**CS683 Project Assignment   
Nutrition Tracker  
Joshua Byrd**

[**Overview**](#_g6igqliy7rm) **2**

[**Related Work**](#_bf21eadgjj29) **2**

[**Requirement Analysis and Testing**](#_9dheewbiht5g) **2**

[**Design and Implementation**](#_312k3b3li0xh) **3**

[**Project Structure**](#_hkcglxnjhrt2) **4**

[**Timeline**](#_tp0jpote18vj) **4**

[**Future Work (Optional)**](#_wx5fnmke6x6g) **4**

[**Project Demo Links**](#_nl6zntsisnrv) **4**

# Overview

For this project, I will be creating an application to track a user’s nutritional intake (calories, protein, carbohydrates, etcetera). Physical fitness and nutrition have long been hobbies of mine, and this will give me the opportunity to work with an idea with which I’m familiar, while creating a challenging and rewarding project.

I have used multiple similar apps over the years, and while they’re often fully-featured and serve their intended purpose, they also frequently come with many drawbacks, such as requiring a user to sign up, requiring a user to sign over the rights to their data, and/or simply being too complicated. With this project, I hope to create an easy-to-use and free nutrition tracker that does not require any sort of sign-up and does not collect user data.

Rather than catering to the fitness professional, my app will be targeted toward general users who simply want a better overview of their food intake, without being overwhelmed by the bells and whistles that some other apps offer.

# Related Work

There are numerous nutrition tracking applications in the Google Play store, including MyFitnessPal, Chronometer, and MacroFactor. They run the gamut of pricing and features, but essentially all allow a user to track their calories and macronutrient intake by letting a user search for foods and then add them to their day. In this, my app will be no different. Users will be able to search a database of foods and add them to their current day, they will be presented with a summary of their intake, and they will be given the ability to search past days to see what they ate. There are three things, however, that will set my application apart from current offerings:

1. My app will be free. While many nutrition trackers offer lots of features beyond the basics, they are often either partially or fully paywalled. Since this is a learning experience for me, this is not a route I’m considering.
2. I will not require users to sign up for anything, and I will not attempt to collect their data. Data collection has become standard practice for mobile applications, and I think it’s something that many users dislike, especially when they just want a utility that works.
3. My app will be simple and easy to use, (hopefully) offering a great experience without being overly complicated. While it’s common to position *simple* as synonymous with *featureless*, I think it’s a good thing in this case. Most people just want to be healthy, and giving them the ability to track their calories, macronutrients, and common predictors such as saturated fat and fiber, is enough. Tracking micronutrients down to the picogram is neither necessary nor productive, and often just takes up space in an already too-busy UI.

# Requirement Analysis and Testing

| *Title* | *View foods eaten and nutrition summary (Essential)* |
| --- | --- |
| *Description* | *As a user, I want to be able to see a list of foods that I’ve eaten on the current day, and and summary of the nutrition (calories, protein, etcetera)* |
| *Mockups* |  |
| *Acceptance tests* | *When the user opens the app, they are immediately presented with an activity displaying a list of foods eaten and nutritional summary* |
| *Test Results* |  |
| *Status* | *The daily display is fully functional. It monitors the state of its ViewModel, updating the list of foods eaten and the summary when they change. It also has a fully functioning DatePicker, allowing a user to select which date to display, and updating the UI accordingly.* |

| *Title* | *Search for foods to add (essential)* |
| --- | --- |
| *Description* | *As a user, I want to be able to search the database for specific foods that I’ve eaten* |
| *Mockups* |  |
| *Acceptance tests* | *Given that the main activity (food list and summary) is displayed on the screen, if a user clicks the “Add Food” button, they are presented with a new activity displaying their most current food choices and a text box allowing them to enter a food and search the database for it. When the user clicks the search button, the app connects to the food database API and returns a list of foods based on their query.* |
| *Test Results* |  |
| *Status* | *The FoodSearch activity is now fully functional. Displayed is a text box where the user can enter a food name and search for it, and a list of the most recently eaten foods below. When a user enters a food, the database is queried and a list of food returned, which are then displayed in the FoodSearchResult activity. Additionally, clicking any of the foods in the recent food list takes the user to the AddFood activity for that food where they can then add it to their day.* |

