Major Changes

Overall view of the features implemented (seen in UI):

- SearchFoodActivity created an activity that searches the food database and allows a user to add from the database.
- AddFoodActivity updated the food activity to add to the database and can only be reached from the searchFoodActivity
- CreateMealActivity allows the user to create a meal (meal is comprised of other edibles)
- We added exercise to the timeline and have an activity as well as changes to the timeline activity.
- Added an account view that allows the user to change their personal information
- Added a goal view that allows for the deletion and addition of goals. As well as allows you to view your current goal status.

Features that were implemented that can't be seen through UI:

- Removal of unused goal factory package (may be readded later if needed).
- The database was implemented this iteration which allows us for persistent storage.
- Exercise logic was introduced in the calculator to account for calories that can be burned.

Developer Tasks

Total Estimated Time: 118

Completed Estimated Time Completed: 95

Incomplete Estimated Time: 23

Team Iteration 2 velocity: 95 estimated hours

Developer Tasks Completed:

6Hr: Show a summary of current goals and current food information (for a day)

10Hr: add search bar to query database (in addfood) #92

5Hr: Create UI for creating a meal

2Hr: Populate the database with foods

1Hr: Create Nav Menu

8Hr: Update Personal Information

2Hr: Make inputs required as needed and check input errors.

6Hr: Access classes tests

2Hr: Create an add exercise activity

2Hr: Don't allow duplicate exercise/goals in a day

5Hr: Add exercise onto the timeline activity.

1Hr: Refactor Goal ids

4Hr: Create the UI for adding, deleting and updating goals

2Hr: Write all queries for db

2Hr: Adding an existing food should not add a duplicate.

2Hr: Clean up Calculator code.

5Hr: Turn Meal into a composite

10Hr: Design HSQLDB

3Hr: Wrap lists and maps in getters and setters

3Hr: Make data structures private in all domain objects

4Hr: Implement Data Access Interface

1Hr: Hotfix DataBase

2Hr: Fix UI for Add Food/Meals

2Hr: Fix UI for Goals

1Hr: Create App Color Theme

Dev Tasks that got postponed:

4Hr: Update UI for total calories in a day

4Hr: Sum up average analytics from a span of days

5Hr: Create a view analytics activity

6Hr: Create Preset Meals to add to a new day automatically

4Hr: Show a summary of goals achieved and food information over a span of days

Outstanding Issues

Iterator for a day’s exercise and goals. Currently while the list of exercise and goals are private in days and there are accessors to add and remove items from these lists we still have a method that returns the lists themselves in order to iterate over them. This should be replaced by returning an iterator as talked about in the design pattern lectures.

fragments for persistence layer - We have some repeated code in the ui that can be reused in fragments.

User Stories

As a user I want to be able to view and modify any day (NEW)

As a user, I would like to be able to see analytics for the food I've documented. (total calories, macronutrients, etc...)
As a user I want to be able to view my total statistics for a day
As a user I want to be able to easily be able to view my total statistics over a span of days (Incomplete)

As a user, I want to be able to track my exercise and view the calories I burned.
As a user I want to be able to track the calories burned in a day

As a user, I want to be able to save my frequently eaten meals along with their nutritional facts.
As a user I want to be able to search for food and add it to my day
As I user I want to create custom meals

As a user, I want to be able to customize different calorie or macronutrient goals to suit my needs.
As a user I want to be able to add, update, and delete my goals
As a user I want to be able to view the analytics for my goals (Incomplete)

Iteration 1 Postponed detailed User Stories

As a user, I would like to be able to save my personal information to my own account.
Allow users to give timely weight updates and make any changes to goals or restrictions at will.

As a user, I want to be able to save my frequently eaten meals along with their nutritional facts.
Allow a user to create default or preset meals that they can use in the future to make logging easier. (Incomplete)

Notes: We decided to allow users to be able to modify previous days. This would be useful if the user is busy during a day and forgot to log their food or just wanted to see what they’ve eaten for that day. We postponed showing advanced analytics and allowing automatic adding of edibles to a day since we felt it was more important to first add the functionality of being able to add edibles to the database, and then adding each of those edibles to a day, including creating new user meals.

Meetings

Meeting 1 – July 14
All Present

- Discussed how we want to do our personal dev log. We decided we would include the
- following:
    - Date the PR was made
    - Date the branch was merged
    - We want to log estimated time, and actual time taken
    - Summary of task
    - Any problems we encountered
    - Who we pair programmed with
    - Debugging
- Chose which user stories we are completing for iteration 2
    - Also breaking down the big user stories into detailed user stories
- Distributed tasks
- Next meeting: July 15

Meeting 2 – July 15
All present
- Discussed if we want to normalize labels for meals, or allow a user to enter custom labels themselves
- Created dev tasks for user stories, assigned priority and assigned a time to teach task.

Meeting 3 – July 26
All Present
- Went over the iteration 1 feedback.
- Discussed design options, whether we want add/remove edible or set edible. We decided to keep the addEdible and removeEdible.
- Discussed design options, whether we need AccessExercise. We decided not to include AccessExercise, as all that functionality would instead be included in AccessAccount.
- Assigned dev tasks.

