

# Bamboo

## Design Document

### Team 2

Ruchira Goel, Ramya Mandyam Anandampullai, Linnea Lindstrom, Hajera Zemy

### Contents

#### Purpose

Introduction	2
Functional Requirements	3
Non-functional Requirements	5

#### Design Outline

High-level System Overview	6
Sequence of Events Overview	7

#### Design Issues

Functional Issues	8
Non-functional Issues	12

#### Design Details

Sequence Diagrams	16
Class Diagram	19
Activity Diagram	23
UI Mockups	24

## Purpose

Many people are interested in keeping track of their eating, exercise, and sleeping habits to monitor their progress with various health goals. While there are many existing health and lifestyle apps, they each have their drawbacks, including having a narrow set of features, a steep learning curve, or premium paywall. Existing apps that help monitor health goals are often specific to one aspect of health, like exercise, making it more work for the user to keep up with multiple platforms and enter a lot of data each day. This is time consuming and inconvenient, leaving most users frustrated with the options currently available.

Bamboo will be an all-in-one mobile application to keep track of your health and wellness-related habits and find out the best way to go about achieving your goals. Users can set multiple highly customizable goals and track how well they are meeting these goals on a daily, weekly, or monthly basis. In addition to this, Bamboo will be tailored to the individual user's needs. There will be functionality to recommend meals and exercise routines catered to the user's health attributes and goals.

After reviewing the existing fitness goal apps, we also found many of them to be visually overwhelming, and with too many features for the casual user to handle. We are instead going for a simple and intuitive design, to make our user experience as smooth as possible.

We are designing a cross platform mobile application to serve this purpose. We will develop a React Native based front end, a MongoDB database, a NodeJS based back end, and a Spring Boot web service to service requests from the front end to the database.

## **Functional Requirements**

### **Account & Personal info**

As a user, I would like to...

1. Sign up for a Bamboo account
2. Set up my account profile with health characteristics specific to me
3. Be able to log in and manage account information
4. Be able to recover my account if I forget the password
5. Be able to access my account on various mobile devices
6. Be able to log out of my account
7. Be able to delete my account and all personal data from the application if needed
8. Be able to view my personal information
9. Be able to update my personal information as needed

### **Daily inputs**

As a user, I would like to...

1. Enter daily exercise information such as number of hours for each activity
2. Enter daily diet and meal information
3. Enter daily sleep information (if time allows)
4. Be able to enter information for previous days if I forgot to
5. Scan product barcodes for easy input of meal information (if time allows)
6. Choose menu items from nearby restaurants for easy input of meal information (if time allows)
7. Add regular workout routines or common meals to a saved list, to save time inputting the same information again on a later date
8. Be able to complete daily input in a short period of time

### **Goals**

As a user, I would like to...

1. Set goals for minutes/hours of daily or weekly exercise
2. Set goals for daily calorie intake
3. Set dietary goals regarding certain nutrients, such as the amount of sodium, or fat in a daily diet
4. Set goals for going to sleep at a specific time (if time allows)
5. Set goals for waking up at a specific time (if time allows)
6. Set goals for getting a certain amount of sleep per night (if time allows)
7. View my current goals
8. Edit and update my current goals

9. Track my progress with my goals

### **Analytics & Recommendations**

As a user, I would like to...

1. Check analytics over time
2. View a graphical representation of exercise over time
3. View a graphical representation of calories consumed over days
4. View weekly/monthly reports containing diet and exercise data and goal progress
5. Find recommended meals that are in line with my set goals
6. Find recommended meals with specific tags, such as high protein or low sodium
7. Set dietary restrictions and allergy information for meal recommendations
8. Find recommended exercise routines to speed up progress towards set goals

### **Notifications & Settings**

As a user, I would like to...

1. Be able to choose my system of measurement (metric or imperial)
2. Receive notifications as reminders to input my exercise or dietary information
3. Receive reminders to exercise on days I consistently forget
4. Receive reminders of my goals based on other behaviour patterns (if time allows)
5. Receive notifications with positive encouragement, such as how long of a streak I'm on of achieving goals
6. Edit notifications settings as needed

### **Development**

As a developer, I would like to...

1. Improve exception handling and produce proper messages for errors
2. Have thorough test cases that test functionality and possible edge cases
3. Ensure the security of the users' data and passwords

## **Non-functional Requirements**

### **Usability**

1. Minimize manual input from the user
2. Simple and intuitive user interface so that all users find it easy to navigate through it
3. Cross-platform application to ensure that people can access it from all mobile devices
4. Track different types of data like exercise and diet, instead of only one aspect of lifestyle

### **Security**

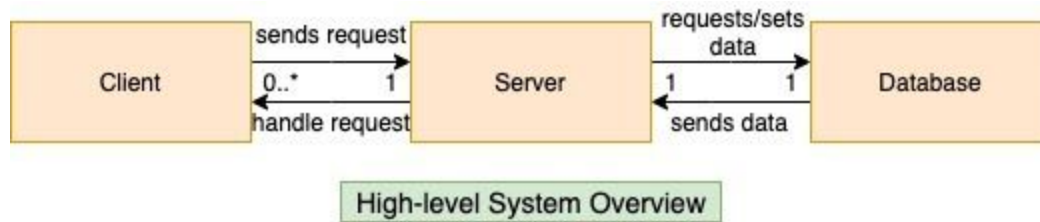
1. To ensure a secure platform, a user will not be able to access the app without being logged in
2. Passwords will be encrypted while handling requests and in the database so that they are not visible to developers and other users
3. Users will be provided with a choice on whether they want to allow location tracking

### **Response Time & Scalability**

1. Application launch time, i.e. time that is required for the UI of the application to load, should be less than 10 seconds
2. Any response to a user action should be completed in less than 1 second
3. For any responses that take longer than 10 seconds, the application will show a loading screen or a progress bar that shows how long the action will take so that users will know what to expect
4. Aim to have Bamboo be able to handle at least 100 simultaneous requests

