Oct. 15, 2021

Phase 0 Progress Report: HiveMind

Specification Summary

In the specification, the reader is given a brief walkthrough of the two main functions of our software - to create food objects that are either perishable or non perishable, and to suggest recipes based on available ingredients.

Scenario Walk-through

The walkover goes through a simple example of using the program to receive a suggested recipe based on what the user has: a potato. First, the food and recipes are loaded by the frameworks and drivers, which passes information about the food (potato) and available recipes to the controllers. The controllers call the handler methods to create the food and recipe objects, and the use cases return a list of recipes based on the foods that the user has inputted. In this scenario, the system will return potato related recipes.

More details can be found in the walk through.

CRC Card Summary

The CRC cards summarize an overview of the design of our recipe-suggestion system. This includes one command line interface, two controllers, two use cases, and three entities. In terms of clean Architecture, CommandGUI, CommandInput and DataParser represent Frameworks and Drivers, FoodController and RecipeController represent Controllers, FoodHandler and RecipeHandler are Use Cases, and Food, PerishableFood, NonPerishableFood and Recipe demonstrate entities.

Skeleton Code

During Phase Zero, we have designed, developed, and coded the following code:

4 entity classes were developed:

- PerishableFood, NonPerishableFood, Recipes, and the abstract class Food.
    - These classes present Food and Recipe objects.

2 use cases were developed:

- FoodHandler and RecipeHandler.
    - These classes both store Food and Recipe objects.
    - RecipeHandler also calculates the Recipe suggestion for the system.

2 controllers were developed:

- FoodController and RecipeController.
    - Calls RecipeHandler with an array of strings

1 basic command line interface was developed:

- Command Input
    - Ensures that only valid inputs were processed.

1 Data Parser

- DataParser
    - Inputs data into 2 csv files
    - Reads data from 2 csv files

What each group member has been working on and plans to work on next

Each of our team members equally contributed to Phase 0 of the Project. More specifically, here is a breakdown of what each group member contributed to the project:

The whole team: The whole team contributed to the making of the CRC cards and the skeleton code.

Current breakdown:

Team member	**Tasks Completed**	**Future Tasks**
Kai	Introduced the organizational platform JIRA to organize our tasks Worked with Nathan to edit the CRC cards. Developed the RecipeController	Continue implementing unit tests for RecipeController Continue implementing code and following design principles
Mark	Created the walkthrough in collaboration with Yvonne Developed the Use Case Recipe Handler	Continue implementing unit tests for RecipeHandler Continue implementing code and following design principles
Nathan	Developed the code framework for the project Developed skeleton code for CommandInput and Dataparser	Continue implementing unit tests for CommandInput and Dataparser Continue implementing code and following design principles
Maggie	Developed the Use Case FoodHandler Developed the code framework for the project	Continue implementing unit tests for FoodHandler, add a method to alert user of expired food Continue implementing code and following design principles
Yvonne	Developed all the entity classes (Food, PerishableFood, NonPerishableFood, Recipe) Developed unit tests for all the entity classes Created the walkthrough and wrote the specification and progress report with Carmel	Continue implementing unit tests for Food, PerishableFood, NonPerishableFood, and Recipe Continue implementing code and following design principles

Carmel	Developed the controller FoodController Wrote the specification and progress report with Yvonne	Continue implementing unit tests for FoodController Continue implementing code and following design principles

What has worked well so far with your design as you have started implementing the code?
So far, the design of our code has allowed us to split tasks for a complex system more efficiently when working as a team. Clean architecture also allowed us to better communicate our program ideas to one another, and organize the entire program so that it follows all of the SOLID principles. In the program itself, it heavily reduces the confusion

We have gone over the Clean Architecture structure of our code a plethora of times to ensure that our code will work well and has good design. (Figure 1)

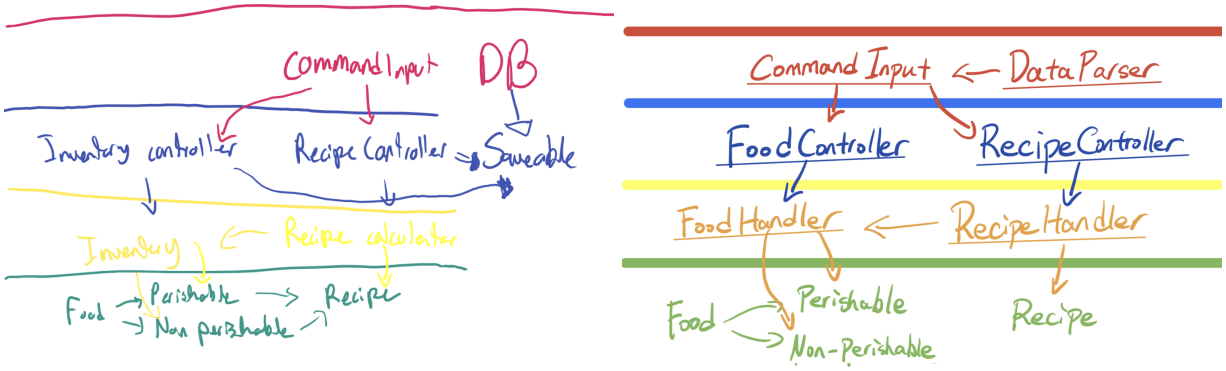

Figure 1. Different drafts of the Clean Architecture of our code **(A)** Our first runthrough of how our code should be separated according to Clean Architecture **(B)** Our final Clean Architecture plan for our code for Phase 0.

Open Questions
How can we incorporate recipe datasets into our program?
How can we update the food when it expires so that we don't constantly check for no reason?