Meeting 4 – July 30
Arvind, Mark, Emmanuel, and Caden Present. Amin absent.
- Finalized Iteration 2 documents
- Discussed tasks we need to complete before handing i

<u>Emmanuel’s Log</u>
Task: Create an AccessLabel object
Completed July 25th
Pull Request was included with the dev task below.
Time took: 1 hour

Task: Implement DataAccess for the Stub Database
Completed July 26th
Pull Request made July 26th
Branch Merged July 27 th
Time took: 2 hours

Task: Write Tests for Data Access
Completed July 27th
Pull Request was included with the dev task above.
Time took: 2 hours (2 devs pair programming with Mark Shinnie)

Task: Write Tests for Copy Constructors
Completed July 27th
Pull Request made July 27th
Branch Merged July 27th
Time took: 1 hour

Task: Create a UI to View, Add, or Remove Goals to a Day
Completed July 28th
Pull Request made July 28th
Branch Merged July 29 th
Created a UI to view a list of goals for a day, and view which goals were passed. Also added popups to
create a new Label, Calorie, Micro, or Macro goal. Added the ability to remove a goal from a day.
Time took: 6 hours

Task: Create a UI to Create a Custom Meal
Completed July 30th
Pull Request made July 30th
Branch Merged July 30 th
Created a UI to build a custom meal. The edibles get added to a checklist, and the user selects which
edibles they would like to add to the custom meal.
Time took: 4 hours

<u>Caden's Log</u>
Task: Create Nav Menu
Completed July 29th
Time allocated: 1 hour
Time took: 1 hour
Notes: Added only to Timeline for now, will be separating it out into its own activity for use on every page.

Task: Update Personal Information
Completed July 30th
Time allocated: 2 hours
Time took: 2 hours
Notes: I decided to only allow edits of certain info that makes more sense to change.

Task: Error Check Inputs (make inputs required)
Completed July 29th
Time allocated: 2 hours
Time took: 2 hours
Notes: In the previous iteration, there were inputs that could crash the app if left blank or if filled out in the wrong format. We now check inputs before submission in every activity for errors. This makes use of the built in error property of EditText.

Task: Create Add Exercise Activity
Completed July 29th
Time allocated: 3 hours
Time took: 3 hours

Task: Don't Allow Duplicate Exercise/Goals in A Day
Completed July 30th
Time allocated: 2 hours
Time took: 2 hours

Task: Add Exercise onto Timeline Activity
Completed July 29th
Time allocated: 4 hours
Time took: 4 hours

Task: Create App Color Theme
Completed July 29th
Time allocated: 1 hour
Time took: 1 hour
Notes: Created a theme in styles.xml for use in all of our activities with properly set colors.

Task: Fix UI for Goals
Completed July 30th
Time allocated: 2 hours
Time took: 2 hours

Notes: made changes to Emmanuel’s Goals UI with the updated theming.

Task: Fix UI for Add Food/Meals
Completed July 30th
Time allocated: 2 hours
Time took: 1 hour
Notes: changed the add food buttons to floating action buttons to be more consistent with the theme of the app.

Amin's Log

Make all data structures private
Pair programmed with Mark Shinnie (Navigator)
Completed July 21st
Time allocated: 1 day
Time took: 5 hours

Populate database with food
Got debugging help from Mark and Arvind
Completed July 28
Time allocated: 2 hours
Time took: 3 hours

<u>Arvind's Log</u>

Task : Design Database
Pair programmed with Mark
Time Allocated: 2 days
Dates worked on: 18 - 21, 23
Total time took: 8 days
Completed on July 26th

Notes: designed the database with Mark. First couple days were spent trying to get HSQLDB to work and going through the initial database and schema design. After finalizing the design, we wrote the SQL script and ran into various issues with actually getting the database to work. After a couple days of debugging, Mark solved the issue with the help of professor braico.

Task: Hotfix DB
Pair programmed with Mark
Time allocated: 4 hours
Time Took: 6 hours
Completed on July 26th

Notes: the code needed to be adapted with the new database and we needed to update the access methods. The log in, add food and sign up features stopped working and I spent time debugging it with mark.

Task: Search Bar Query Food DB
Time allocated: 10 hours
Time took: 4 hours
Worked on July 28th

Notes: also created a pop up for adding food in the UI in this time as well.

Task: Access Class tests
Time allocated: 6 hours
Time took: 4 hours
Worked on July 29th

Task: Populate the database
Worked with Amin and Mark
Time allocated: 3 hours
Time took: 2 hours
Worked on July 29th

Notes: spent time debugging Amin's script with mark.

Task: Create a way to transition between days
Time allocated: 6 hours
Time took: 2 hours
Worked on July 30th

Notes: created a calendar view to switch between which days we need to track

Task: Cleaned up the code
Time took: 1 hours
Worked on July 30th

Notes: general code cleaning; filled in empty comments, refactored with android studio etc. Created a helper class for changing activity to reduce code.

