# Better Bulking Design Documentation

10/24/2020

Anthony Yip

## System Overview

Better Bulking is an app designed for fitness enthusiasts, simplifying the overall process of weekly meal planning. The app allows users to directly search or import recipes, specify required calorie requirements, and generate a compiled shopping list.

## Design Considerations

### Assumptions and Dependencies:

This web-application will be dependent on the spoonacular API.

### General Constraints:

Currently using the spoonacular API free plan, which only allows for 50 points/day. As such, will try to develop app with minimal API requests when possible. Depending on app design and number of users, may have to upgrade to either the starter ($1/month for 150 points/day) or cook ($29/month for 1,500 points/day) plans.

### Goals and Guidelines:

Maintain a minimal and easy to navigate user interface.

### Development Methods

I will be using Rapid application development (RAD) primarily because I am the only developer.

## System Architecture

This web-app will be built using the React framework. The main components include:

* Search meals
  + Returns a list of recipes based on user’s search query and filtered by dietary restrictions
* Import Meals
  + Imports a recipe (including ingredients and instructions) based on a user provided external link
* Custom Import
  + Allows user to create their own meal
  + Search the spoonacular API ingredients endpoint and add to meal
* Meal Planner
  + Summarizes user selected meals.
  + Organized by daily/weekly meals.
* Shopping list
  + Compiles are ingredients, combining when appropriate, and generating a single shopping list
  + Ingredients will be organized by category (e.g., meat/poultry, vegetables, spices, etc.) for easier navigation when shopping

## Policies and Tactics

Describe any design policies and/or tactics that do not have sweeping architectural implications (meaning they would not significantly affect the overall organization of the system and its high-level structures).

## Detailed System Design

Most components described in the System Architecture section will require a more detailed discussion. Other lower-level components and subcomponents may need to be described as well.

**Top-level states**:

* User specified daily calorie requirement = (daily, weekly)
* Current daily macros = {M: {protein: integer, fat: integer, carbs: integer, etc}, Tu: etc}
  + function setDailyMacros( ) dependent on selectedMeals state
* Current daily calories status = {M: integer, Tu: integer, W: integer, etc}
  + function setDailyCalorieStatus( ) will be dependent on selectedMeals state
* Current weekly calorie status = integer
  + function setWeeklyCalorieStatus( ) will be dependent on daily calorie status
* Selected meals = {

Monday: {

Breakfast: {},

Lunch: { },

Dinner: { },

Snack (optional and can be added by user): { }

},

...

}

**Subcomponents**:

* Search meals
  + States:
    - User search input
    - API search return JSON
  + Propped functions:
    - prop.setSelectedMeals
  + Subcomponents:
    - searchResultsContainer
      * Picture, title, calories, macros, description
* Import Meals
  + States:
    - User search input (link)
    - API search return JSON
* Custom Meal
  + States:
    - Title
    - Ingredient search input
    - API search result JSON
* Meal Planner
  + States:
    - prop.selectedMeals
  + Subcomponents:
    - Generate shopping list
    - Email to user?

**Shared components:**

* DisplayMeal (shared by Search meals, import meals, meal planner)
  + States:
    - Title
    - Ingredients
    - Calories
    - Macros
    - Instructions