
Dithi Saxena
Catherine Penquite
Abhishek Gunasekar
Christopher Yu
Vivek Nair
Team 6

Project Design Document

19th February 2021

Werk  It

TABLE OF CONTENTS

Purpose	3
• Functional Requirements	3
• Non Functional Requirements	5
Design Outline	8
• Components	8
• High-Level Overview	9
• Detailed Overview of Client/Server/Database Architecture	9
• Sequence of Events Overview	10
Design Issues	11
• Functional Issues	11
• Non Functional Issues	13
Design Detail	16
• Data Classes Diagram	16
• Description of Classes and Models	17
• Database Design	18
• Sequence Diagrams	19
• UI Mockups	24

Purpose

Exercise is an essential component for a healthy mind and body. However, working out today, given the current pandemic, has become hard and lackluster. This is because physical and social isolation can cause lethargy and a lack of motivation. People seeking to improve their physical fitness have varying ranges of experience when it comes to exercise, and tracking one's exercise goals has become tedious and cumbersome.

Tech companies like Apple and Google have created apps such as Apple HealthKit and Fitbit, which track and monitor health data. These technologies, while great for collecting data such as calories burned or steps taken, don't have the option of creating and tracking workouts. One of major motivational elements of exercise is being to see progress, and to be able to share it with friends.

The main purpose of this project is to design and develop a cross-platform application that makes working out from anywhere easy. *Werk It* provides an easy way to customize workout plans and communicate one's progress with friends. Not only does it track progress through previous workouts, it also monitors and collects health data. *Werk It* is unique from its competing products such as Apple HealthKit and Fitbit in its adaptability and ability to personalize workouts and statistical graphs.

Functional Requirements

1. Users can create a *Werk It* account or login.

As a user,

- a. I would like to be able to register for a *Werk It* account on the web app.
- b. I would like to be able to login to my *Werk It* account on the web app.
- c. I would like to be able to sign up on the mobile app.
- d. I would like to be able to login to my *Werk It* account on the mobile app.
- e. I would like to have face ID login once my credentials are saved.
- f. I would like to have touch ID login once my credentials are saved.
- g. I would like my password to be reset if I forget it.
- h. I would like my device to remember me until the next time I log out.

As a developer,

- a. I need to display an account creation form if the user does not have an account.
- b. I need to display an error message if at least one of the user's credentials is incorrect.
- c. I need to display a password/email reset form if the user has forgotten their password/email.

2. Users can easily access the application and manage their profile page.

As a user,

- a. I would like to be able to easily access the *Werk It* landing page.
- b. I would like to be able to easily access *Werk It* from my mobile device.
- c. I would like to be able to easily navigate the dashboard.
- d. I would like to be able to set my profile picture.
- e. I would like to be able to view my app history on the web app.
- f. I would like to be able to have my data persist within the mobile app.

3. Users can set up and customize their workout routines at any time.

As a user,

- a. I would like to be able to select the type of workout I plan to do (lifting, running, swimming, etc.).
- b. I would like to be able to select individual types of lifts.
- c. I would like to be able to create a new type of lift if it does not already exist.
- d. I would like to be able to set the duration of my run.
- e. I would like to be able to set the speed for my run.
- f. I would like to be able to set the number of laps for my swim.
- g. I would like to be able to set a custom type of workout.
- h. I would like to be able to set the exercises for my personalized workout.
- i. I would like the option to choose set/reps/weight for my custom workout.
- j. I would like to be able to create my own workout plans.
- k. I would like to be able to set a workout schedule for the week.
- l. I would like to be able to choose workouts that help me achieve my body goal.

4. Users can record their workout and food consumption data on the mobile application.

As a user,

- a. I would like to be able to input the number of sets I did for each type of lift.
- b. I would like to be able to input the weight I did per set of each type of lift.'
- c. I would like to be able to keep track of my caloric intake.

5. Users can view graphical representations of their personal workout data on the web application.

As a user,

- a. I would like to see my workout time per week on a histogram.
- b. I would like to see a line graph comparing my workout time with my friends.
- c. I would like to see a progress bar at the start of the week indicating how much of the weekly goal is accomplished.
- d. I would like to be able to easily comprehend the visualizations generated based on my workout statistics.

6. Users can receive reminders and recommendations based on their workout patterns.

As a user,

- a. I would like to receive workout suggestions based on my activity.
- b. I would like to be reminded if I am inactive for prolonged periods of time.
- c. I would like to be reminded when I have an upcoming workout scheduled.

7. Users can compete and communicate with others online

As a user,

- a. I would like to be able to connect with my friends.
- b. I would like to be able to send my workout plan to my friends.
- c. I would like to be able to send fitness challenges to my friends.
- d. I would like to post my activities to social media applications.

8. Users can use additional features of the mobile app and web app that were built to enhance the user experience and engagement.

As a user,

- a. I would like to see a motivational quote whenever I access the app.
- b. I would like to be able to stay motivated through a workout streak counter.
- c. I would like to be able to view the web app in dark mode.
- d. I would like to be able to view the mobile application in dark mode.
- e. I would like to be able to connect my music streaming platform.
- f. I would like to have music that syncs to the rhythm of my workouts.
- g. I need to display a loading symbol if requests take longer than a second so that the user does not think the app has frozen.

Non Functional Requirements

1. Architecture Requirements

As a developer,

- a. I would like to have the startup landing page be built with Bootstrap, describing the product intent, purpose, and to attract potential users.
- b. I would like to use React Native technologies to create flexibility between the web application and the mobile application.
- c. I would like to use Node.js as the backend technology, which connects to a Firebase that stores the user information.
- d. I would like to create a comprehensive workout status page for individual users by displaying graphical visualizations using D3.js.
- e. I would like to use Restful APIs to enable users to share their accomplishments on social media.

2. Performance Requirements

As a developer,

- a. I would like both the web app and the iOS app to support 100 concurrent users.
- b. I would like both the applications to support 1000 simultaneous requests to the database.
- c. I would like to limit the fetch request of an API to have a response time of no more than 500 milliseconds.

3. Security Requirements

As a developer,

- a. I would like to encrypt the users' profile information and login credentials when stored on the database.
- b. I would like to use a hash function with a hash table of length 31 to encrypt the user information before storing it in the database.

4. Usability Requirements

As a developer,

- a. I would like to develop a modern and simplified user interface for the web application.
- b. I would like to present the information in the visualizations and landing page in manageable chunks.
- c. I would like to follow a standard color scheme for the frontend web pages.
- d. I would like the web application to be dynamic and resizable across screens that are 768 px (phone), 992px (tablets) and 1200 px (large computer) wide.
- e. I would like to use exception handlers to handle server errors in our backend.
- f. I would like to follow the same usability guidelines that was used for the web application for the mobile application.
- g. I would like both the web application and the iOS application to be accessible 24 hours a day and 7 days a week.

5. Testing and Hosting Requirements

As a developer,

- a. I would like to use Git as the version control and GitHub to host the codebase for both the applications.
- b. I would like to use tools such as Postman and DigitalOcean to diagnose any backend issues during the development process.

6. Deployment Requirements

As a developer,

-
- a. I would like to deploy the web application on GitHub pages that is linked to our repository on GitHub.
 - b. I would like to run the mobile application using Expo CLI to run locally on an emulator or physical device.