# Design Outline

## High-Level System Overview

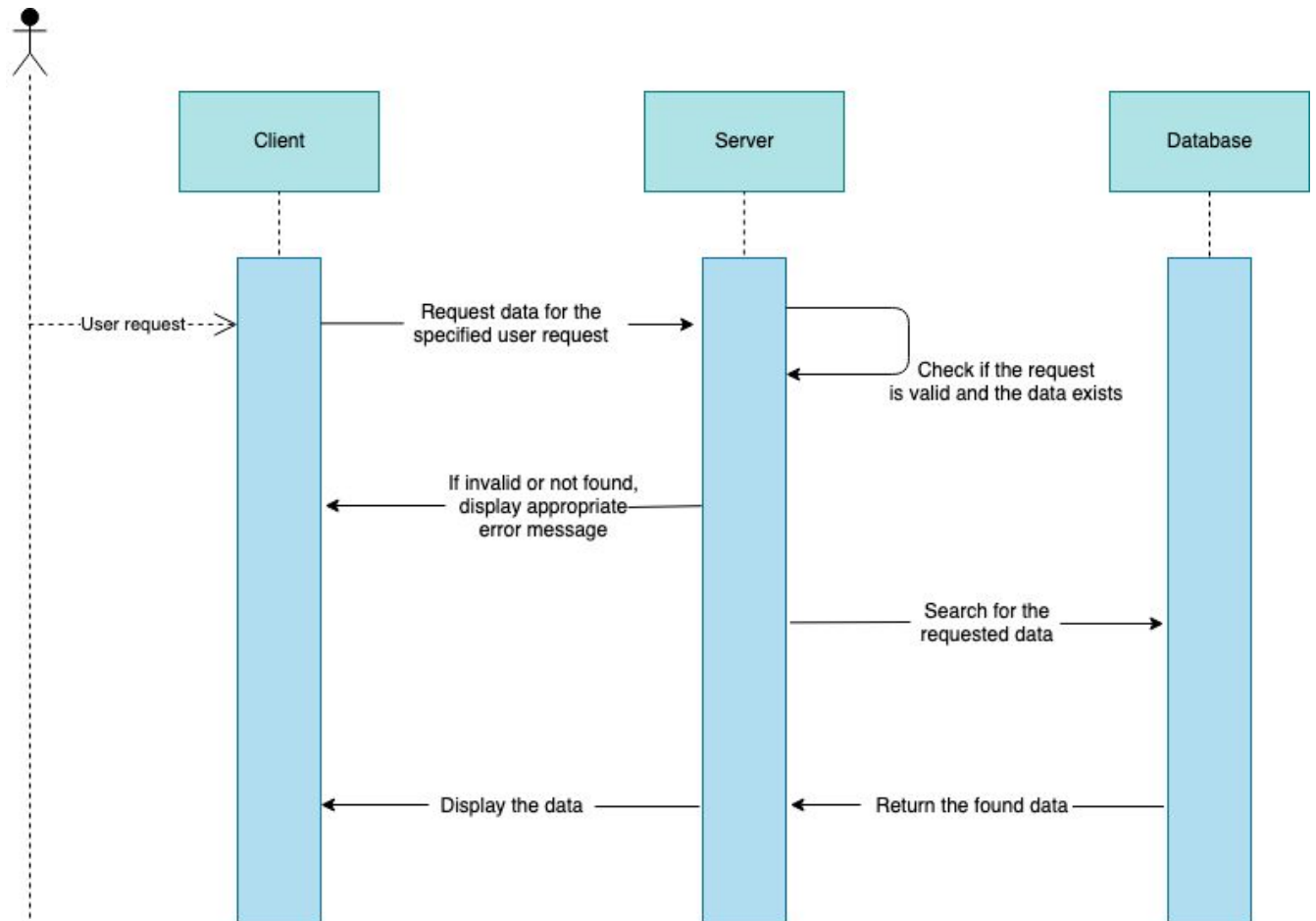


Our system will use the client-server model, comprised of these components:

1. Client (React Native)
  - a. Displays the frontend, providing a user interface
  - b. Send requests to the server and receives responses
  - c. Renders the display
2. Server (Spring Boot)
  - a. Receives and handles requests from the client
  - b. Send queries to the database to add, update, or request data
  - c. Forwards necessary data to the client
3. Database (MongoDB)
  - a. Stores user data
  - b. Handles queries by the server and sends requested data
  - c. Adds, updates, and deletes data as requested

## Sequence of Events Overview

The following figure illustrates the general outline of the involvement of the client, server and database when the application is being used:



# Design Issues

## Functional Issues

1. What information is needed when making an account?

Option 1: email and password

Option 2: username, email, and password

Choice: Option 1

Justification: A lot of apps that require accounts have the user specify a username, which they are henceforth referred to as, and use to log in. Most of the people on the team felt that remembering usernames was almost always a hassle, especially when the users can just log in with their email. Also, since the users would not be interacting with each other, there would not be the need for a username. Hence, we chose option 1.

2. What information is needed when logging in for the first time?

Option 1: personal information, at least one goal, notification settings

Option 2: personal information

Choice: Option 2

Justification: Option 1 would be good to help the user set up the app and tailor it to their preferences right away, as well as introduce them to features they may not have known about. However, requiring too much information to be inputted right away can be tedious. Option 2 would minimize time in setting up the app, and allow the user to be able to start exploring it as soon as possible. Personal information, which includes age, sex, height, and weight, should be enough to get started on using the app. Also, the user may not have a specific goal in mind yet, or know what to expect from notifications. In its most basic form, Bamboo can be used for just tracking daily data, and does not need any goals from the user.

3. What information can users input for food?

Option 1: all nutrition facts (exact weight and numbers for calories, protein, fat, carbohydrate)

Option 2: Meal information (for eg. one serving of pasta with alfredo sauce)



Option 3: Product barcodes

Option 4: Select from nearby restaurants' menus

Choice: Option 2 (also 3 and 4, if time allows)

Justification: It would be very convenient for us as the developers to go for option 1, as we can just take what the user inputs and not perform any additional calculations. However, it would be unreasonably difficult for the user to calculate the exact number of calories, or the exact number of grams of carbohydrates in their meal. This approach would be an inconvenient and unusable user experience. Options 3 and 4 seem very appealing from the user's perspective, as it minimizes time spent on input and allows them to select exact items that they ate. However, just these options do not account for users that cooked their meal. Also, these features would be technically challenging for us to implement, and we were not sure our timeline would allow us to do this. Instead, we went for a compromise in option 2, where the user does input some information about what they ate, but not in excessive detail. We then calculate the average nutritional info for the food items they entered. We plan on implementing options 3 and 4 if time allows, but our priority for the meal input section is option 2.

4. What information can users input for exercise?

Option 1: duration, activity

Option 2: duration, activity, calories burned

Option 3: combine activities to form a routine

Choice: Options 2 & 3

Justification: The most important factors in recording exercise data are - what the activity was, and the time spent doing it, which should be the bare minimum input. The initial concerns we had with option 2 were that users may not know how many calories that they burned, although it would be useful information for them, so we decided we will include it but make it an optional field. If the user chooses not to provide the calories burned, we can calculate a rough estimate of the amount of calories burned using the activity and number of hours. If they used an exercise machine (e.g. a treadmill), often it will display that information to them, and they could then easily input it into the app. The users will also be able to combine activities to form an exercise routine, which can be saved in the

app. This would be a useful feature for people who have a recurring exercise routine that they do.

