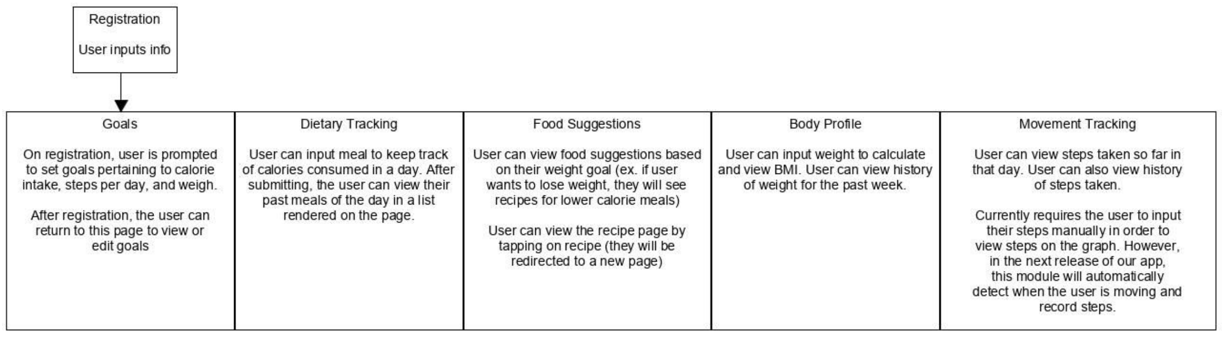
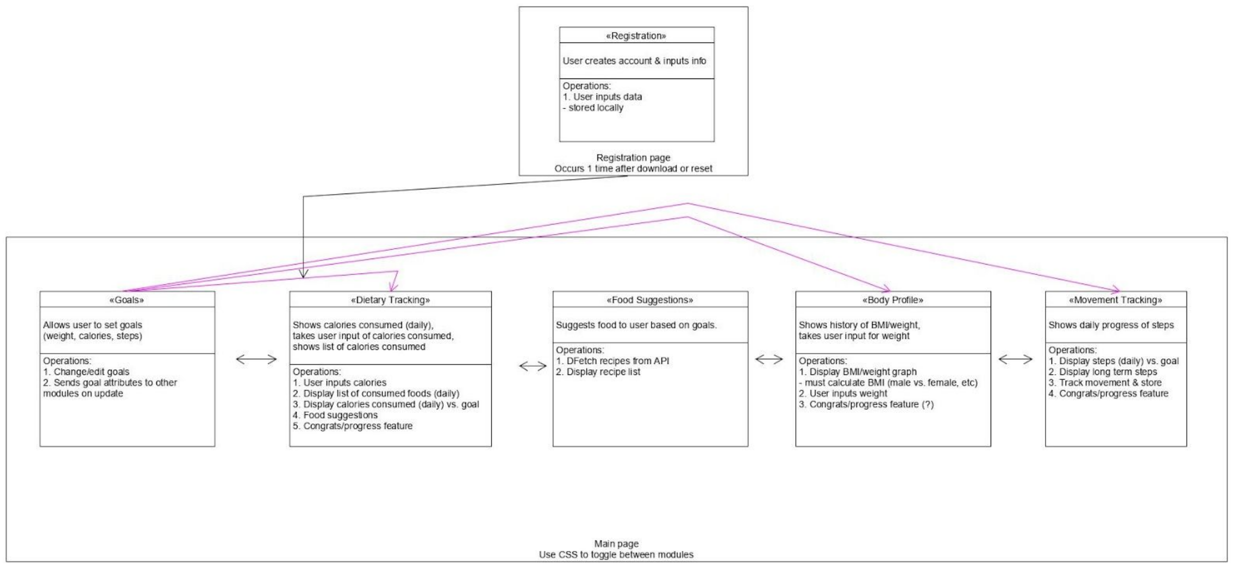
Sendit Fitness - Design and Architecture

We each developed a page of the application individually. All the pages are bounded by the module SwipeNavigator that allows for swiping motions to navigate between pages.

App flow:


Registration
User inputs info
Goals
On registration, user is prompted to set goals pertaining to calorie intake, steps per day, and weigh.
After registration, the user can return to this page to view or edit goals
Dietary Tracking
User can input meal to keep track of calories consumed in a day. After submitting, the user can view their past meals of the day in a list rendered on the page.
Food Suggestions
User can view food suggestions based on their weight goal (ex. if user wants to lose weight, they will see recipes for lower calorie meals)
User can view the recipe page by tapping on recipe (they will be redirected to a new page)
Body Profile
User can input weight to calculate and view BMI. User can view history of weight for the past week.
Movement Tracking
User can view steps taken so far in that day. User can also view history of steps taken.
Currently requires the user to input their steps manually in order to view steps on the graph. However, in the next release of our app, this module will automatically detect when the user is moving and record steps.