Design Outline

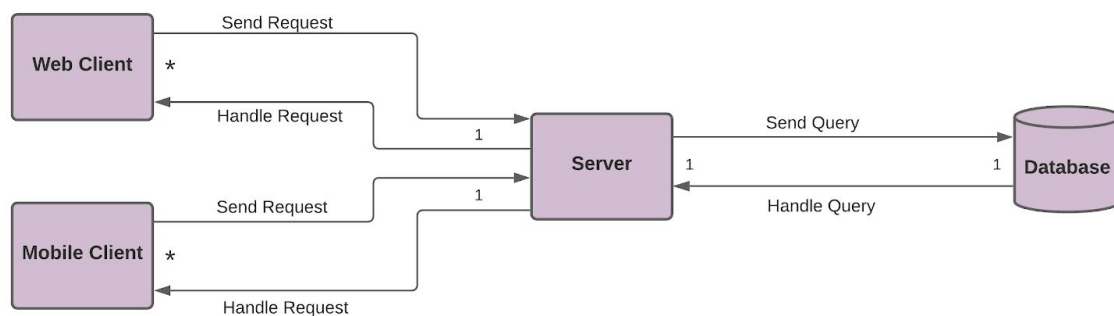
This project is a cross-platform application that tracks and stores workouts and progress along with user health data, which is used to generate graphical visualizations for display. The project has two main components, a mobile and web application. The mobile application operates on a Client-Server model, where the mobile client collects the data and sends it to the server to make a database transaction. The web application operates on the same Client-Server model, where the web application makes a server request for data from the database, and then displays that data in a visual format. Although the mobile and web applications have their separate clients, they are connected to one server which links to a central database.

Components

1. iOS Client
 - a. The iOS client will be the mobile interface for our system and how the user will interact with the system.
 - b. The iOS client will send and get data from our web server using HTTP requests.
 - c. The iOS client communicates with the database to track and record user workout data.
2. Web Client
 - a. The web client will be the web interface for our system and how the user will interact with the system via the internet.
 - b. The web client will display the user workout statistics via graphical visualizations using D3.js.
 - c. The web client will also depict the leaderboard for each user, once the user logs in, comparing the user's workout activity with that of his or her's friends.
3. Server
 - a. The server will accept HTTP requests from mobile and web clients.
 - b. The server will query the database using an HTTP request for information requested from the client.
 - c. The server will return the data requested to the respective client in JSON format.
4. Database
 - a. A relational database that will store all the information for the cross-platform application.
 - b. The database will respond to any queries from the server and send requested data back to the server.
 - c. Typical data stored in the database include user profile, workout data, and etc.

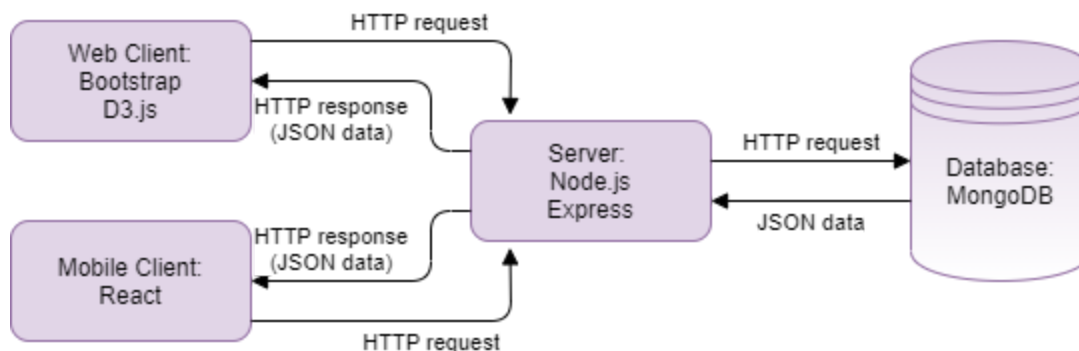
High-Level Overview

This project will have a many-to-one client server architecture. The client will send requests to the server whenever new workouts need to be created and when workout data needs to be inputted. The server will accept requests from the client and query the database for information. The server will also write data to the database when information needs to be stored. The server will be updating visualizations upon each rendering of the web application.



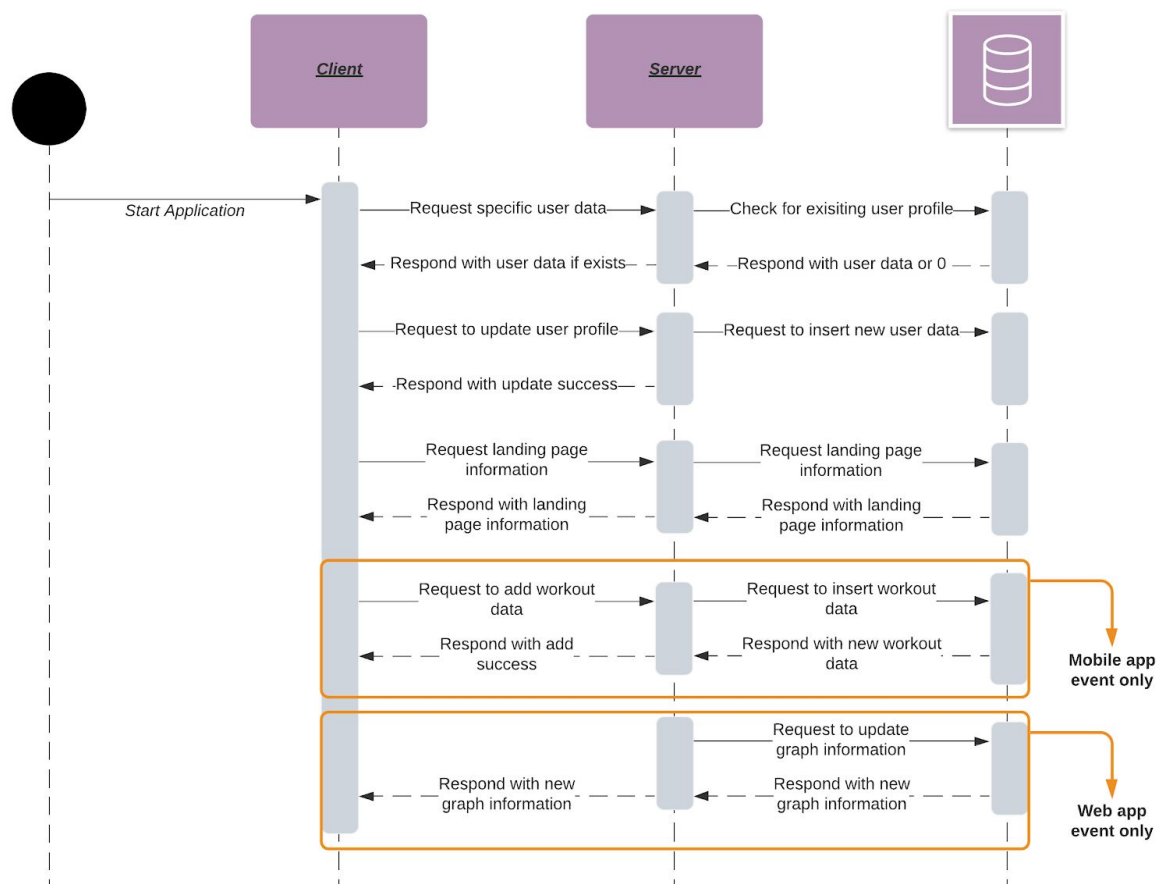
Detailed Overview of the Client/Server/Database Architecture

The web client, mobile client, and server will all make requests and respond to requests using the HTTP protocol. Data responses will be sent in the JSON format, which the database will also comply with. The web client will be made primarily using Bootstrap and the data visualizations will be created using D3.js, while the mobile client will be made using React. The server, made of Node.js, will contain the API to make requests to the database and handle HTTP requests from the clients, specifically through using the Express framework, where our database will be hosted by MongoDB. The server will be listening for requests through a local, private port.



Sequence of Events Overview

This diagram shows the typical sequence of events when using this cross-platform application. The sequence begins when the user starts the web or mobile application. When the user logs in, the respective client sends a request to the server which will handle the request. The server will send a query to the database and receive the data which will be sent back to the client where the client will respond with the appropriate action for the user profile. Once logged in, the client will send a request for generating the landing page data, and the server will send a query to the database to acquire the appropriate data to display. If the application was accessed through the mobile app, the client can also create and input workout data to which the server will respond with an “add success.” If the application was opened through the web app, the client will be able to see graphical visualizations of workout data (updated upon each web app rendering).



