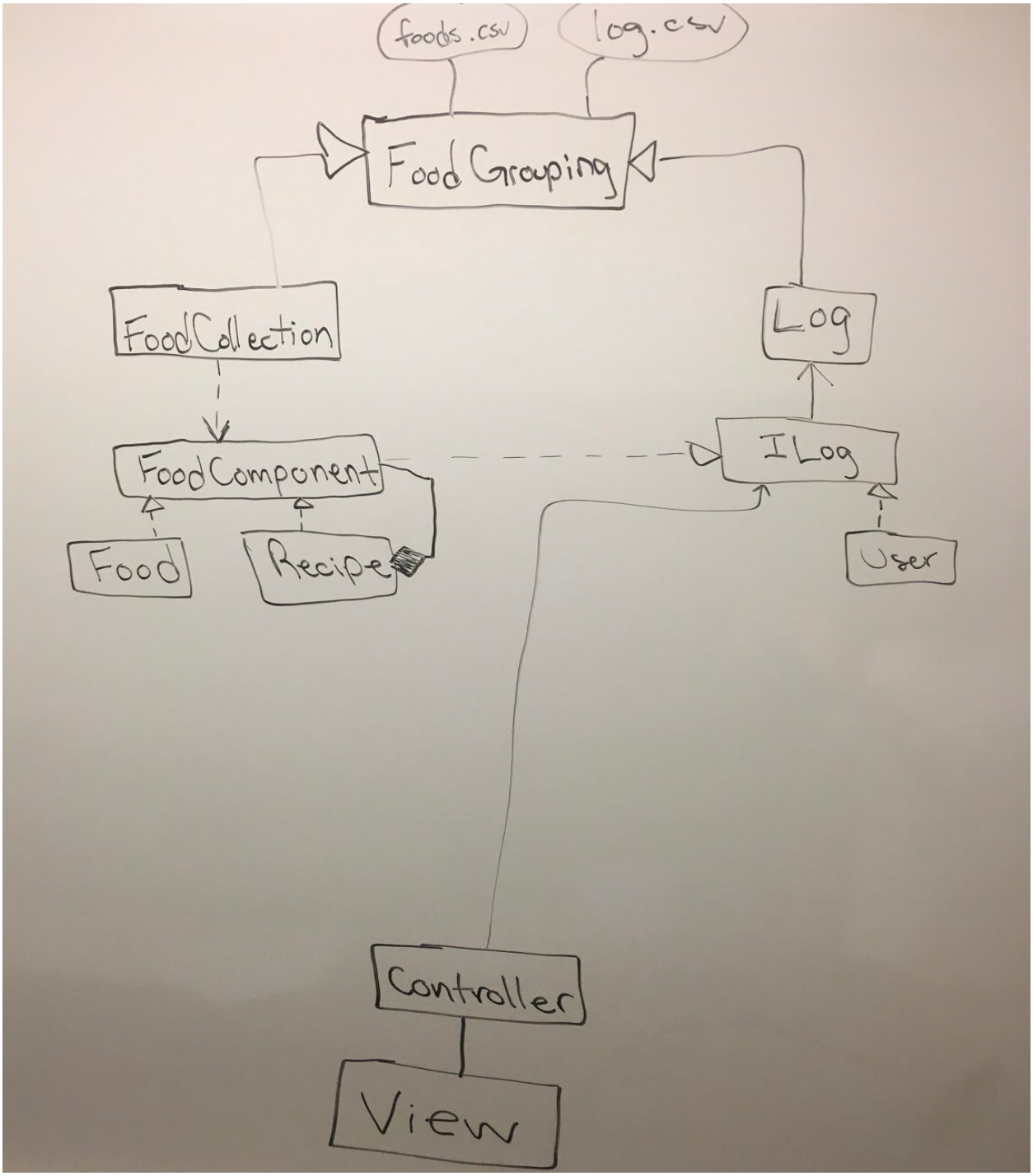
DESIGN SKETCH

10/15/2019

Team B

