Many people with dietary restrictions and medical conditions need to be very cognizant of the nutrients they are consuming. For example, people with high blood pressure need to avoid sodium, and people with POTS need to consume more sodium. Most nutrition guides use a one-size-fits-all approach in nutrition guidelines, but people's bodies have diverse needs.

When I began creating this app, this was the problem I wanted to solve. I wanted to make a user friendly app where it was easy to input what you'd eaten and know how it fit into your nutrition goals. I created a list of every single item that can appear on a nutrition label with the intention of making the goal customizable. I also included calorie and weight tracking support, as most users would likely expect that out of an app that tracks meals.

Apps already exist that can track your nutrition, but most either focus more on tracking fitness and physical activity or are very difficult to navigate and enter data for. As a result, I wanted to make an app that specialized in making it easy to log what you'd eaten and look at a summary of your information. Specifically, I wanted meal entry to be quick, the user's nutrition information to be customizable and straightforward, and recipes to be easy for the user to save and use later.

Unfortunately, my deliverable is more of a demo of what most of the app could look like with its full functionality. The skeleton is there; I did not have the resources to flesh it out. With this in mind, I can discuss what approaches I used and why, and which others I considered. I can also discuss what I would have needed to complete to make the app function the way I wanted to, and how it would have worked.

Most apps have a sidebar that pops out in order to navigate the main categories of an app quickly, so that is what I built my app around. I implemented a DrawerLayout for this purpose; each other page is a fragment that is loaded into the main view when its item in the list is selected. This seemed like the most intuitive way for the user to navigate the app.

I chose the MPAndroidChart API to chart the user's weight and calorie intake over time, as well as to provide a pie chart showing the proportion of carbohydrates, protein, and fats the user ate that day. They are not visible in the app, because they are set to be invisible until the user has entered data; the functionality to enter that data and display it was not implemented, but the objects exist and are waiting. There were better and more robust APIs available, but those ones cost money to use, and based on some research I did early on, this was the best free one available.

To store persistent user data, I used Room. I have tables established for the user's weight, breakfasts, lunches, dinners, and snacks. I didn't want there to be too many tables taking up space with redundant data that would need to be recalculated and refreshed anyway; this seemed like the simplest approach. With timestamped meal data, it would be possible to pull all food eaten on that date. It would also make it possible to sort suggested foods by how recently they were eaten while keeping them separated by category (no suggestions of a steak dinner as a snack, for example). Keeping a nutrients table or calories table would require a lot of recalculation any time any data was ever modified, as well as being bug prone, as it would be very easy to forget to update every table affected by one update in another table. By calculating the number of calories and nutrients consumed on an as-needed basis, fewer table updates would be required, and the data shown to the user would be guaranteed to be fresh and reliable every time.

Before I designed the search or results pages for food, I needed to know which API I would be using. I initially thought I could use the government's public food API, but this proved to be impossible: the data would have needed a ton of sanitizing that I was not equipped to do. Therefore, I had to start looking at other APIs. Unfortunately, any I found that would meet my needs cost money. I found one that would have been free for me and one other person to use, but if both the instructor and another grader tested out the app (as I was expecting), I would have been charged $300. (The API in question is Nutritionix.) At this point, I was beginning to run out of time, so I focused on what I could do without the API; a better alternative likely exists, but I did not have the time to find it.

Some other features I was entirely unable to work on were notifications allowing the user to log the same meal as the previous day from the notification itself and setting timed notifications for each type of meal. The second udacity course we were supposed to go through had a segment on this, so the main difficulty in implementing it would have been ensuring the app had the correct fragment loaded with the proper modal up. A preferences fragment would have been appropriate for the notification settings.

Ultimately, while there is not much of substance to interact with in the final product, I feel I accomplished a lot toward its completion; the main things I needed in order to complete it were an API that provided the information I needed, to create a search page with settings reflecting the API's functionality, provide results to the user with what information the API delivered, and populate a page that would allow the user to interact with that food. Then, I would be able to interface that data with the local database to populate the tables and pull real data from there too; the data pumping classes to populate lists for the user already exist, I just needed data to pump in the first place. While working on this project, I learned about Room and SQLite, multiple nutrition APIs, drawer views, fragments, collapsible lists, and more. I was just unable to implement everything I learned.