Design Issues

Functional Issues

1. Are users required to login in order to access our application's features?
 - Option 1: No login credentials required
 - Option 2: Use Spotify, Google, or Facebook account to login
 - Option 3: Allow the user to create a unique username and password to login to our application.

Decision: Option 3

Justification: Setting up an account with a username and password is essential to identify a user. More importantly, in order to allow a user to reset their password, it is pertinent to verify the identity of the respective user, and keeping track of the username and password is the most optimal way to do so. Moreover, using Facebook, Google or other third party services is problematic because sometimes their servers are down, and it is rather impossible to predict whether they are able to reach the user in time to allow the user to reset his or her's password.

2. How long should the workout data be tracked for each user?
 - Option 1: For as long as possible
 - Option 2: For 1 year
 - Option 3: For 3 months

Decision: Option 2

Justification: Tracking data as long as possible produces the issue of overloading the database, and the unknown amount of time can make it difficult to display comprehensive data visualizations. Tracking the data for 3 months prevents the database overloading, but does not provide the long-term data review promised by the application. Tracking the data for one year is a compromise between the two previously mentioned, allowing the application to provide yearly reviews of the user's progress while not saving too much data to the database.

3. How can we effectively update graphical visualizations?
 - Option 1: After a workout is created, automatically update all visualizations
 - Option 2: Update graphical visualizations every time the web app is rendered
 - Option 3: Automatically update all visualizations every 24 hours

Decision: Option 2

Justification: Every time the web app is opened or refreshed, it should automatically update all graphical visualizations in order to ensure the user the data was collected. This makes most logical sense since a user expects to see real-time data updates to check their progress. It would not be useful if the user had to wait 24 hours to see updates on their goal progress, and updating visualizations after every workout is created will be extra work for the server. Therefore, it will be simplest that whenever the web app is refreshed, update all graphical visualizations.

4. When should the mobile app send the collected workout data to the database?
 - Option 1: Every time a new data value is logged.
 - Option 2: At the end of the workout.
 - Option 3: Whenever the user closes the app.

Decision: Option 2

Justification: The mobile app should make a database transaction to store collected data at the end of a workout. This option is the best because it keeps the number of database requests to a minimum since all the data will be grouped together. Sending data every time a value is logged would result in an excessive number of database calls, which could potentially cause issues with runtime since database transactions are slow. It would also be problematic to send data when the user closes the app because that runs the risk of data being lost before the transaction is completed. For these reasons, it makes most sense to send the workout data at the end of each workout.

5. What should be done when the 1-year database limit on user data has been reached?
 - Option 1: Discard the current data in the database
 - Option 2: Store the data in a compressed form in the database
 - Option 3: Give the user the option to download the data onto their device

Decision: Option 3

Justification: Discarding the current data would leave the user unhappy if they still wanted to have access to that data after the 1-year period, which rules out Option 1. Storing the data in a compressed form in the database could potentially overload the database in some situations or make the database operations slow, which rules out Option 2. This leaves Option 3, which would give the user a choice in keeping their 1-year data on their device, while also freeing up the database. Therefore, the final decision here is to give the user the option to download the data onto their device.

Non Functional Issues

1. What frontend framework should our team use to develop the web application?

- Option 1: vanilla HTML + JavaScript
- Option 2: Bootstrap
- Option 3: Foundation by ZURB
- Option 4: Semantic UI

Decision: Option 2

Justification: First of all, using a framework would help the reusability of our code and save time during the development process. Hence, using option 1 would not be a feasible approach to follow. Given that a framework must be used, Bootstrap was undoubtedly the best framework because Bootstrap is a popular and widely used framework in industry, with extensive documentation and examples. Moreover, Abhishek Gunasekar, one of the team members, has prior experience in developing web applications using Bootstrap, and since he's handling the front-end aspect of the web app, we decided as a group that Bootstrap was the most logical and wise choice.

2. What backend framework should our team use to develop the web application?

- Option 1: Firebase
- Option 2: MongoDB Realms

Decision: Option 2

Justification: MongoDB Realms allows communication between clients and the backend to be done in JavaScript; in addition, queries can be done in JavaScript instead of SQL, which simplifies the implementation efforts required during the Sprints. MongoDB Realms also provides files to design the User Interface for both Web and Mobile clients, which would make development easier by having everything in one place. With Firebase, we would have to do most of this manually, and would have to set up a web and application server separately, which would take more time to implement, making MongoDB Realms the most appropriate choice.

3. What platform should our team target for the mobile client?

- Option 1: iOS
- Option 2: Android

Decision: Option 1

Justification: Given that our team decided to use MongoDB Realms as the backend framework, we would be easily able to target either iOS or Android. Since the majority of team members have a mobile device running iOS, we have decided that iOS would be the best initial platform to target.

4. What development environment or tool should our team use?

- Option 1: Vim
- Option 2: Visual Studio Code
- Option 3: Eclipse EE

Decision: Option 2

Justification: Our team has worked with all of these development tools in some capacity. Our project has multiple components to it (e.x. MongoDB Realms, clients, backend), which would not make Vim a suitable choice due to the difficulty in managing these multiple parts at the same time. Since the project isn't at the level of an Enterprise Application, Eclipse EE would not be suitable due to the long download and installation time, and also the various workstation and directory setups (not needed due to the small size of our project). This leaves Visual Studio Code, which is lightweight, has built in JavaScript support, and has extensions for using other languages and technologies, which makes this our final choice.

5. What technology should we use to generate workouts?

- Option 1: Randomly combine all of the activity types that the user has logged
- Option 2: Get suggestions from friends or professionals who work out
- Option 3: Collect data on the relationship between workout type, time vs health (an aggregate of health data), resulting in a function $f: R^2 \rightarrow R$ such that $f(\text{workout type}, \text{time}) = \text{health}$. Optimize this function using a technique such as Lagrange Multipliers, in a way that maximizes *health*

