PROJECT / RELEASE

Project Design Document

TWNN

Lowell Pence <lxp3901@rit.edu>

Jack Old <jxo4940@rit.edu>

Fred Amartey <fda9891@rit.edu>

Fengyi Chen <fxc1494@rit.edu>

# 

# Project Summary

This project aims to make the process of logging food consumption and weight changes streamlined. Diet Manager is a software designed to help users meet their diet goals by keeping them up-to-date on exactly they are consuming every day. Users will have features like weight and calorie consumption tracking, adding foods and recipes, and displaying a summary of the nutrition data logged with graphs. This will make diet managing and progression tracking simple for the user.

A food library will allow the user to save commonly consumed foods or complex recipes so that they can easily log them in the future. All user entered data and progress tracking is automatically saved to the user’s computer when the program closes, and can be manually saved at any time as well.

# Design Overview

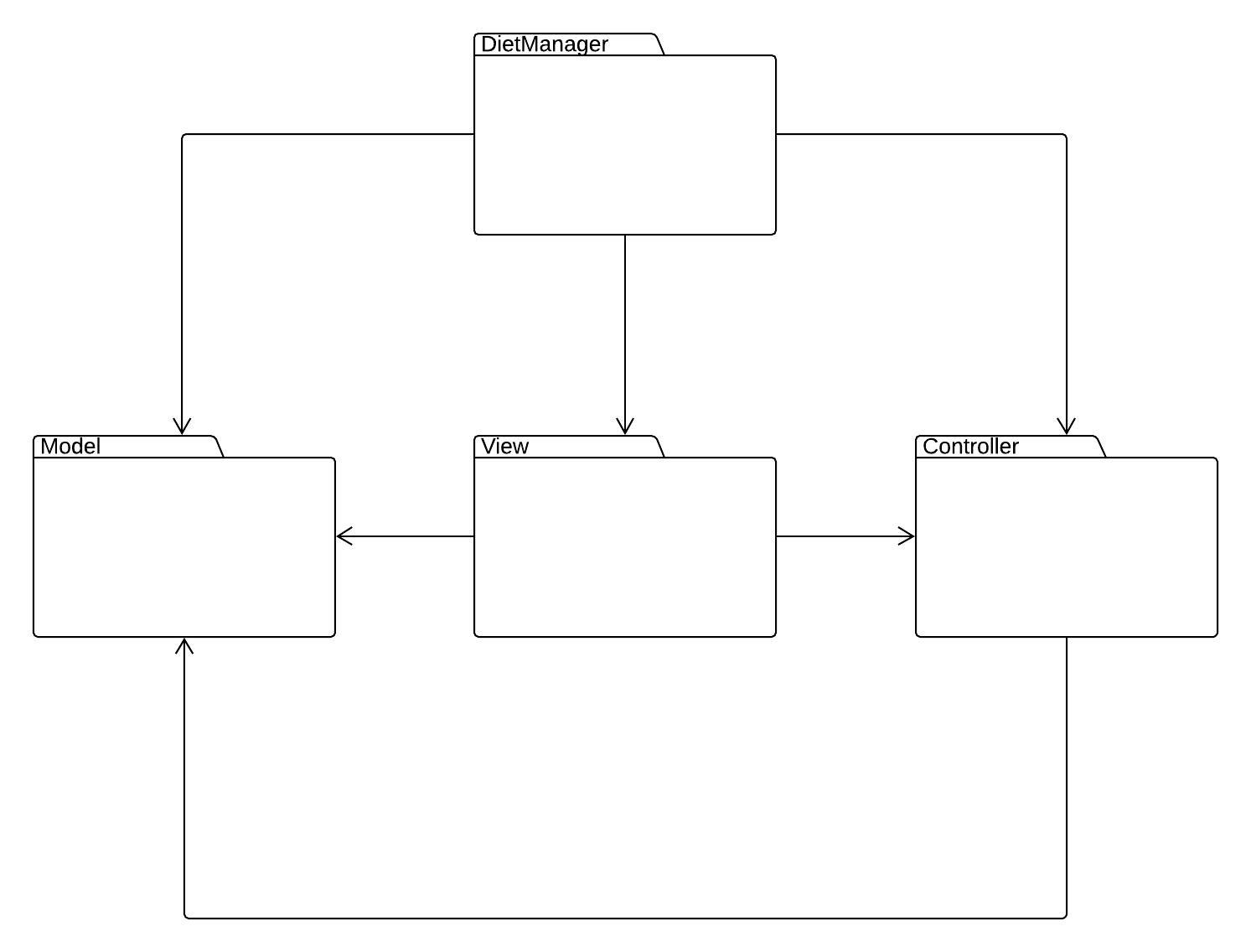
To make this streamlined as possible, we decided to implement the model-view-controller design pattern to achieve separation of concerns within our application. This also aims to lower coupling and increase cohesion as well.