| *Title* | *Add food to current day (Essential)* |
| --- | --- |
| *Description* | *As a user, after searching for a food and obtaining a list, I want to be able to add the foods that I’ve eaten to my list for the current day.* |
| *Mockups* |  |
| *Acceptance tests* | *Given a list of foods that a user has searched for (or the list of most recent foods), when a user clicks on a specific food, they are given the option to add it to their list of daily foods. If they click this add button, the food is added to their list of foods for that day.* |
| *Test Results* |  |
| *Status* | *The AddFood activity is fully functional, displaying the details of the food that is passed to it, and giving the user a text box to enter the number of servings of the food they’d like to add. When the ‘Add’ button is clicked a new DailyEntry is made in the database and the user is taken back to the main DailyDisplay.* |

| *Title* | *View foods eaten on past days (Essential)* |
| --- | --- |
| *Description* | *As a user, I want to be able to see a list of foods that I’ve eaten and a nutrition summary for a specific day.* |
| *Mockups* |  |
| *Acceptance tests* | *Given that the main activity is displayed on the screen, if the user clicks the “View journal” button, they are presented with a new activity that allows them to search for a specific date. If they enter a specific date, they are then presented with the main activity populated with data from the specified date.* |
| *Test Results* |  |
| *Status* | *The ability to display past days is now fully functional. Using a DatePicker, when the date in the top bar is clicked, the user is presented with a calendar that they can use to select a day. When a day is selected the UI updates to information about that day.* |

| *Title* | *Create custom meals (Desirable)* |
| --- | --- |
| *Description* | *As a user, I’d like to be able to create named custom meals, so that I can add all the foods at once instead of having to search for each item* |
| *Mockups* |  |
| *Acceptance tests* | *Given that the main activity is displayed on the screen, if the user clicks the ‘Custom Meals’ button, they are presented with a list of custom meals and the option to create new ones. When a meal is selected, they can click another button and add all foods to the current day*  *Given that the user has searched for a food, when a user clicks on a food, they are presented with an option to add the food to a custom meal. If they choose this option, the food is added to the specific meal.* |
| *Test Results* |  |
| *Status* | *Not implemented* |

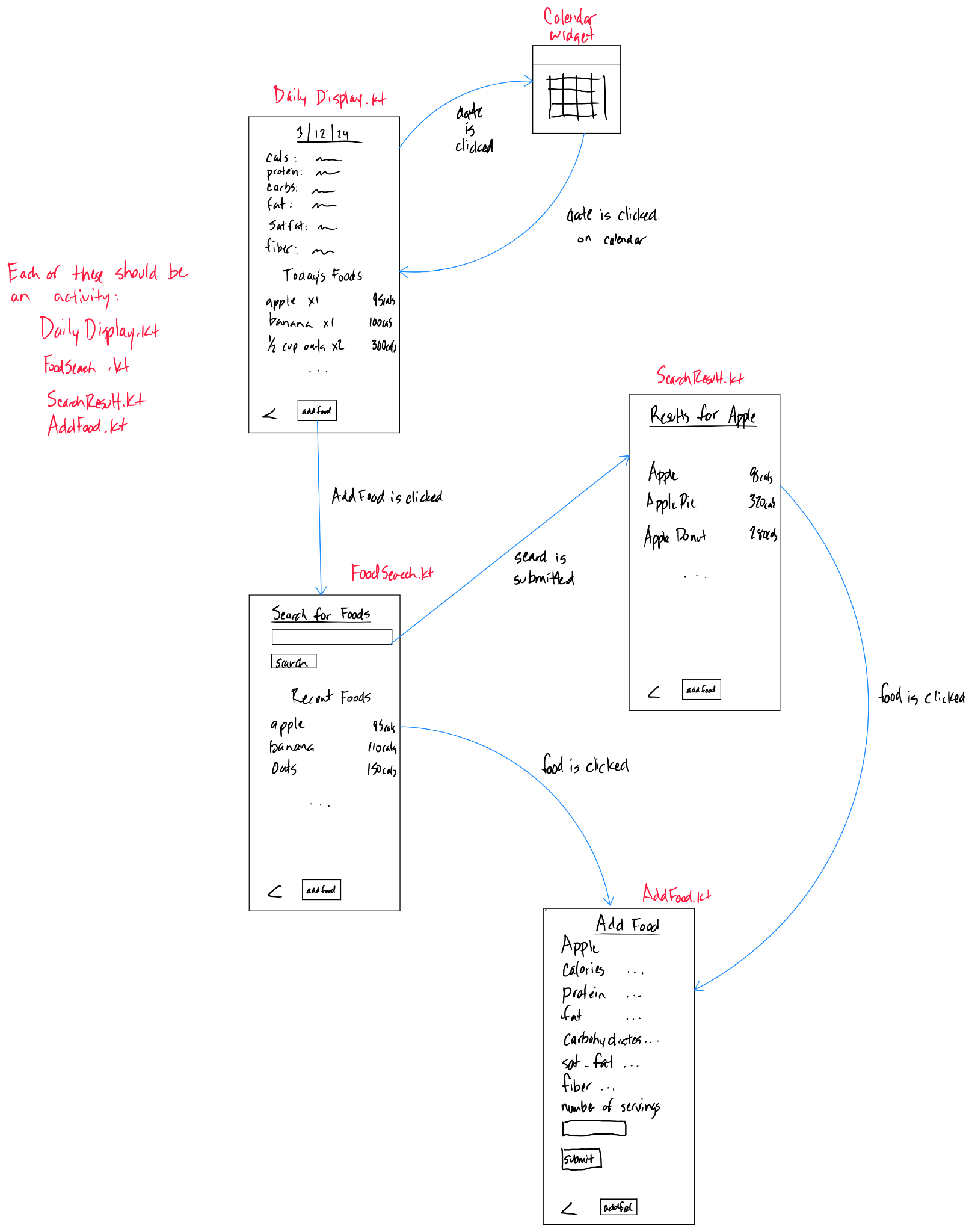
| *Title* | *Track biometric information (Desirable)* |
| --- | --- |
| *Description* | *As a user, I’d like to be able to track my weight, bodyfat percentage, and, and other body measurements* |
| *Mockups* |  |
| *Acceptance tests* | *Given that the main activity is displayed on the screen, if the user clicks “Add biometric information”, they are presented with a new activity that allows them to enter body information. When entered, this information is stored for later retrieval* |
| *Test Results* |  |
| *Status* | *Not implemented* |

