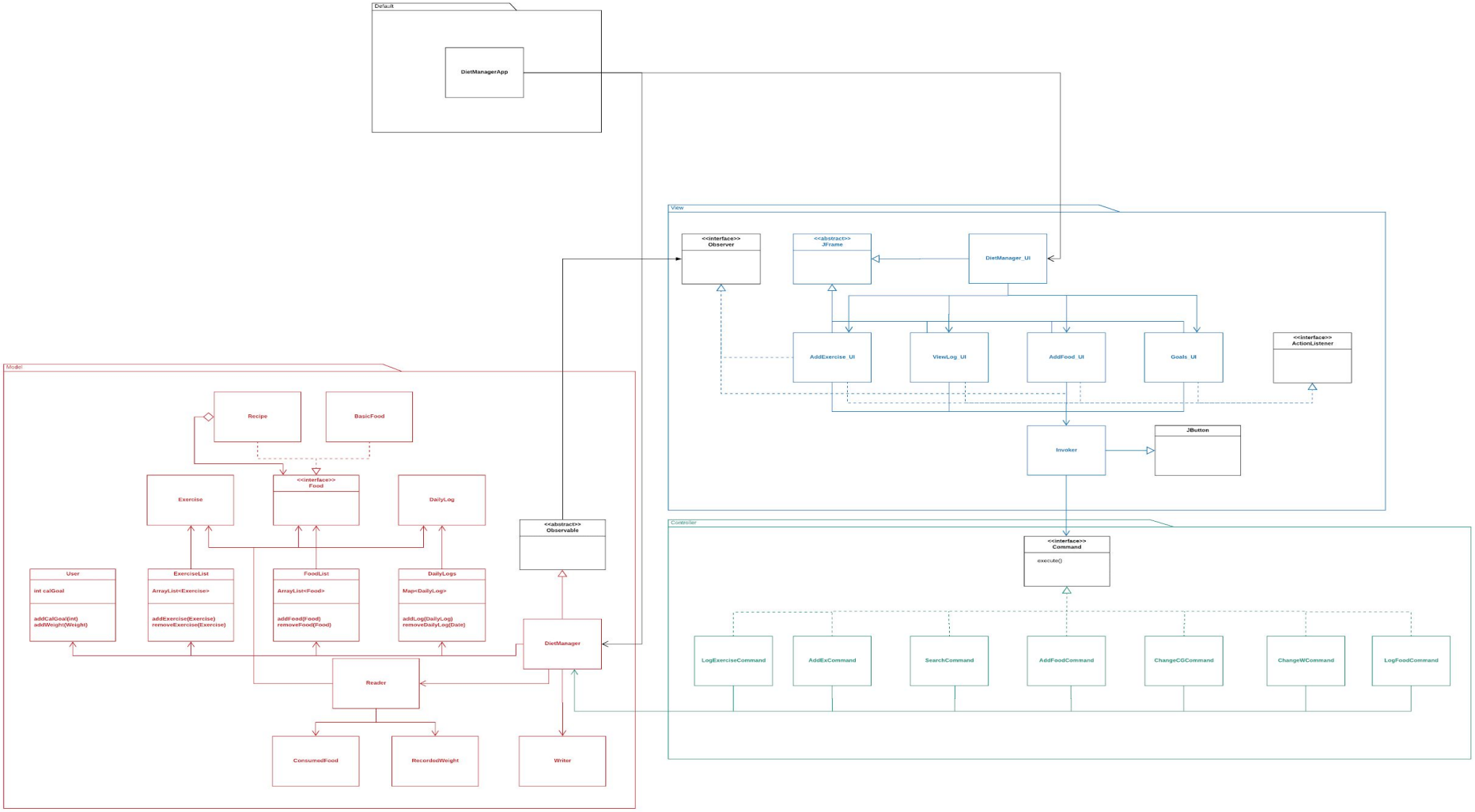