The model will store the system state when the program is running. This means the data from the user will be held and manipulated in the model.

The view is going to be responsible for displaying logged data, their respective graphs, saved foods/ recipes, and other data relevant to the user on a given date. It will have a well designed UI that allows a user to enter new foods, recipes and user information such as weight and caloric goals that will ultimately be handled by the model.

The controller is going to serialize the user input and allow us to manipulate the data into our a format in which the model can understand. The controller will update the model when the view changes, and also update the view is certain instances. The view will be updated by the model as well. With each component orchestrating one functionality, it will foster high-cohesion and lead to classes only interacting with entities that are needed to function; which promotes low-coupling.

The current design is assuming that the program runs by somehow starting each piece of the MVC, and that the user is interacting with a command line interface.Subsystem Structure



The Model updates the view with relevant data as necessary and stores data

The View is constantly being updated by the Model as data changes

The Controller is constantly updating the model and sometimes updates the View when necessary

# Subsystems

## **Subsystem - Model**

|  |  |
| --- | --- |
| **Class** EntryCollection | |
| **Responsibilities** | Support access to the media in the library collection.  Add, find, delete an existing media in the collection.  Provide virtual collection for the entire library consortium collections. |
| **Collaborators**  **(uses)** | Media - the basic type for all different media in the collection.  network.Consortium - consortium communication (network subsystem). |

|  |  |
| --- | --- |
| **Class** LoggableItem (interface) | |
| **Responsibilities** | Provide functionality for a food or recipe to add itself to the log. Anything meant to be logged must conform to this interface. |

|  |  |
| --- | --- |
| **Class** Log | |
| **Responsibilities** | Take in data and store it within its internal data collection.  Write the data to external media(csv, JSON, etc.).  Read data from external media(csv, JSON, etc.).  Return data from within its internal data collection. |
| **Collaborators (inheritance)** | LogEntry - uses functionality for writing and reading from external media.  LoggableItem - Log is dependent on interface LogabbleItem to receive requests to receive data to be added to the internal data collection. |

|  |  |
| --- | --- |
| **Class** FoodCollection | |
| **Responsibilities** | Take in FoodComponent objects and store them within its internal data collection.  Write the data to external media(csv, JSON, etc.).  Read data from external media(csv, JSON, etc.).  Return list of foods from FoodCollection’s internal data collection |
| **Collaborators (inheritance)** | FoodFactory - uses functionality for writing and reading from external media.  FoodInterface - FoodCollection is dependent on the FoodInterface interface to interact with the data structure of stored foods and recipes. |

|  |  |
| --- | --- |
| **Class** FoodInterface (interface) | |
| **Responsibilities** | Provide functionality to interacting with a recipe or food object. Anything object added to the data structure must conform to the constraints of the interface. |

|  |  |
| --- | --- |
| **Class** BasicFood | |
| **Responsibilities** | Contain a food object that contains data about the food, including the name and nutritional values. |
| **Collaborators (inheritance)** | FoodInterface- SimpleFood implements the interface FoodInterface for operations and attributes regarding its name and nutritional data that is assigned to that object.  LoggableItem - Implements interface LoggableItem to be able to submit |

|  |  |
| --- | --- |
| **Class** Recipe | |
| **Responsibilities** | Contain food objects and other recipe objects that contains data about the food or recipe, including the name and nutritional values.  Add or remove components associated with that Recipe object. |
| **Collaborators (inheritance)** | FoodInterface - Recipe implements the interface FoodInterface for operations and attributes regarding its name and nutritional data that is assigned to that object. |