| *Title* | *View data summaries graphically (Optional)* |
| --- | --- |
| *Description* | *As a user, I’d like to be able to view graphical summaries of my information, such as macronutrient percentages as pie charts or line charts of biometric information.* |
| *Mockups* |  |
| *Acceptance tests* | *Given that the main activity is displayed on the screen, if the user clicks the “View statistical information” button, they are presented with a new activity with various options such as “View nutrition information for the last month” and “View weight change over the last six months”. Clicking one of these will present the user with the appropriate chart.* |
| *Test Results* |  |
| *Status* | *Not implemented* |

# Design and Implementation

Update 3/28/2024 (Iteration 1) - As the course has progressed, and we have learned more about the Model - View - ViewModel (MVVM) architecture, it has become clear that this is the standard in Android development, and so this is also the architecture that I will use. Most of the basic functionality will remain the same, with some notable exceptions below.

**Update 4/13/2024 (iteration 2)** - At this time, the app and all its main functionality has been implemented using an MVVM architecture. The database has been created and all UI activities are updated through the ViewModels and their associated repositories.

* ***Functionality*** -The user will be presented with the main activity upon opening the app. This will display their foods eaten that day, as well as a summary of their nutritional intake. Additionally, there will be a small menu that allows them to add food, view past days’ intakes, and possibly add biometric information and create custom meals (these are desirable features at this time). **Update 4/13/2024 (Iteration 2)** - All main functionality has been implemented. I had hoped to get to some of the desirable features in the final iteration, but delays due to bugs have likely made this impossible.
* ***Activities*** -In addition to the main activity (DailyDisplay.kt), there will be activities for the search page (FoodSearch.kt), for the results of a search (FoodSearchResults.kt), for displaying a food to add (AddFood.kt) and for choosing a day in the past to display (the main activity can be reused to display any particular day’s information). If biometric and custom meals are implemented, there will be activities for these as well. Update 3/28/2024 (Iteration 1)- In the course of planning for the app architecture, I have created the following state diagram showing each of the activities and how they relate to one another: **Update 4/13/2024 (Iteration 2)** - All Activities are functional in accordance with the state diagram above.
* ***Android-specific features*** - To view past days’ information, the app will feature a calendar that allows the user to choose a specific date which will be used to query the database and populate the activity with the appropriate data.
* ***Data Storage*** - User-specific data will be stored locally in an SQLite or Room database. The database will contain a table for storing foods eaten (with an accompanying date for querying) and recent foods for displaying on the search page, as well as potentially tables storing biometric information and custom meals. General food information will not be stored locally.

