Intelligent Meal Planning

Anaz Mohammed, David Nazaretyan, Francisco Cedeno, Pouria Salekani
Department of Computer Science
California State University, Northridge
Northridge, CA 91330
{anaz.mohammed.985, david.nazaretyan.554, francisco.cedeno.35, pouria.salekani.474}@my.csun.edu

***Abstract*—Meal planning plays a crucial role in maintaining a healthy lifestyle. Despite the abundance of meal planning apps available, many lack personalization and integration with grocery shopping. This project aims to address these gaps by developing an automatic meal planning app that generates personalized meal plans based on users' dietary preferences, age, weight, height, and fitness goals, allowing users to replace any meal as desired. Our solution incorporates an online grocery ordering feature to streamline the process further. Experimental results demonstrated that our app effectively generates personalized meal plans and simplifies the grocery shopping experience, leading to increased user satisfaction and adherence to healthy eating habits. In conclusion, our automatic meal planning app bridges the gap in existing solutions by offering a personalized and seamless meal planning experience that caters to users' diverse needs and preferences.**

***Index Terms*—meal planning, personalization, dietary preferences, algorithm, grocery shopping, app development.**

I. Introduction

Author: Anaz Mohammed

The importance of meal planning cannot be overstated, as it plays a pivotal role in promoting healthy eating habits, managing dietary restrictions, and fostering better food choices. Research has shown that individuals who engage in meal planning have a higher likelihood of consuming a well-balanced diet, leading to improved overall health outcomes and reduced risk of chronic diseases [1]. Despite the availability of numerous meal planning apps, many users report that existing solutions often lack personalization and fail to cater to individual dietary preferences, lifestyle, and health goals. Additionally, these apps seldom integrate with grocery shopping platforms, resulting in a fragmented and inefficient user experience. This project aims to address the gaps in the current landscape by developing an automatic meal planning app that not only generates highly personalized meal plans but also streamlines the grocery shopping process to encourage adherence to healthy eating habits.

Author: David Nazaretyan

Our proposed project seeks to create an intelligent and user-friendly meal planning app that generates personalized meal plans based on an individual's dietary preferences, age, weight, height, and health goals. Our innovative solution incorporates machine learning algorithms and a comprehensive food database to provide a diverse range of meal options tailored to the user's needs. Furthermore, our app seamlessly integrates with online grocery platforms, allowing users to order the required ingredients for their meal plans with just a few clicks. The outcome of our project is a highly efficient and convenient meal planning experience that encourages users to maintain a healthy lifestyle.

The major contribution of our project lies in its ability to bridge the gaps in the current meal planning landscape by offering a holistic, personalized, and streamlined solution. Our app not only generates meal plans tailored to individual preferences and goals but also simplifies the grocery shopping process by integrating with online retailers. As a result, our innovative approach to meal planning has the potential to significantly improve user engagement, promote healthier eating habits, and contribute to better overall health outcomes for individuals.

Author: Francisco Cedeno

The rest of this paper is organized as following: Section II discusses related work; Section III presents the framework of the project; Section IV illustrates the implementation details; Section V presents the experimental results; and Section VI concludes the project and discusses future work.

II. Related Works

Author: Anaz Mohammed

In this section, we will introduce the structure of related work and discuss the major contributions in the field, categorized by their focus. The related work can be divided into three primary subsections: (1) meal planning algorithms and techniques, and (2) user interface and experience design in meal planning apps.

A. Meal Planning Algorithms and Techniques

Author: Pouria and David

In order to provide the user with an efficient algorithm to recommend recipes/meals based off of their goals, a look into another work had to be done. A bunch of data (specifically ingredients) was imported from *food.com* which then were turned into instances [2]. By using a SVM (support vector machine) and other methods to classify these ingredients, a cuisine teller was made, which automatically tells the user the recipe/meal based off of the ingredients. Since each ingredient is an instance, a value is assigned; 1 being it exists and 0 being it does not (*i.e. looking at this meal does meal[x] exist or not).* This sophisticated algorithm is very helpful to the user which can make the user follow their diet in an easy way. This is because this algorithm would detect ingredients when looking

at meals, now, this algorithm can be taken further by cross-referencing those ingredients and combining their **rough** total calories and macronutrients.