5. How can we minimize daily input required from the user?

Option 1: save their previous inputs and allow them to select from the list

Option 2: simplify inputs (e.g. only name and number of calories for food)

Option 3: make certain inputs optional

Choice: Options 1 & 3

Justification: There is a tradeoff between minimizing the amount of input required (the benefits of option 2) and maintaining the accuracy of data (where option 2 falls short). If we simplify the inputs, we would not be able to provide very accurate and holistic data regarding diet and exercise to the users. We will instead not require the user to fill out every field, however the more information they can give us regarding what they ate and what exercise routines they did, the more beneficial our app will be to them. By saving their previous inputs and allowing them to select them again in the future, over time it will become more convenient to complete their daily inputs.

6. What kind of goals can users set?

Option 1: general health goals (like overall fitness/weight control)

Option 2: specific goals geared towards targeting specific health aspects

Option 3: no goals

Choice: Options 2 & 3

Justification: Option 1 would be the most personalizable for the users, but as developers, it will be quite difficult to have users set goals relating to general fitness or weight related concerns, because such goals would be too vague to be able to track. Another important thing to consider is that, such information should only be given out by licensed medical professionals. Since we will only be using information that is freely available on the internet and APIs for our application, Bamboo should be used specifically to help achieve targets, and not for self-diagnosis or as a substitute for medical treatment.

7. What notifications will the app have?

Option 1: reminder to input daily data

Option 2: reminder to input daily data, reminder to exercise, goal streak

Choice: Option 2

Justification: At first, we thought we would just have notifications to remind the user to input their daily food and exercise data, as we do not want to bombard them with notifications. Then we decided we would make notifications customizable and users could disable the ones they did not want, so there would be no drawback to offering notifications for multiple things, such as a notification for consecutive days of goal achievement. It would encourage the user to keep up with their goals.

## Non-functional Issues

### 1. What technology will we use for the frontend?

- Option 1: React Native
- Option 2: Angular
- Option 3: Flutter
- Option 4: Android Studio
- Option 5: Xcode

Choice: Option 1

Justification: Several members of our team have worked with Android Studio before, making option 4 seem like the right choice initially. However, this would only allow us to develop an Android application, and we want our app to be accessible to as wide an audience as possible. We thought we could use Xcode as well and make an iOS application too. Android Studio and Xcode have the advantage of being native technologies, meaning that it would be easy to have a native user interface and user experience for both platforms. However, developing our app separately for Android and iOS would mean a lot of duplication of code, and would likely require more time than we have for this project.

Instead, we started to explore cross platform technologies like React Native, Angular, and Flutter. With this approach we could work on a single codebase for both Android and iOS applications. Of these options, we ruled out Angular JS because it does not offer the same ease of making a native UI/UX. React Native and Flutter however are frameworks that allow us to create completely native apps like we would with Android Studio or Xcode. We would have access to all the Android and iOS APIs to give our app a native appearance and flow. Between these two options, we ultimately decided to go with React Native because a member of our team has worked with this framework before, and we thought this experience would be valuable to us.

### 2. What technology will we use for the backend?

- Option 1: Node.js
- Option 2: Spring Boot

Choice: Option 2

Justification: All of our team members are skilled in Java, and Spring Boot allows for application development easily with the language. It supports different databases, so we could still be open to any database, both SQL and noSQL, after this choice of backend was made. Spring boot also minimizes the use of xml files for configuring the web application. These advantages do not carry over to options like Node.js, which is JavaScript based, and would have required more time for our team members to get comfortable with.

3. What database will we use?

Option 1: MongoDB

Option 2: MySQL

Option 3: NoSQL

Choice: Option 1

Justification: MongoDB works well with React Native's Javascript objects and is easy to learn and use. The installation process is also simple, especially for people new to the technology. The database can also handle large sets of data, which is important to consider for the development of a multifaceted application intended for multiple users.

4. What platform will we host the server on?

Option 1: Heroku

Option 2: Azure

Option 3: AWS

Justification: AWS has a service called Elastic Compute Cloud, an infrastructure that needs to be set up before deployment and managed afterwards. In contrast, Heroku is a more ready-to use platform that takes care of the details. Similar to AWS, Microsoft Azure is also a Cloud-hosting service. Azure is very flexible and scales well, but Heroku still makes deployment easier and faster. Because our needs from our server are relatively limited and we have a time-constraint, we decided to go for the option that was easier to use.

5. Which APIs will we use for nutrition and diet-related information?

Option 1: Spoonacular

Option 2: CalorieKind Food

### Option 3: Edamam APIs

Choice: Option 1 & 2

Justification: The most optimal solution for an API would be to use an all-in-one API that is also relatively easy to use. However, most APIs that we found did not provide all the functionality that we expect our application to have. So, we found that using a combination of both Spoonacular and CalorieKind's APIs would be our best option, where we can use Spoonacular's variety in functionality, but we can also use CalorieKind for the simpler functionality without much hassle. Option 3 would require us to use multiple APIs for nutrition analysis, meal recommendations and other such functionality, so we decided not to go with that.

### 6. What type of architecture will our system have?

Option 1: client-server (fat-client)

Option 2: web application (thin-client)

Choice: Option 1

Justification: A client-server application will likely be faster, as the interface would be installed on the mobile device. A web application would be more scalable, and performance would be unlikely to decrease even with large numbers of users. A web application may also take longer to develop than a client-server application, and will be more difficult to test. We chose option 1, as the approach will better guarantee our non-functional requirements for speed, and we do not need to support an unlimited amount of users, which would be the main benefit of option 2. Option 1 will also be better suited to our abilities; we feel it will be easier for us to develop, test, and secure than option 2.

### 7. How should users receive notifications?

Option 1: mobile push notifications

Option 2: email notifications

Choice: Option 1

Justification: Push notifications are more likely to be immediately seen by the user, but they are short and simple. The benefit of email notifications is that they can be more detailed, and deliver content. We decided to go with option 1, since

push notifications better suit the type of notifications we plan to provide: quick reminders and positive encouragement, rather than data reports. Push notifications will also feel more personal than email notifications.

