HealthNCare App

Team Charlie

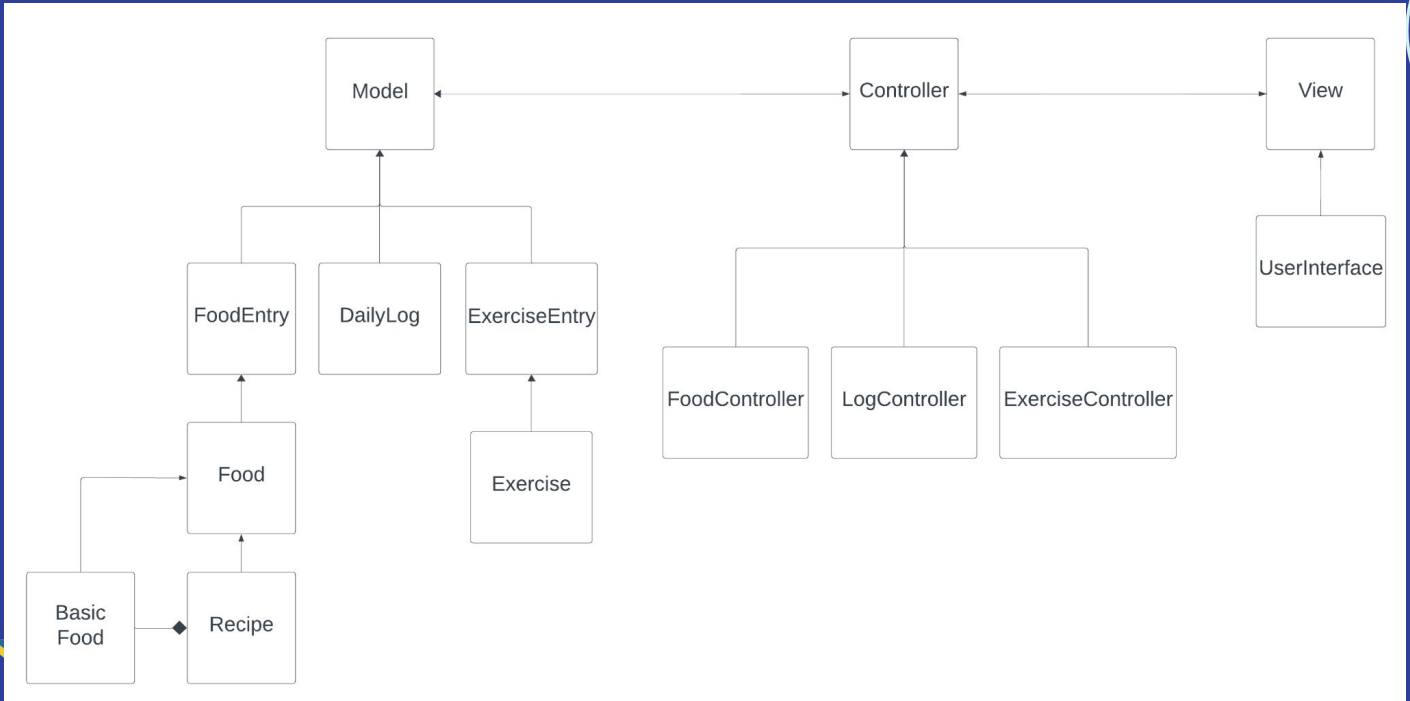
Zaher Asad, Allan Flores, Alfred Franco, Ethan Gapay, Alex Tedesco, & Nolan Wira

