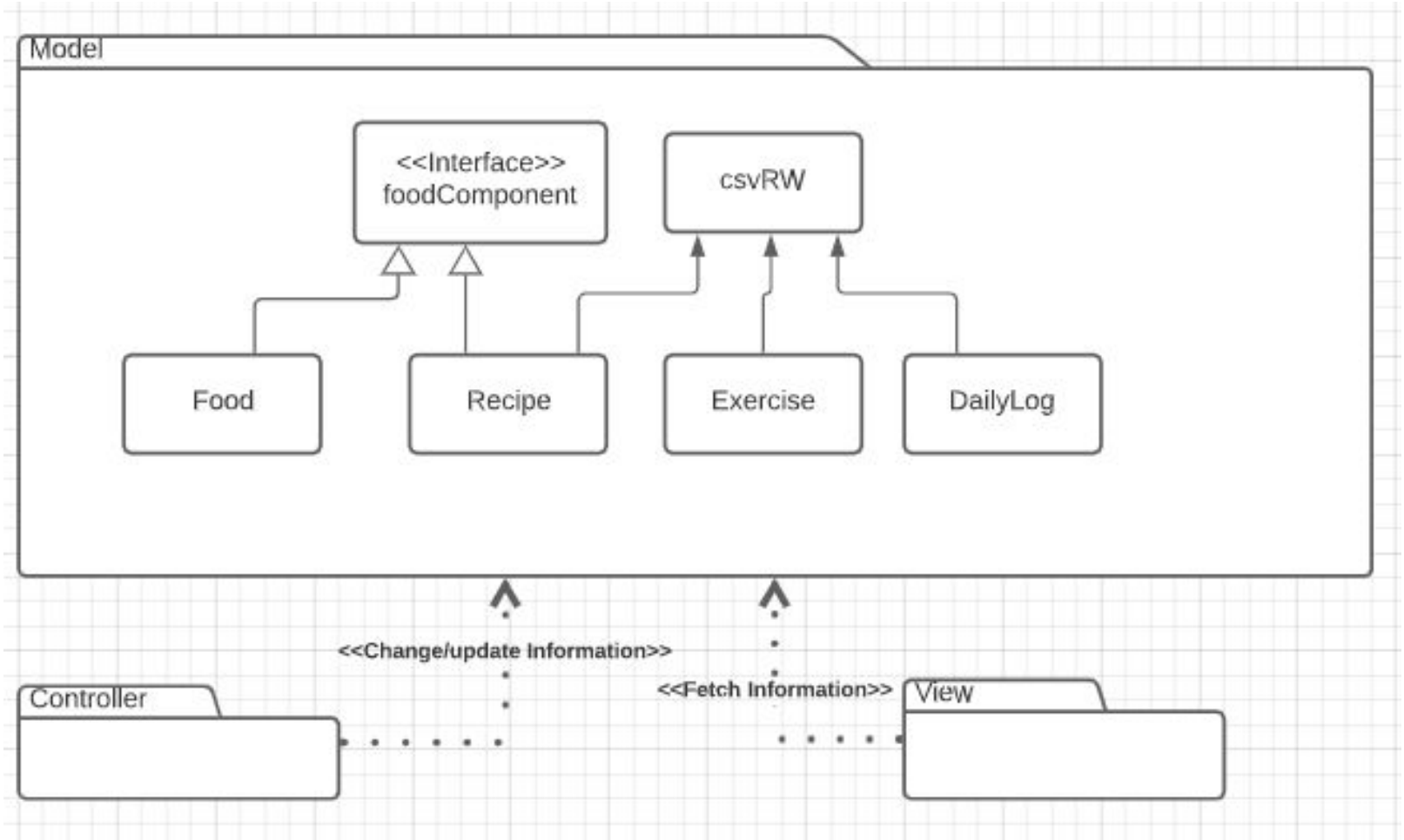