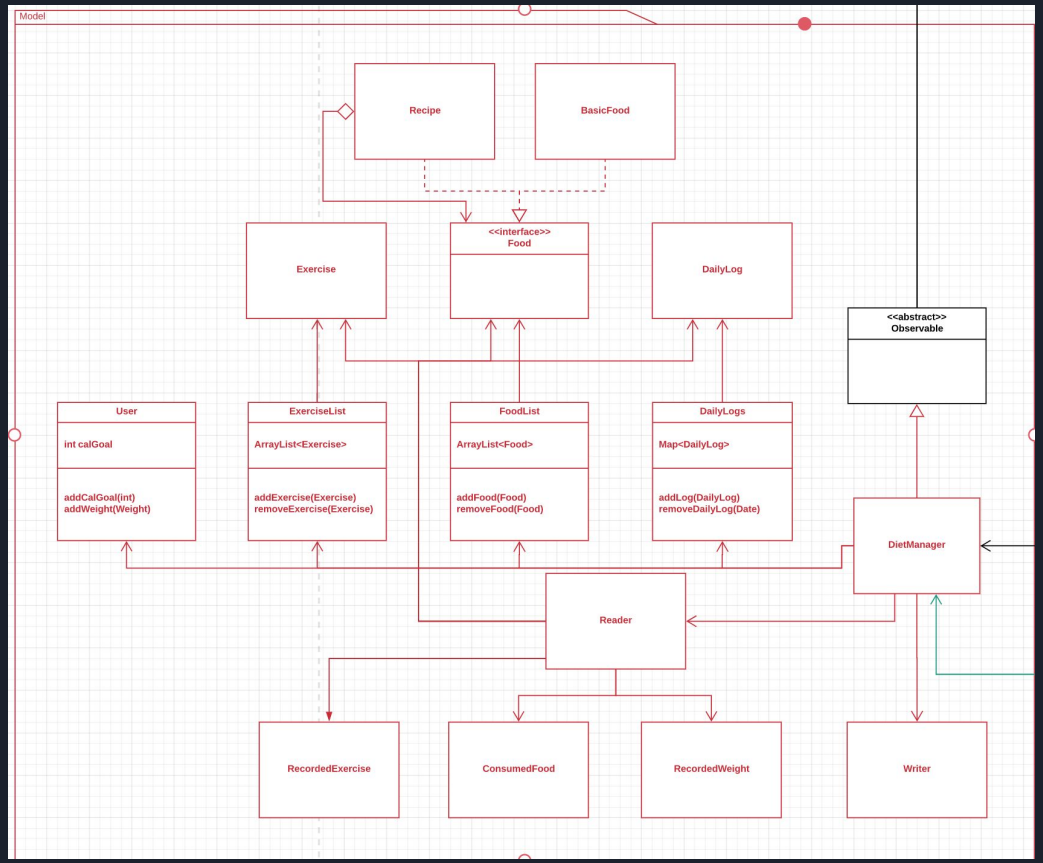
Team 1 - Phase 2 Presentation