The algorithm that we use is a mixture of *cross-referencing* and *cuisine-classification*. The *Spoonacular API* would provide the recipe/meals information, then, when the user tries to sign-up with the app, they would need to enter their information. Looking at the specific metrics: height, weight, and age; one can derive the

$BMR = 66.47 + (13.75 \times \text{weight in kg}) + (5.003 \times \text{height in cm}) - (6.755 \times \text{age in years})$

for the user, then *multiply that by their activity level*:

- Sedentary (little to no exercise + work a desk job) = 1.2
- Lightly Active (light exercise 1-3 days / week) = 1.375
- Moderately Active (moderate exercise 3-5 days / week) = 1.55
- Very Active (heavy exercise 6-7 days / week) = 1.725
- Extremely Active (very heavy exercise, hard labor job, training 2x / day) = 1.9

which then gets the *daily caloric intake* (DCI) for the user. If the user has a DCI of 2,000 calories, that would mean the user would need to consume 2,000 calories to maintain their weight. Furthermore, cross-referencing diet preferences for the user with the Spoonacular API and the DCI, this will make sure to display meals that will fall within the user's DCI *(if the user were to selection a "lose weight" option, their DCI would be at a deficit*). By doing this, it is ensured that the user will have an easy time to pursue their goals since they will get recommended the most accurate information.

B. User Interface and Experience Design in Meal Planning Apps

Author: Francisco and Anaz

This subsection focuses on the user interface and experience design aspects of meal planning applications, examining the features and functionalities of popular meal planning apps and their effectiveness in engaging users and promoting healthy eating habits.

One example work in this area is the MyFitnessPal app [3], which offers a user-friendly interface and integrates various features such as meal tracking, and exercise logging. However, its meal planning functionality may be limited, requiring users to manually search and add individual meals to their plan. In contrast, our solution is to offer a more streamlined and automated meal planning process, generating personalized meal plans that cater to users' dietary preferences, health goals, and convenience needs.

III. Design

Author: Anaz Mohammed

As shown in Fig. 1, this project is composed of 4 components: Login/Authentication, Meal Plan Generator, Search and Favorite Meals, and Order Missing Ingredients. Component 1 handles the registration and login for the user, and authenticates them into the app. Component 2 generates the meal plans for the user and displays it on the screen. Component 3 gives

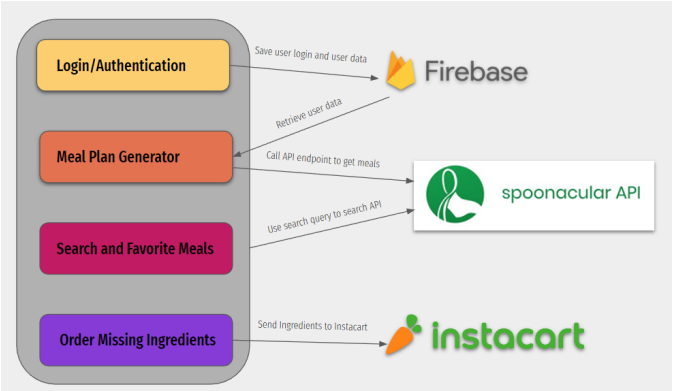

Fig. 1. Framework of the Project [**?**]

the user the ability to search for any meals using keywords and allows them save their favorite meals. Component 4 allows the user to push ingredients to a grocery list so they can easily order them from an online grocer.

A. Login/Authentication

Author: Anaz Mohammed

The first major component is the login and authentication feature. This component is essential to ensure that users can easily create and acces their personalized meal plans, as well as manage their account settings and preferences.

