

Group: 203-2

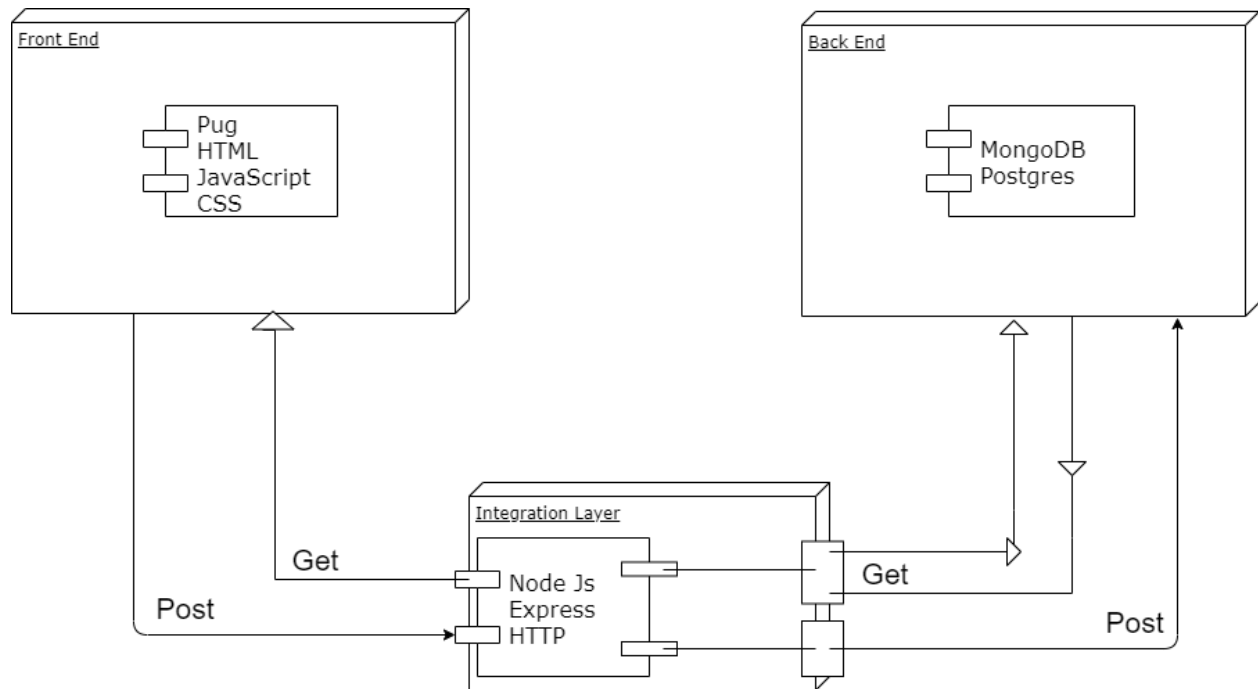
<https://sreeshanath.github.io/Project%20Milestones/Project%20Milestone%204/index.html>