Update 3/28/2024 (Iteration 1) - Basic food information might be stored locally. See below.

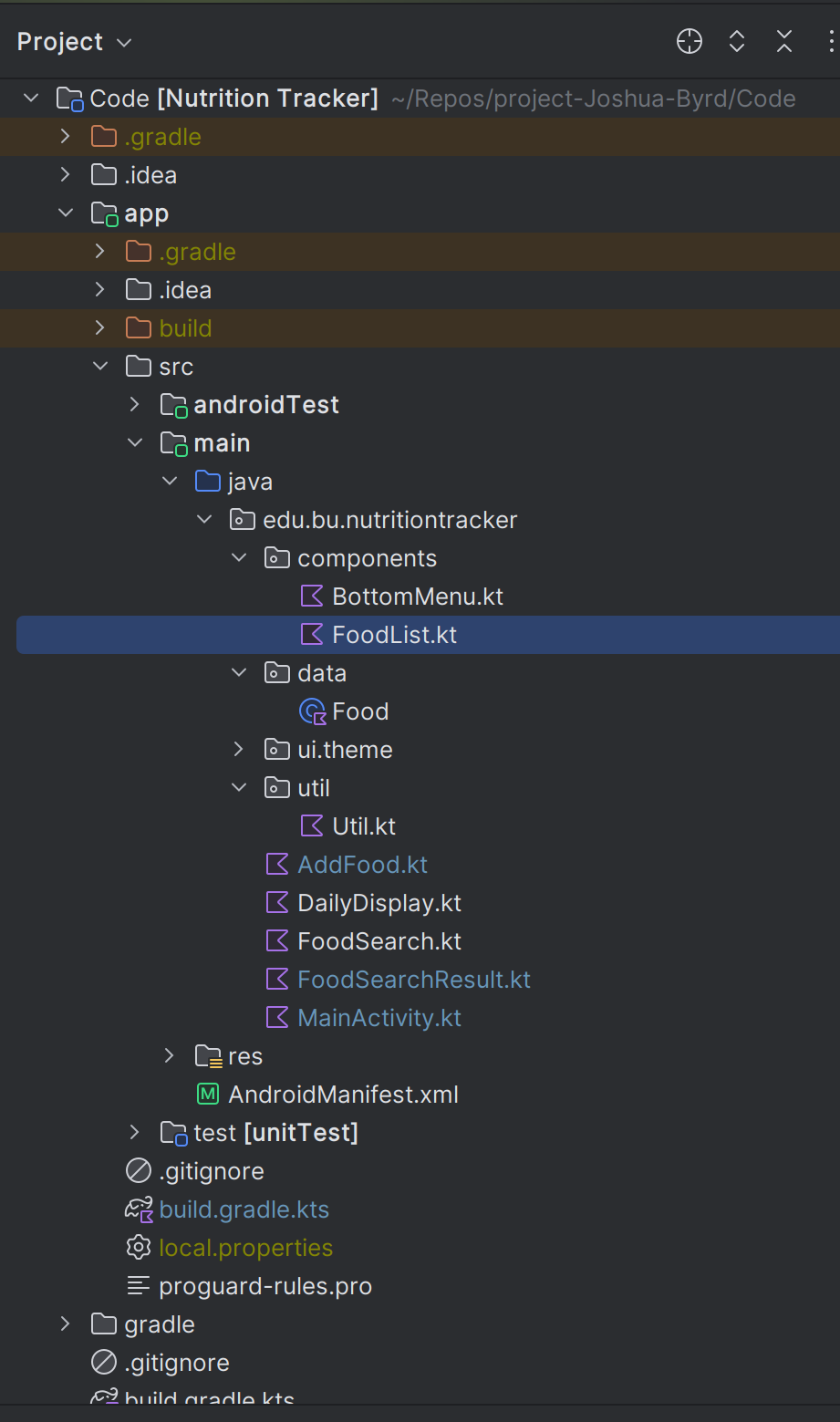
**Update 4/13/2024 (Iteration 2)** - All information is stored locally. The database contains two tables, *Food* and *DailyFoods*. Fittingly, the *Food* table contains data about individual food items, with the following fields: foodId, name, serving, calories, protein, totalFat, carbohydrates, saturatedFat, and fiber. The *DailyFoods* table contains entries that a user has made for a specific food on a specific date. It contains the following fields: foodEntryId, foodId (a foreign key to *Foods),* numServings, and date. I have implemented an initializer function to read food items from a local CSV file and prepopulate the *Food* table; However, this is **not**functional at this time, and all entries in this table have been made manually to have some data to work with. In the next iteration, I will look into why this function isn’t working.