There are two parts to this component. The first is user registration where new users are prompted to create an account by providing their email address and a secure password. Once the account is created, users are then required to fill out a form that collects essential information for customizing their meal plans. This information includes body metrics (such as age, weight, and height), fitness goals (weight loss, maintenance, or muscle gain), and food preferences (vegan, vegetarian, etc). The second part is user login where the user uses the email and password they registered with to login.

In summary, the login and registration component serves as the foundation for our meal planning application, as it enables users to create and manage their personalized accounts securely. This component, combined with the user information gathered during registration, allows the application to generate customized meal plans that cater to each user's unique needs and preferences.

B. Meal Plan Generator

Author: Francisco Cedeno

The design for the second significant component, the meal plan generator, follows a structured approach. We intend to fetch user data from our database, leveraging the information provided during registration, to calculate their daily caloric needs. This will be combined with their food preferences to recommend a tailored diet plan. To facilitate easy access to

the recommended recipes for the day, we will display the recipe name, image, and macronutrient breakdown on the user's dashboard.

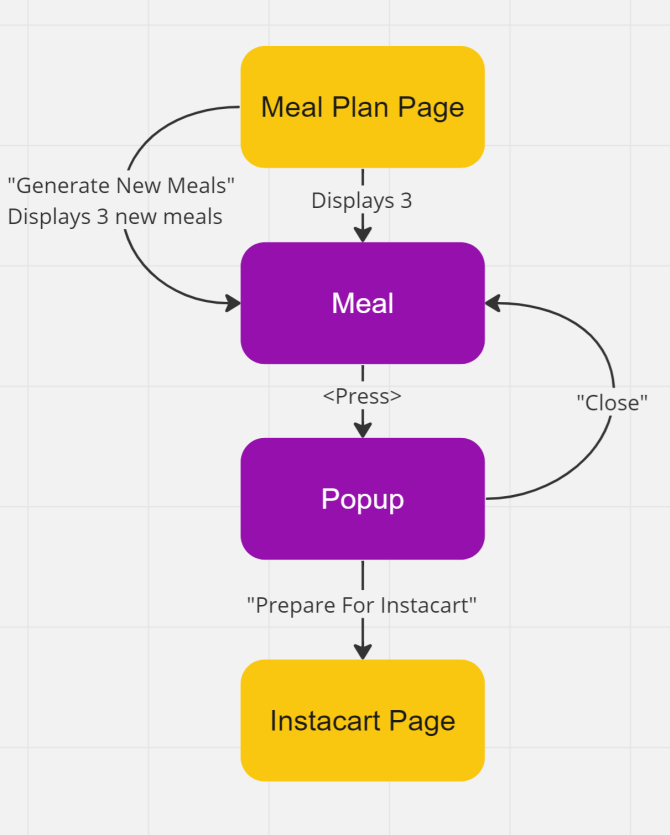

Fig. 2. Design for Component 2.

As we sought to display additional information while ensuring that the user remained on the same page, we chose to implement a popup. This would enable users to view any critical information quickly and exit the popup in a way that allowed them to continue viewing other meals on their dashboard. For the popup, we decided to include ingredients and instructions, considering the variations that come with each meal. We incorporated a scroll feature, enabling users to view long ingredient or instruction lists easily.

With the basic setup in place, we concluded that the daily meals page would display only three meals at a time. This layout would allow us to maintain the desired simplicity and prevent information overload.

C. Search and Favorite Meals

Author: Pouria Salekani

The third major component is the *Search and Favorites* component (which is packaged into one component since they work together).

The design for component 3 is shown in Fig. 6. There are 3 types of users and 5 major functionalities that are supported by this system.