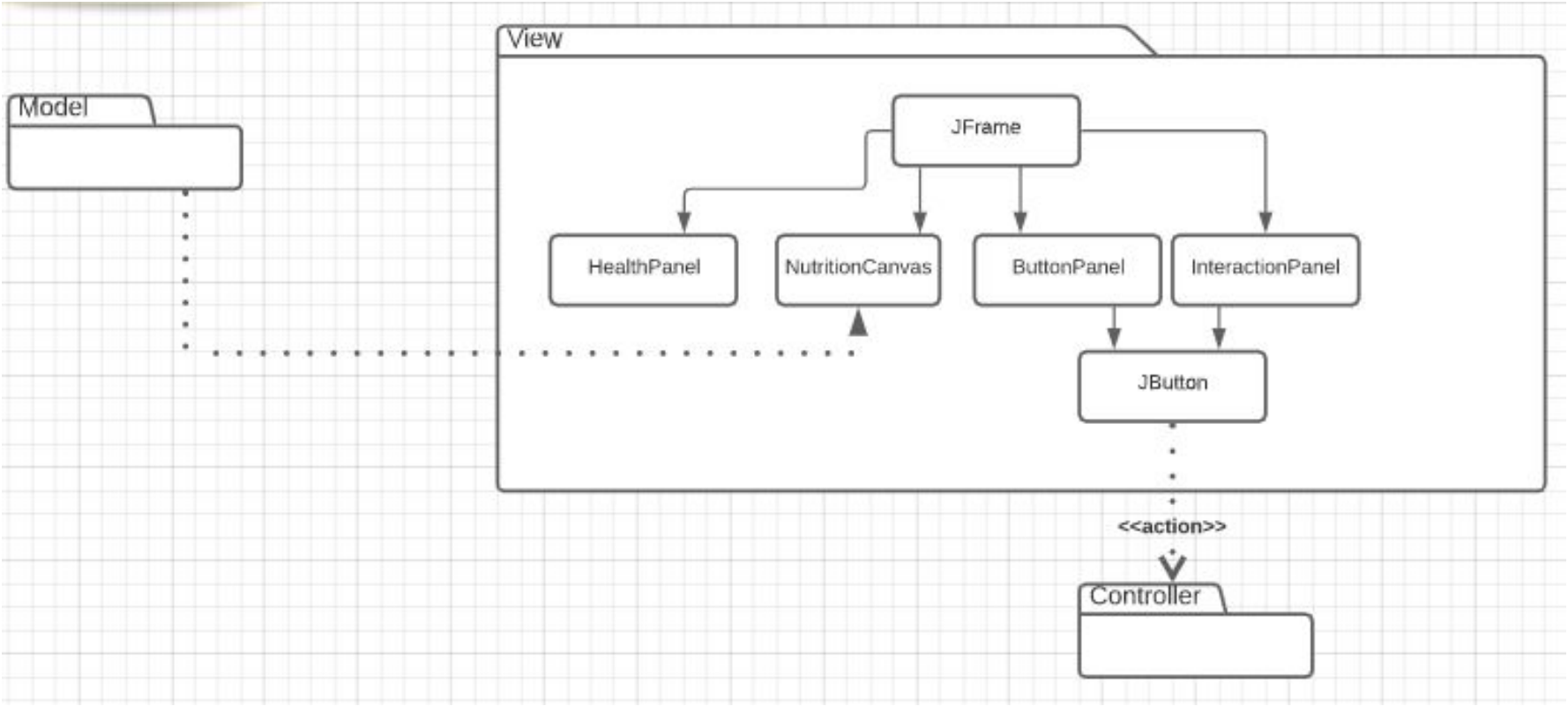
Team DeepBlue: Wellness Manager