## **Revised List of Features**

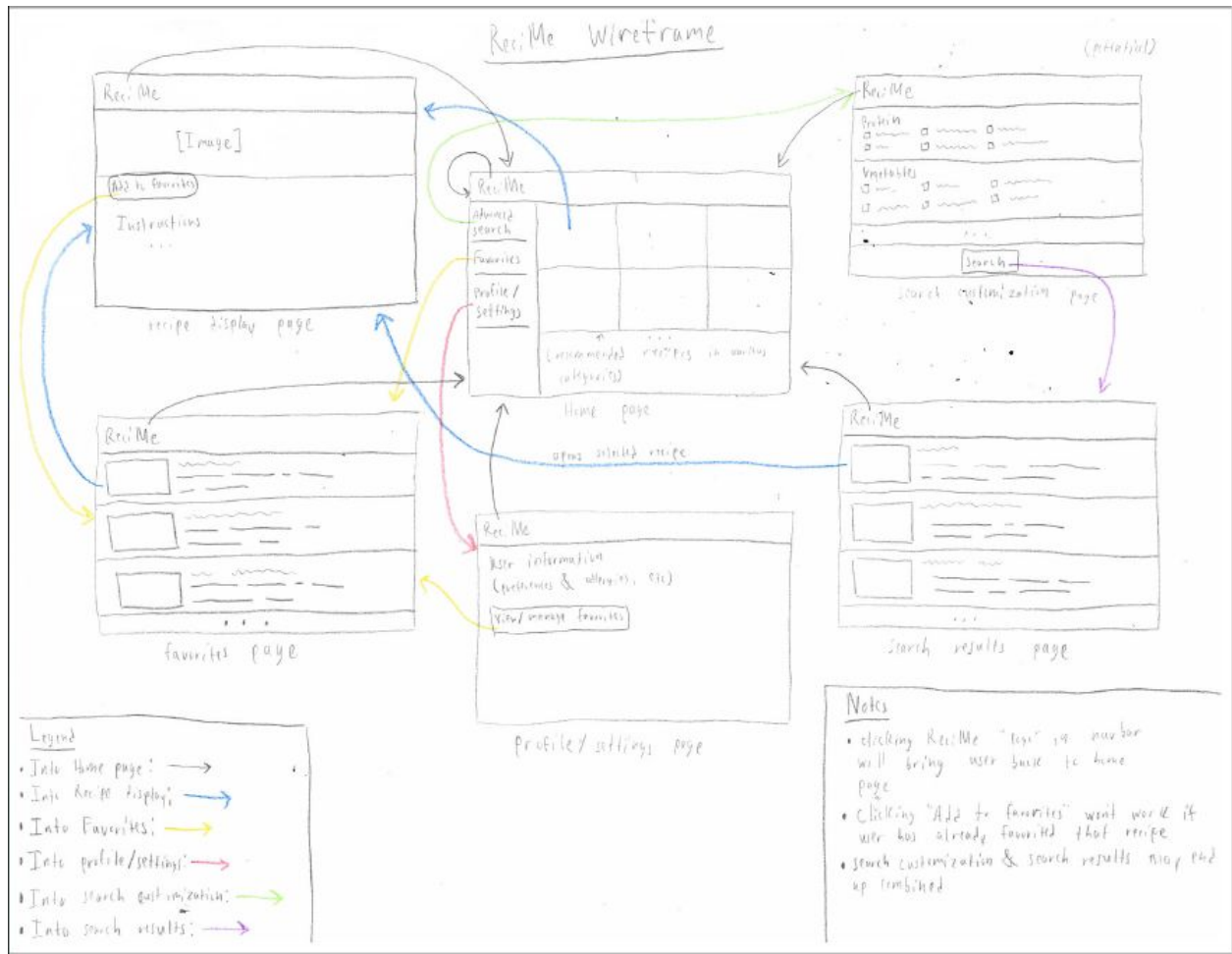
**Features are listed in the order of their priority and how they will be completed.**

- Navigation Bar
  - Will allow users to revert to previous state of application or to enter a new page.
  - Easiest and most useful feature for navigation. Will be quick to implement and very meaningful to the project.
- Search by Image (AKA grid search)
  - A way to browse through recipes by letting images be your guide. Will display an image of a recipe along with its' name so you know how delicious it is before even looking at its ingredients.
  - Most important feature at the moment. Requires integrating front end with backend and api. Also means functionality of search bar
- Recipe Display
  - Fully expand selected recipe in order to view its' requirements. User will be able to find prep/cook time, health rating, ingredient choice, and instructions for how to make the meal here.
  - Very important to project. Without this, users won't be able to make these wonderful recipes
- Ingredient Filter (Allergies, Food-Waste)
  - Will allow the user to filter out any ingredients based on food allergies, and gives the user the option of using their own food in the refrigerator to save costs and minimize food waste.
  - One of the best ways to filter recipes so it is up there on priority
- Sign up/Sign in
  - Necessary for favorite recipe feature. Will allow user to create and log into their own account from website.
  - We will need to create a user database and sign in/up feature in order to implement but it is necessary to do the favorite recipes feature.
- Favorite Recipes
  - A simple toggle star next to recipe entries that will save a certain recipe to a users' favorited folder. This can be accessed quicker and easier next time they wish to make the same thing; not to mention, saving them the headache of forgetting what the recipe was called in the first place.
  - A critical feature but it will take awhile to implement.
- Meal Filter (Breakfast, Lunch, Dinner)
  - Depending on which meal of the day you would like to enjoy (breakfast, lunch, or dinner), you can use the meal filter to look at recipes belonging to that description. For example, if the user chooses breakfast, they will only see breakfast recipes.
  - Low priority, will do if we have time.