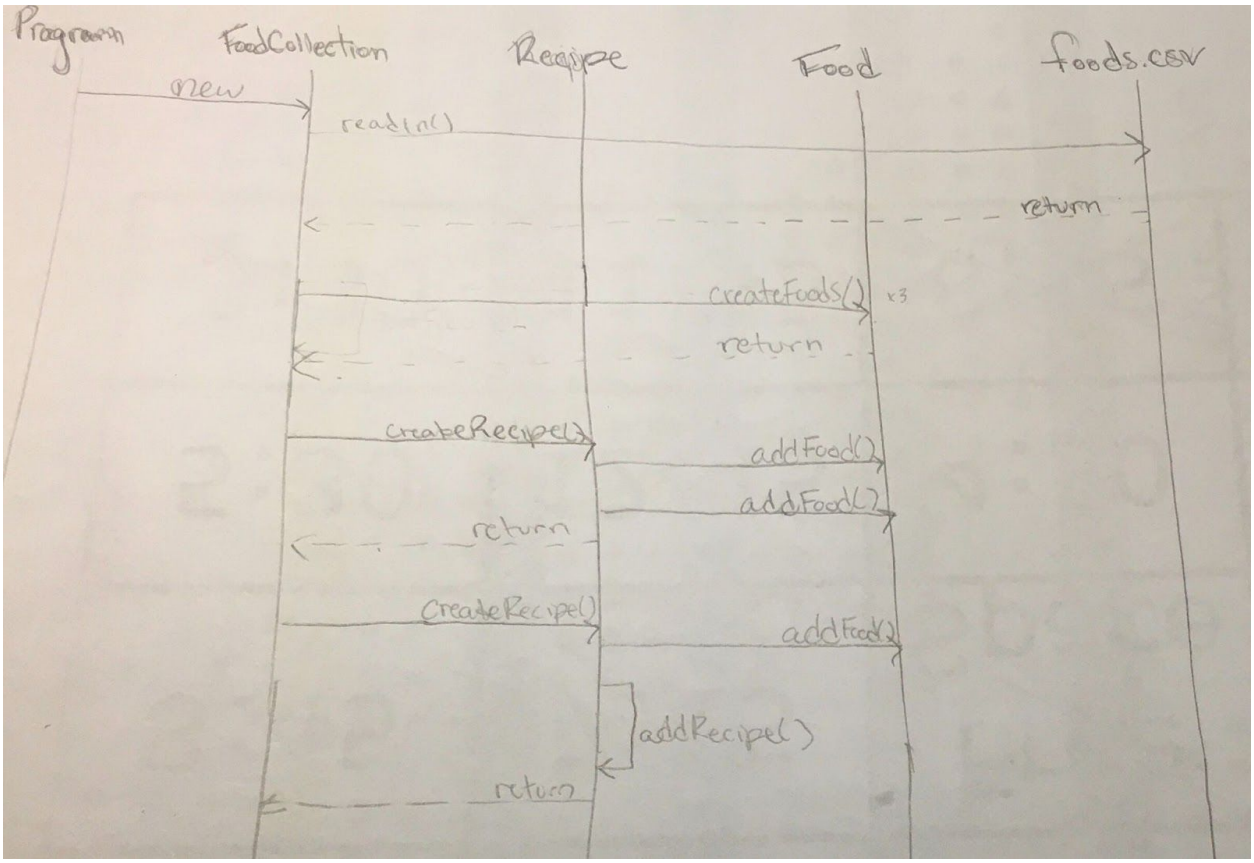
Jack Old
Fengyi Chen
Fred Amartey
Lowell Pence

