Phase 2 - Nancy Group A

Not Always Neatly Coding… Yet.

Introductions

Jordan Rabideau - Team Coordinator

Kelly Appleton - Requirements Coordinator

Tessa Zakroczemski - Testing/QC Coordinator

Yun Yiu Cheng - Configuration Coordinator

Although these were our assigned roles, we tended to equally collaborate on each portion of the project.

Principles We Abided By

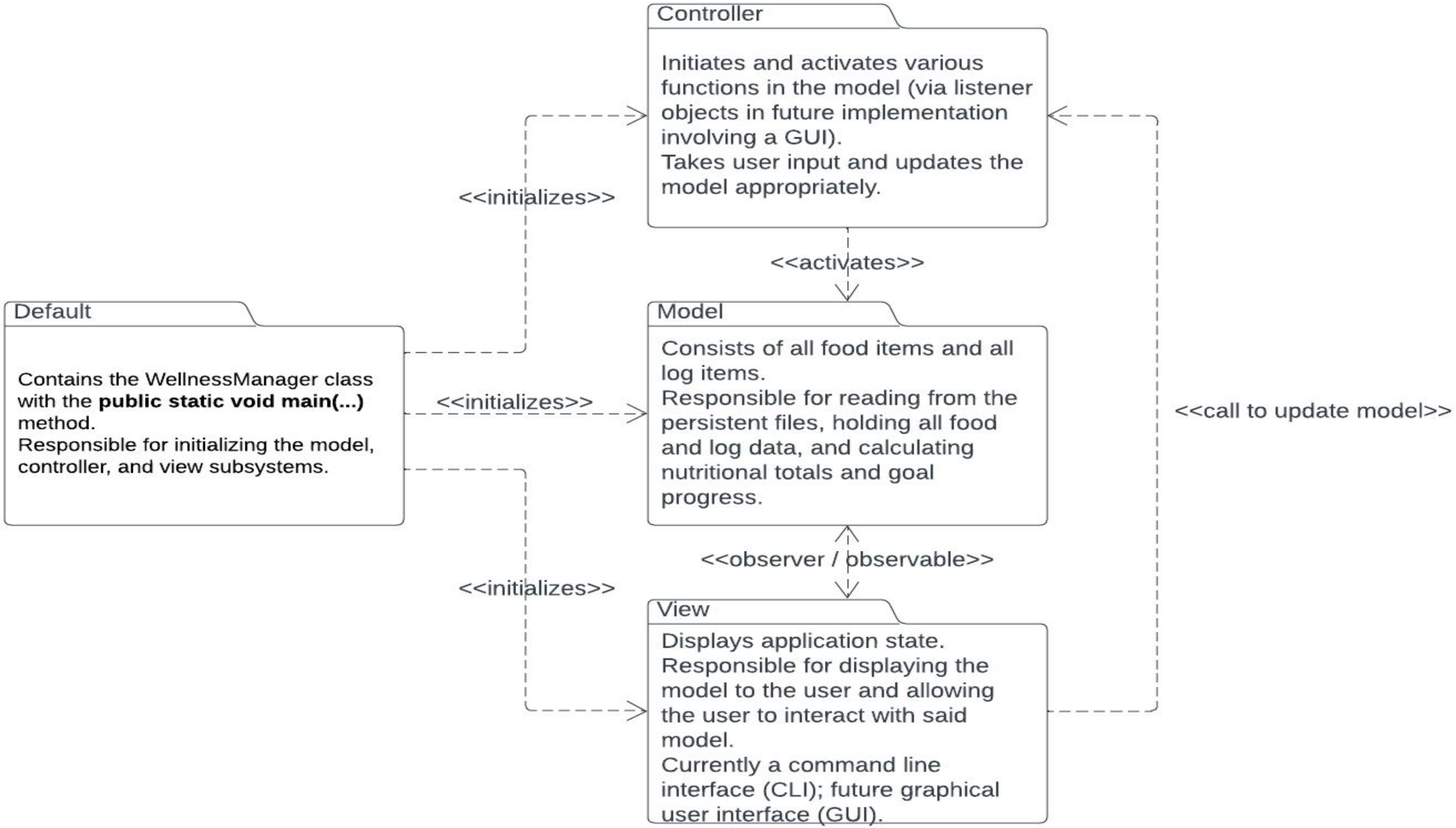
Principles of Good Design

- Low Coupling & High Cohesion
  - there is a small degree of interdependence between classes (only when necessary)
  - each component addresses a single issue
- Allowed us to implement the GUI efficiently (past the learning curve)