When first displaying the *Search page* it shows no meals since there is nothing connecting it to the endpoint. The design of

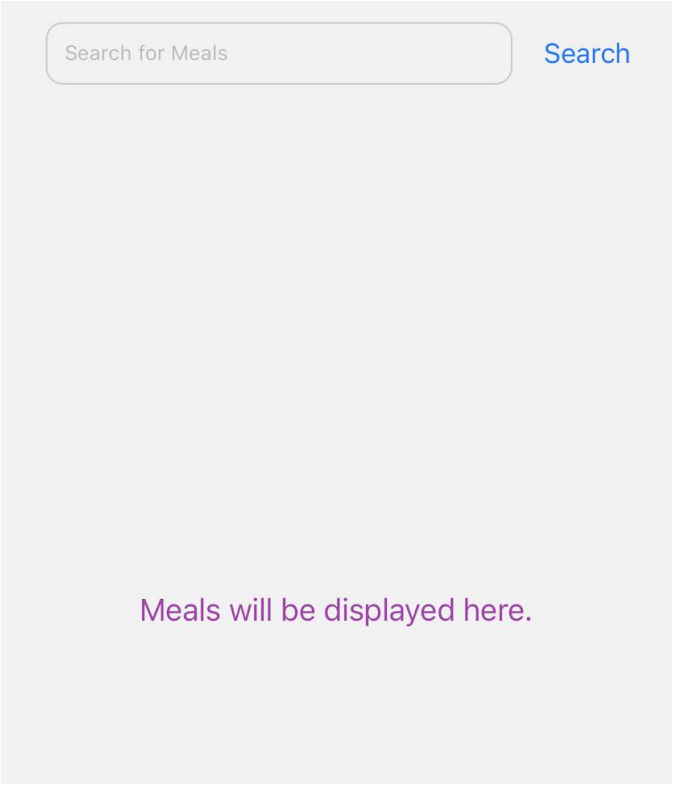

Fig. 3. Design for Component 3.

this component is fairly simple since if the user wants to reach their goals, path of least resistance is a good idea. The user would be prompted to enter any meal they like which **is or is not included in the main meal section (B. Meal Plan Generator**, then the text gets automatically read and sends a query to the *Spoonacular API* to which then the Spoonacular API returns a bunch of recipes pertaining to the user inputted text. Each recipe includes: total calories, macronutrients, recipe image, recipe instructions, recipe ingredients and a *favorites button* will send information over the *"Favorites.js"* AKA *Favorites page*.

The *Favorites page* would display favorites meal that the user favored in the *Search page*. At first, assuming no favorites, the page will be blank (also do remember that you get recommended different meals everyday by default) unless the user adds a favorite meal/recipe to the page. **The favorite recipes/meals are NOT affected by the daily refresh** so the favorite meals always stay there. Furthermore, after the *Favorites page* reads data from the *Search page*, it would then display the same values as done in the *Search page*; total calories, macronutrients, recipe image, recipe instructions, recipe ingredients.

D. Order Missing Ingredients

Author: David Nazaretyan

The fourth and final component of the app is the ability to order missing ingredients from any given recipe using Instacart, an online grocery delivery service. This feature was

created as a concept to break away from other meal planning apps, as none provided a service with freshly delivered ingredients from grocery stores.

Initially, this component was going to be designed using the Instacart API, but upon investigation, it was determined that this method was not possible. The nature of the Instacart API is designed for grocery stores, and as such, this app would have difficulty gaining approval to use it. With this in mind, an alternative solution was theory crafted, namely, the usage of Selenium to automate the web browsing of users to automatically order the ingredients from Instacart's website itself.

To facilitate automatic orders, a page on the app was designed in order to allow for ingredients to be passed on to the Selenium automation. Additionally, the ability of users to select which ingredients they needed from the list was necessary, therefore, removal options were added to each of the ingredients as listed on the app.

IV. Implementation

A. Login/Sign-Up

Author: Anaz Mohammed

Firebase was utlized to implement the login and registration feature. Firebase is a popular, secure, and scalable backend service that enables seamless user authentication, data storage, and real-time data synchronization. Upon successful registration, the collected user information is securely stored in Firebase, which allows for efficient retrieval and updating of user data whenever required. The user authentication process ensures that only authorized users can access their personalized meal plans and account settings. Users can log in to the application using their registered email address and password, and Firebase will handle the authentication process, verifying the provided credentials against the stored user data.