8. What type of navigation will the home page have?

Option 1: bottom navigation bar

Option 2: grid navigation menu

Option 3: drawer navigation

Choice: 2 & 3

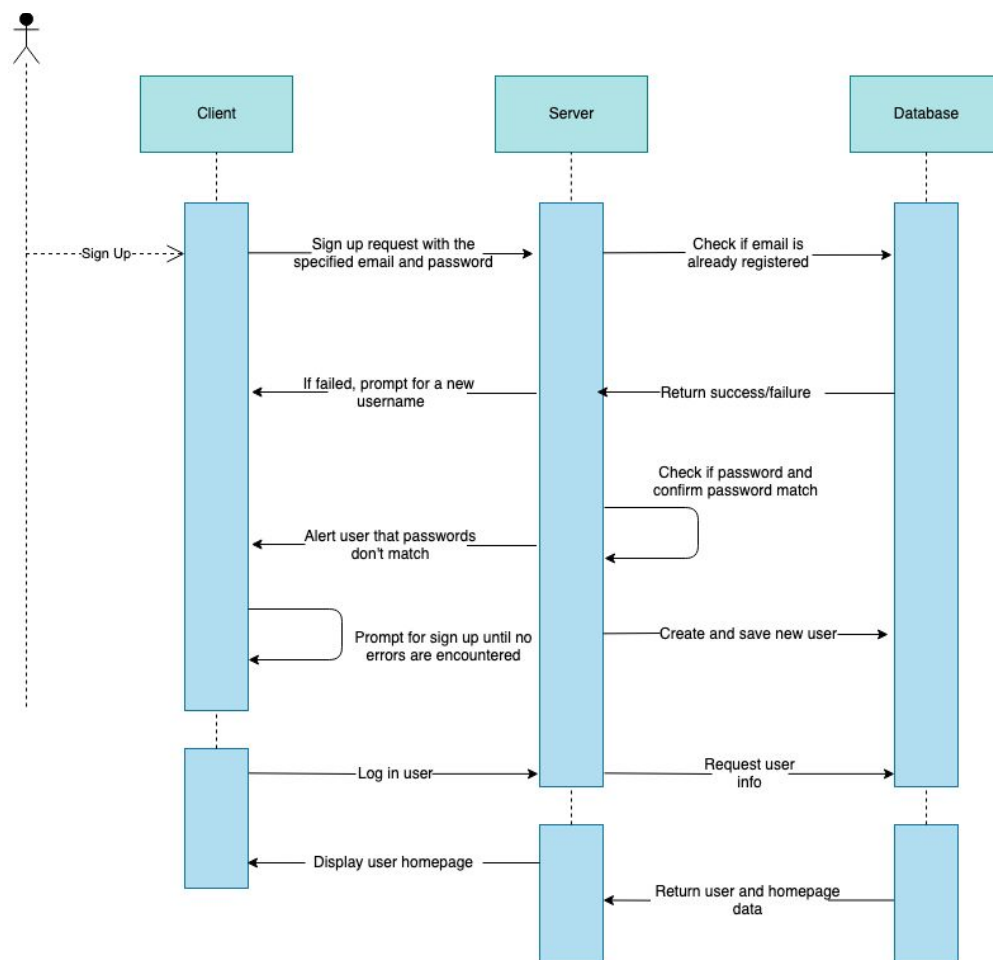
Justification: The bottom nav bar is a classic and popular choice among modern apps. A particular benefit is it enables the user to easily navigate one-handed. It is a good choice for apps with few function pages. The grid navigation menu is intuitive and offers immediate visibility of core functions. It can, however, be overwhelming with too many choices. Drawer navigation, i.e. the hamburger menu, is a fine choice when many options exist, as it saves display space and avoids clutter. The main drawback is low discoverability, and as thus is usually used for secondary navigation. We decided on the grid nav menu, in order to show the user all their choices on the home page, as soon as they log in. We will have a hamburger menu in the corner that will expand into secondary functions (e.g. profile, settings, and logout).

# Design Details

## Sequence Diagrams

### 1. Sign Up

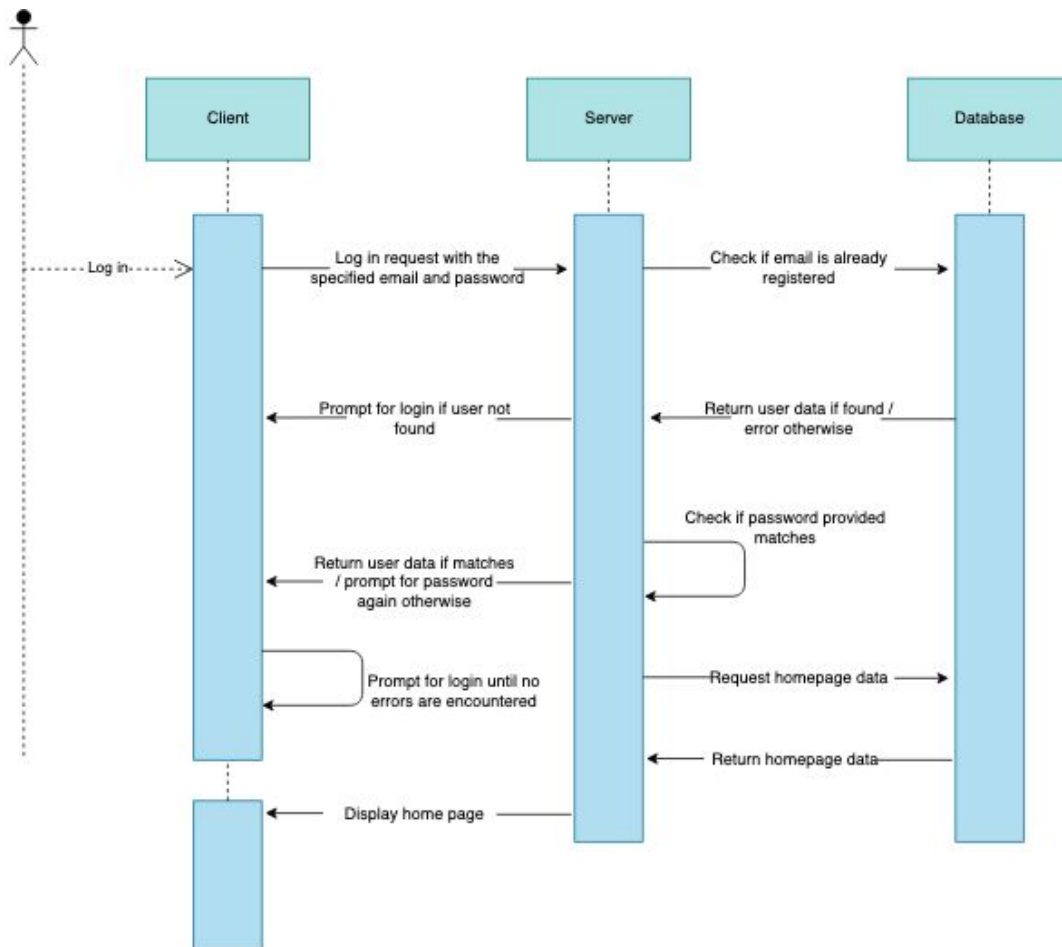
The figure below shows how the client, server and database are involved in a user's sign up request. A user who does not yet have an account will be able to sign up for one. They will input their email and create a password, and their account will successfully be created if the provided email is not yet tied to an account. Their account will then be logged in and the user will be shown the home page.