Mutual Respect

- Abide by the team contract from the beginning
- Respect each others commitments outside of the course
- Meet regularly to make sure we're on the same page

Architectural Overview - Subsystems



Controller

Initiates and activates various functions in the model (via listener objects in future implementation involving a GUI).
Takes user input and updates the model appropriately.

Default

Contains the WellnessManager class with the **public static void main(...)** method.
Responsible for initializing the model, controller, and view subsystems.

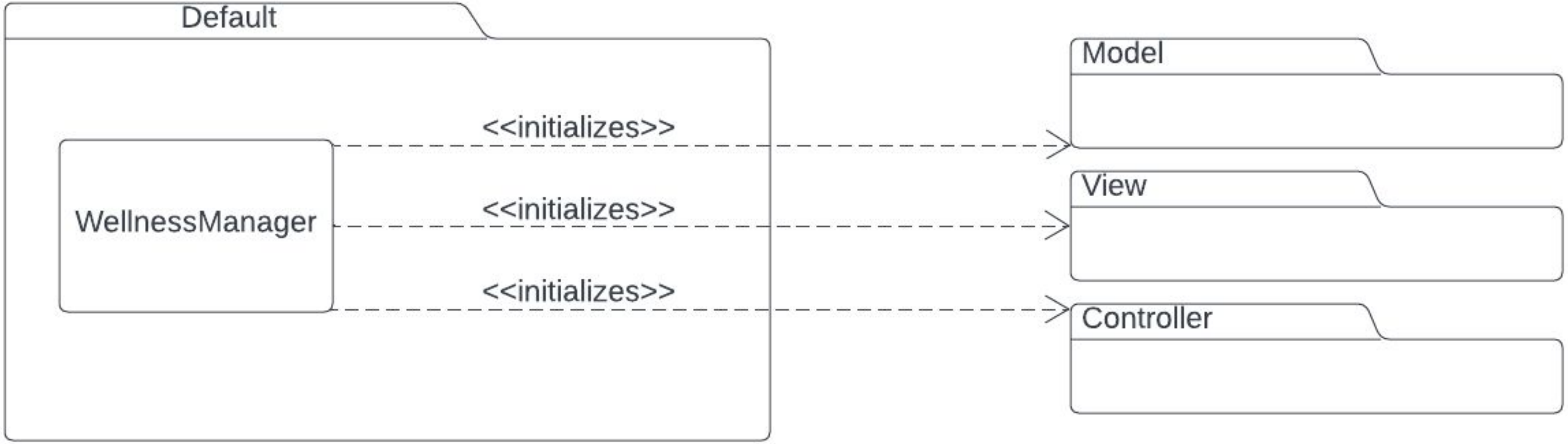
Model

Consists of all food items and all log items.
Responsible for reading from the persistent files, holding all food and log data, and calculating nutritional totals and goal progress.