Table of Contents

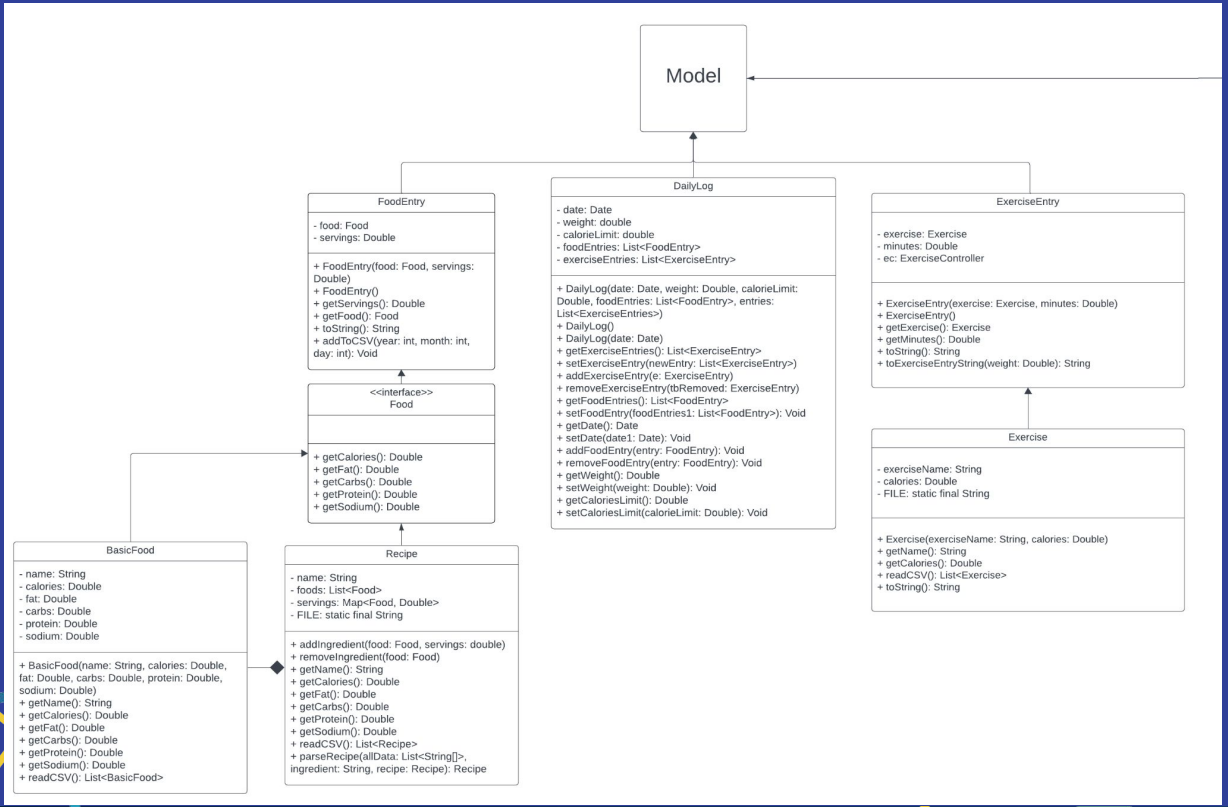
01

Final System Design Overview

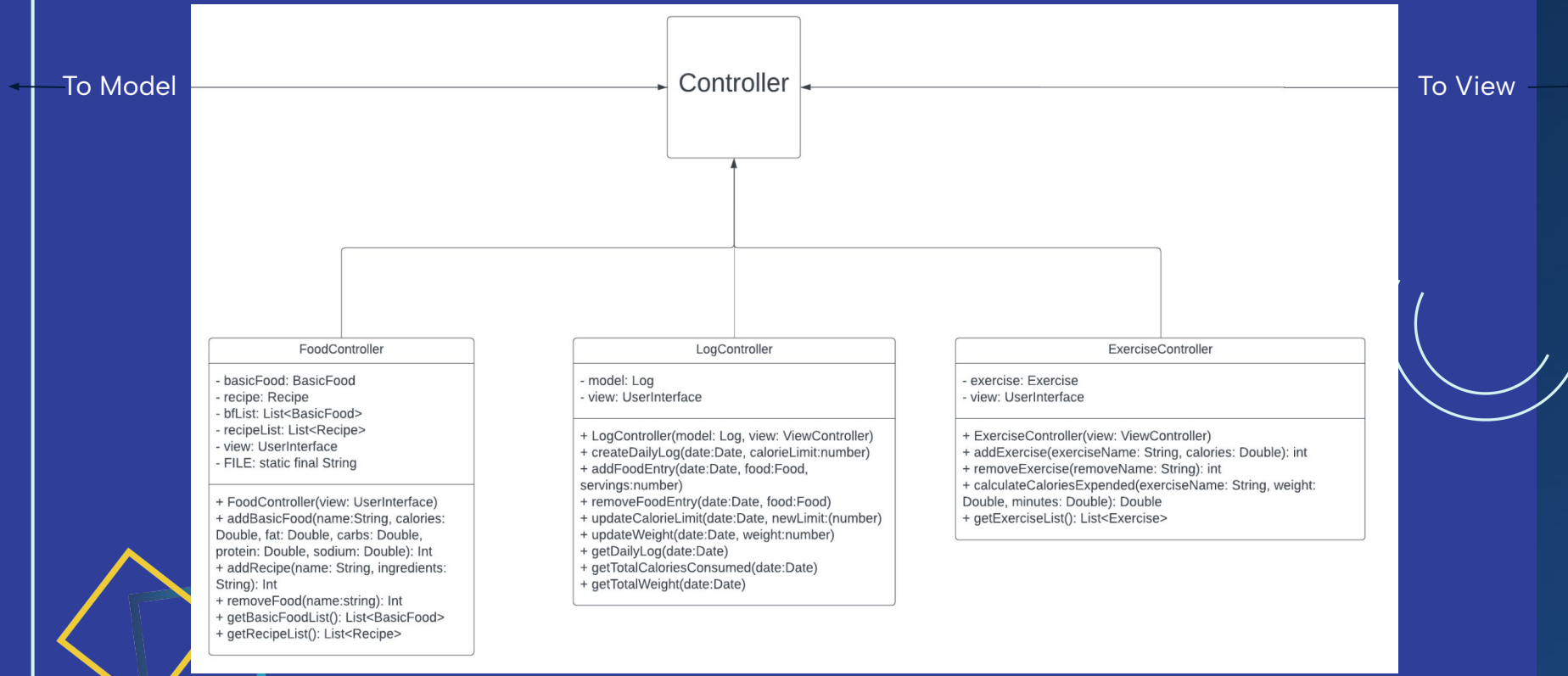
Domain Diagram