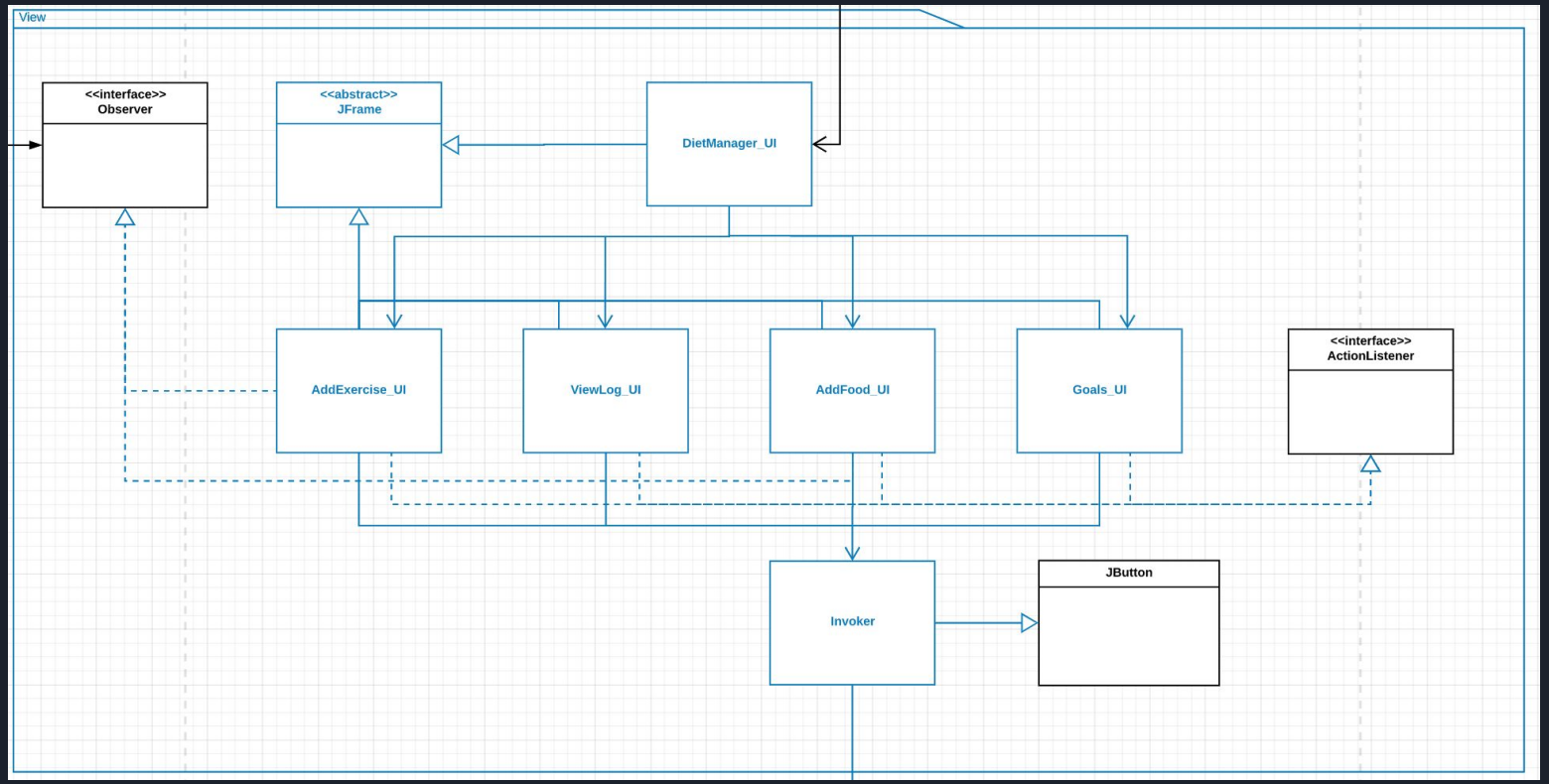
Abbey Sands, Anna Jacobsen, Tim Endersby, John Hill, Xuting Zhang