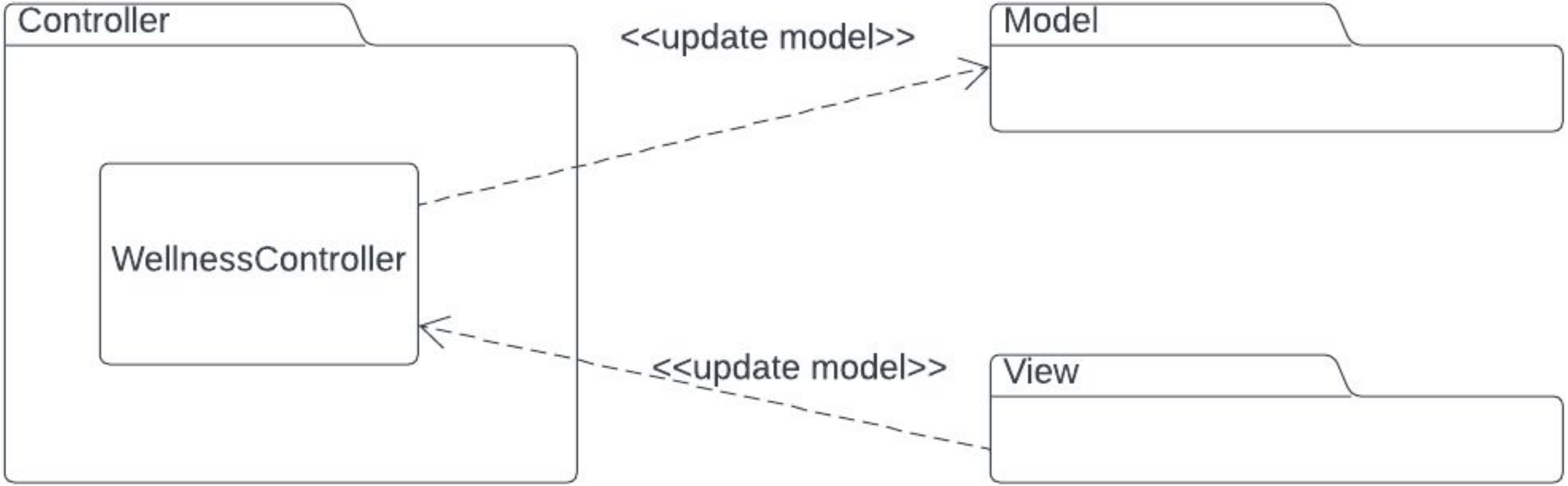
View

Displays application state.
Responsible for displaying the model to the user and allowing the user to interact with said model.
Currently a command line interface (CLI); future graphical user interface (GUI).

