
    		

SINGLE WORKOUT PAGE

Workout Details

The Dorito

Bench Press

12x4

Single Arm Bicep Curl

4x10

Deadlift and Scream

5x5

Add an exercise...

+


Add to calendar


+

+

PROFILE PAGE (DEFAULT)

Andy Plank





97%

5%




+7

Consistency

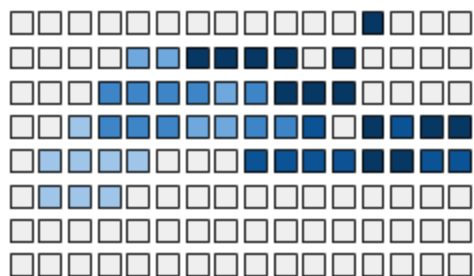
Trend






Weight

Edit Profile


Jan Feb Mar Apr May Jun Jul




PROFILE PAGE (WORKOUTS)

Andy Plank





97%

Consistency




5%

Trend

+7

Weight

Edit Profile










Workouts

The Dorito12 Exercises>


Olive Pants9 Exercises>


Shlarms11 Exercises>

PROFILE PAGE (ACTIVITY)

Andy Plank





97%

Consistency




5%


Trend

+7


Weight

Edit Profile









Andy has not done legs for 964 days!




Andy has just finished his 1000th arm day!



Andy hit a new max bench of 15 lbs!



Andy shared a new workout on the global page.



Andy shared a new workout