Peter Schwarzkopf, Evan Hiltzik, Vincent Sze, Tianpeng Kuang, Yongheng Mei

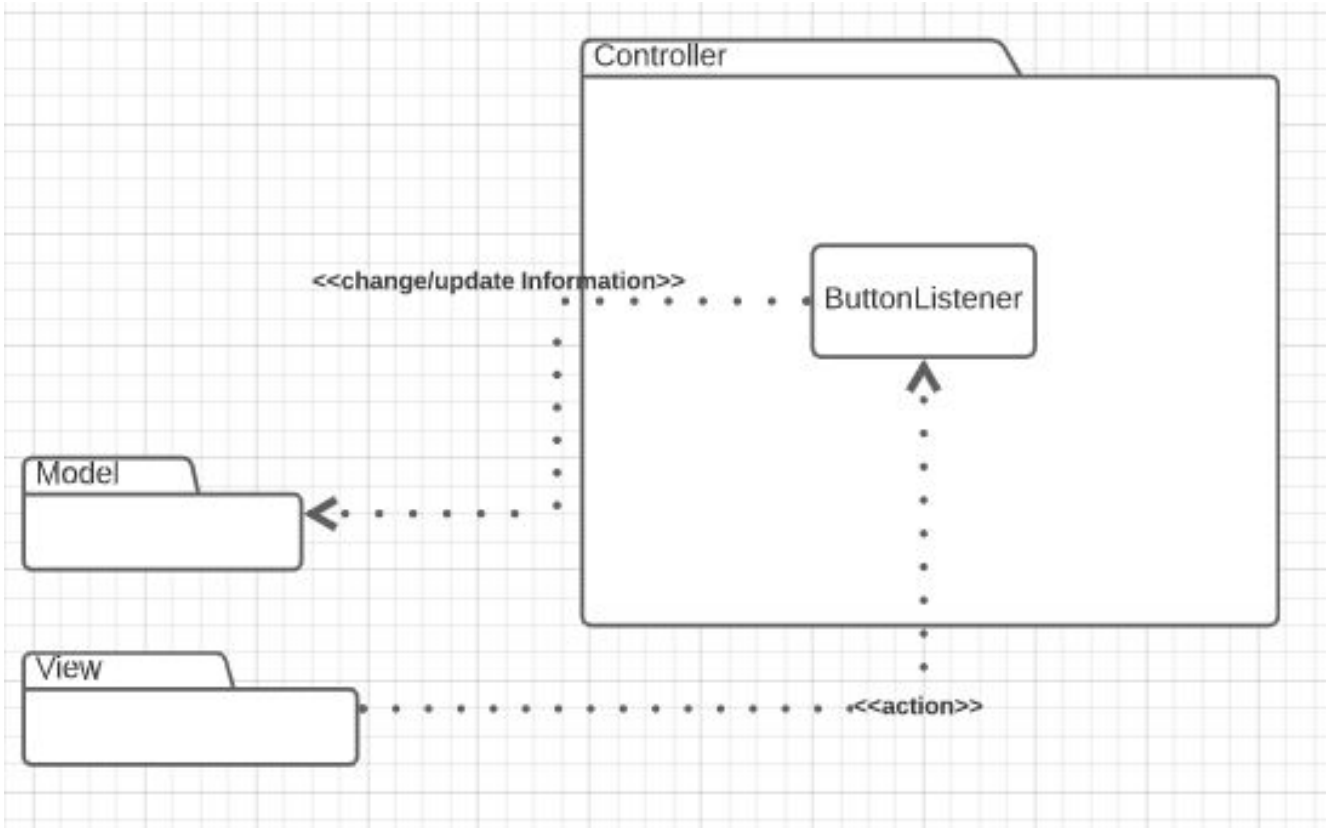
Static Model - UML (Model)