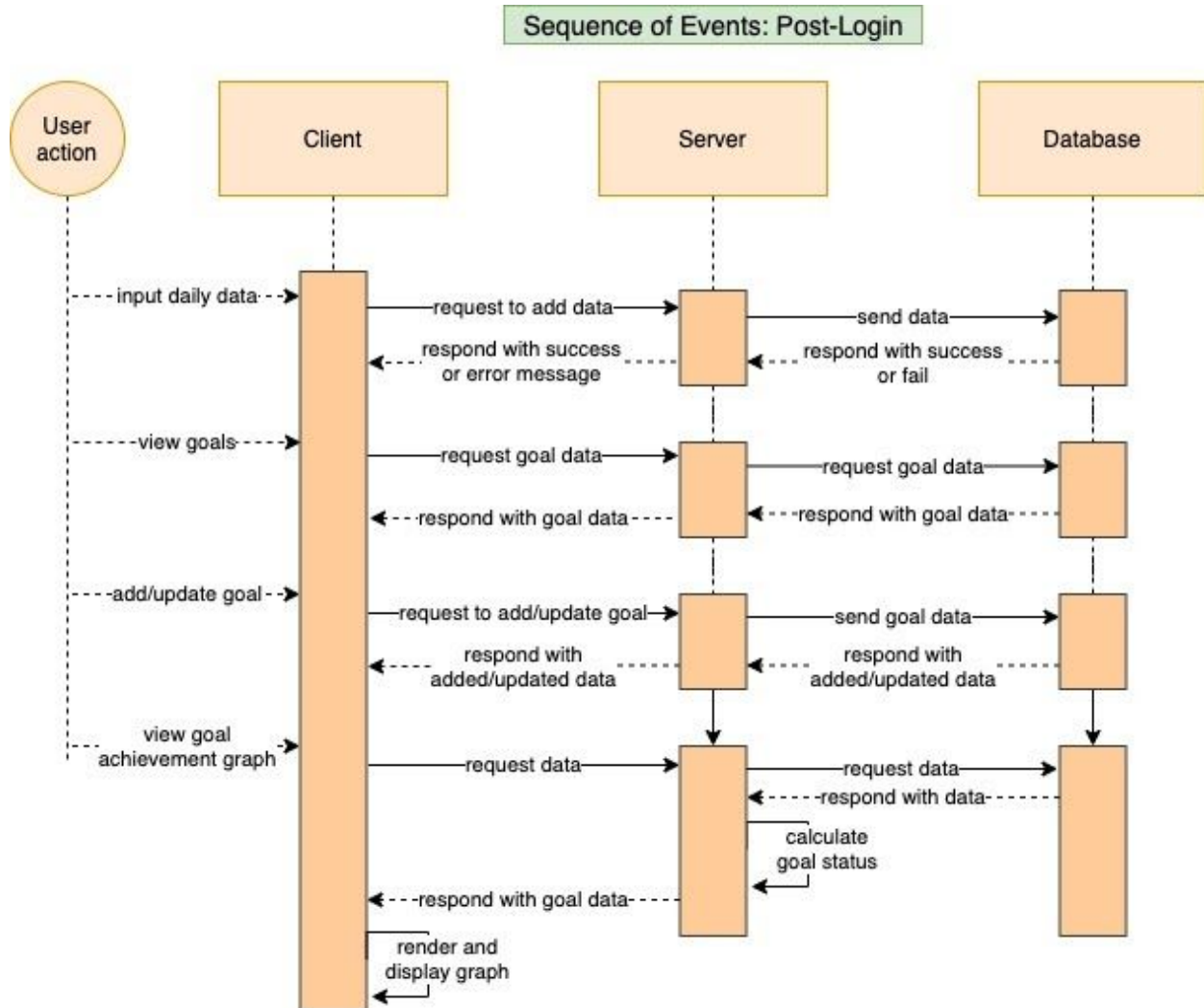
## 2. Log in

The figure below shows how the client, server and database are involved in a user's login request. Users that already have an account can log in by inputting their email and password. The login attempt will fail if the username does not exist or the password is incorrect, and the user will stay on the login page. When the user correctly types their login information, they will successfully be logged into the app and be shown the homepage.