Mark's Log
Task: Clean up calculator code/Turn Meal into a composite
Task Started July 15
PR Open: July 15
PR Merged: July 18
Time taken: 6 hours
Time allocated: 7 hours (5hr Meal into composite) (2hr clean calculator code)
Assigned to: Mark (driver) and Amin (navigator)
Notes: Previously there were a lot of code smells in the calculator class. We had to check if an edible was a food or a meal and handle each one accordingly. We fixed this by adding a get macro/micro getter in the abstract goal class, so we can get the grams polymorphically, eliminating the need to cast. Furthermore, we used to return the Enum int map of micros or macros to which the client could do whatever they wanted. Now this map is private with public getters and setters that only allow you to set the int value to valid amounts. For example, you can set the amount of iron to 0 but you can't remove the iron int pair from the map anymore. Furthermore, we made Meal iterable so we can loop over it. Since meal is a composite object, we don't iterate recursively over it but only one layer deep.

Task: Design HSQL Database
Task Started July 15
July 15: 2 hours designing relational schema
July 16: 5 hours Write initial script/attempting to get the db. to work on app pair (Mark/Arvind)
July 17: 13 hours write sql queries for db. Refine schema
July 18: 4 hours Further refine sql queries
July 20: 2 hours validating queries
July 21: 1 hour Get Sql connected on device
July 23: 3 hours get a minimal working query in data access object to work
July 24: 9 hours integrate queries written previously into the java sql interface and hsql
July 25: 1 hour clean up code and add comments
July 26 1 hour integrate with turning days into having meals instead of lists of edibles
PR Open: July 24
PR Merged: July 26
Time taken: 41 hours
Time allocated: 10 hours
Assigned to: Mark and Arvind
Notes:: This task was grossly underestimated. The problem is there are so many things that come up that you weren't expecting at first. Mapping the objects to a relational schema took a lot of time, research, and experimenting to get right. This was further complicated by the fact that meals are difficult to map into a relational schema since it is a composite object. This makes it difficult to store a multi nested meal in the sql database. Also getting the hsql db. to work took a lot of time and debugging. Eventually with the help of the professor we were finally able to get it up and running. Crafting all the sql queries and getting them all to work was painstaking as it required getting the sql right, conforming to the java sql interface, and finally troubleshooting any errors that came up whether it be by a syntax error or a simple spelling mistake. I will now talk about the way we designed the db. and why we did it that way
DB Schema
For enums we choose to store these values as integers in the db. as it is easy to convert to and

from an enum in an integer. For the goals table we choose to put all classes of goals into one table and simply tell store which class they were as a string in the db. This is useful as it cuts down the number of tables and we treat all the goals as the same when storing them, so it made sense.
Accounts has been merged with user Info into its own table as user info is essentially just a container. Since each account has a list of days the days table is a weak entity of accounts since a day is always associated with an account. Likewise, the exercise and goal tables are weak entities of a specific day in an account. This lets us do things like keep track of the goals of that day so if the goals change, we have a record of what they used to be. There is also a table that keeps track of the edibles in each day which is also a weak entity of days.
The edible class was split up into 2 tables (not including the table that tracks the edibles in a day). There is an Edible, and an Edible int pair table. We combined the edible and food class into just the edible table. The meals are contained in the edible table and the edible_int_pair tables. This helps to limit the number of tables by keeping combining edibles and food into one. In the edible table we keep track if the edible is a meal in a Boolean value. The edibles in the meal are contained in the edible int pair table. This table stores what is the parent meal, what edible is contained, and how many of this edible is contained. This allows us to easily get any edible by querying the edible table, and if it is a meal, we could easily query the edible int pair table to find the edibles in the meal. To store all the labels, we have a label table. We have a separate table for storing which edibles have which labels, since it is a many to many relationships. Since we have both meals and food in edibles, we only need one table for storing which edibles have which labels.

Task: Adding existing food should not add duplicates
Task Started July 24
PR Open: July 24
PR Merged: July 26
Time taken: 3 hours
Time allocated: 2 hours
Assigned to: Mark
Notes: The way we solved this is instead of having days contain a list of edibles it now contains a meal object. This solves multiple issues. One is, since Meal, holds a list of edibles it can act as a list. Furthermore since meal already encapsulates its data and only allows you to access its edibles in certain ways this lets day just return the meal and allow the client to interact with the meal as they wish. It also lets us keep track of how many of a certain edible is in each meal time during a day (breakfast, lunch, dinner, snacks) as Meal already has this functionality.

Task: Implement Data Access Interface
Task Started N/A (See Emanuel's Dev log)
PR Open: July 26
PR Merged: July 28
Time taken: Mark - 4 hours
Emanuel - See his dev log
Time allocated: 4 hours
Assigned to: Emmanuel
Notes: Additional testing revealed further bugs in sql database that needed to be addressed. I

also did some pair program navigating while he wrote the integration tests for the data access interface.