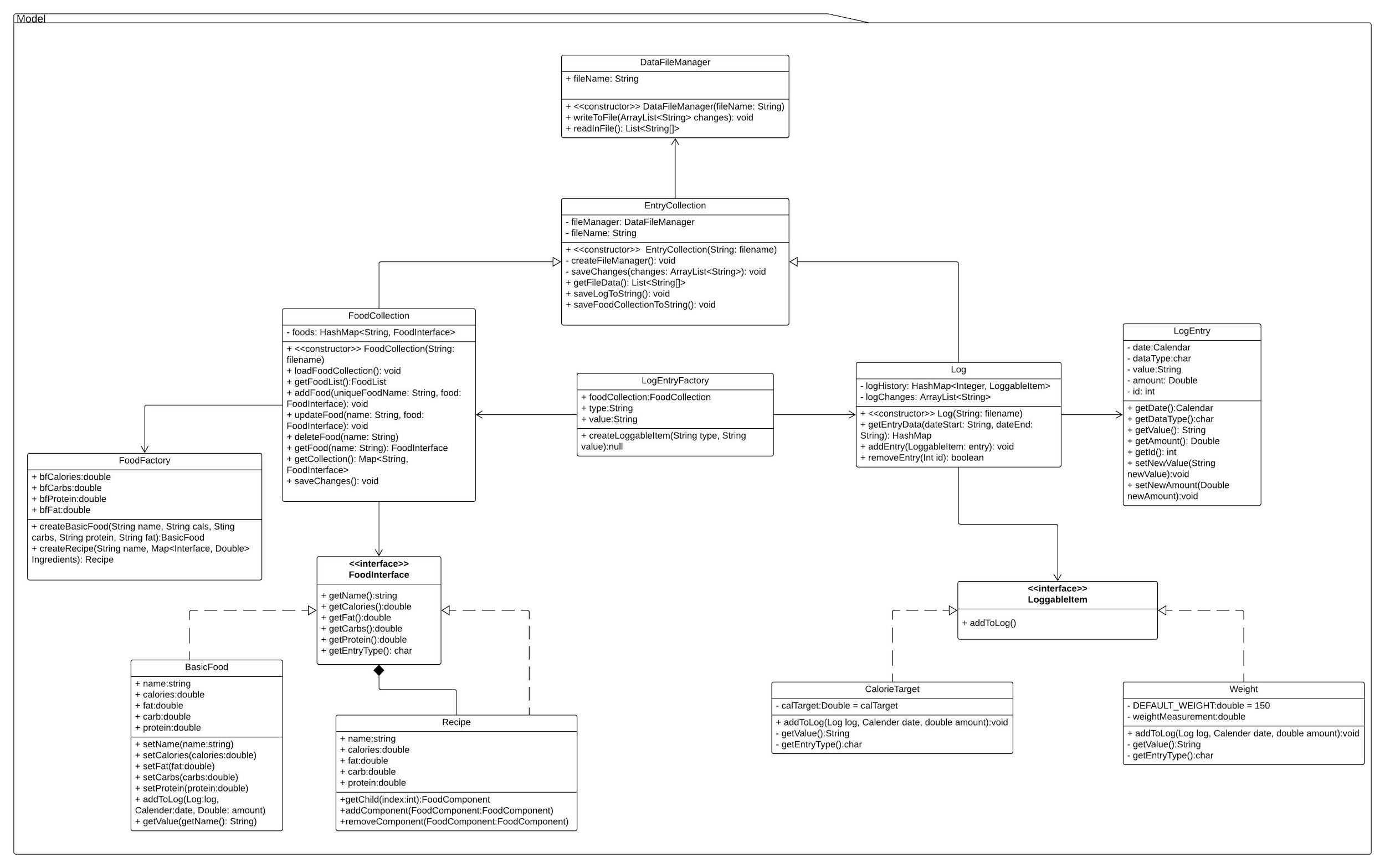
|  |  |
| --- | --- |
| **Class** FoodFactory | |
| **Responsibilities** | Takes the responsibility of creating the concrete food and recipe objects when needed by FoodCollection. |
| **Collaborators (inheritance)** | Is used by FoodCollection |