React navigation was used to change the screens once the user logs in. The app was set up to check the user context (whether they are logged in or not). If the user is logged in the they are routed straight to the home screen, but if the user is not logged in then they can only access the login screen or registration screen. This ensures that only users that are logged in can access the app, and this also ensures individual user data is private and only accessed by the user it belows to.

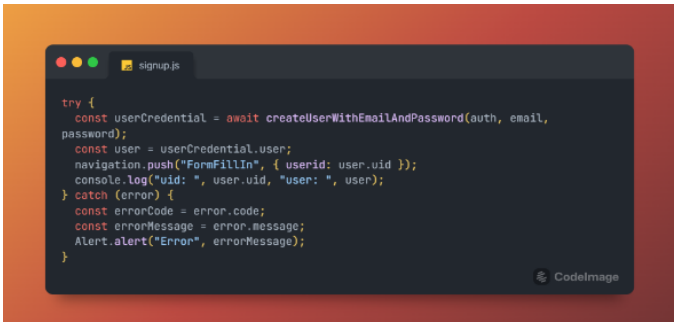

Fig. 4. Code for "signup.js"

B. Generate Meals

Author: Francisco Cedeno

To determine the appropriate caloric intake for a user based on their goals, we retrieved their information from Firebase. This calorie count was then passed to the generate meal plan endpoint, along with any dietary restrictions, to generate a plan of three meals for the day. However, the information retrieved from the endpoint only provided a general overview of the recipe, including the name, source URL, and ID, which we used in two additional endpoints, nutritionWidget.json and information. The former provided the macronutrient breakdown of the meal, while the latter retrieved the image of the recipe. A Promise.all method was employed to combine all the information retrieved into a single use state, facilitating its use.

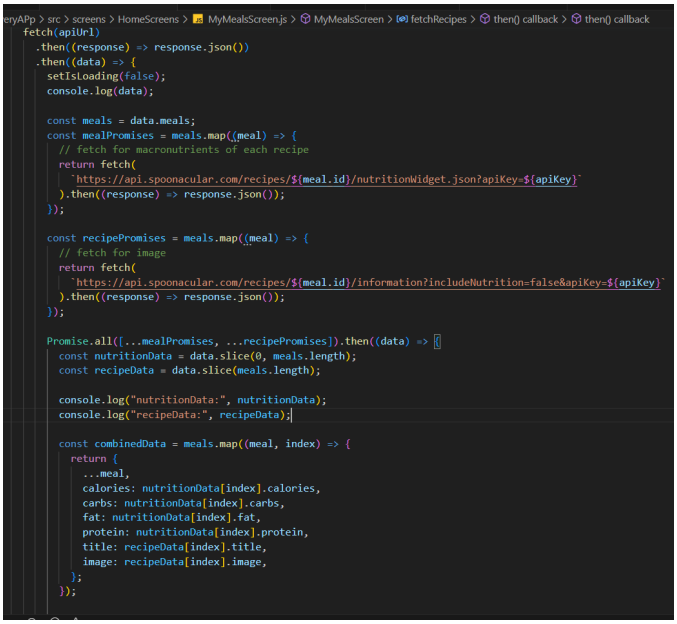

Fig. 5. Code for "MyMeals"

In the subsequent part of the code, we utilized a touchable opacity object, colorizing it purple to enhance its aesthetic appeal and making the text white for greater legibility. We mapped through the recipes, creating boxes for each that neatly displayed all the stored information. The focus of these boxes was to provide the most critical information, such as the recipe title and macronutrient breakdown. Upon clicking on a box, a popup displayed the instructions and ingredients, both of which utilized a Flatlist. The popup also featured a "Prepare for Instacart" button that enabled users to send the ingredients directly to the Instacart page.