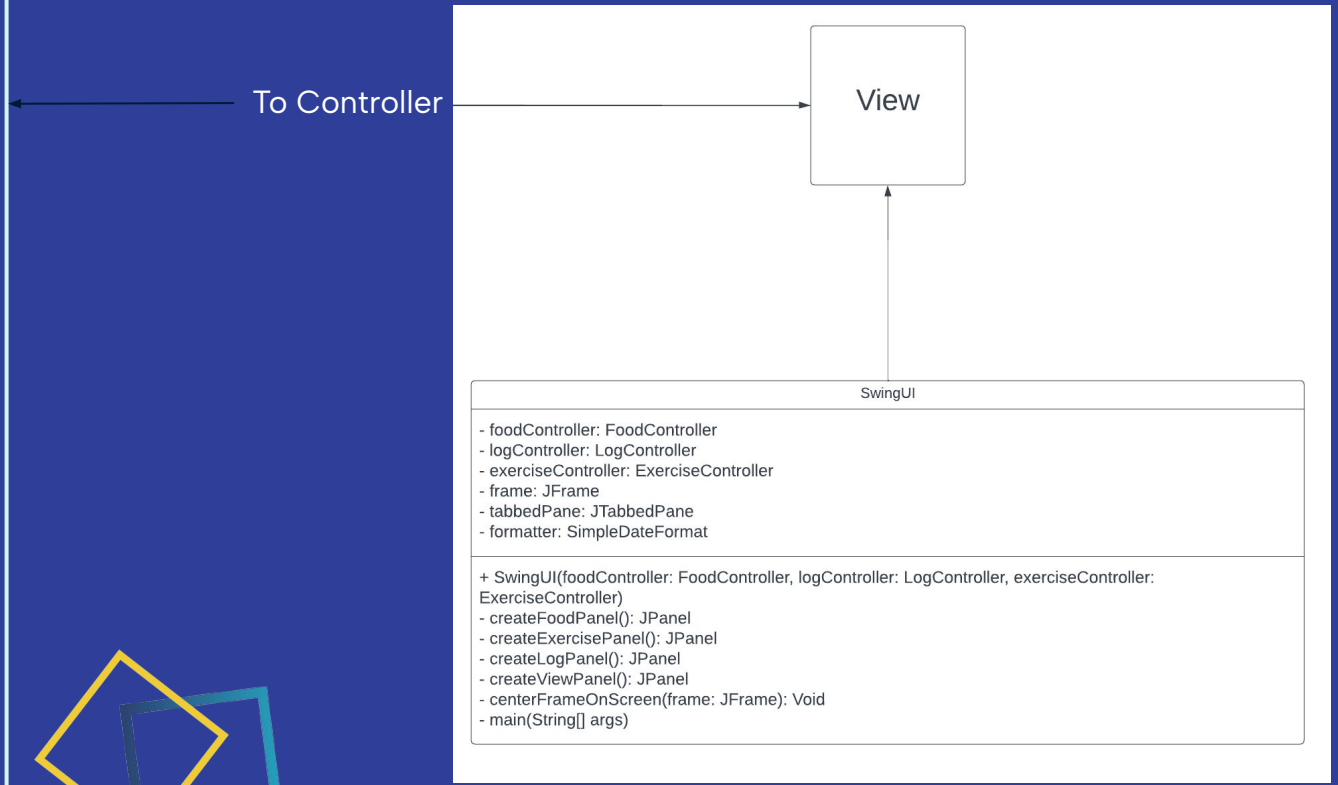
Class Diagram - Model



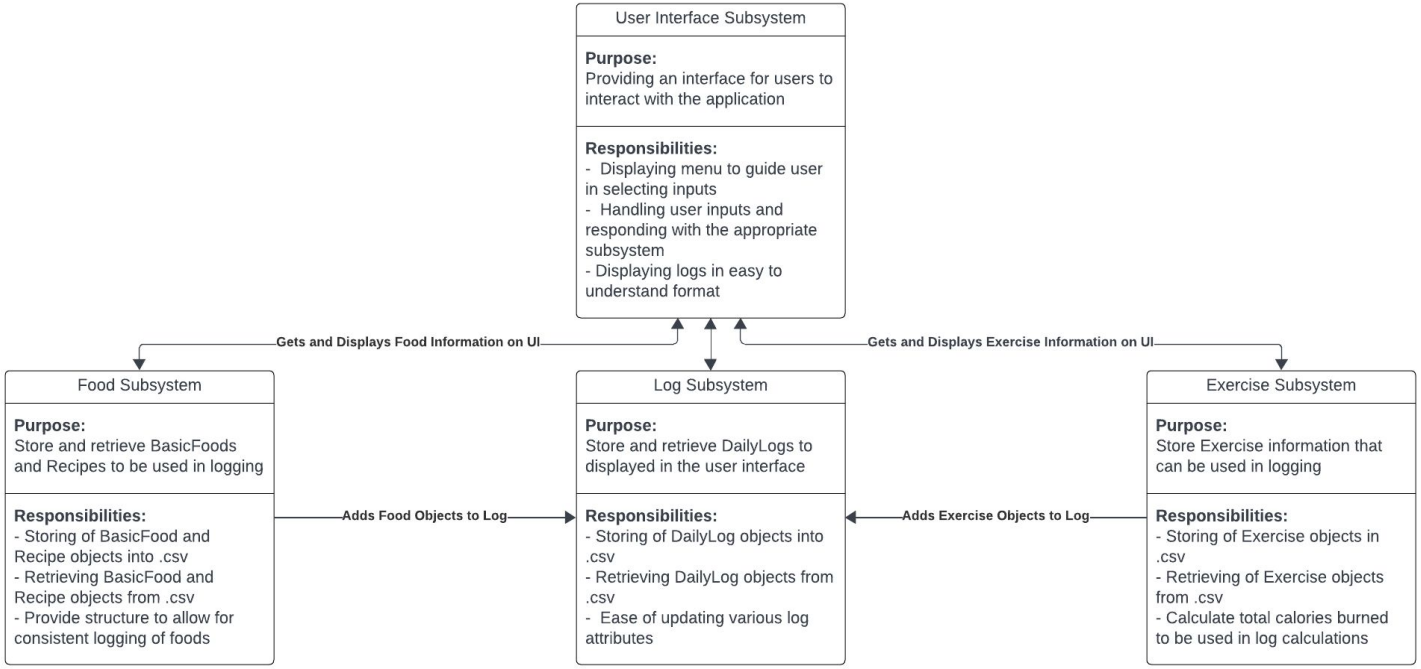
Class Diagram - Controller



Class Diagram - View

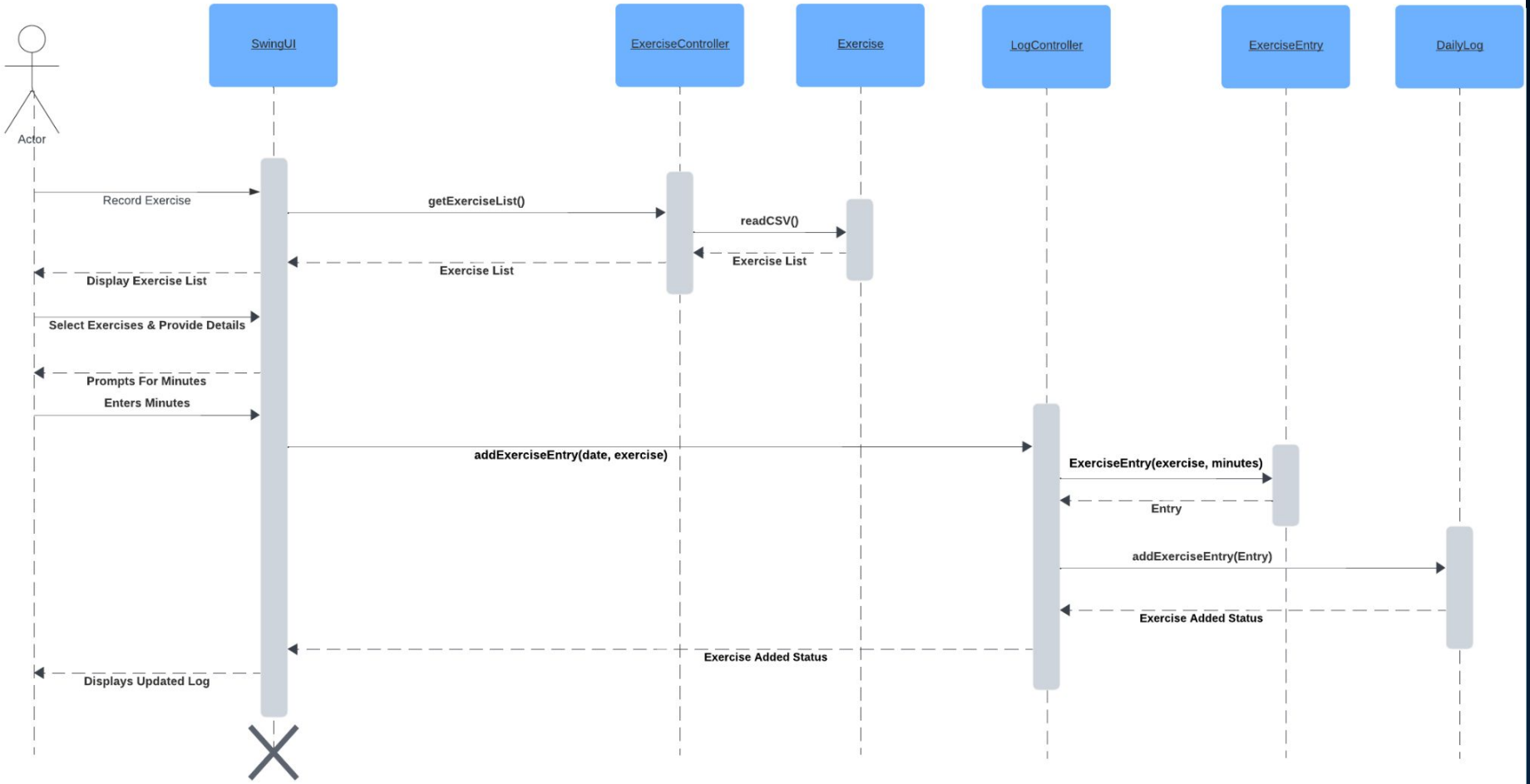