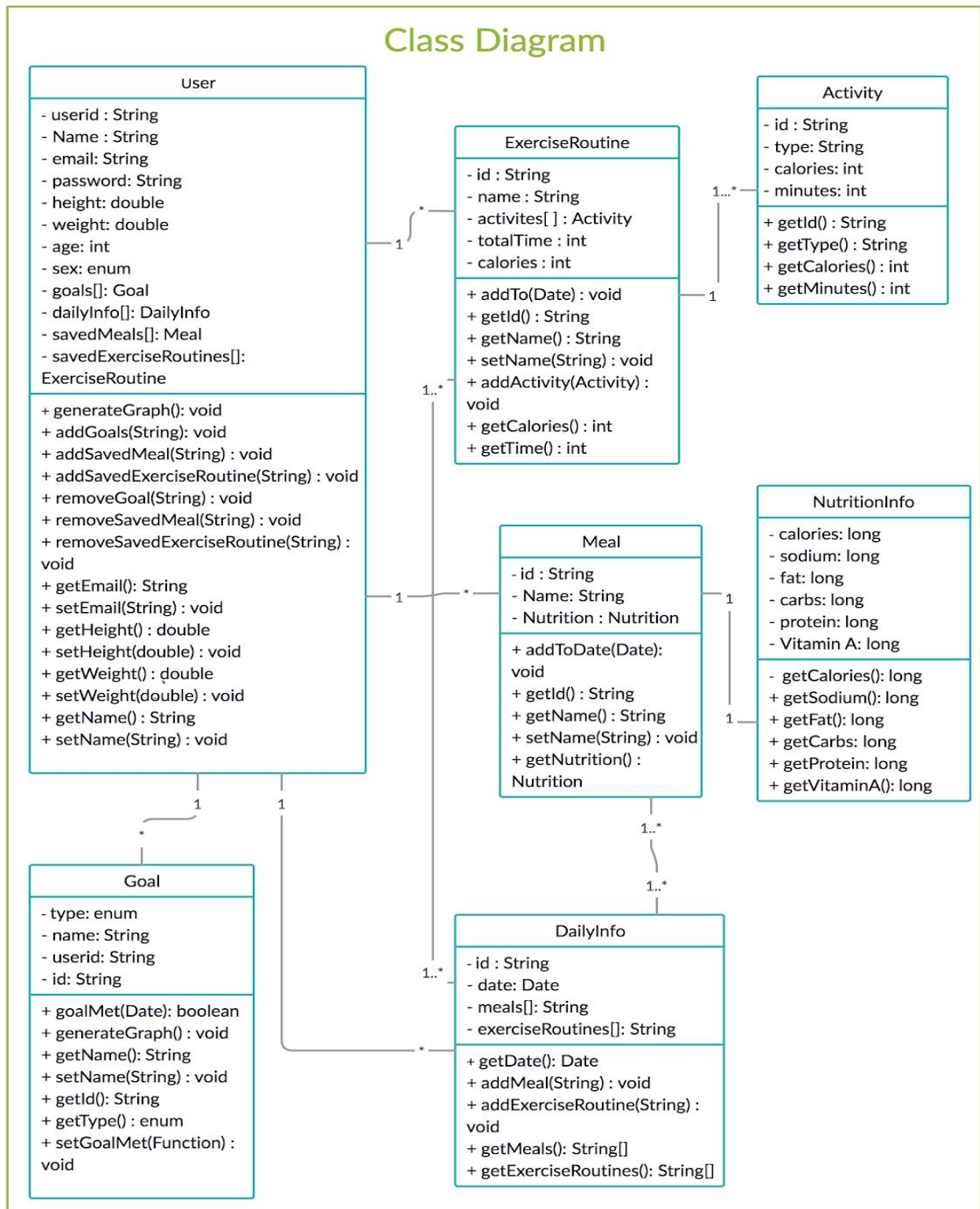


### 3. Post-login actions

The below diagram details a few potential user actions after logging into the app, and the interactions between the client, server, and database to service the user actions. A typical sequence of events for a user using the app each day would likely be to first input their daily data, take a look at their goals and perhaps update or add to them, and then take a look at their analytics, such as a graph displaying their consistency in achieving their goals. The diagram shows how actions dealing with data will have the client making a request to the server, which in turn interacts with the database. Ultimately, the client renders the display to show the data to the users.



## Class Diagram



## Description of Data Classes and their Interactions

### User:

- Represents a user of the app
- Fields:
  - Account information - userid, name, email, password (encrypted)
  - Health characteristics - height, weight, age, sex
  - Goals - an array of ids of Goal objects. These are the fitness goals that the user is currently tracking
  - DailyInfo - a hashmap of ids dailyInfo objects. The key is the date that the dailyInfo object refers to, and the value is the id of the dailyInfo object. These dailyInfo objects store the actual daily diet and exercise information of the user
  - Saved Meals - an array of ids of Meal objects. These meals are the ones that should appear in the user's saved meals list
  - Saved Exercise Routines - an array of ids of exerciseRoutine objects. These exercises are the ones that should appear in the user's saved exercise routines list.
- Methods:
  - generateGraph() - this generates a summary graph of the user's goal progress
  - addGoal(String) - add a new goal id to the user's goals array
  - addSavedMeal(String) - add a new meal id to the user's savedMeals array
  - addSavedExerciseRoutine(String) - add a new exerciseRoutine id to the user's savedExerciseRoutines array
  - removeGoal(String) - remove the goal id given as an argument from the user's goals array
  - removeSavedMeal(String) - remove the meal id given as an argument from the user's savedMeals array
  - removeExerciseRoutine(String) - remove the exerciseRoutine id given as an argument from the user's savedExerciseRoutines array
  - Getters and setters for some fields - getEmail(), setEmail(String), getHeight(), setHeight(double), getWeight(), setWeight(double), getName(), setName(String)

### Goal:

- Represents a user's fitness goal that they're tracking

- Fields:
  - Identifying information - id, userid, name
  - type - this will be either exercise, diet, or sleep. This is to know what section this goal should be displayed under
- Methods:
  - goalMet(Date) - checks if this goal was met for the date given as an argument.
  - generateGraph() - generates a summary graph for this specific goal
  - setGoalMet(Function) - This is to modify the goalMet(Date) function if the user edited the goal
  - Getters and setters for some fields - getId(), getName(), setName(String), getType()

### Meal:

- Represents a meal that the user has inputted
- Fields:
  - Identifying information - id, name
  - Nutrition - a nutrition object that stores the nutritional information (like how many grams of carbohydrates, for example) of this meal
- Methods
  - addTo(Date) - add this meal to the dailyInfo object corresponding to the date given as an argument
  - Getters and setters for some fields - getName(), setName(String), getNutrition()

### Nutrition:

- This represents the nutritional information of a meal
- Fields:
  - Calories, carbohydrates, protein, fat, vitaminA, sodium ...
- Methods:
  - Getters for all fields

### ExerciseRoutine:

- This represents an exercise routine that the user has inputted
- Fields:
  - Identifying information - id, name
  - activities - array of activity objects that make up this routine. These might be things like, 10 minutes of jogging

- totalTime - the total minutes this routine takes
- calories - the total calories that this routine burns
- Methods
  - addTo(Date) - add this exerciseRoutine to the dailyInfo object corresponding to the date given as an argument
  - addActivity(Activity) - add an activity object to the activities array
  - Getters and setters for some fields - getId(), getName(), setName(String), getCalories(), getTime()

### Activity:

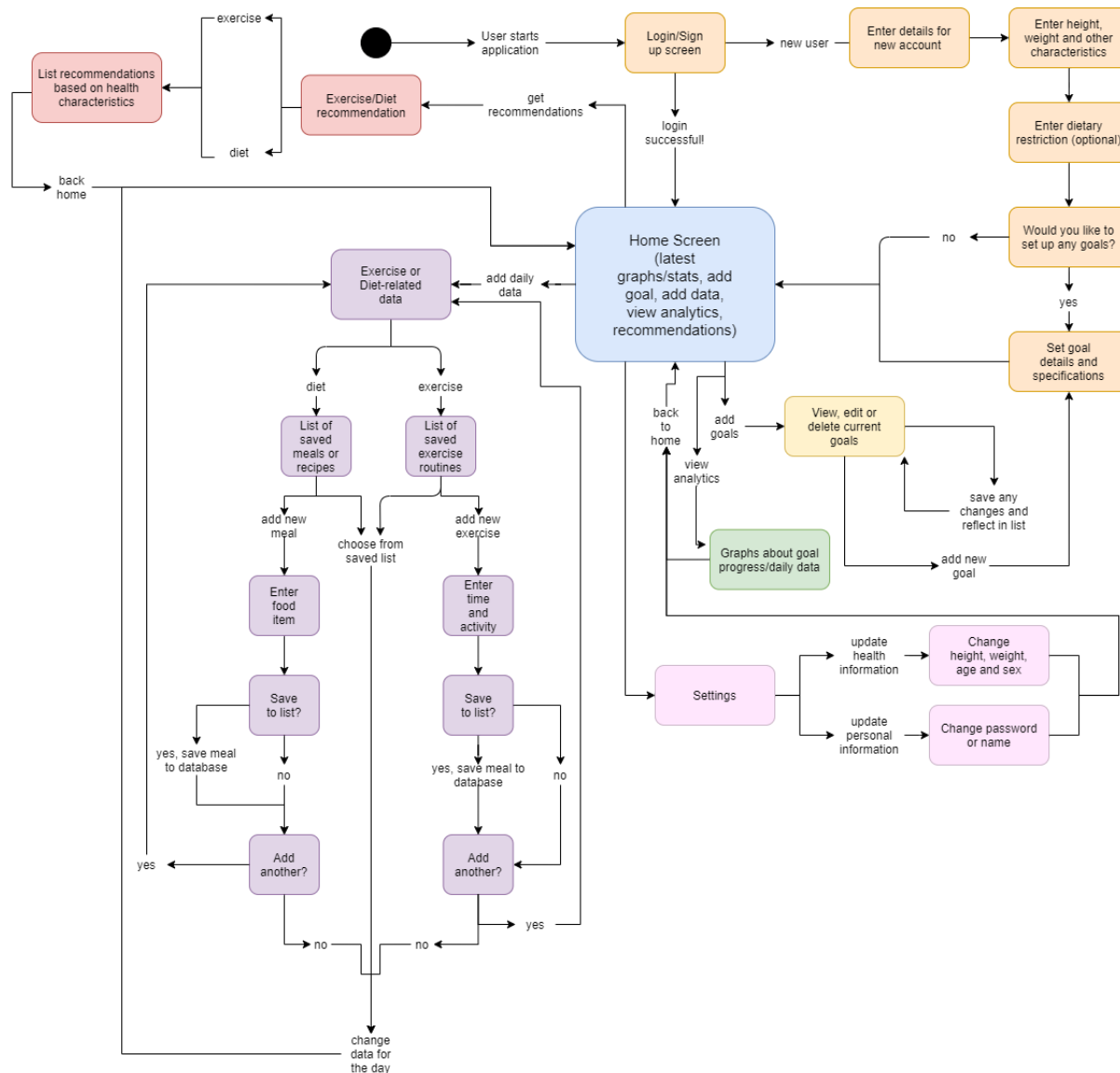
- This represents a specific activity that is part of an exerciseRoutine
- Fields:
  - id
  - type - a string to identify what type of physical activity this is, like “jogging” or “swimming” etc.
  - time - the number of minutes the activity is done for
  - calories - the calories burnt doing this activity for the specified amount of time
- Methods:
  - Getters for some fields: getId(), getType(), getCalories(), getMinutes()