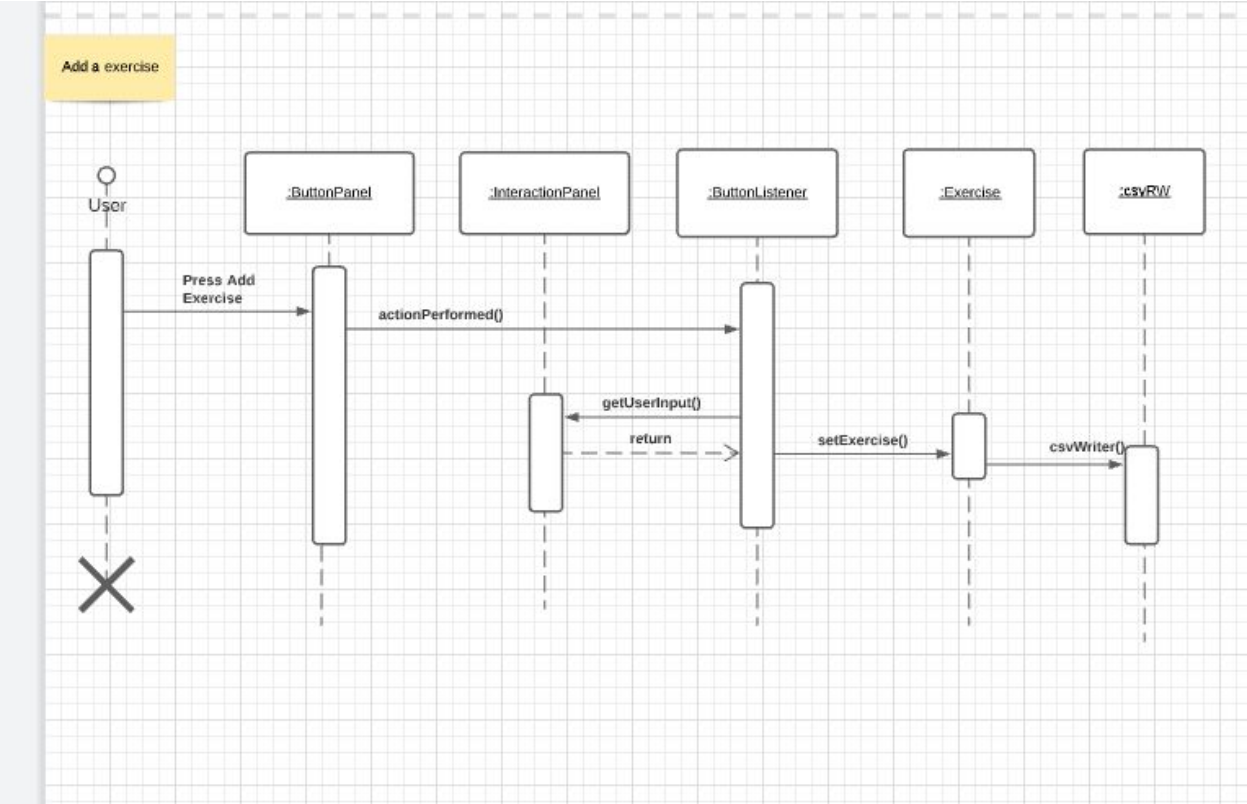
Static Model - UML (View)