Subsystem Diagram



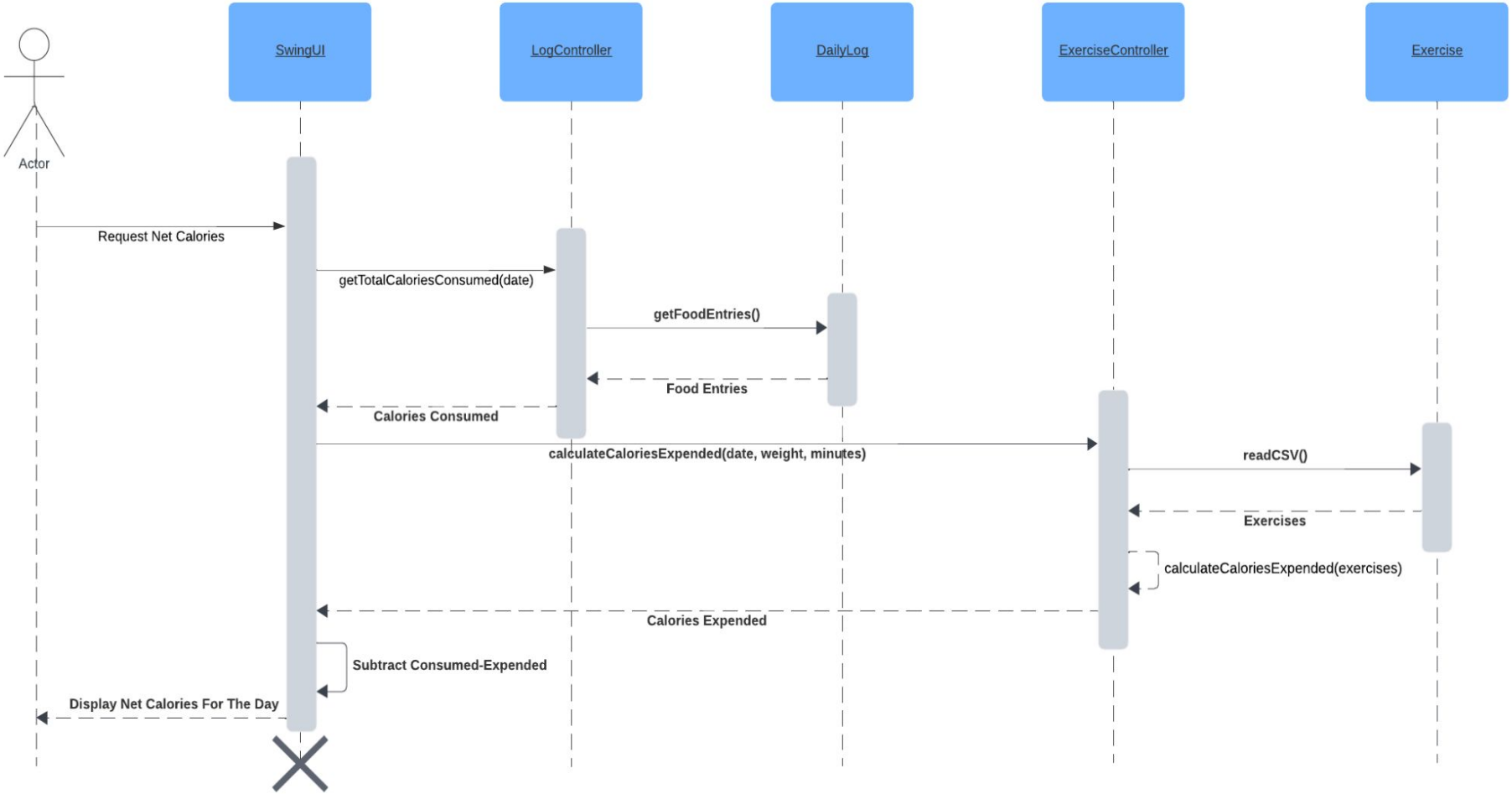
Sequence Diagram #1

Recording Exercises


SwingUI
ExerciseController
Exercise
LogController
ExerciseEntry
DailyLog
Actor
Record Exercise
getExerciseList()
readCSV()
Exercise List
Exercise List
Display Exercise List
Select Exercises & Provide Details
Prompts For Minutes
Enters Minutes
addExerciseEntry(date, exercise)
ExerciseEntry(exercise, minutes)
Entry
addExerciseEntry(Entry)
Exercise Added Status
Exercise Added Status
Displays Updated Log