Default
DietManagerApp
View
<<interface>>
Observer
<<abstract>>
JFrame
DietManager_UI
AddExercise_UI
ViewLog_UI
AddFood_UI
Goals_UI
<<interface>>
ActionListener
Invoker
JButton
Model
Recipe
BasicFood
Exercise
<<interface>>
Food
DailyLog
<<abstract>>
Observable
User
int calGoal
addCalGoal(int)
addWeight(Weight)
ExerciseList
ArrayList<Exercise>
addExercise(Exercise)
removeExercise(Exercise)
FoodList
ArrayList<Food>
addFood(Food)
removeFood(Food)
DailyLogs
Map<DailyLog>
addLog(DailyLog)
removeDailyLog(Date)
DietManager
Reader
ConsumedFood
RecordedWeight
Writer
Controller
<<interface>>
Command
execute()
LogExerciseCommand
AddExCommand
SearchCommand
AddFoodCommand
ChangeCGCommand
ChangeWCommand
LogFoodCommand

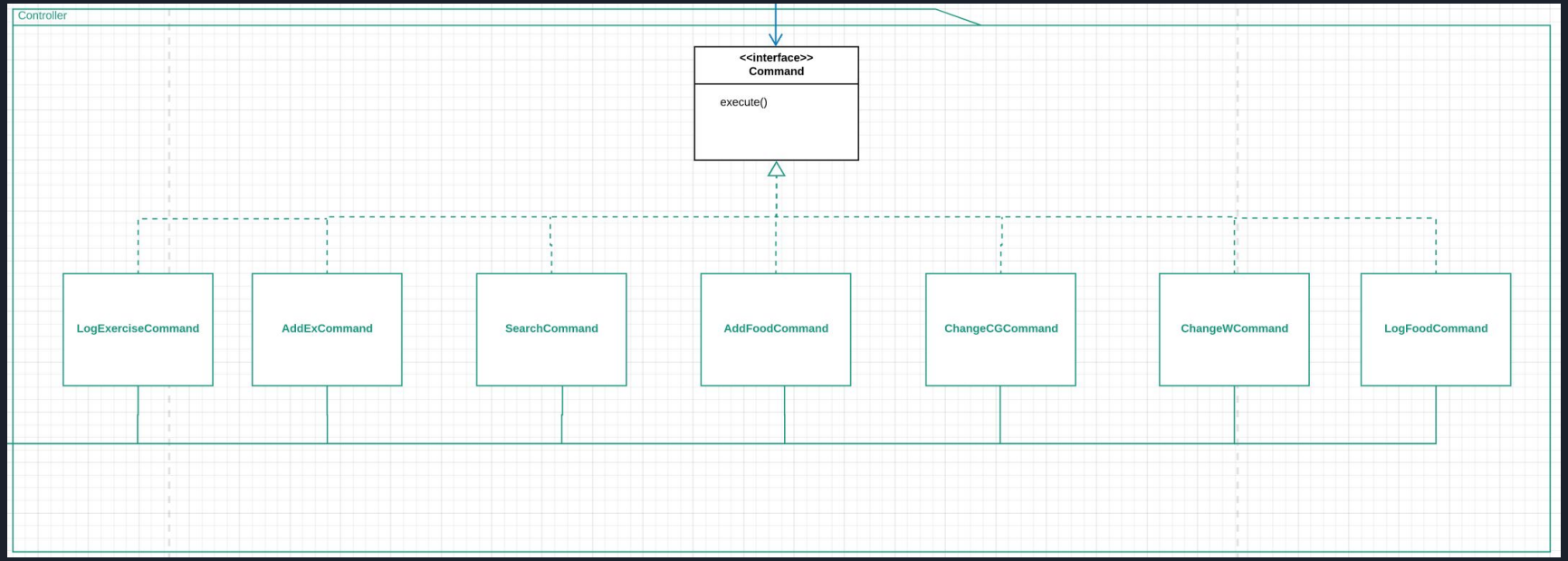
Model


Model
Recipe
BasicFood
<<interface>>
Food
Exercise
DailyLog
<<abstract>>
Observable
User
int calGoal
addCalGoal(int)
addWeight(Weight)
ExerciseList
ArrayList<Exercise>
addExercise(Exercise)
removeExercise(Exercise)
FoodList
ArrayList<Food>
addFood(Food)
removeFood(Food)
DailyLogs
Map<DailyLog>
addLog(DailyLog)
removeDailyLog(Date)
DietManager
Reader
RecordedExercise
ConsumedFood
RecordedWeight
Writer

View


View
<<interface>>
Observer
<<abstract>>
JFrame
DietManager_UI
AddExercise_UI
ViewLog_UI
AddFood_UI
Goals_UI
<<interface>>
ActionListener
Invoker
JButton

Controller


Controller
<<interface>>
Command
execute()
LogExerciseCommand
AddExCommand
SearchCommand
AddFoodCommand
ChangeCGCommand
ChangeWCommand
LogFoodCommand

Design Overview - Why Did We Go With This Design?