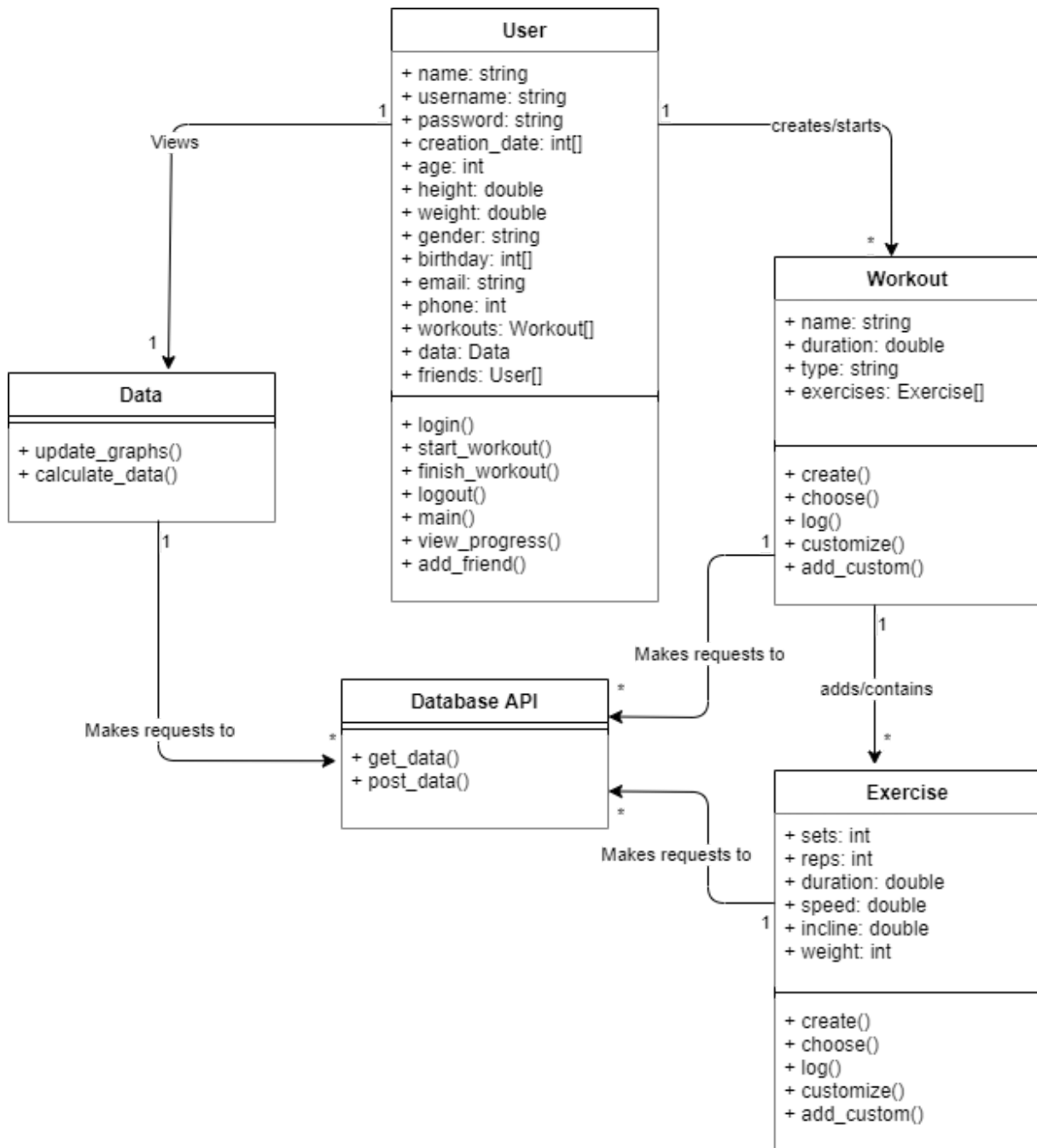
Decision: Option 2

Justification: A random combination of all activity types that the user has done would not be specific to the user's needs, and would not guarantee overall consistent improvements, which makes Option 1 an unsuitable choice. Collecting data, performing multivariable regression, and optimizing the multivariable function would be a very involved task; furthermore, detailed research would need to be done in order to account for aspects such as lurking and confounding variables. Therefore, Option 3 would be a good feature to implement in the case of extra time, but is practically not suitable for the

expected time frame of the project. This leaves Option 2, which would be easy for everyone to do (e.g. send out a survey), and would give us access to more relevant data while not making major assumptions about the relationships between the variables. Therefore, the final decision is Option 3.

Design Detail

Data Classes Diagram



Description of Classes and Models

- **User**

- A user is created when someone signs up for our application.
- A user will have a unique username and password combination to log in with.
- The user can complete their profile by filling in their age, height, weight, gender, birthday, email, and phone number.
- Each user has their own list of workouts, which can be viewed in both the mobile and web app, but edited only in the mobile app.
- Each user has their own list of friends that they can add by username, and each friend is also an instance of a user.

- **Workout**

- A workout object will be created when a user creates a new workout in the User class
- Each workout will have a unique name to be identified by, and the creation of the workout will require the name to be unique
- Each workout will have a duration time, a type of workout, and an array of Exercises created by the user.
- Workouts can be selected, started, and logged to if they are already created.
- New workouts can be created and have exercises added to them.

- **Track Data**

- A data class will handle the data tracking and calculation for the user.
- Graphs and visualizations will be updated through the data class.
- The necessary data calculation will be done in the data class using the information from workouts and personal information from the user.

- **Database API**

- An API class will be responsible for communication between the database and the core program that is made up of these classes.
- The API routes will be handled in this class.
- Getting data as well as storing data to the database will occur in this class.
- Many of the other classes including Data, Workout, and Exercise will utilize this class for the functionality mentioned previously.

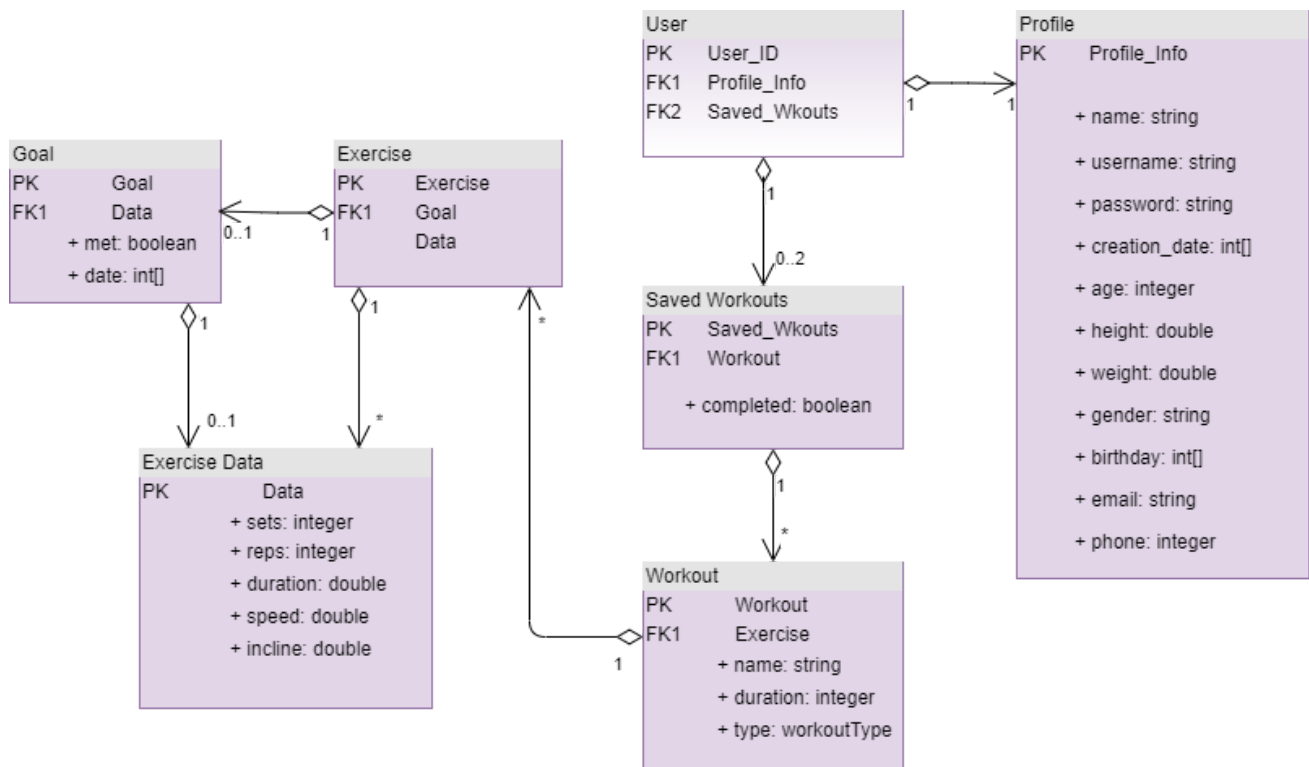
- **Exercise**

- An exercise is created when the user introduces a new exercise into the database.
- An exercise can have any of the following: sets, reps, weight, duration, speed, or incline associated with it based on the exercise.

- These exercises are added to a list of exercises held by a workout.