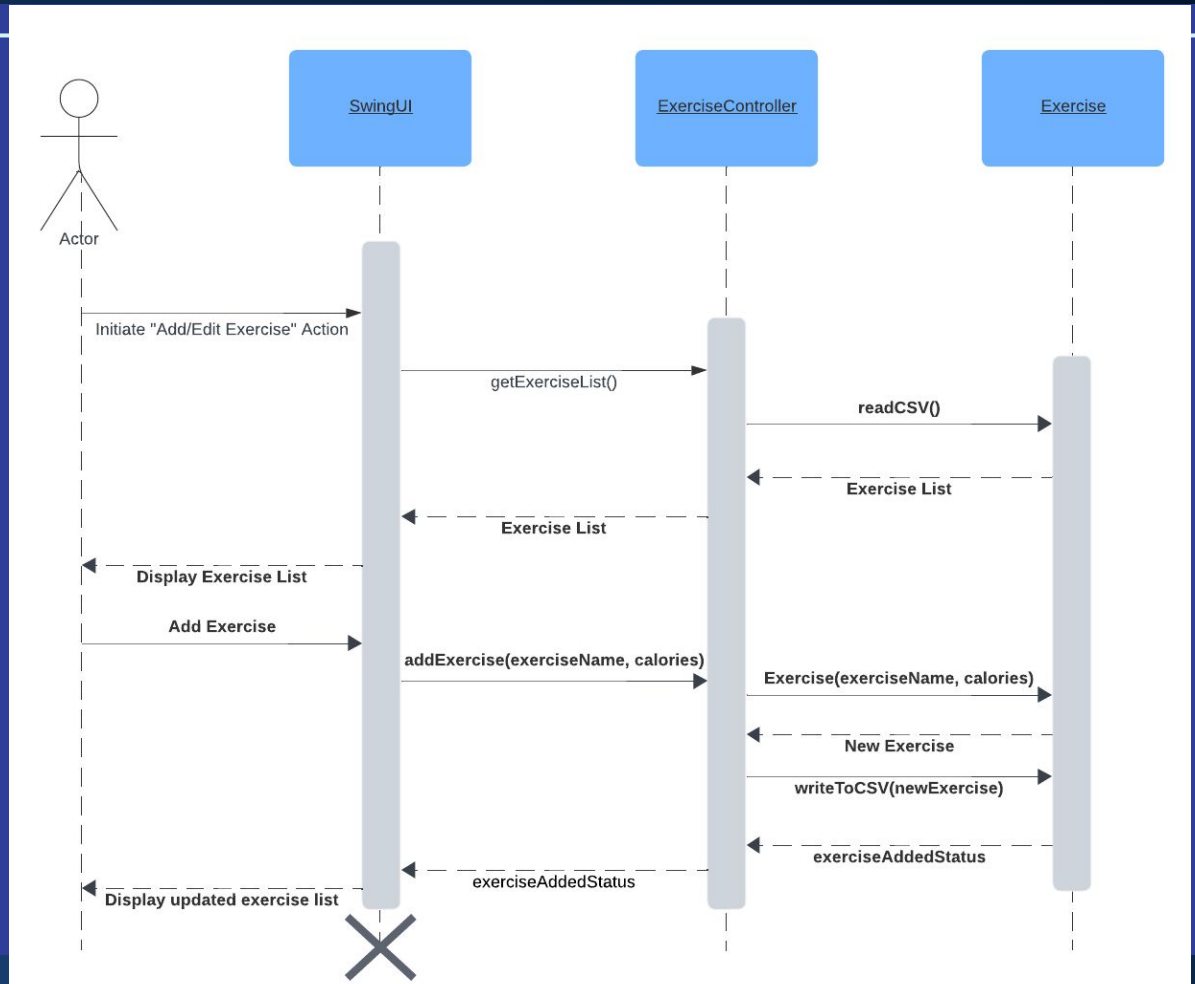
Sequence Diagram #2

Subtracting calories depleted from calories consumed to compute net calories per day.


SwingUI
LogController
DailyLog
ExerciseController
Exercise
Actor
Request Net Calories
getTotalCaloriesConsumed(date)
getFoodEntries()
Food Entries
Calories Consumed
calculateCaloriesExpended(date, weight, minutes)
readCSV()
Exercises
calculateCaloriesExpended(exercises)
Calories Expended
Subtract Consumed-Expended
Display Net Calories For The Day

Sequence Diagram #3

Adding Exercises


SwingUI
ExerciseController
Exercise
Actor
Initiate "Add/Edit Exercise" Action
getExerciseList()
readCSV()
Exercise List
Exercise List
Display Exercise List
Add Exercise
addExercise(exerciseName, calories)
Exercise(exerciseName, calories)
New Exercise
writeToCSV(newExercise)
exerciseAddedStatus
exerciseAddedStatus
Display updated exercise list

Changes From Version 1.0

Refined Log/Food Subsystems

Log and Food subsystems redone to adhere to MVC architecture

Enhanced CSV Operations

Improved CSV reading and writing capabilities.
(OpenCSV)

Exercise Subsystem

Implemented the Exercise Subsystem (Exercise.java + ExerciseController.java)

CLI to SwingUI

Transitioned from a CLI application to Swing/AWT GUI application.

Bar Chart Integration

Implemented BarChart for CSV data.
(JFreeChart)

Updated Diagrams

Made improvements to our Class and Domain Diagrams for better implementation.