|  |  |
| --- | --- |
| **Class** LogEntryFactory | |
| **Responsibilities** | Generates a loggableItem object for the use by FoodCollection and Log classes. |
| **Collaborators (inheritance)** | Uses FoodCollection and Log |

|  |  |
| --- | --- |
| **Class** LogEntry | |
| **Responsibilities** | Encapsulates the formatted object that will be sent in from log to entry collection. |
| **Collaborators (inheritance)** | Is used by Log |

|  |  |
| --- | --- |
| **Class** CalorieTarget | |
| **Responsibilities** | Generates the CalorieTarget object when the user wants to create a new calorie target. |
| **Collaborators (inheritance)** | Implements LoggableItem interface |

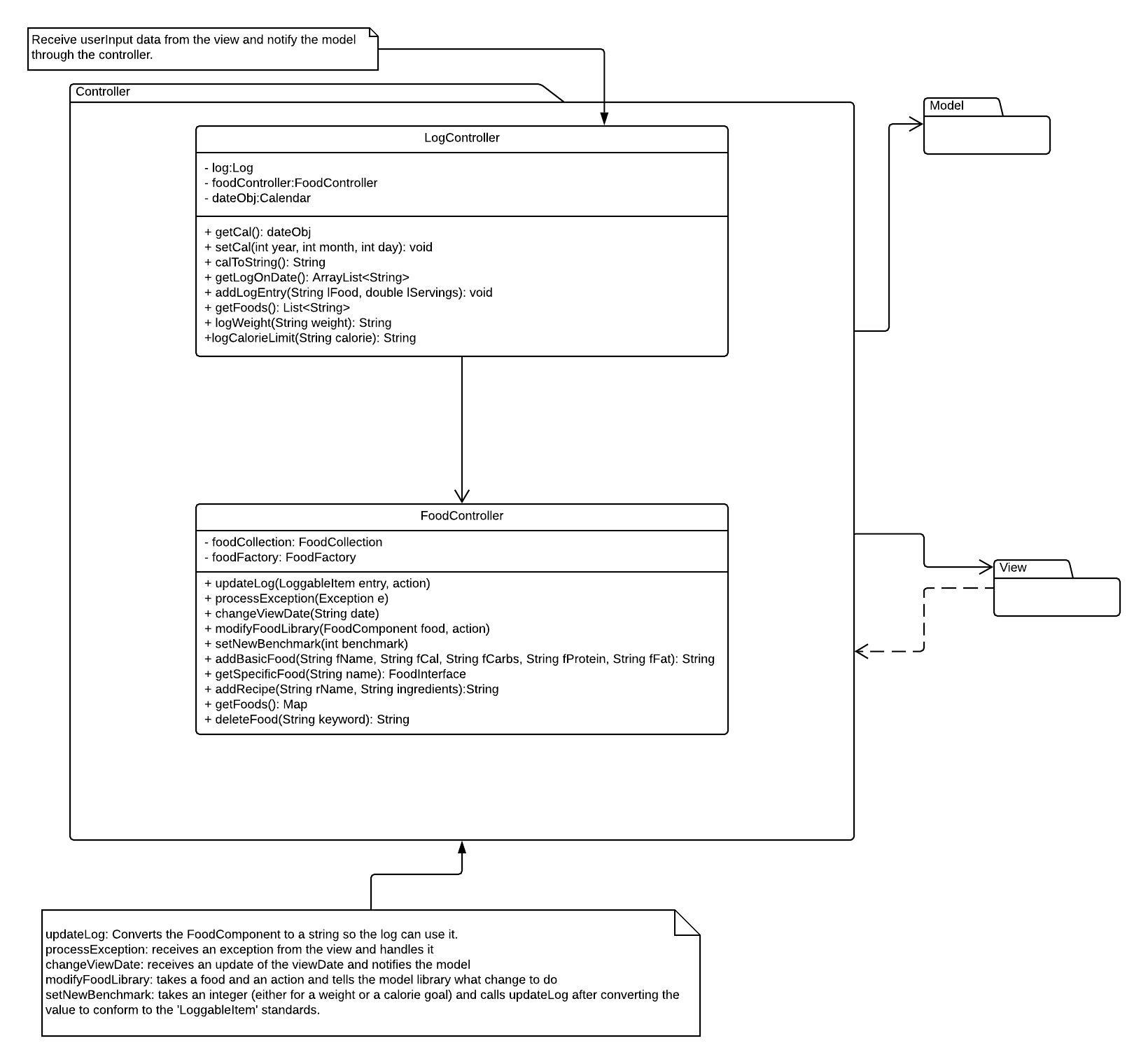
|  |  |
| --- | --- |
| **Class** Weight | |
| **Responsibilities** | Generates the Weight object when the user wants to add a current weight. |
| **Collaborators (inheritance)** | Implements LoggableItem interface |