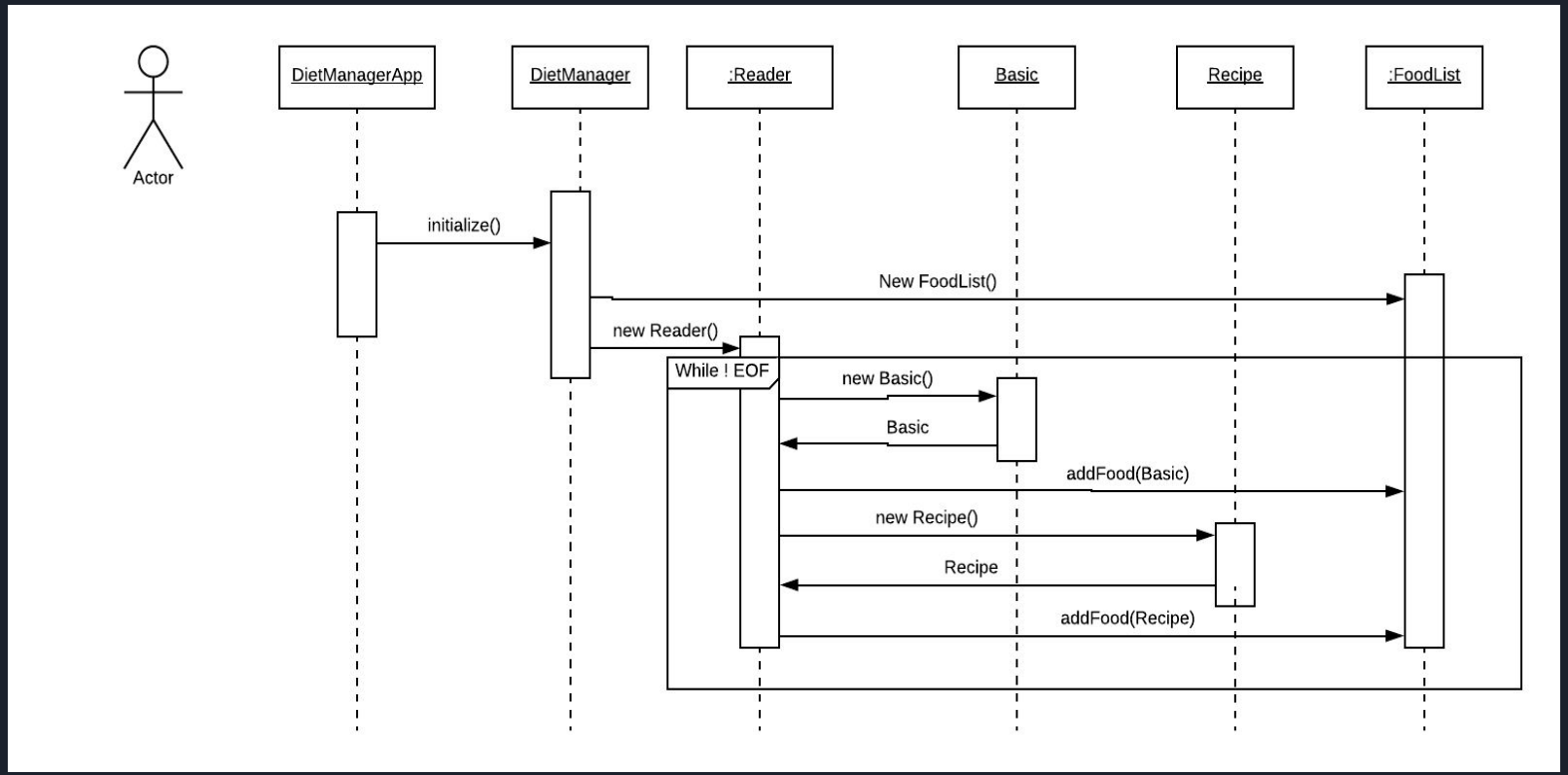
The most noticeable change between our final design and our design from phase 1 was the implementation of the command pattern.

We chose to implement the command pattern in our design because it provides a level of abstraction between the view and the controller. It also gives the client the ability to more easily extend the program later on if he chooses to add more functionality. Lastly, our team simply wanted to implement the pattern for practice.