02

The Good

The Good

Effective Use of MVC Architecture

Displaying separation of concerns, breaking program responsibilities up

High Cohesion and Low Coupling

Splitting each functionality into its own subsystem
High cohesion within a subsystem
Low coupling between subsystems

Easily Understandable Flow

Using the MVC architecture allowed for a clear understanding on where certain processes took place
Helped developers easily understand what classes/methods to call

Team Communication

Constant communication between developers working on different parts of MVC
Team members understood what was to be completed

03

The Bad

The Bad

Poor Timing

Our group did not set deadlines for what had to be done for the project
This led to last minute rushes to complete

Design Implementation

After designing our initial UML, we realized that actually implementing it posed problems
A lot of changes had to be made from the UML to the code

Initial Understanding

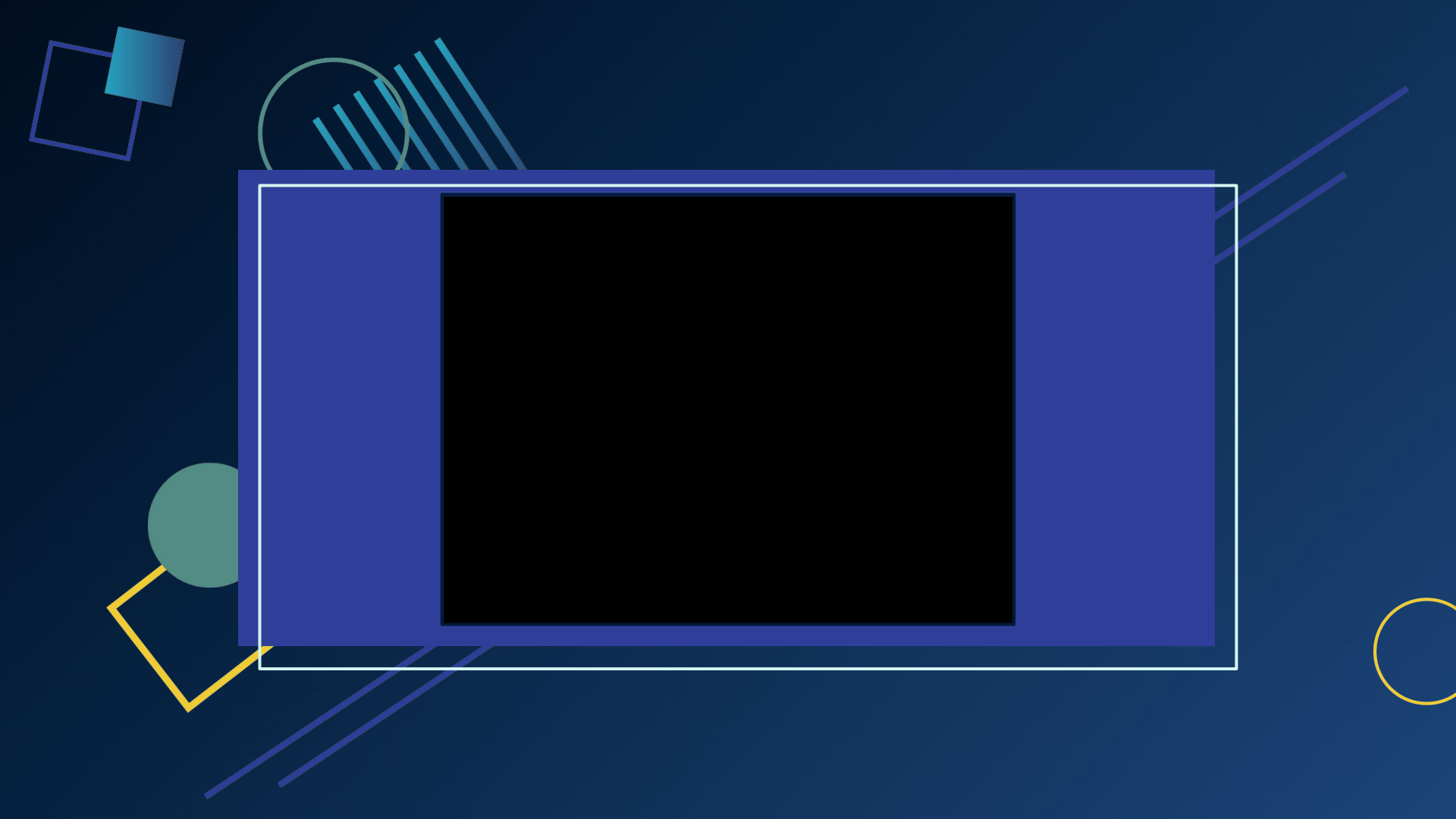
Starting of the project, team members still were unsure about how the program would flow
Not until implementation that design started to click

Design Conflicts

Differences in developers methodologies led to code that was logically different throughout

04

HealthNCare Demo



05

Conclusion

Conclusion

Important takeaways:

- Have all team members understand the flow of the application during initial planning
- Set and enforce deadlines for each component of the application
- Keep strong communication between members for future projects
- Include code design during initial design phase

06

Q&A Session