There were no new Big User Stories, User Stories, or Developer Tasks introduced in this iteration because we felt that the priority was to finish up what we had.

Big User Stories/User Stories/Developer Tasks not mentioned here were DEPRECATED. This list represents the final working-features that were released with Iteration 3. If a feature is not on this list, it is because it was scrapped due to time constraints.

Big/User stories that were completed in this iteration:

Detailed User Story: Enter Meals

Description: Allow the user to enter details of a meal eaten using foods from an pre-existing list, and entering serving size.

Priority: High (Since creation)

Cost: (see developer tasks) 4 days (changed from 8 days in Iteration 1)

Created: Iteration: 0

Completed: Iteration 3

Discussion: We rehashed the layout view for entering foods into an entering-meal screen. We felt that the user only really needs to enter a food and a portion size. The rest of the calculations would be handled by us.

Violations from Previous Plan: We did not give the user many options to enter as was planned. This felt too complicated so we minimized the options. We also consequently did not allow new foods to be entered because we felt the database was populated enough. Additional information would only introduce unnecessary bugs that would be too expensive to fix with the given team velocity.

Developer Tasks: See wiki

Detailed User Story: Display Eaten Meals

Description: Display the records of all previously entered meals, separated temporally.

Priority: High

Cost: (see developer tasks) 8 days

Created: Iteration: 0

Completed: Iteration 3

Discussion: Queries the database and extracts information based on the date the food was entered. That is, all foods within a date range are listed with their corresponding nutrient values (split between the pie chart elements).

Violations from Previous Plan: The list is not interactable as was previously planned. It is just a list of foods that were eaten in a date range. The meals no longer have a time entered either. Foods are not viewable after 30 days and date ranges are fixed to today (day), the last 7 days (week), and the last 30 days (month). Many possible functionalities were scrapped in order to deliver a working product.

Developer Tasks: See wiki

Big User Story: Pre-Loaded Foods

Description: "As a user, I want to use foods that are already in the application for dietary tracking."

Priority: High

Cost: (see developer tasks) 10 days (changed from 3-4 days in Iteration 2)

Created: Iteration 0

Completed: Iteration 3

Discussion: The pre-loaded foods were very difficult to implement because of the CSV files and the inconsistencies with the Canadian Nutrient File database.

The tests for this class involve outputting the contents of the memory to file per change. Android also requires a second table called Android_MetaData be present in the database, which gave a lot of errors working with.

There was also the problem of how to use the database between contexts. This was saving the latest database instance of DatabaseDefintion into a static member of DatabaseDefinition. This way, the business objects (which have trouble accessing contexts) can still access the database without having to pass some kind of context or throw around the same DatabaseDefinition object.

The number of columns in the tables were shortened from 150 to 60 as well by removing uncommon nutrients.

Violations from Previous Plan: None

Developer Tasks: See wiki

Big User Story: Push Notifications

Description: “As a user, I want push notifications to remind when I’ve made a goal or when I’ve forgotten to enter data.”

Priority: Low

Cost: (see developer tasks) 3-4 days (changed from 8 days in Iteration 1)

Created: Iteration 0

Completed: Iteration 3

Discussion: Push Notifications use a factory class in the HTNotification to produce a message using a member function .throwMessage(). The notification will display in the notifications area as a dismissable expandable list-item.

Violations from Previous Plan: Notifications were originally supposed to tie together with Goals and throw notifications when an entered food contained nutrients that exceeded some watching-value. However, this proved too difficult to complete. We instead re-used the class to give a notification on new meal entry.

Developer Tasks: See wiki

Detailed User Story: Display Notifications

Description: The application will notify the user (through the action bar) when a user has met or exceeded their daily/weekly goals.

Priority: High

Cost: (see developer tasks) 4 days (changed from 10 days in Iteration 1)

From Iteration: 0

Completed: Iteration 3

Discussion: This user story simply represents a medium for where notifications are to be displayed. That is, notifications could have been through Toasts or Snackbars, but the customer requested it be in the Action Bar, so we did that.

Violations from Previous Plan: None

Developer Tasks: See wiki

Big User Story: Statistical Information

Description: “As a user, I want to be able to see a breakdown of the foods I have eaten and recorded based on qualifications I desire to see like carbs, protein etc.”