To pass the ID to the popup file when a box was clicked, a method was implemented. The popup file itself relied on the information endpoint for retrieving the ingredients and analyzedInstructions endpoint for the instructions.

C. Search and Favorites

Author: Pouria Salekani

The first paragraph will talk about the *search feature*. The search features first starts off with a search bar and asks the user to input a name of a recipe. Suppose the user types in *"steak"* then the search bar would send a **query** to the Spoonacular API endpoint and the endpoint would return *x recipes containing the word"steak"*. The endpoint would feature the macronutrients: protein, carbs, fat and it will also include the total calories for each recipe. Then in order to grab the attributes, a *useState* was called which it would then return the data *in the form of JSON*, after that, in order to use output the data, in the React Native *return (.....);*, the data returned was then mapped. In the end, to access the data, it is like accessing a dictionary, **key-value pair**, so in order to access the returned data, something like this was created: *recipe.nutrients[protein]*. Lastly, the search page can display 5 meals at a time, in order to change that, modifying the endpoint would just suffice.

The second paragraph will talk about the *favorites feature* which works in conjunction with the *search features*. In order to favorite a recipe from the *search page*, a button would there as a prompt, after clicking on the **favorite** button, the information about the recipe is stored inside of React's *navigation* system. The navigation system would then send the information over to the desired screen, in this case, the *favorite page*.

```
const handleButtonPress = () => {
  navigation.navigate('Favorites', {recipe_id, calories_x, title_x, protein_x, image_x, carb_x, fat_x });
};

const openModal = (id, calories, title, protein, carb, fat, image) => {
  recipe_id = id
  calories_x = calories
  title_x = title
  protein_x = protein
  image_x = image
  carb_x = carb
  fat_x = fat

  setModalVisible(true);
};

const shutModal = () => {
  setModalVisible(false);
};

const handleAndShut = () => {
  shutModal()
```

Fig. 6. Code for "Search.js"

As seen above, the *navigation.navigate("Favorites," .....)* will send the information over to the *favorite page i.e. "Favorites.js"*, the second argument takes in what wants to be passed to the other screen. After retrieving the information in the *"Favorites.js"* page, the process to display everything is fairly simple. Since this time what is being passed are the actual values and **not recipe.id or a dictionary** then a simple *Text tag* can display the values of the favored recipe.

D. Instacart

Author: David Nazaretyan

The usage of Selenium to automate the Instacart ordering section was a necessity as previously mentioned due to the incompatibility of the app with Instacart's API. This created some implementation questions regarding how Selenium would be able to take the ingredients from the app and order them automatically while avoiding the robot detection on Instacart's website.

To receive ingredients from the app, a separate page was created where ingredients for any given recipe were listed, with removal options if the user already had them in possession. From there, once a user would determine their list to be complete, they would select the "order from Instacart" option, which would pass the ingredients to the selenium.js file as an array in its args. Once the ingredients were received, they would then be iterated through a loop, and Selenium would search for each one on the Instacart website. After all items in the order were placed in the cart, the user would simply check out, or change the cart as necessary before doing so.

To avoid bot detection, the Selenium code was implemented in a way that allowed the user to log in to Instacart on their own. This way, the user would have secure login through the website itself, and any "are you a robot" prompts could be answered by the human user. Once the login phase was completed, Instacart would no longer prompt the user for bot-behavior, meaning that the speed of ingredients being searched for was instantaneous.

V. Results

To evaluate our project, we conducted 3 experiments. First was to test the meal plan accuracy,the second was to test efficient ways to call API endpoints from react native, and the third was Selenium Automation.

A. Meal Plan Generation Accuracy

Author: Anaz Mohammed

In order for the app to fulfill its promise, it needs to accurately generate meals based on the users needs. We ran multiple tests by generating new meals to observe if the meals returned matched the caloric needs and preference of the user. There was a 90% chance that meals were generated correctly and in the remaining 10% the caloric needs were not hit. However the preference of the user was always accounted for.