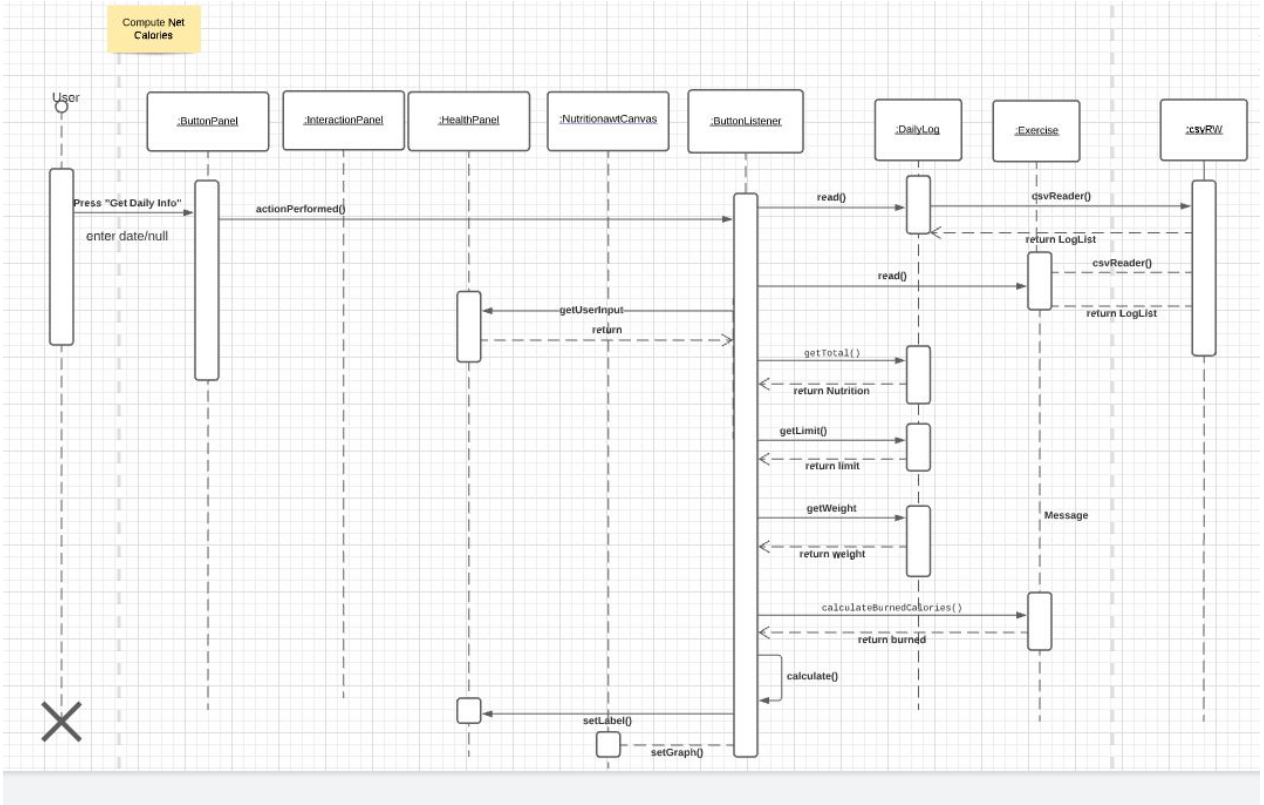
Static Model - UML (Controller)



Sequence Diagram #1 - Add exercise



Sequence Diagram #2 Update Panel



Implementation Process

- New skeleton
- Adding new classes and replacing unnecessary ones
- Changing Associations
- Patterns

Patterns Used

Composite Pattern:

Leaf: Food.java

Composite: Recipe.java

Component: FoodComponent.java (interface)

Patterns Used

MVC Pattern:

Model: csvRW.java, Recipe.java, Food.java, dailyLog.java, exercise.java FoodComponent.java

View: ButtonPanel.java, HealthPanel.java, InteractionPanel.java, NutritionCanvas.java

Controller: ButtonListener.java

Patterns Justification/Explanation

Composite: Food items are basic - like a slice of bread - and so their title and nutrition info can be stored simply. Recipes, however, can consist of both basic Food items and sub-Recipes, making the Composite Pattern a wise choice; Recipes can store both themselves and basic ingredients, and ingredients and recipes can both reference the FoodComponent interface for shared information.

Model/View/Controller: The separation of our software concerns naturally led us to the MVC pattern. The Model, consisting of the CSV and the storage structure of our Composite pattern, could be separated from our View, the UI, by a Controller, which interprets user input from the View. This prevents overly coupling our classes, and keeps our internal file structure organized. Our View consists of an AWT-based UI, rather than the text-based interface used previously. Our Controller has changed as well - rather than a CommandController detecting input from the command line, we implemented our ButtonListener, which picks up both text input and button presses, and delegates the proper tasks to our Model, which now contains the new Exercise and dailyLog classes for the exercise functions of the Wellness Manager.

Successes

- Took the feedback from the previous phase and spent more time on the design document and diagrams for greater reference material
- Good use of the MVC structure to separate classes & allowed for a complex but understandable file structure
- Bar graph and UI integration went smoothly

Challenges

- Our main challenge was converting what we had in our diagrams and design document into code
    - We can plan, but have a hard time implementing
- GUI optimization
- Time management and work delegation definitely still affected us for this phase

Time for the Demo!

Question ?