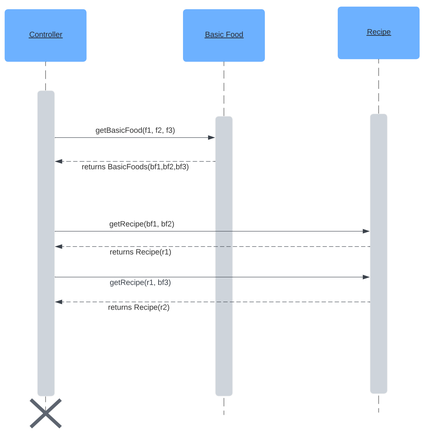
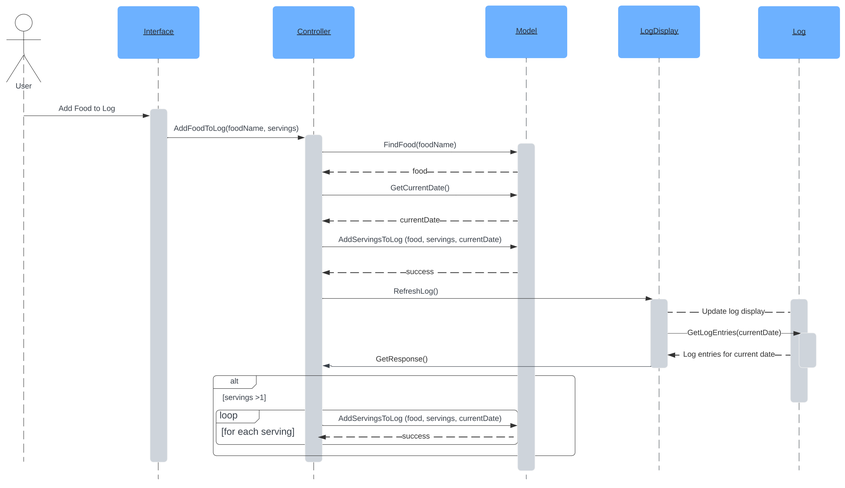
Sequence Diagram Scenario #1



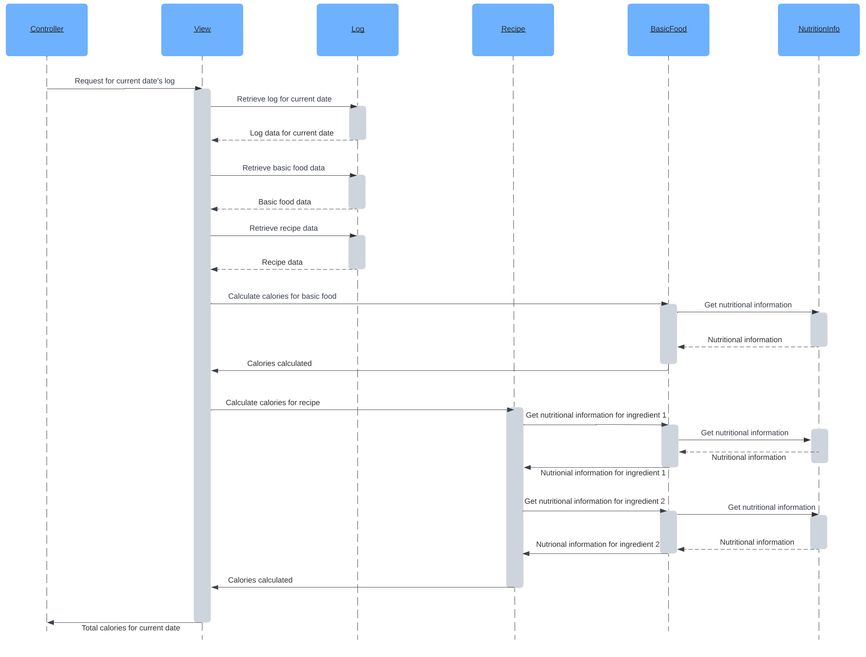
Read in a food database consisting of three basic foods, a recipe that contains two of the basic foods, and a recipe that contains the first recipe and the remaining food.