## **Subsystem - Controller**

|  |  |
| --- | --- |
| **Class** FoodController | |
| **Responsibilities** | Receive state change information from the view subsystem and update the appropriate model objects.  Format user entered data so that model objects can use it. (ex. a user inputs a food name and the food library will know how to interpret that)  Handle exceptions/errors that the user may run into. |
| **Collaborators**  **(uses)** | The view subsystem will use the controller to update the model with changes.  The controller will collaborate with the model when it needs to update it with changes from the view. The controller will update the view under certain circumstances. |

|  |  |
| --- | --- |
| **Class** LogController | |
| **Responsibilities** | Receive state change information from the view subsystem and update the appropriate model objects.  Format user entered data so that model objects can use it. (ex. a user inputs a food name and the food library will know how to interpret that)  Handle exceptions/errors that the user may run into. |
| **Collaborators**  **(uses)** | FoodController - LogController allows for logging of foods from the foodController |



## 

## **Subsystem - View**

|  |  |
| --- | --- |
| **Class** LogPanel | |
| **Responsibilities** | Receives user input of weight, calorie goal, or a food to add or delete, to their log. |
| **Collaborators**  **(uses)** | EventHandler - Implements EventHandler interface to check for user input when user is interacting with panel. |

|  |  |
| --- | --- |
| **Class** CLI | |
| **Responsibilities** | Receives user input of Adding a new food to food library, removing a food from food library |
| **Collaborators**  **(uses)** | EventHandler - Implements EventHandler interface to check for user input when user is interacting with panel. |

|  |  |
| --- | --- |
| **Class** DailySummary | |
| **Responsibilities** | Displays daily summary of data to user including:  Calorie intake goal, current weight, and current amount of daily calories consumed |
| **Collaborators**  **(uses)** |  |

|  |  |
| --- | --- |
| **Class** ProgressTracker | |
| **Responsibilities** | A dynamic view of the changing user data such as weight and calories consumed/ goal. |
| **Collaborators**  **(uses)** | dailySummary, EventHandler - implements EventHandler interface to check for user input when user is interacting with panel. |