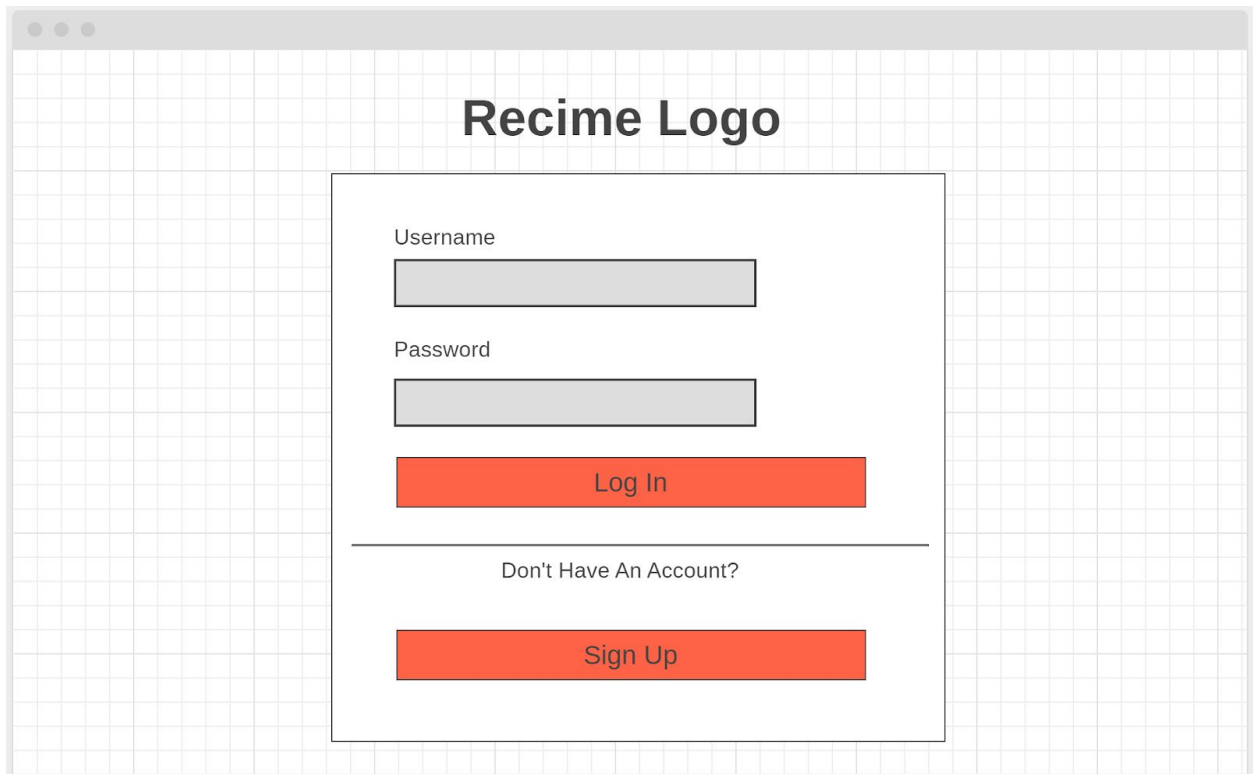
## Architecture Diagram



## Front End Design



## Login Page:



A login page mockup with a light gray grid background. At the top center is the text "Recime Logo". Below it is a white rectangular form with a thin black border. Inside the form, the text "Username" is followed by a gray input field. Below that, "Password" is followed by another gray input field. Under the password field is a red button with the text "Log In". A horizontal line separates this from the text "Don't Have An Account?". Below the line is another red button with the text "Sign Up".

