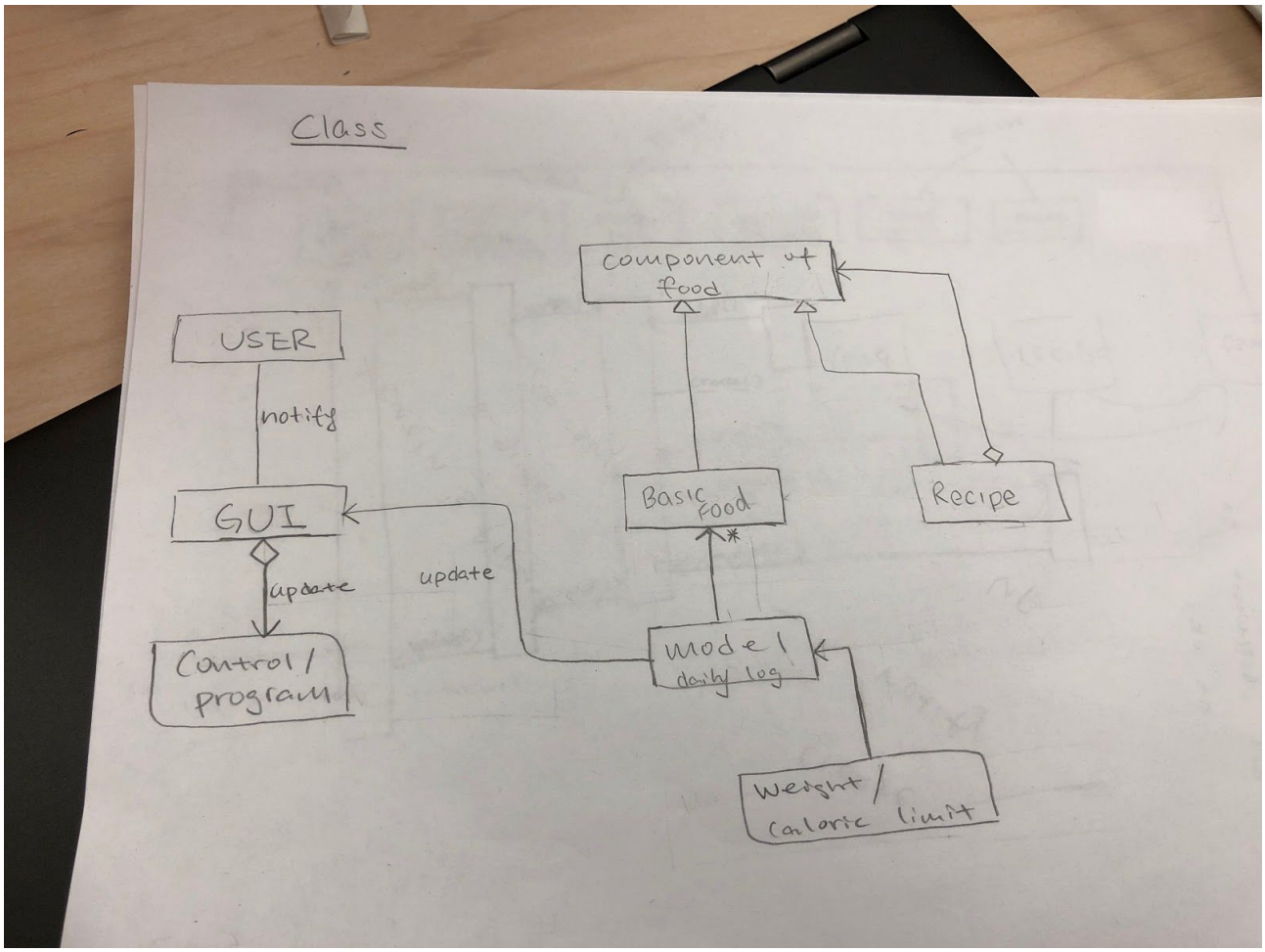
Design Sketch

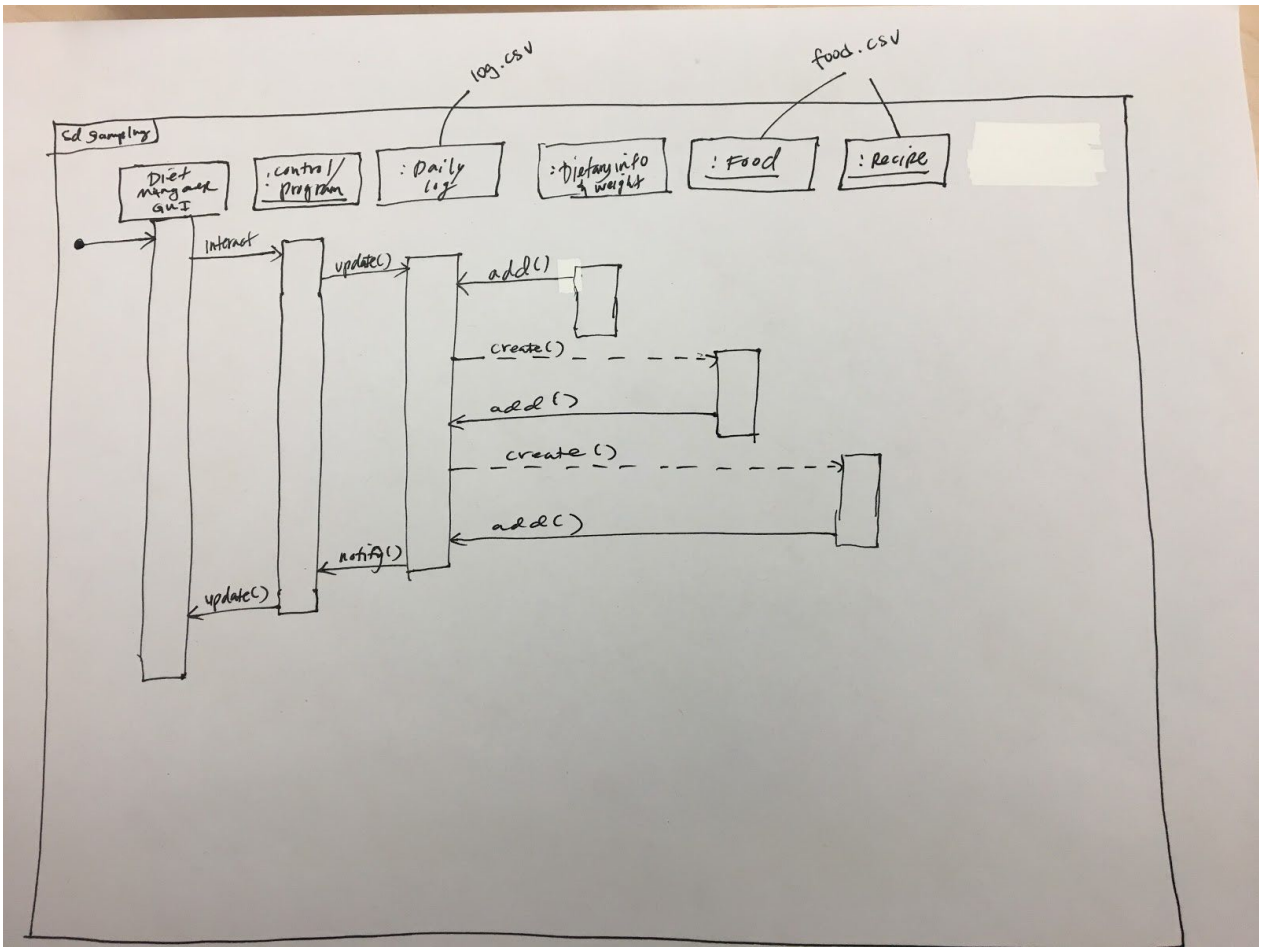
Date: 10/18/2018

Team F: Daniel Cox, Chad Cummings, Tenzin Dhondup, Zhimin Lin, Chandler Michel

Diet Manager Class Diagram



Diet Manager Sequence Diagram



Recipe

1. The Attributes are Name, Other Foods (Basic Food, SubRecipe), Other Foods' Number of Servings.
2. Responsible for entering values into the CSV file (food.csv) as well as the (dailylog.csv)
3. Indicator by ("r").

Food

The Attributes are Name, Calories, Fat, Carb, Protein. In which the calories, fat, carb, protein are in grams.

Responsible for entering values into the CSV file (food.csv) as well as the (dailylog.csv)
Also responsible for passing values such as calories, fat, carb, and protein for the other classes to access.

Daily Log
Takes in the values enter in on the food.csv and also enter the values in the log.csv.
Values of Dietary Info such as weight, calorie limit are also entered in the log.csv.

DietaryInfo
User input besides food/recipe related, such as weight, calorie limit and passes those values in Daily Log.

Program
Takes in all the user inputs and stores values in designated CSV files. Also can access the data through the program and display for a user. Connects Recipe, food, Daily Log, Dietary Info, and the CSV files together.

We are going forward with this design using the composite pattern into consideration which aligns with the requirements and deliverables of this project. In order to provide the best experience and design for our users, we will also implement a GUI and is included in our design plan. In order for the client to use the program, an aggregation of control options is needed such as button inputs. As the user interacts with the data inputs, the program will send the data input into the daily log (such as foods and recipes).

We decided to pick the recipe class as the composite class because it stores the grouping of foods and ingredients, and it will be able to make sub-groups of foods within the recipe composite class. This makes it possible to set up the leaf classes which the individual pieces of food/ingredients will go into.

By putting emphasis on the composite pattern, we can achieve ease of class implementation and potential scalability based on the sequence and workflow required to process this program. By associating tasks into specific groups of classes, we can reduce the possibility of technical debt caused by weak design diagrams and not covering all the basic requirements. We should also note that this design, while not preferred, is subject to minor changes as we learn more about the project via the client or during the development process.