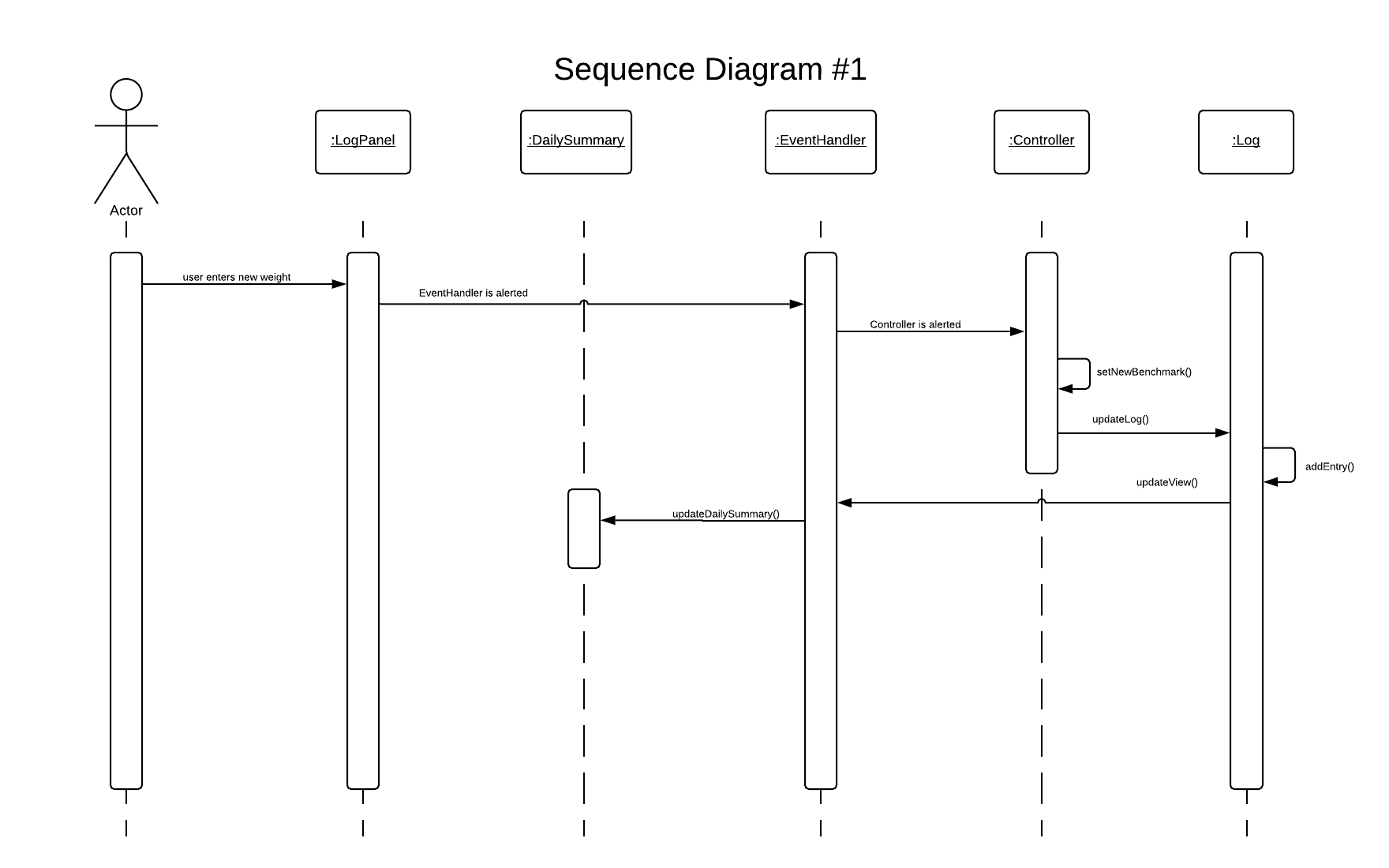
|  |  |
| --- | --- |
| **Class** EventHandler (Interface) | |
| **Responsibilities** | Takes in a listener for user input on view panels. Event listener must conform to the EventHandler interface. |

# 

# 

# Sequence Diagrams

## **Sequence Diagram #1 Description**

The user is adding a new weight measurement to their DietManager log. The user will be able to see their inputed value once it has been added to the log.