* ***Third Party APIs***  *-* When a user searches for a food, the application will connect to a third party API to query for food information. The results will be displayed to the user and information about a user’s chosen food will be extracted, and transformed into data to be inserted into the local database (as a daily entry).

Update 3/28/2024 (Iteration 1) - As I’ve begun to lay out the basic infrastructure of the project, planning out how data will be constructed and interacted with, I’ve started to question the wisdom of using a third party API for accessing food data. It occurred to me that no matter how I go about it, I will always have to store food items - recently accessed food items, daily food items, etc. While I could take a food item obtained from an API and transform it into a Food object in the app and insert it into the database, this automatically creates some referential integrity and redundancy issues. Because a person might eat (for instance) an apple many times over the course of a year, there would be that number of “apple” entries in the database, without referencing a specific “apple” entity. Although I’m not certain that this would lead to problems in the future, I’m not very comfortable with this ambiguity.

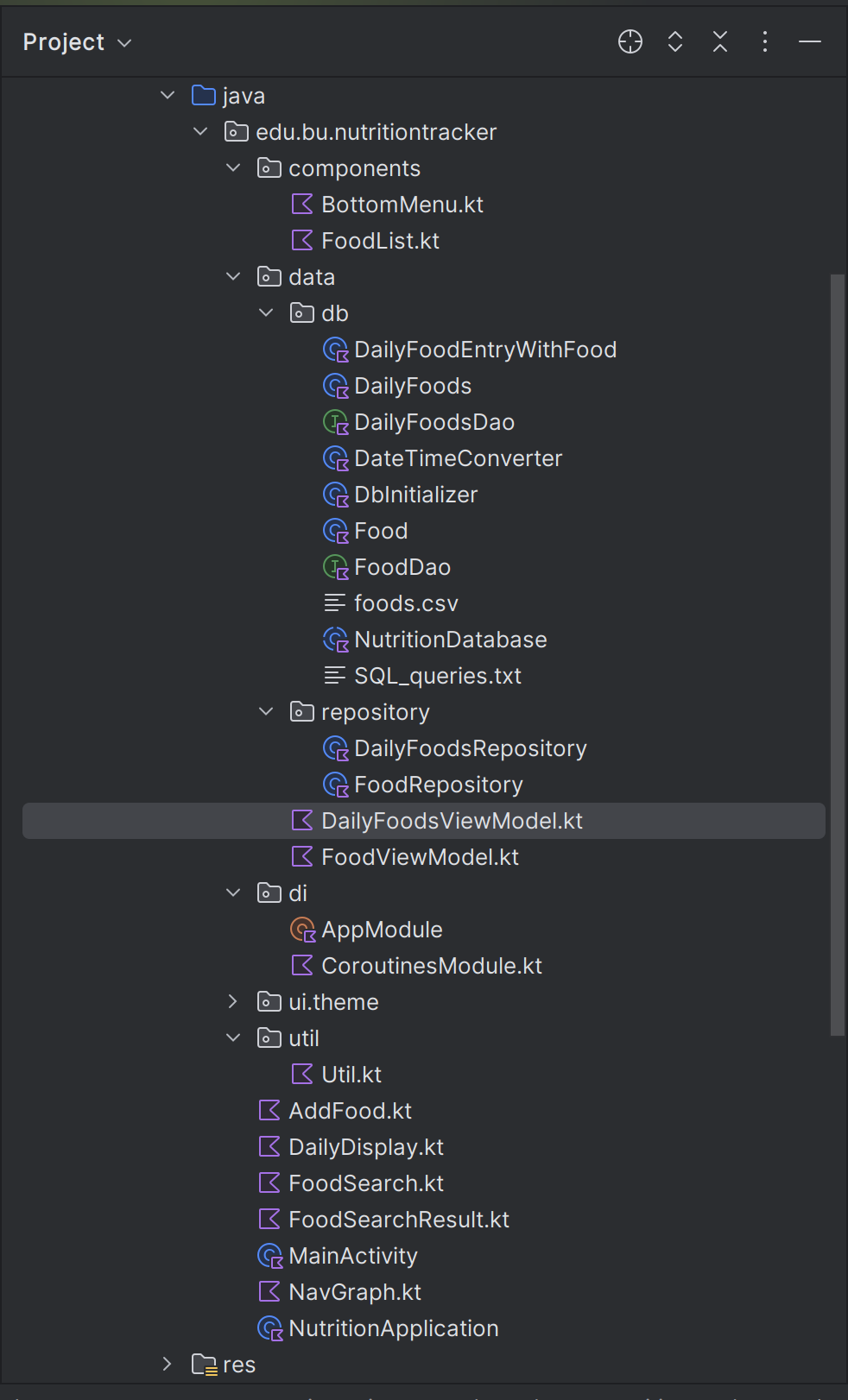
Of course, the big advantage to using a third-party API is that the user doesn’t have to store a potentially massive database of food items. However, I think that given that this is text data, and each entry would only consist of seven fields, a reasonable amount of food data could be stored without severely encroaching on the user’s storage space. At this point, I’m not sure what is the correct way to go about this, and I would welcome any input from the facilitator or professor. **Update 4/13/2024 (Iteration 2)** - The app does not utilize a third-party API.