Rough Class Diagram

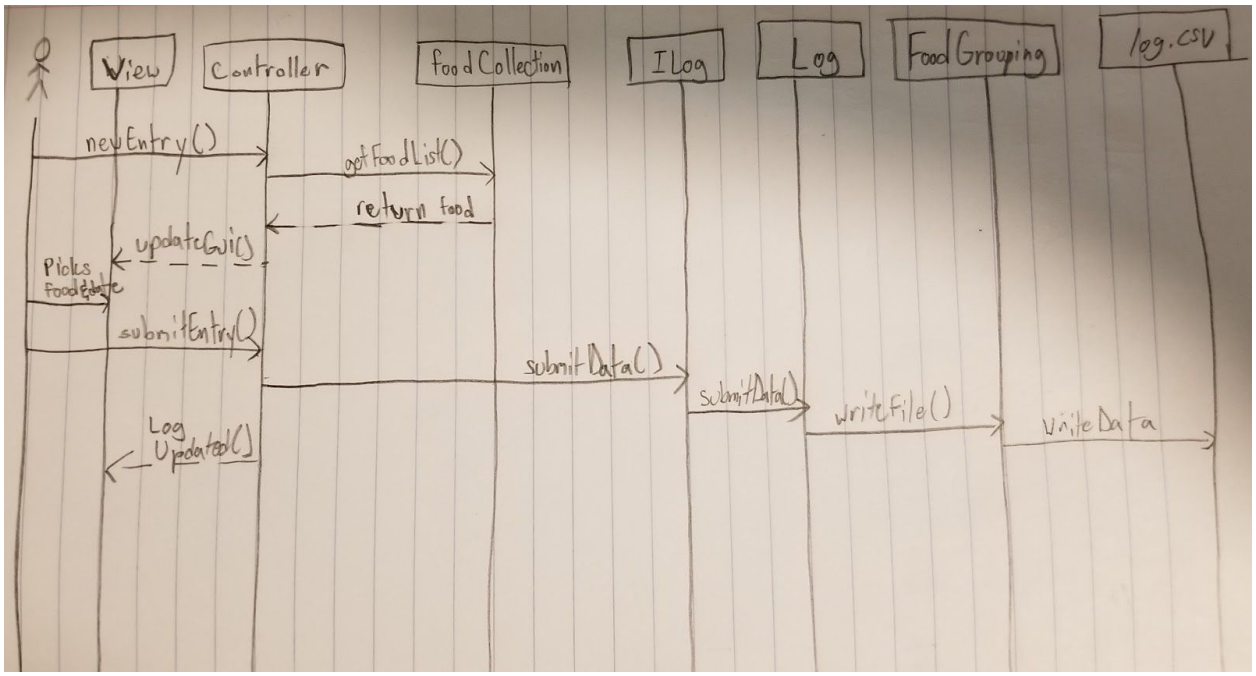


Sequence Diagrams

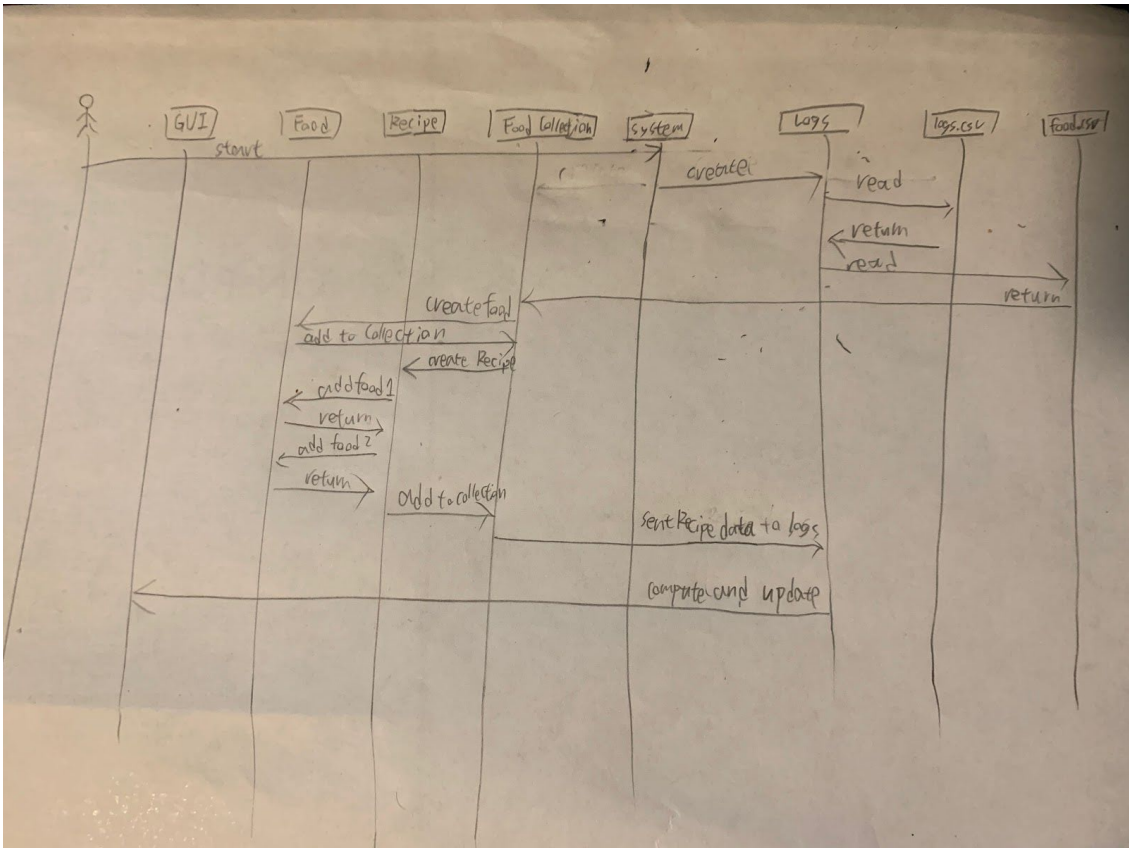
1. Read in a food database consisting of three basic foods, a recipe that contains two of the basic foods, and a recipe that contains the first recipe and the remaining food.



2. Add two servings of a food to the log entry for the current date.



3. Compute the total number of calories for the current date, assuming the log consists of a basic food and a recipe consisting of two basic foods.



4. A narrative in which the team provides:
   A brief description of each of the application classes other than those in the UI. The description must include each class's responsibilities (what the objects in the class know and what they can do).

   FoodGrouping: An abstract class that provides functionality for opening and reading a CSV file. Only knows about itself and how to open/ read files.

   FoodCollection: Stores all the foods and recipes from the foods.csv file so that they can be easily accessed when the program is running. Knows about FoodComponents, the FoodGrouping parent class and the given CSV file.

   FoodComponent: An interface (the component) for providing common functionality for both the food objects and recipe objects. Knows about the log.

   Recipe: A concrete (compound) class for recipes. A recipe object has a list of associate food objects, a name, etc. Knows about FoodComponents.

   Food: A concrete (leaf) class for a simple food. Holds the information that makes up a food. Knows about FoodComponents.

   Log: Loads the log data from the log.csv file and takes in new log changes while the program is running. Saves the changes to the file when the program exits. Knows about FoodGrouping parent class and the given CSV file.

   ILog: An interface that takes in 'getData()' requests to the Log class from external classes like from foodComponent, User and the controller. This done to practice the P2I design pattern, and to reduce coupling directly to the Log class.

   User: Holds relevant user information like weight. Will provide "future functionality" for if the client wants multiple users or more unique user information such as favorite recipes/ foods. Knows about the log.

Structuring the food grouping into 'Food Collection' and 'Log' allows for easy logging and relations between the groups. It also provides one place for the file reading functionality to be specified. Logging each food item or recipe's data (calories, fat, protein, etc) is more accessible and the implementation of the ILog interface allows for adapting any food components with the appropriate logs and data attributes. Perhaps the coupling is still too high and could be improved upon.