Database Design

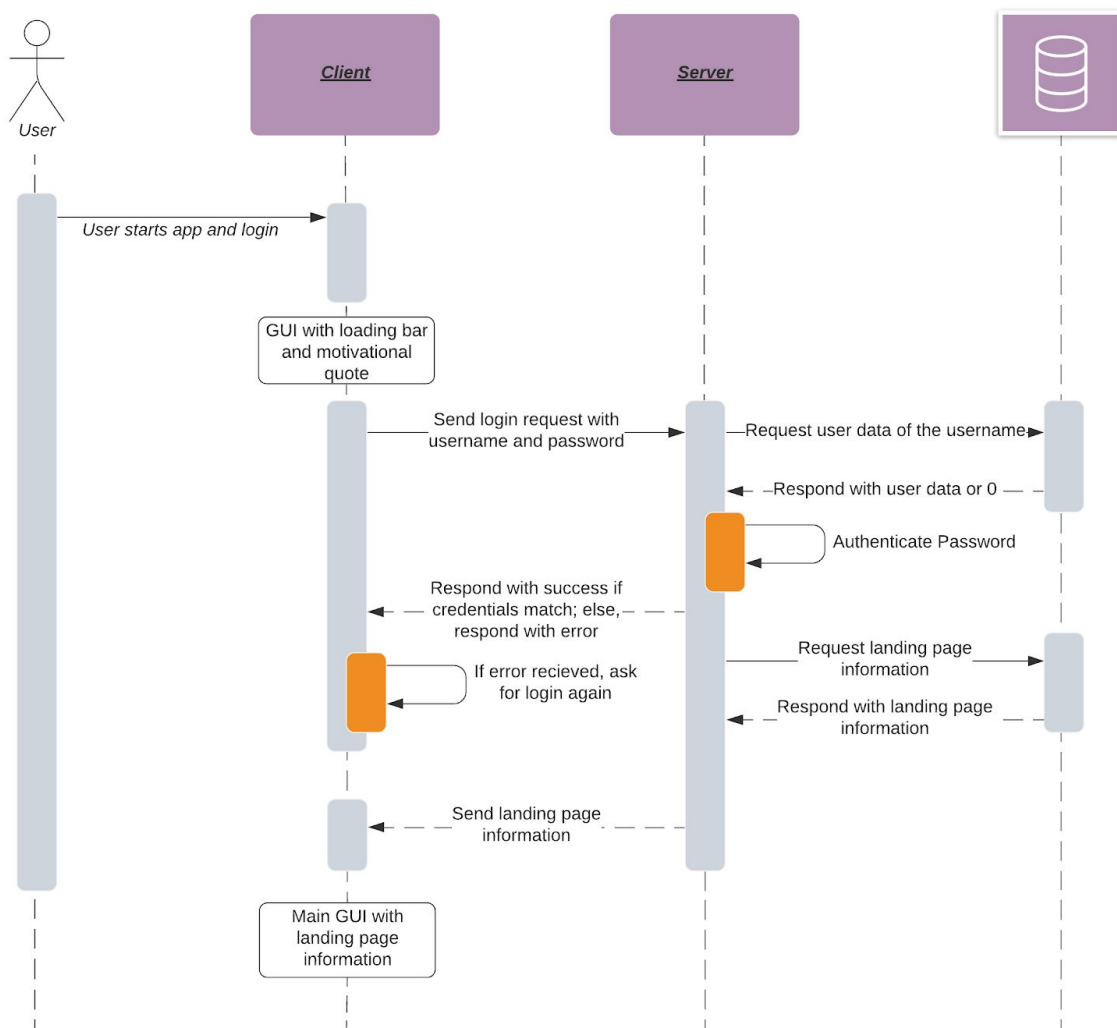
The database is a particularly important part of this application because it is responsible for tracking the data to be reported by the web app, which is a highly marketed feature of this application. The database will be implemented using MongoDB, as it is easily compatible with both Node.js and React. The tables in the database are related to the classes in the class design, with the root table being User. Tables are identified using a Primary Key, and related tables found under their Foreign Key. Express, a Node.js web app framework, will be utilized by the server to provide clean, efficient API routing to the database to request and store data.



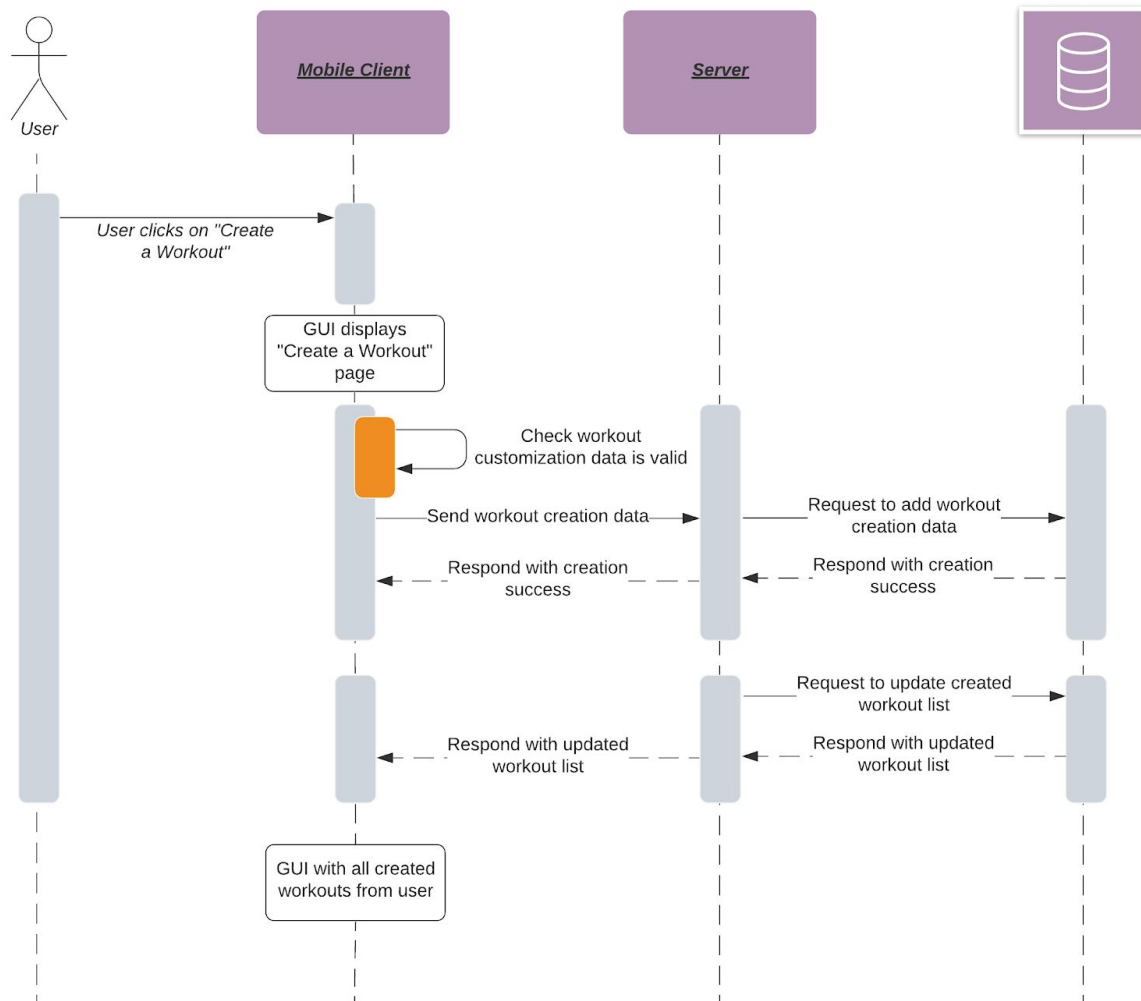
Sequence Diagrams

The following diagrams depict the major events of the cross-platform application. This includes user login/creating accounts, creating a workout, inputting workout data, updating graphical visualizations, and generating a leaderboard. This sequence shows how our application events are handled in the client-server architecture. The client will allow the user to perform actions that will be sent to the server to be handled and queried. The client will also display the appropriate UI.