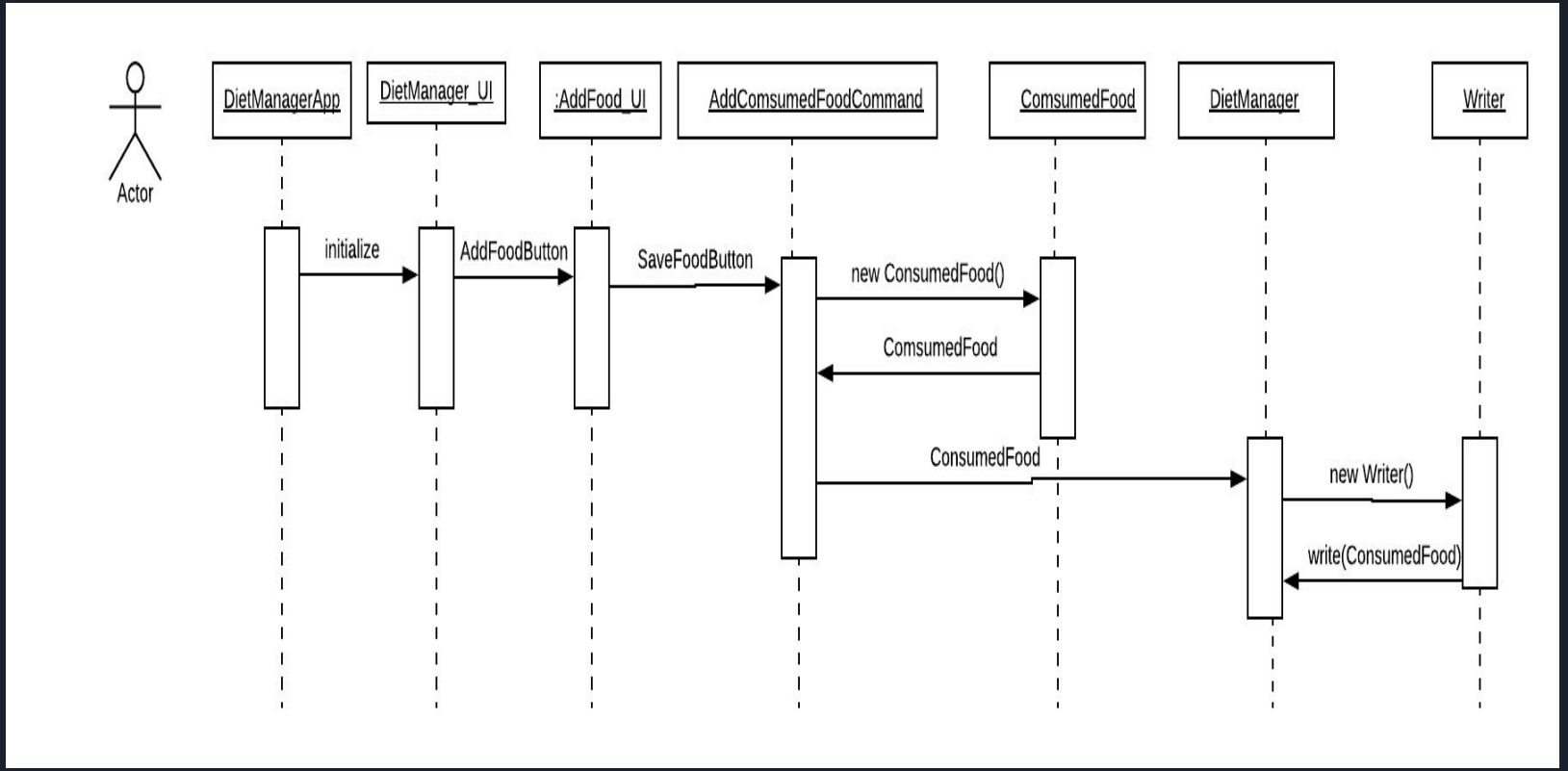
Other than that we focused on cleaning up a few of our loose ends, like removing the SubRecipe class, improving upon the previously implemented observer pattern, and prioritizing separation of concerns. This prioritization can be seen in the new classes ExerciseList, FoodList, and DailyLogs.

This iteration of the design was worked on equally by the entire team, and we believe it is a large improvement upon our phase 1 design.