High-Level Design:


«Registration»
User creates account & inputs info
Operations:
1. User inputs data
- stored locally
Registration page
Occurs 1 time after download or reset
«Goals»
Allows user to set goals (weight, calories, steps)
Operations:
1. Change/edit goals
2. Sends goal attributes to other modules on update
«Dietary Tracking»
Shows calories consumed (daily), takes user input of calories consumed, shows list of calories consumed
Operations:
1. User inputs calories
2. Display list of consumed foods (daily)
3. Display calories consumed (daily) vs. goal
4. Food suggestions
5. Congrats/progress feature
«Food Suggestions»
Suggests food to user based on goals.
Operations:
1. DFetch recipes from API
2. Display recipe list
«Body Profile»
Shows history of BMI/weight, takes user input for weight
Operations:
1. Display BMI/weight graph
- must calculate BMI (male vs. female, etc)
2. User inputs weight
3. Congrats/progress feature (?)
«Movement Tracking»
Shows daily progress of steps
Operations:
1. Display steps (daily) vs. goal
2. Display long term steps
3. Track movement & store
4. Congrats/progress feature
Main page
Use CSS to toggle between modules

Overall

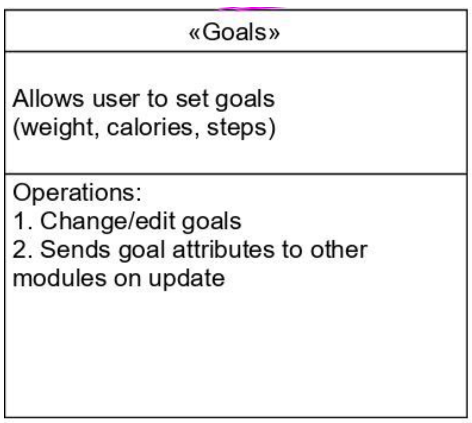
Upon first downloading and opening the app, the user is taken to the Registration page, then the SetGoals page, then the home page which utilizes the SwipeNavigator. After registering once, the user will always be taken to the home page. The SwipeNavigator is used to combine the different components we built including Goals, Dietary Tracking, Movement Tracking, and Body Profile. These four components are rendered in one page, but the SwipeNavigator only shows one at a time and the user can swipe the screen left or right to switch components.

Register

- The register page was implemented with the help of an npm library tcomb-form-native, which helped with specific user inputs such as strings, numbers, and dates
- Upon first opening the app, the user is taken to the registration form and fills in their info which includes their name, weight, height, birth date, and gender
- Once the user submits their info, they are taken to the goals module

Goals

- The goals page was also implemented using tcomb-form-native, and the user first sets their goals. This includes whether they want to lose, gain, or maintain their current weight, the max calories per day, and the max steps per day
- After setting their goals, the user is able to view their goals as part of the swiper component
- Finally, the user is also able to edit their goals which will take them back to the initial set goals page


«Goals»

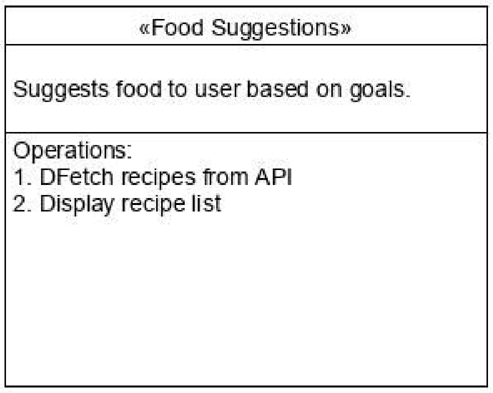
Allows user to set goals (weight, calories, steps)

Operations:
1. Change/edit goals
2. Sends goal attributes to other modules on update

Dietary Tracking

Food Suggestion

- There is a outer container holding all of the components inside of the module. Inside that container is a Flat list.
- This element is linked to a list that represents our food suggestions.
- When a specific function is called, it will make a request to an recipe REST API.
- Inside of the call, there are specific breakfast, lunch and dinner categories so that the resulting food suggestions will return a breakfast, lunch, and dinner line up.
- Also, the API call will search for a recipe that are within a specific calorie range so the resulting suggestions lineup will add up to an amount of calories that the user has specified in the registration
- Finally, we parse the API response and update the food suggestion list. This will result in the Flat list element dynamically getting updated.