# Project Structure



Update 3/28/2024 (Iteration 1) - In this iteration, I have begun to lay out the basic project structure. I have created four activities that will serve the bulk of the functionality: AddFood.kt, DailyDisplay.kt, FoodSearch.kt, and FoodSearchResult.kt. Additionally, I have created a components directory which is currently storing a composable for a bottom menu used throughout the app, and a composable that displays a list of foods that is also used multiple times. I have created a data directory that currently contains the Food class, but will eventually all house Models and ViewModels, as well. Finally, I have created a util directory and util.kt file that contains utility functions that are used elsewhere in the app (at this, they are mostly functions creating lists and maps of foods for testing the displays).

**Update 4/13/2024 (Iteration 2)**

****

At this time, all MVVM structure has been implemented. Basic structure consists of the following packages:

* *data*, there are two ViewModels correlated to the tables in the database, and subdirectories for the database (*db*), and repositories (*repository*).
  + *db* contains the entities and data access objects for the database as well as classes for initializing the database and converting DateTime objects (because SQLite does not have a date data type). There is also a class DailyFoodEntryWithFood that represents a join operation between the two tables. This is necessary to be able to display a given food with the date it was on which it was entered.
  + *Repository* contains two repositories associated with the ViewModels
* *Components* contains reusable composables. However, the bottom menu is no longer in use as I have implemented the Activities using Scaffolds
* *di* contains the modules used for Hilt dependency injection
* *util* contains a file that houses several utility functions.
* Finally, the main package contains all Activities.

# Timeline

As I am new to Kotlin and Android development, construction of the project will follow the course curriculum. That is, UIs will be implemented generally, then data storage and APIs, and finally ancillary features such as the calendar for selecting days to display. Testing will be conducted throughout.

**Update 3/28/2024 (Iteration 1)**  - At this time, the timeline remains the same. I have worked on all goals that were set for this iteration, and completed most of them, so for now, I will continue to proceed as planned. As we are coming up on module 4 and data persistence, over the next iteration, I will be working on building the database, finishing the navigation, and implementing the ViewModel/state functionality.

**Update 4/13/2024 (Iteration 2)** - At the beginning of this iteration, I implemented the database entities and DAOs, as well as the Hilt dependency injection. However, the project came to a complete standstill after this, as I ran into a problem with my Gradle build file that took nearly a week to resolve (many thanks to my facilitator and professor for their help with this). After a three day sprint of work, I have nearly caught up with my original timeline, and at this time, all of the essential functionality is working. In my original timeline, I had left this final iteration for implementation of the API functionality, but since all data is now stored locally, I can spend that time tweaking and testing the app and improving the UI. Having lost so much time, it’s not realistic to think that I’ll get to the desirable features, but if time permits, I will try. I would rather have a professional, and complete app with less functionality than to have multiple half implemented features.

| Iteration | Application Requirements  (Essential/Desirable/Optional) | Android Components and Features to be used |
| --- | --- | --- |
| 1 | Implement Activities/UIs for all essential features  Research potential APIs to connect to to obtain food information.  Research calendars in Android development. | Activities and composables |
| 2 | Implement SQLite database and all tables.  Create functions to insert food entries and query the database for past days’ information.  Implement a calendar for selecting past days. | Database  Android calendar  navController/navGraph  ViewModel |
| 3 | Implement functionality for connecting to API.  Time Permitting, implement custom meals and biometric activities and tables.  Perform final testing. | Food/nutrition API? |