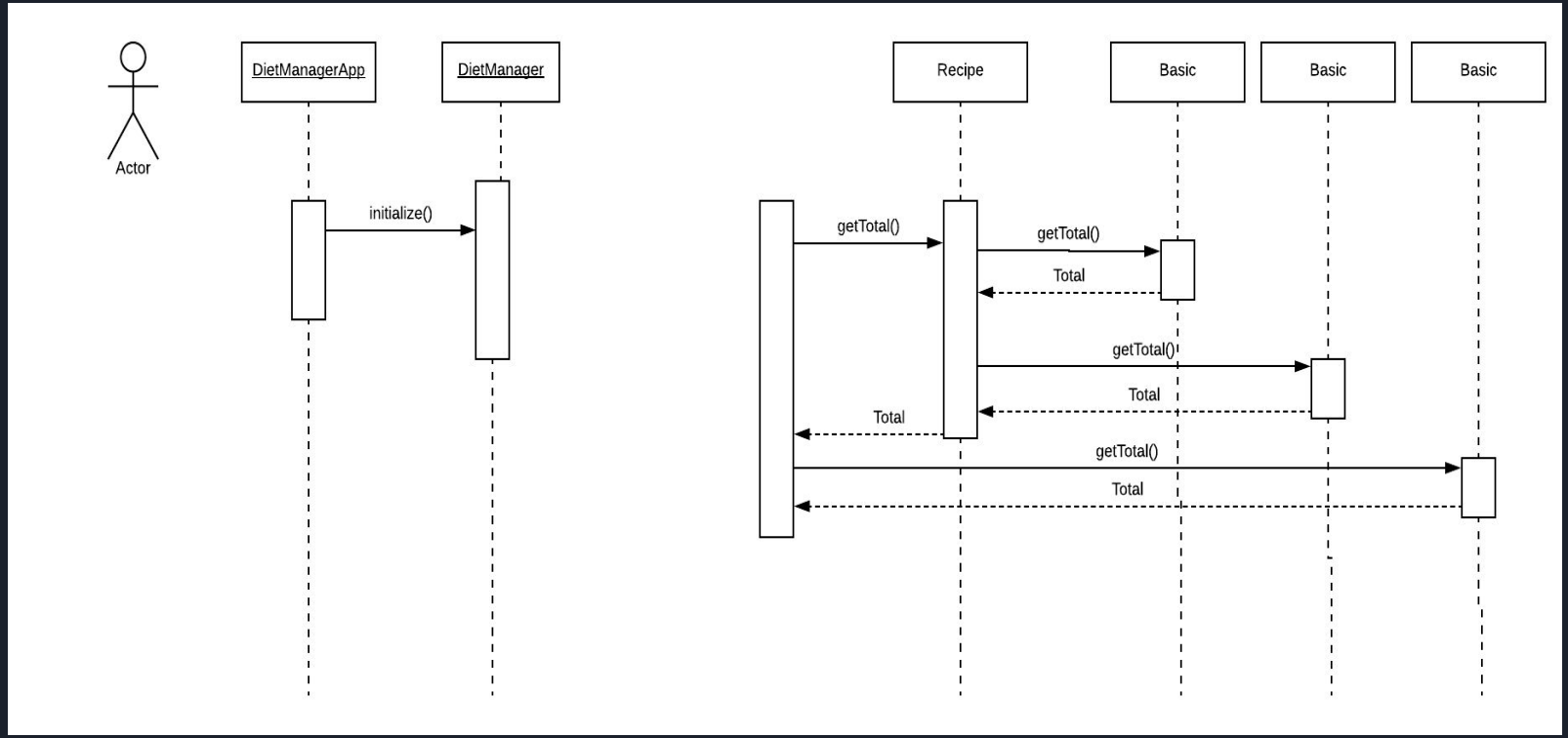
Sequence Diagram 1:



Sequence Diagram 2:



Sequence Diagram 3:



The Good - What Went Well?

Some things we believe went well for Phase 2 include:

- We kept MVC separated very well in our design. This allowed us to have separation of concerns, low coupling, and high cohesion in our program. Other things we implemented in our design include the command pattern, and the observer pattern which also helped lower coupling. We achieve a higher cohesion by having all of our classes focus on a specific part of the program instead of trying to multitask.
- Teamwork and communication improved during phase 2. Team design sessions allowed everyone to see what was going on with the uml and get the big picture. We remained committed to meeting three times per week, were open with our discussions, and kept our Slack and Trello pages very well updated/maintained.

The Bad - What Could Have Gone Better?

Some things we believe could have gone better during Phase 2 include:

- The design got stuck at times, which delayed the start of the implementation, causing a bit of frustration.
- Other classwork was a bit of a distraction, sometimes making it hard to focus on this project.
- The team is still a little shaky on sequence diagrams as a whole.

A Short Demonstration...

Please enjoy this brief demo of our program.

Questions?

Any at all?

Fin

Thank you for listening!