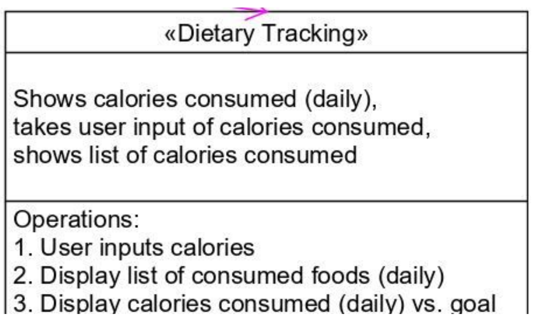

«Food Suggestions»

Suggests food to user based on goals.

Operations:
1. DFetch recipes from API
2. Display recipe list

Dietary Tracking

- Dietary tracking relies on asynchronous storage to keep track of meals. Each day, 2


«Dietary Tracking»

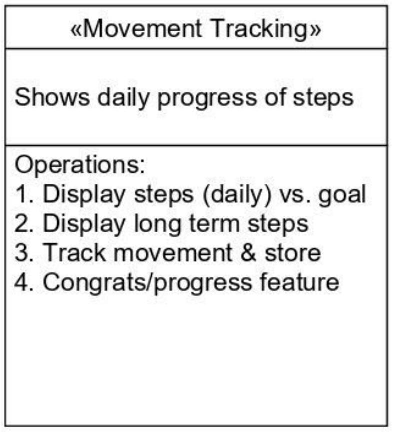
Shows calories consumed (daily), takes user input of calories consumed, shows list of calories consumed

Operations:
1. User inputs calories
2. Display list of consumed foods (daily)
3. Display calories consumed (daily) vs. goal

    new keys are generated to store a list of meals and the number of consumed calories.
    - To accept input, the user types in the name of the meal and the caloric amount. One the user presses the submit button, the app stores the new meal and calories consumed in storage and renders the new meal onto the list of meals displayed on the page.
    - To render the list of consumed meals, we parse the meal list from storage into a JSON object

Movement Tracking

- The movement tracker allows the user to enter in steps that they have taken. Step data is stored using asynchronous storage in JSON format.
- The movement tracker uses asynchronous storage to retrieve the user's step goals. This allows the user to dynamically set any goal for the movement tracker to use.
- The movement tracker uses React Native's built-in alert functionality to alert the user when they have reached half of their step goal and when they have completed their step goal.
- The movement tracker utilizes the LineGraph component, as well as the react-native-svg-charts module (see references below) for visualizing step data.
    - Step data for the day is visualized as a "progress circle" showing how many steps left to go for the day.
    - Step data for the past week is showed as a line graph with the Y-axis labeling amounts of steps for each day.



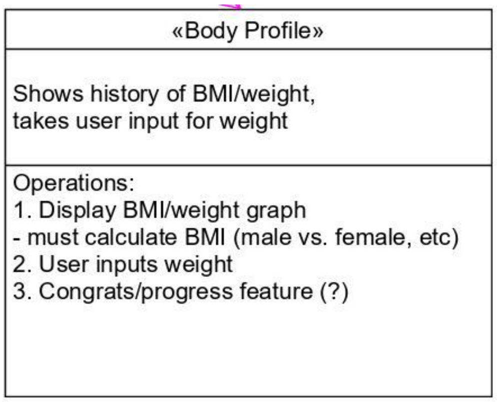
«Movement Tracking»

Shows daily progress of steps

Operations:
1. Display steps (daily) vs. goal
2. Display long term steps
3. Track movement & store
4. Congrats/progress feature

Body Profile

- The weight page was implemented with the help of an npm library tcomb-form-native, which helped with specific inputs, namely number for the weight.
- Asynchronous storage was used to store the JSON data of the body profile. The JSON data contains {weight: [array of numbers], height: number, BMI: [array of numbers]}.
- Once the weight is input, BMI is calculated using the input weight and height stored from the registration page. Weight will be stored into the array of weight in the asynchronous storage.
- Once BMI is calculated, the new BMI is stored in the asynchronous storage in the array BMI.
- The newly computed BMI will be updated to the graph, but if the user inputs multiple weights in same day, only the latest BMI will be stored. I also store the date the weight was changed, and if the date stored and the date when the user inputs the weight is the



«Body Profile»

Shows history of BMI/weight,
takes user input for weight

Operations:
1. Display BMI/weight graph
- must calculate BMI (male vs. female, etc)
2. User inputs weight
3. Congrats/progress feature (?)

same, the last element of the array changes. Otherwise, the weight is appended to the weight array.

LineGraph

- We created a LineGraph component as a wrapper for a component provided by the react-native-svg-charts module (see references below).
- This component was designed in a modular way so that it can be incorporated into any other page to display a line graph of custom data.

References

- react-native-svg-charts: https://github.com/JesperLekland/react-native-svg-charts