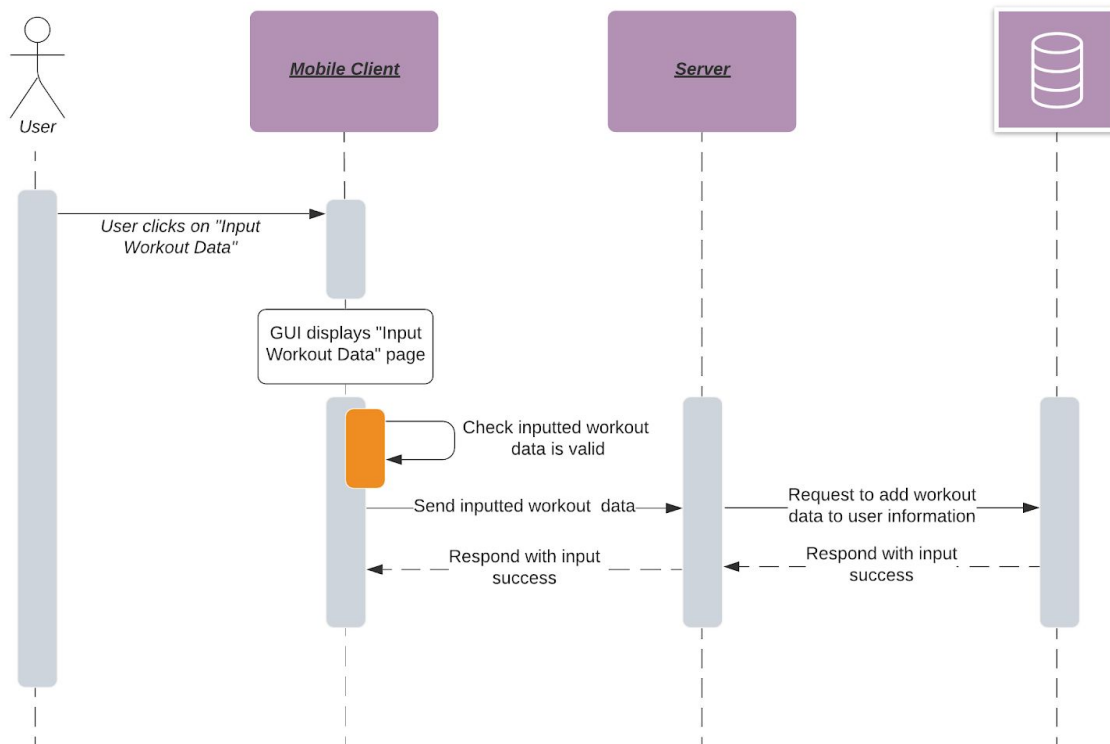
1. Sequence of events when the user starts the app/login



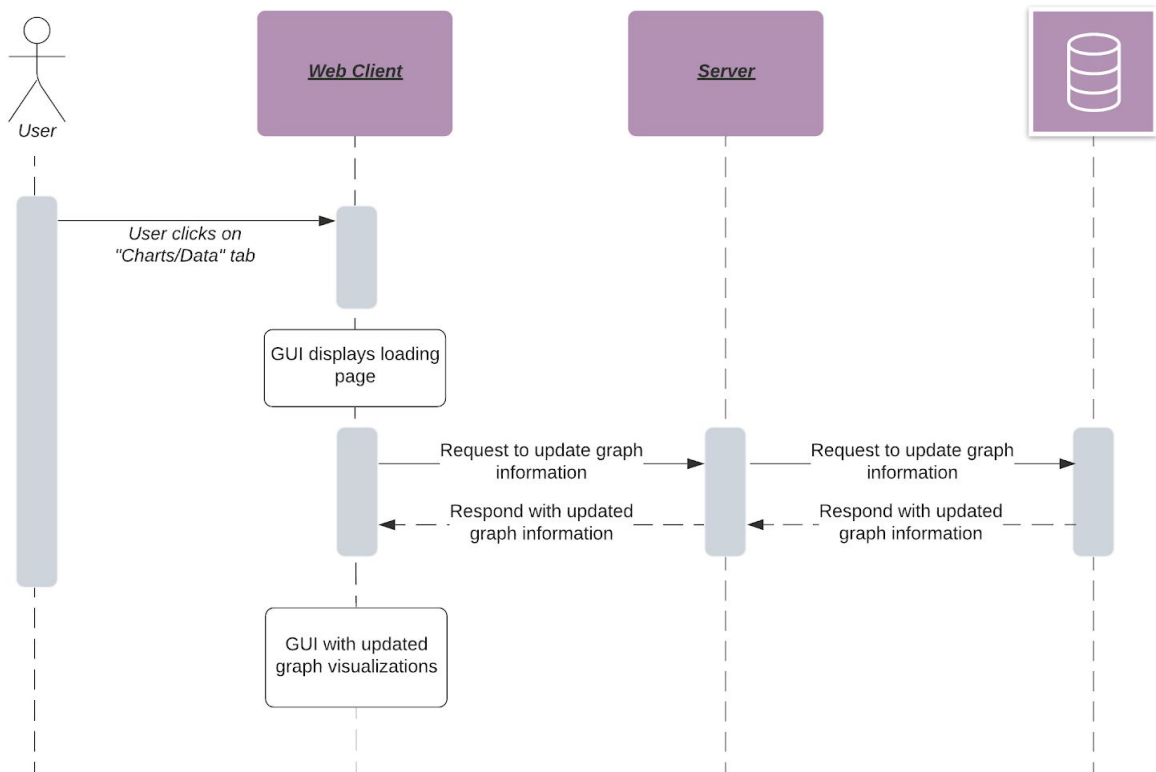
2. Sequence of events when the user creates a workout



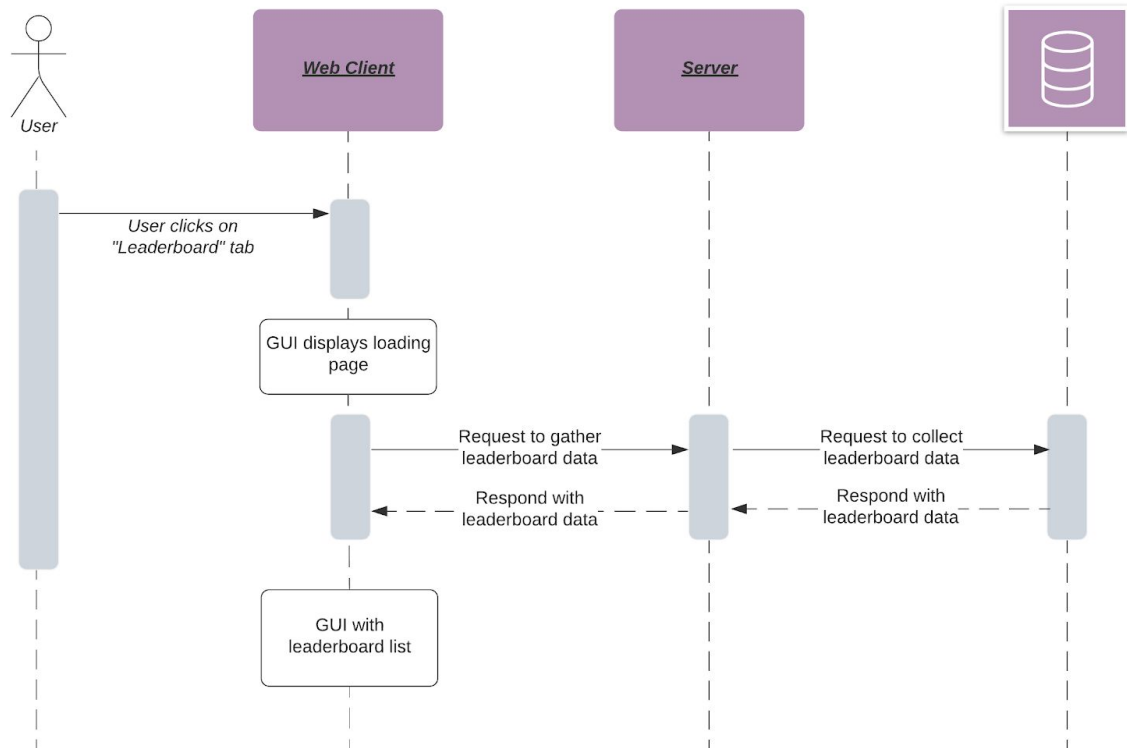
3. Sequence of events when user inputs workout data