Sequence Diagram Scenario #2



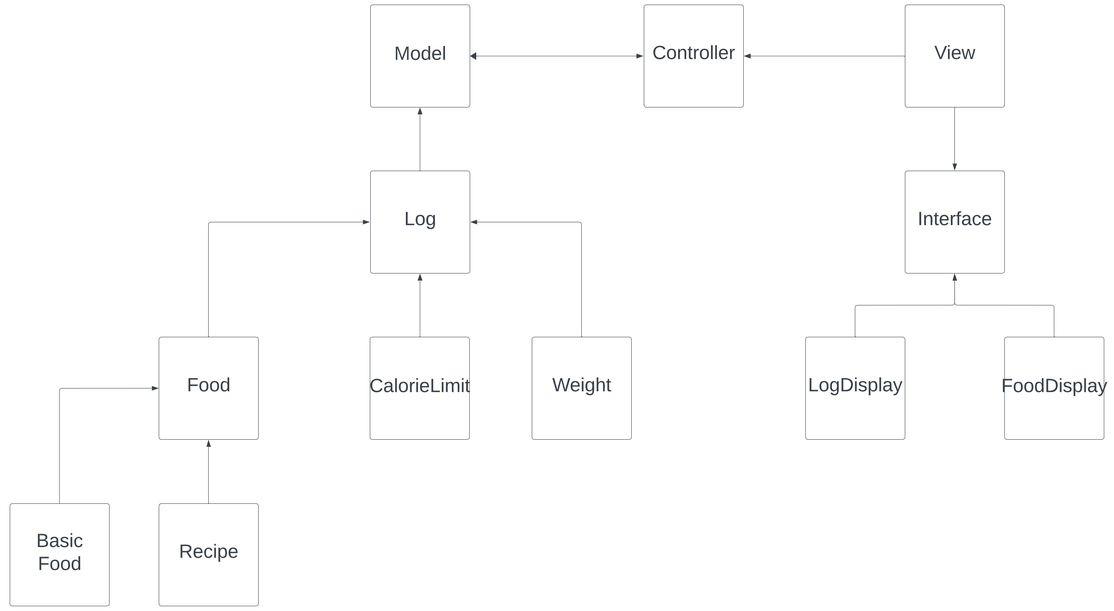
Add two servings of a food to the log entry for the current date.

Sequence Diagram Scenario #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.

Class Diagram



Controller: Acts as the intermediary between the View (UI) and the Model (data). It processes user inputs from the View and translates them into actions to be performed by the Model. Responsibilities include fetching basic food items, generating recipes, and adding food to a log based on user requests.
Model: Represents the system's data structure and logic for handling food items, recipes, and nutritional information. It is responsible for storing and manipulating the data, such as adding servings to a log, calculating calories, and retrieving nutritional information for both basic foods and recipes.
Log: Manages entries related to food consumption. It can add servings to a log, refresh the log display, and retrieve log entries for a specific date. Its primary responsibility is to track the user's food intake over time.
Basic Food and Recipe: These classes are part of the food database. Basic Food represents individual food items with their nutritional data, while Recipe represents a combination of these items into a meal, including the calculation of total nutritional values from its ingredients.

The organization of the system into separate classes for handling the UI, data processing, and data storage (Model-View-Controller or MVC architecture) allows for a clear separation of concerns, making the initial implementation more straightforward and future maintenance or upgrades easier to manage. This separation enhances the system's scalability and adaptability, as changes to one component (e.g., the user interface design) can be made independently of the others (e.g., the data model). However, this approach may introduce complexity in managing the interactions between components, especially for large-scale applications, and could require more effort in the initial design phase to ensure that all components communicate effectively