### DailyInfo:

- Represents one day’s exercise and diet information for a user
- Fields:
  - id
  - Date - the date that this object corresponds to
  - meals[] - array of ids for meal objects that the user inputted for this date
  - exerciseRoutines[] - array of ids for exerciseRoutine objects that the user inputted for this date
- Methods:
  - addMeal(String) - add the meal id given as an argument to the meals array for this object
  - addExerciseRoutine(String) - add the exerciseRoutine id given as an argument to the exerciseRoutines array for this object
  - Getters for some fields: getDate(), getMeals(), getExerciseRoutines()

## Activity Diagram

From the activity diagram below, we can see that the design is very intuitive, and navigating from one page to another is very fluid. Since Bamboo works on user data and tracking daily input, it is important for a user to have an account before beginning to use the application. This way, the app can be used across different devices. On sign up, a few short questions are asked in order to accumulate data, and then the user can get started with getting recommendations and adding daily data. Every page has a next button that will allow the user to move to the next page easily. The home page has tiles that link to every functionality that is provided by the application. It will also have a sidebar with options like settings, changing health attributes, logout, delete account and notification settings.



## UI Mockups

Login/Sign Up page:

### Bamboo.

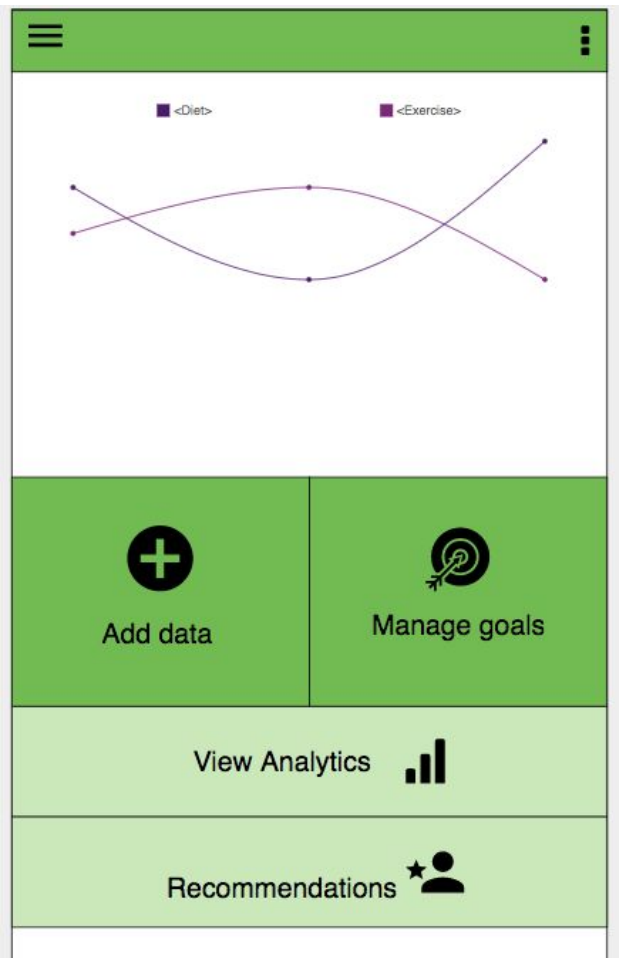
Log in

Or

[Sign up](#)

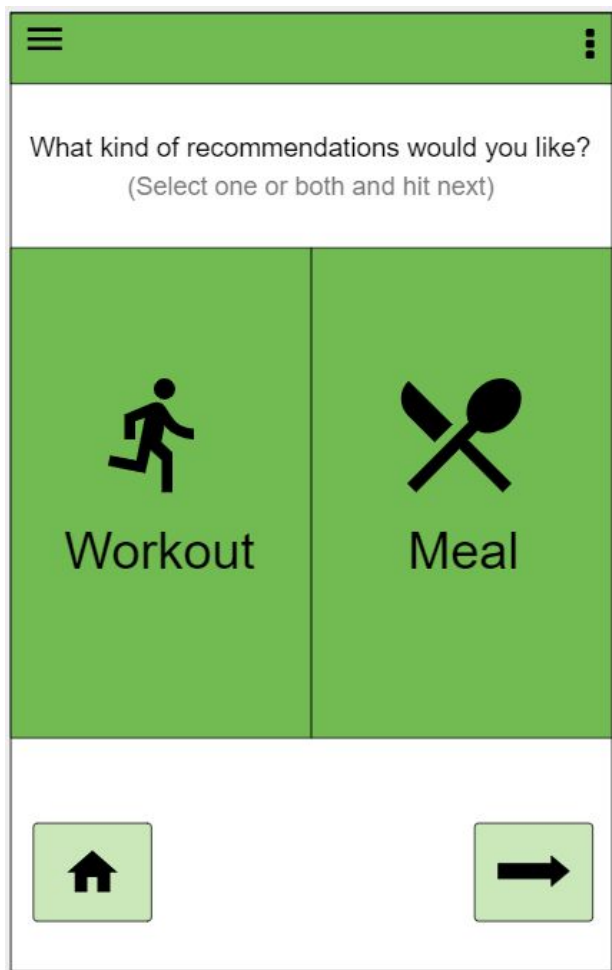
[Forgot Password?](#)