Recime Logo

Username

Password


Log In

---

Don't Have An Account?

Sign Up

## Signup Page:



Join the Recime Family!

First Name:

Last Name:

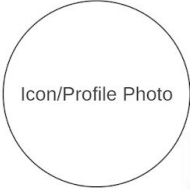
Username:

Password:

Confirm Password:

Already on Recime? [Sign in](#)

## Profile Page:

Recime Logo	Home	Favorite Recipes	<input type="text" value="Search"/>	<input type="button" value="Search"/>	User
		User's Name			
<input type="button" value="Edit Photo"/>					
<b>General Information</b>			<input type="button" value="Edit Profile"/>		
Name: User's Name					
Username: username					
Email: email@something.com					
Birthday: MM-DD-YYYY			<input type="button" value="Reset Password"/>		
<b>Eating Preferences:</b>					
List if vegan, vegetarian, etc.					

# Search Results

Recipe profile	admin	spec
Filter options	recipe	recipe
~		
~	recipe	
~		
~	recipe	
~		

# home page

onClick

recipe display

Recipe	profile	Meal	Search
recipe		Meal	
recipe		Meal	
recipe		Meal	

recipe display

Recipe	profile	Meal	Search
materials	image	ingredients	
instructions		health info	



## **Web Service Design**

The web service our team is using is Unirest. We're using Spoonacular API, which provides a list of functions that deal with recipes, food, and nutrition. For example, "search for recipe" and "Get recipe information" are two functions we will embed in our application. The "search for recipe" and "Get recipe information" will be get request to the API. The API will receive the name of the recipe from the front end which will be the input to the "search for recipe" get request then return all recipes that have the name included in the title. The "search for recipe" returns the name of the recipe, a picture of the recipe, and a unique recipe id that are inputs for other get requests. The server will then push the name of the recipe and the picture to the web page. Once the user has selected which recipe they want the server will then find the matching id number of the recipe and use the recipe id as input for the "Get recipe information" get request. The API will then return a JSON that gives all relevant information about how to make the recipe as shown in the database design section.

Spoonacular get request return multiple data types, including boolean, integers, strings, and arrays. Diet and cost descriptions are documented using boolean (ex. If a recipe is not vegan, Vegan = False). Serving size, and preparation and cooking times are represented as integers. Additionally, the ingredients and instructions of each recipe are represented as arrays and strings respectively.

## **Database design**

For ReciMe, we will be using a MONGO database. The database will store recipes that the user has favorited. The reason why we decided not to choose a relational database is because each recipe has a different amount of ingredients and so the data is not static making a relational database not an optimal choice for us. In addition, the API outputs its data in a JSON file and so to minimize parsing we decided to use a database that stores JSON files.

### **Schema definition for Recipe: JSON format**

```
{  
vegan: bool,  
glutenFree: bool,  
dairyFree: bool,
```

```

veryHealthy: bool,
cheap: bool,
veryPopular: bool,
sustainable: bool,
ketogenic: bool,
preparationMinutes: Integer,
cookingMinutes: Integer,
sourceUrl: website link showing recipe information,
extendedIngredients:
  [An array of ingredients with each element as its own JSON as shown below],
title: name of recipe,
readyInMinutes: Integer,
servings: Integer,
image: website link to image of recipe
imageType: jpg or png,
diets: [ An array that shows diet preferences that the recipe fits into. Ex: gluten-free],
instructions: String that describes how to cook the recipe
}

```

### Schema definition for Extended Ingredients: JSON format

```

{
image: website link to image of ingredient,
name: name of ingredient,
original: String that describes amount needed. Ex '1/2 cup chicken broth',
Amount: Integer. Shows amount of ingredients that is needed.
unit: String the shows the unit. Ex: 'cup'
}

```