B. API Endpoint Testing

Author: Pouria Salekani and Francisco Cedeno

There are two factors that go into this section; finding the best endpoint that will generate the *most accurate* meals for the user and finding multiple endpoints to acquire the meal/recipe information.

First is finding the best endpoint. The testing was conducted on a bunch of dummy users, around 10, by creating profiles for them with some similar preferences and goals and some that were different. It is safe to say that the best endpoint that most accurately returned meals/recipes that lined up with the user's preferences and goals. was the *meal-generate* endpoint. Of course, there had to be implemented some sort of filtration to get rid of unnecessary meals or meals that did not align with the user's goals.

When evaluating the type of information to be displayed, it became evident that certain endpoints did not meet our criteria. After careful consideration of the most critical information, we concluded that it was imperative to showcase the image, title, and macronutrient breakdown, including calories, fat,

carbs, and protein. While this served as the initial display, we recognized that users may desire additional information such as ingredients and preparation instructions. As a result, we integrated a total of three endpoints, namely nutritionWidget.json, analyzedInstructions, and information, to provide a comprehensive view of the recipe. For the search page, we utilized the endpoint complexSearch. Despite testing various search methods, we opted for the query parameter as it proved to be the most user-friendly and straightforward to implement.

C. Selenium Automation Testing

Author: David Nazaretyan

Initial tests of the Selenium automation ran into one small issue, namely, the opening of search recommendations which obscured the results of the search from Selenium, and stopped the automation process. To solve the issue, after searching for an ingredient, Selenium then clicked on an empty portion of the website to close the search recommendations. Following this change, the code was run 50 times with varying ingredients that were described using different keywords. Throughout testing, there were no major errors in the automation. One problem that occurred during testing was that sometimes an ingredient would not accurately be added to the cart, due to improper recommendations from Instacart. One such example was when the search term ”apple” was searched for in some grocery stores, the resulting cart addition was apple juice, as that was the top search result. However, most of the time, the ingredients added to the cart were accurate.

VI. Conclusion and Future Work

Author: Pouria Salekani

To conclude, this project because a healthier lifestyle is a happy lifestyle, which can be attained by intelligent meal planning. We also want people to pursue their lifestyle goals and look their best by holding themselves accountable and using an app that enables them to better track what to food, their favorite food, and to basically automate their desired diet for them. Automating is a big part of this too, it works like a second brain. Recommending recipes based on the user's goals, preferences, and diet, will automate it so it will be easier for them to adhere to the goal. Moreover, the Instacart implementation allows the user to order ingredients of the recipe that they liked, this ensures a path with the least resistance but a smart path.

There is still lots to do with the project, for instance, we want to build a dedicated website for this project (possibly) and for sure, we want to deploy this project to mobile. While we will save those for last, there are some implementations that are yet to be done. One is replacing the meals. Right now the users can favorite a meal that they like, however, if they wanted to replace one of the *main meal(s) page* then they cannot do so. Moreso, once they do replace one of the meals then it'll be *saved for the current day*. Another feature we want to implement is global user states. Having global user states by using Redux, for example, will make it easier to track the user object thus saving us work and time. The last feature we want to implement is having an *order history page* on this page, users can see their ordered ingredients and those ingredients will be mapped to a specific recipe so if they want, they can just click on that recipe and order the ingredients *remember, users cannot really order the recipes off of Instacart, only ingredients*.

After all of those future implementations, we are confident that the app will be at its tip-top shape and while that is the future work that we have determined for now, hopefully along the journey, there will be more exciting future work to come.

References

[1] V. A. G. I. B. A. E. K.-G. S. H. Pauline Ducrot, Caroline Méjean and S. Péneau1, *National Library of Medicine*, 2017.

[2] C.-T. L. M.-K. S. H. Su, T.-W. Lin and J. Chang, “Automatic recipe cuisine classification by ingredients,” *Adjunct Publication*, p. 565–570, 2014.

[3] “Myfitnesspal.” [Online]. Available: https://www.myfitnesspal.com/