Priority: High (changed from medium in It0)

Cost: (see developer tasks) 10 days

Created: Iteration 0

Completed: Iteration 3

Discussion: We had a lot of trouble displaying the right percentage points for the different pie-chart elements and trying to factor the units into the calculation, but we were able to successfully implement this feature. Clicking the pie-chart elements brings up more details about the foods contributing to that food (based on the legend). The “Other” section has all the other nutrients that, if displayed on the pie chart, would be below a threshold value for being so small (eg alcohol 0.1grams when proteins is 210grams). The chart spins too, which is a neat aesthetic bonus of using the mpchart library.

Violations from Previous Plan: None.

Developer Tasks: See wiki

Detailed User Story: User Information Submission Form

Description: Allow the user to submit information about their weight, age, and height in a form.

Priority: Medium

Cost: (see developer tasks) 3 days (changed from 8 days in Iteration 1)

Created: Iteration 0

Completed: Iteration 3

Discussion: The user-information screen is within the Settings activity. It is pre-populated with information that the user can change if they choose to.

Violations from Previous Plan: The user’s information are not used in any way in this app. They are there for record keeping – information that is readily available to display and change but serves no real purpose besides being consistent between application runs. The original plan was to use the gender, height and weight measurements with the calculations to produce values that would interact with the goals and set soft-limits on how much calories you should eat or if your BMI says you are underweight, for example. This proved to be too expensive given our team velocity and, since it was a low priority feature, we decided to minimize its functionality.

Developer Tasks: See wiki

Detailed User Story: Display Nutritional Information Graphically

Description: Display the user's statistics in graphical format (pie charts and bar graph).

Priority: High

Cost: (see developer tasks) 3 days (changed from 15 days in Iteration in Iteration 1)

Created: Iteration 0

Completed: Iteration 3

Discussion: This user-story was a desired implementation of how to display nutrient information. That is, the customer requested that the information not be given to them as difficult-to-digest raw text. They did still want information about the nutrients, but not too much. They wished for the information to be displayed graphically, so we used the mpchart library to create pie-charts to render the data. Clicking the pie-chart elements brings up more detail about the contributing foods to that nutrient pie-chart sum.

Violations from Previous Plan: None

Developer Tasks: See wiki

Big User Story: Button Sounds

Description: "As a user, I would like to hear something, some sort of sound, when I click on the buttons of the app."

Priority: Low

Cost: (see developer tasks) 2 days

Created: Iteration 2

Completed: Iteration 3

Discussion: The customer requested that they hear some sound-verification that an option was clicked when they saw the Iteration 2 release. We were able implement this feature into the app but not without several bugs emerging that needed immediate attention. Sometimes the sound would continue playing in the background, even when the screen was off and the app closed. The first iteration of button sounds were also very "jumpy" sounding to the customer, so we lowered the volume and changed it to a more pleasant sound. The customer was the one that chose this sound.

Violations from Previous Plan: This user story came really late into Iteration 2 and we lost a lot of time trying to implement it when bigger user stories needed completing. Despite the priority being low, the team did feel that a sound verification on button clicks was missing from the app to make it nicer to use, so we implemented it sooner despite the lower priority.

Developer Tasks: See wiki

Big User Story: Splash Screen

Description: "As a user, I would like to see a nice customizable pop-up screen as I run the app."

Priority: low

Cost: (see developer tasks) 1 days

Created: Iteration 2

Completed: Iteration 3

Discussion: The logo for the app spins on start-up for the app.

Violations from Previous Plan: The logo was only meant to be a pleasant opening screen but it turned into a dependency for the project. In the onCreate method for the Splash screen, the database is loaded and saved. Without this, the MainActivity, which loads the pie chart and the pie-chart values, began crashing. The crash was due to Android threading the two Activities (MainActivity layout inflating and loading the database) and requesting values from the database before they have even been loaded.

But by loading the database during the Splash screen, the app stopped crashing. Recall that the database is external and on first launch, needs to be OutputStreamed into the right app database path. Our guess is Android does not thread activities if they are more than 2 clicks away, as if buffering the immediately clickable activities. Because of the database loading on Splash screen, the first run of the app will take several seconds before loading.

Developer Tasks: See wiki