Home Page:

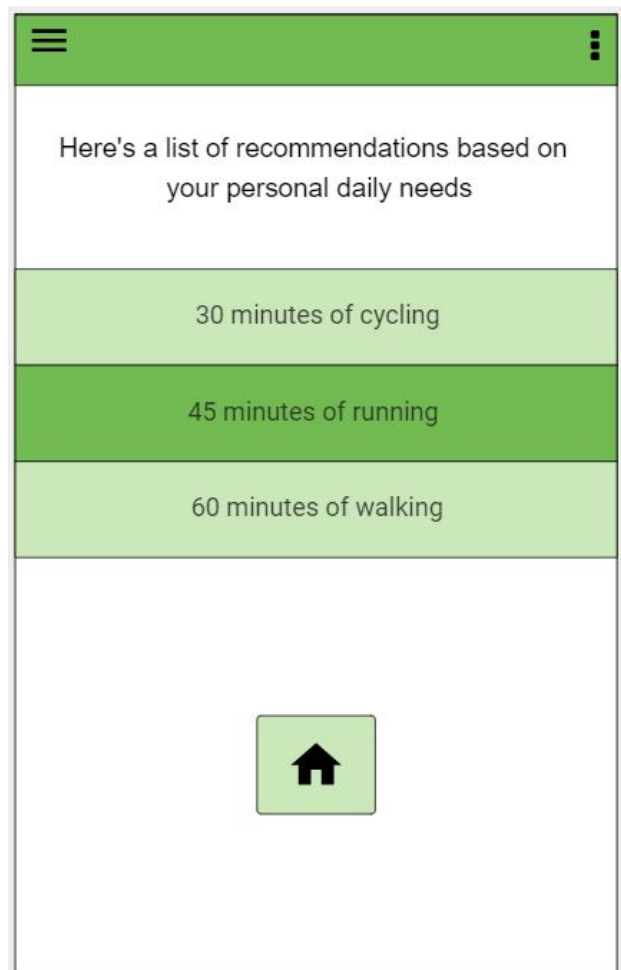




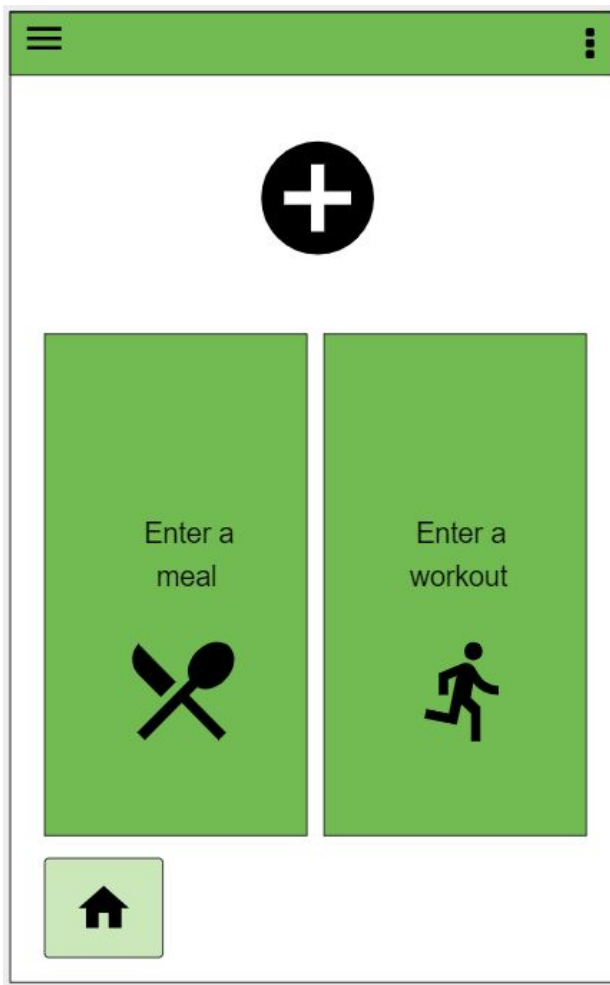
Recommendations:



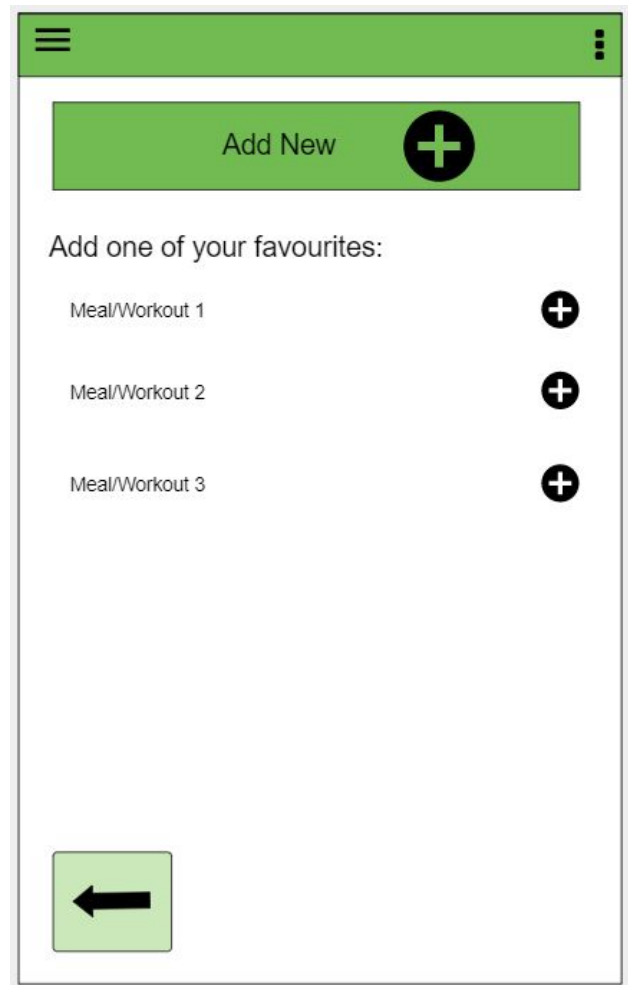
If user selects workout recommendation:





Enter daily data:



Next page for daily input :





Detailed entry for meal:




Enter meal name:



Enter number of servings:

Add Another 

Add to favourites 




Detailed entry for workout:





Enter an exercise activity:

Enter the duration:



:

Add Another 


Add to favourites 





View/ Edit Goal:




Here are all your goals!



More than 3 hours of exercise per week

About 20 g of protein per day





Set goals:



Select the type of goal:

Diet

Exercise

Less than

More than

About

Grams

Calories

Minutes

of

Grams

Calories

Minutes

per

Day

Week