4. Graphical visualizations are created/updated

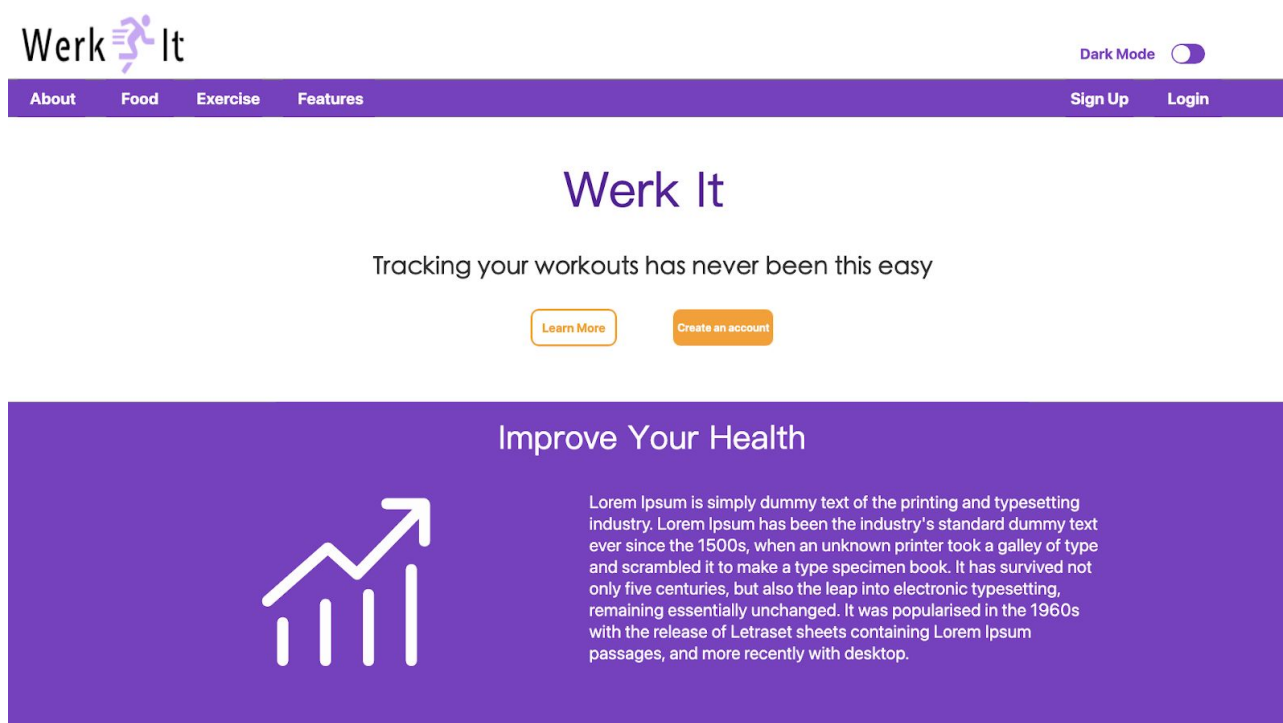


5. Generating leaderboard data

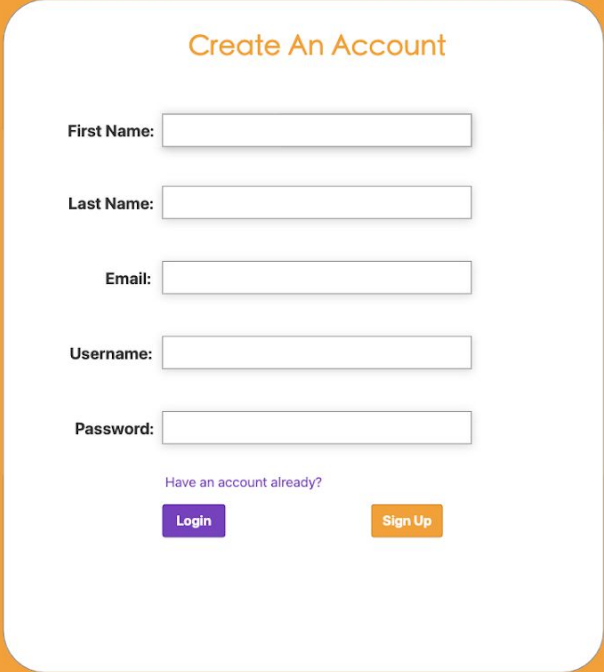


UI Mockups

Our product design includes a navbar that is present in all the webpages. This allows the user(s) to toggle between important features of the application such as signing up or logging in. The navbar also provides an optional switch that the user(s) can use to switch between dark mode and light mode, a feature that we hope to implement in our later sprints. The logo present at the top left of the navbar can be used to quickly access the homepage of our application.



This is the landing page of our web application. This is where the user(s) can learn more about the features of our application and decide to create an account or login.

A screenshot of a web application's 'Create An Account' form. The form is centered on a solid orange background. It features a white rounded rectangle containing the title 'Create An Account' in orange text. Below the title are five input fields, each preceded by a label: 'First Name:', 'Last Name:', 'Email:', 'Username:', and 'Password:'. At the bottom of the form, there is a link 'Have an account already?' in purple text, followed by two buttons: a purple 'Login' button and an orange 'Sign Up' button.

Create An Account

First Name:

Last Name:

Email:

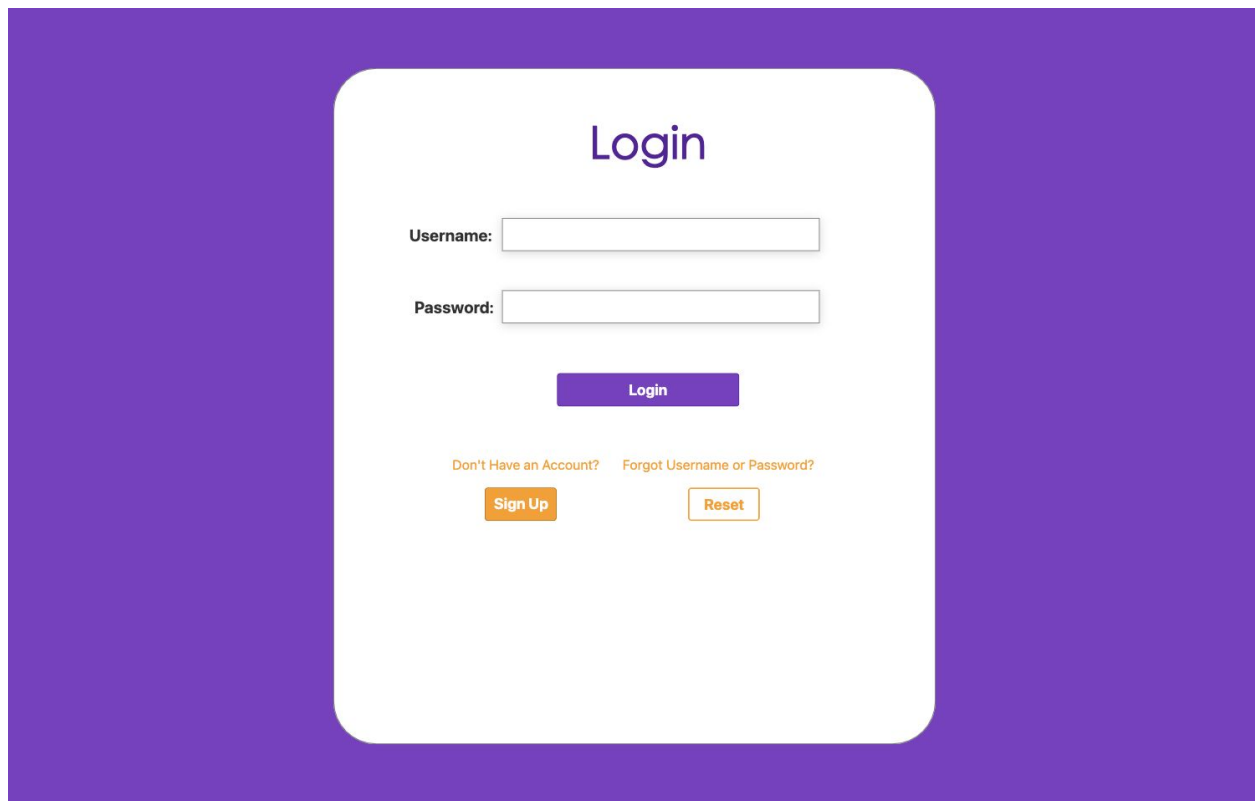
Username:

Password:

[Have an account already?](#)

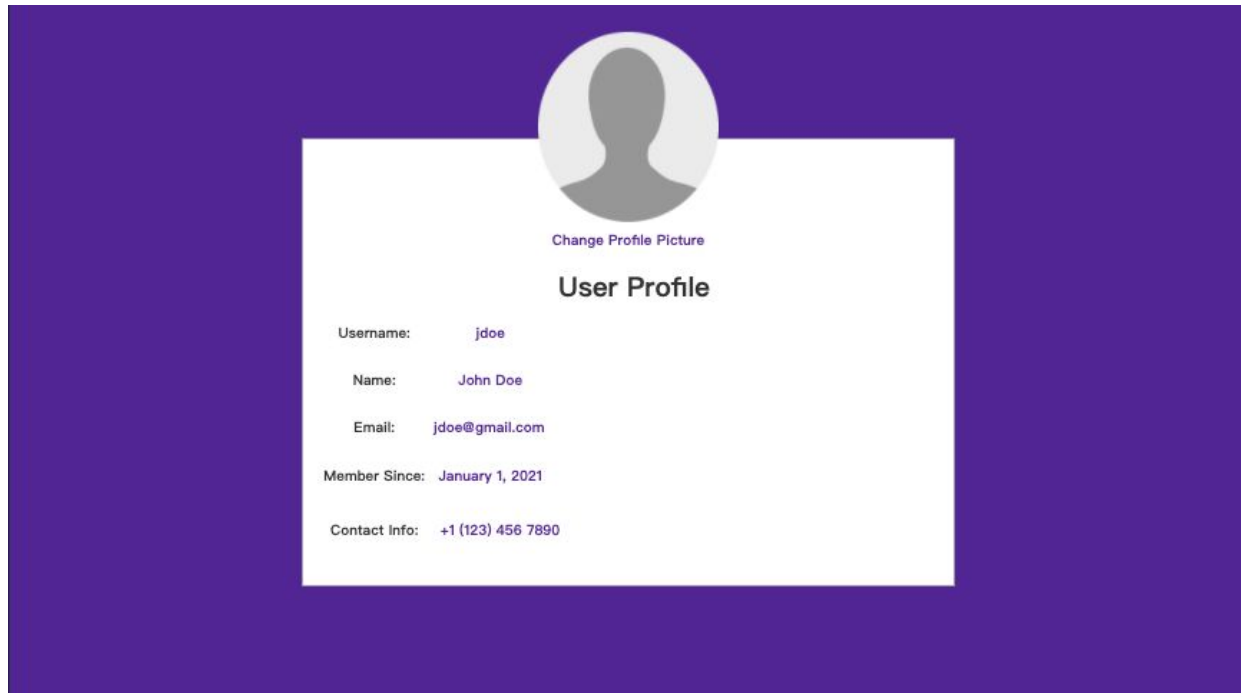
[Login](#) [Sign Up](#)

This is the sign up page for the web application that requires a new user to enter their first name, last name, email, username, and password. The bottom left button allows the user to login if he or she has a *Werk It* account already, or choose to sign up. An additional feature that our team hopes to implement is to include an additional check whether a particular user decides to sign up twice. If a user decides to sign up twice with the same email, then our application shouldn't allow him or her to do so. The mobile application will have a sign up page with a similar layout.



The image shows a login page for the 'Werk It' web application. The page has a solid purple background. In the center, there is a white rounded rectangle containing the login form. At the top of this rectangle, the word 'Login' is written in a large, dark purple font. Below it, there are two input fields: 'Username:' followed by a white text box, and 'Password:' followed by a white text box. Under the password field is a purple button with the text 'Login' in white. Below the 'Login' button, there are two links in a smaller, orange font: 'Don't Have an Account?' and 'Forgot Username or Password?'. Under the first link is an orange button with the text 'Sign Up' in white. Under the second link is an orange button with the text 'Reset' in white.

This is the login page of the *Werk It* web application. There is a reset button for the user to reset their username or password in case they forgot it. There is also a sign up button for the user who has not signed up for the application yet. The mobile application will also have a login page that looks very similar.



This is the profile page that shows up after a particular user has signed up for our application. The user at any point can change his or her's profile picture, username, name, email, and contact information.



Dashboard

Weekly Goal



Leaderboard

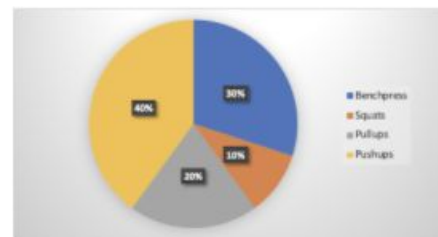
Exercise:

Rank	Person	Repetitions
1	You	100
2	Friend 1	95
3	Friend 2	80
4	Friend 3	50
5	Friend 4	40

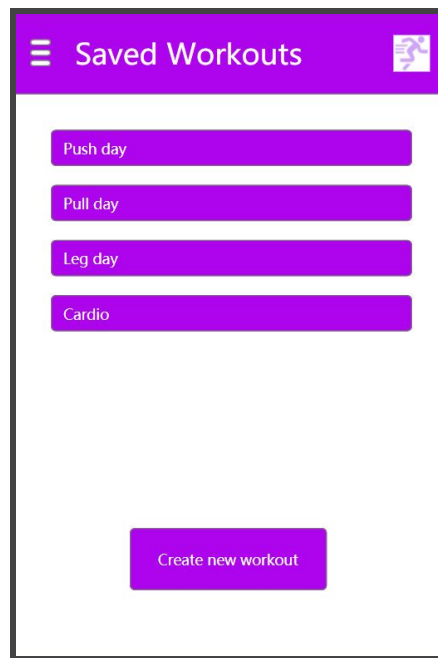
Choose Friend to Compare



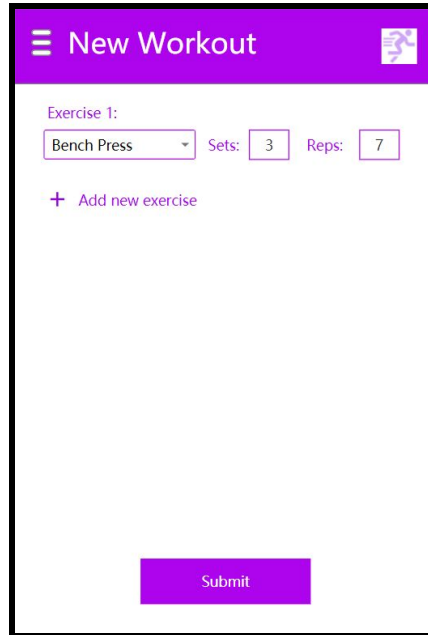
Workout Routine by Exercise



This is the main dashboard page that displays the workout data corresponding to each user. The dashboard will house the leaderboard that compares the user's performance with his or her's friends, a progress bar indicating how far the user has progressed in achieving the weekly goal, and other sophisticated yet beautiful visualizations that illustrate the user's workout patterns and routines.



This is the saved workouts page that keeps a record of all the workouts the user has created and done before. These workouts can be reviewed, edited, and deleted at the user's discretion.



New Workout

Exercise 1:

Bench Press Sets: 3 Reps: 7

+ Add new exercise

Submit

This is the workout editor page, which will probably be the most often used page on the mobile app. Here the user can create their own workout by selecting exercises from a list.