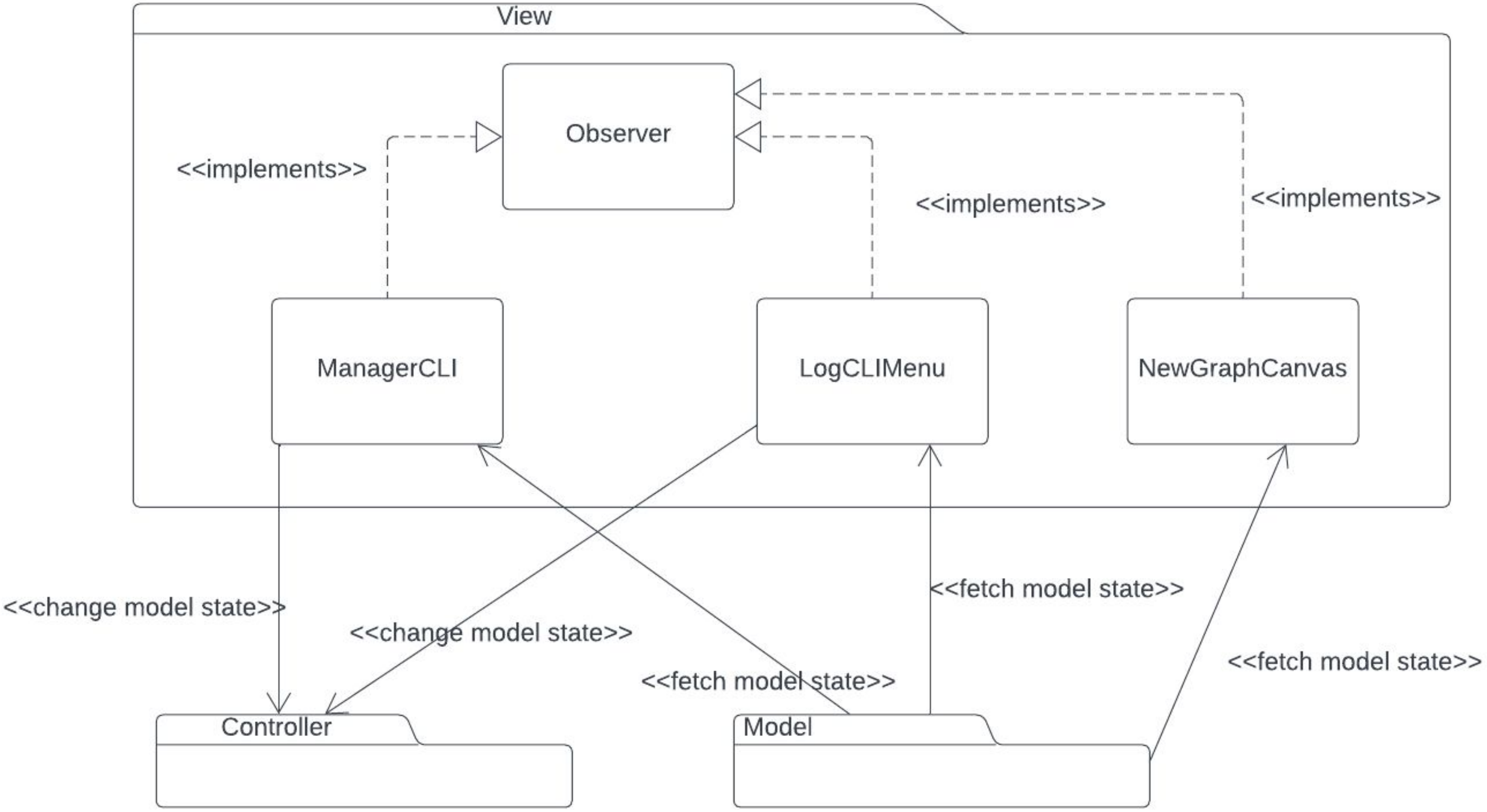
Design Overview - Default



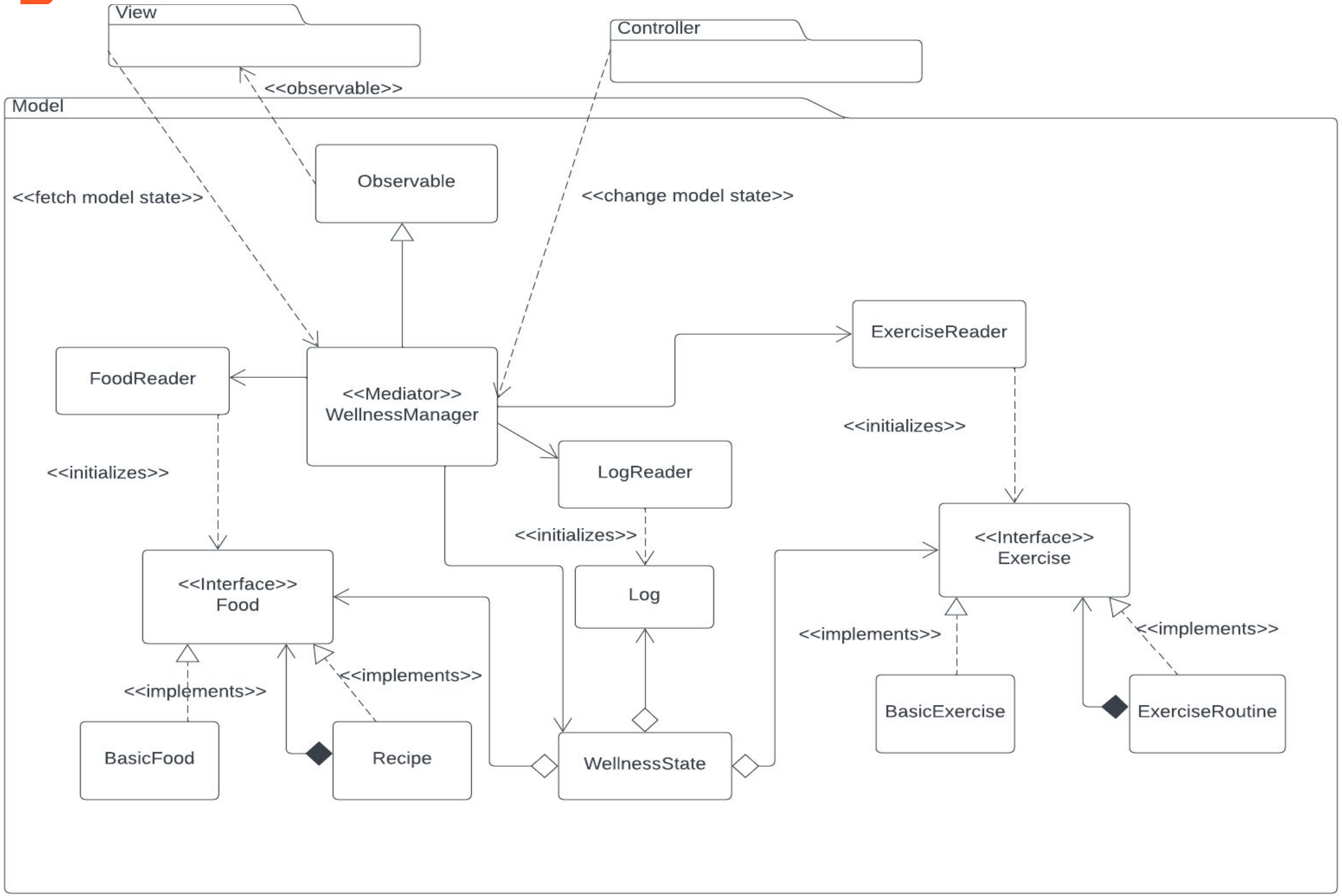
Design Overview - Controller



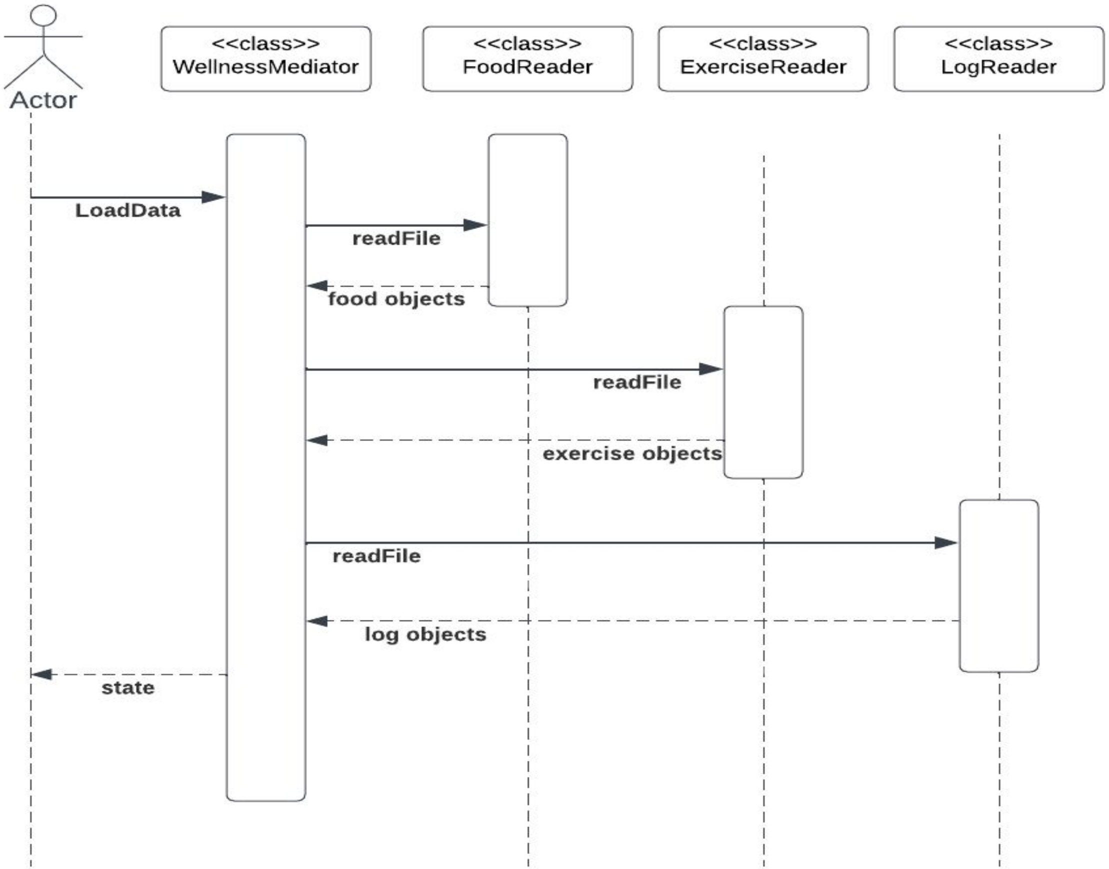
Design Overview - View



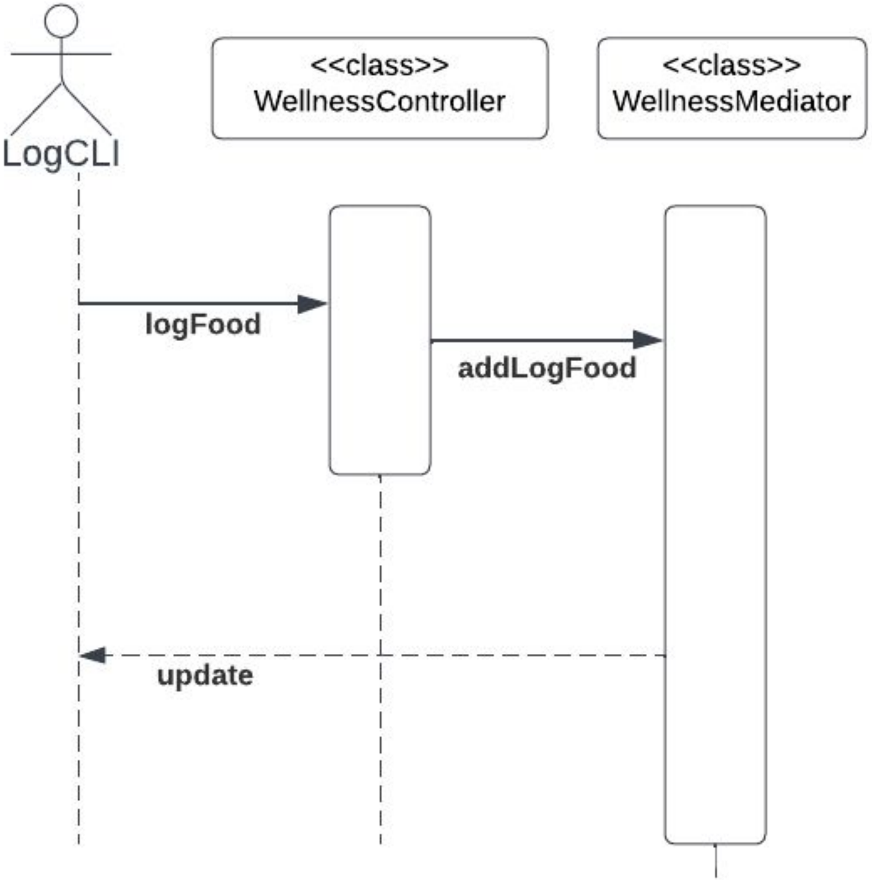
Design Overview - Model



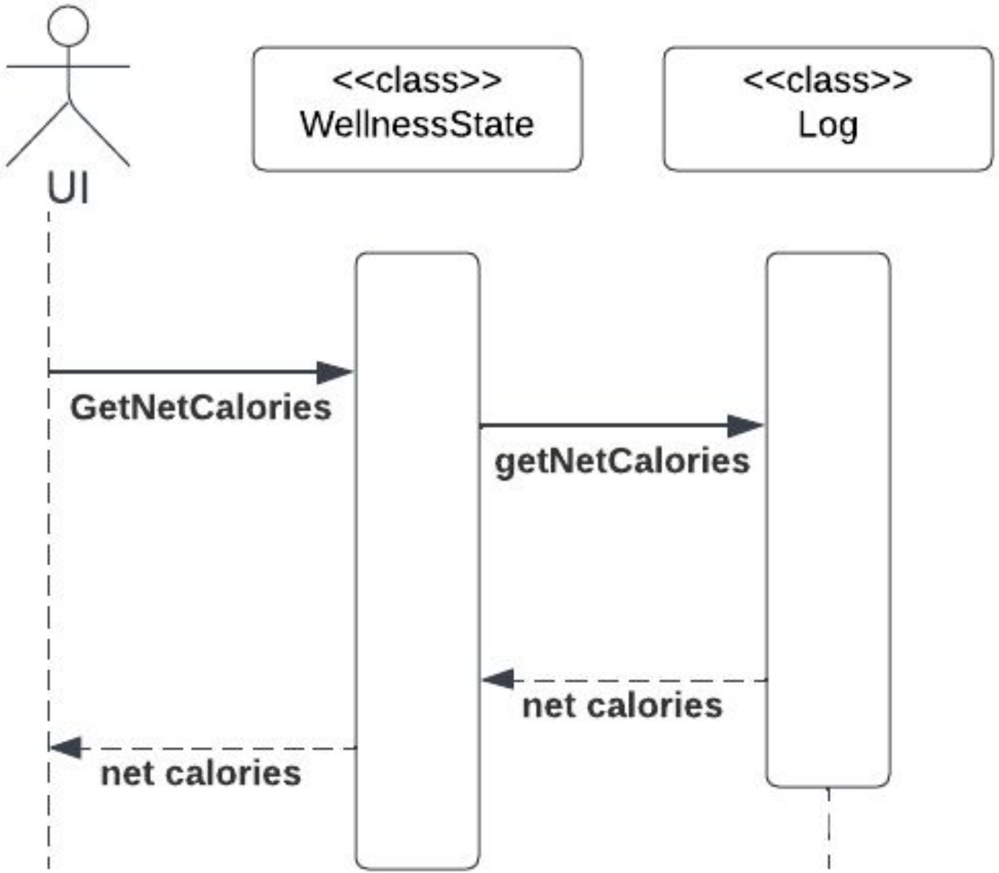
Sequence Diagram - Initialize Objects



Sequence Diagram - Add Food to Log



Sequence Diagram - Get Net Calories



Strong Points

- Exercise routines are ready to be implemented (extensibility)
- Switching from CLI to GUI was relatively straightforward
- All view classes update automatically when Model updates
- Team-wise
  - All willing to do work / help each other
  - Always asking “What can I do?”

Design Improvements & Weak Points

- Code organization
- Time management
- Observer pattern implementation
  - WellnessState object is coupled to many other classes
- Class bloat
  - WellnessState and WellnessMediator classes became larger than they should have been
- GUI inefficiency
  - We are relatively unfamiliar with Swing, which caused progress to slow
- Class names, especially after switching from CLI to GUI
  - Wanted to switch “CLI” to “UI” but did not have time / did not want to break anything

Lessons Learned

- Allocate time for a learning curves (Swing)
- The initial discussion of program patterns and justifications is crucial
  - It is much more difficult to implement patterns once you’ve started coding
  - Beyond good design, it helps the overall organization of the program as well as team coordination
- Technical debt is inevitable, but having too much of it will hurt the team

Demo

A demo of our program will now be given...