## 

## 

## 

## 

## 

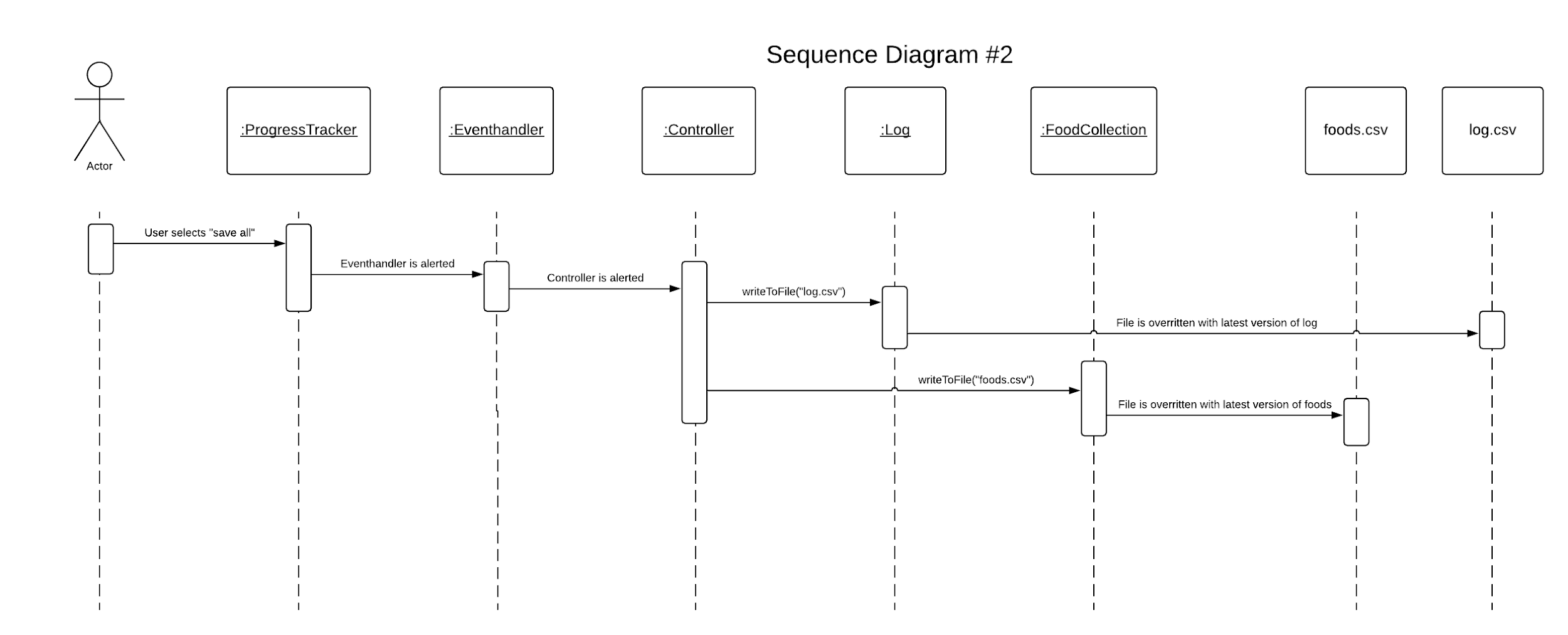
## 

## 

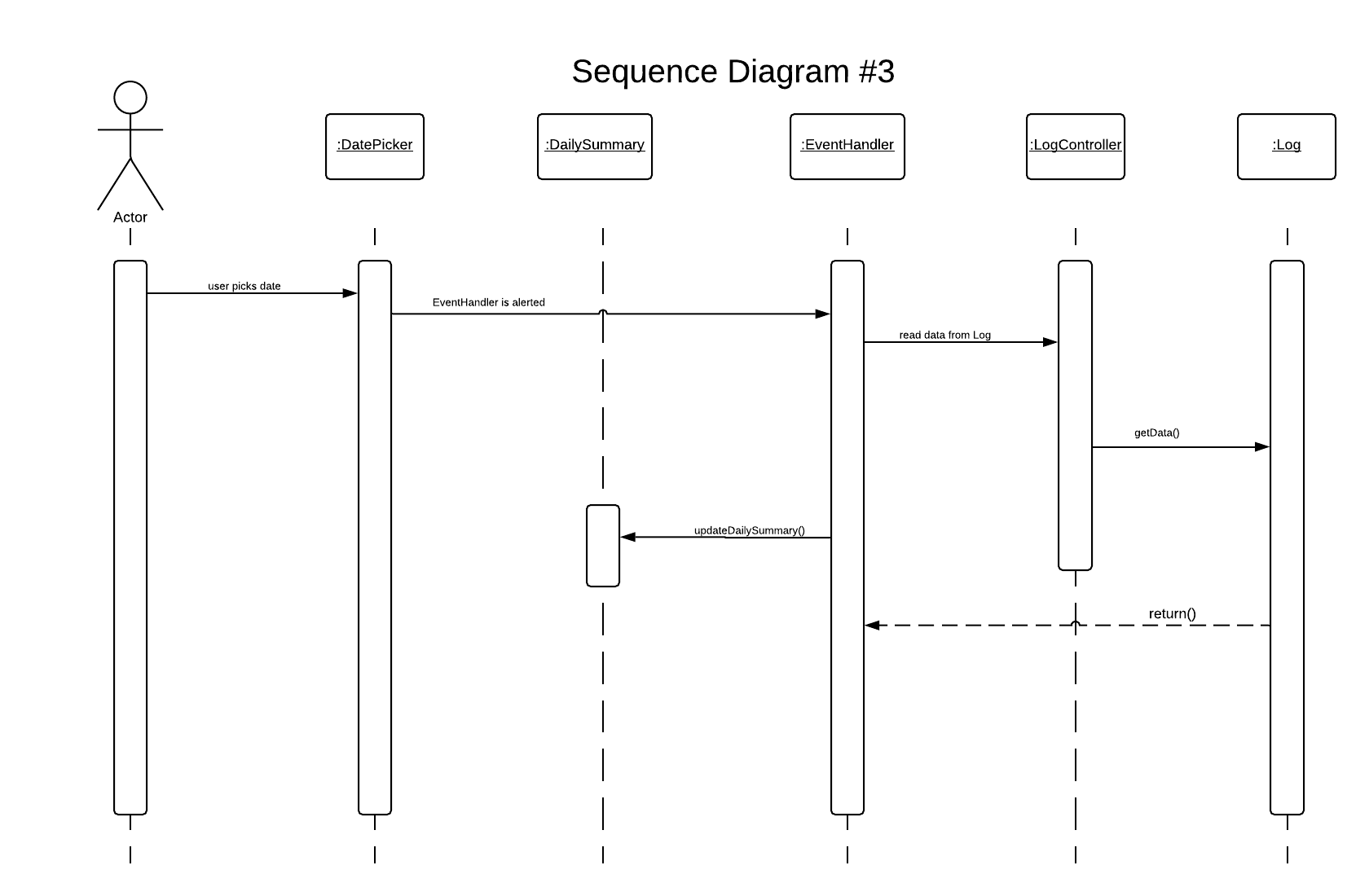
## 

## **Sequence Diagram #2 Description**

The user is saving all changes made since the last save to the appropriate data files.



## **Sequence Diagram #3 Description**

The user has initiated a date change to pull up log data from a previous date. 

# 

# Pattern Usage

**Pattern #1 MVC**

|  |  |
| --- | --- |
| **MVC Pattern** | |
| **Model** | FoodCollection  Log |
| **View** | CLI  GUI |
| **Controller** | FoodController LogController |

**Pattern #2 Composite**

|  |  |
| --- | --- |
| **Composite Pattern** | |
| **Component** | FoodInterface |
| **Composite** | Recipe |
| **Leaf** | Food |