PROJECT / RELEASE

Project Design Document

Team B

Amanda Bui, axb1006@rit.edu

Daniel Chung, dec8768@rit.edu
Ikemefuna Chukwunyerenwa, ijc3093@rit.edu
Francisco Paliouras, fxp6816@rit.edu
Matthew Russell, mmr8027@rit.edu

Project Summary

Our mission is to develop an application and make adjustments to improve the output of the project. We, as a team, are creating an application that allows users to register their own recipes, daily consumption of food and track their progress of their current diet. The users should have control to add their own basic food and recipes. Not only just adding their food and recipes, they will also have the access to view or keep track of their consumption throughout the day.

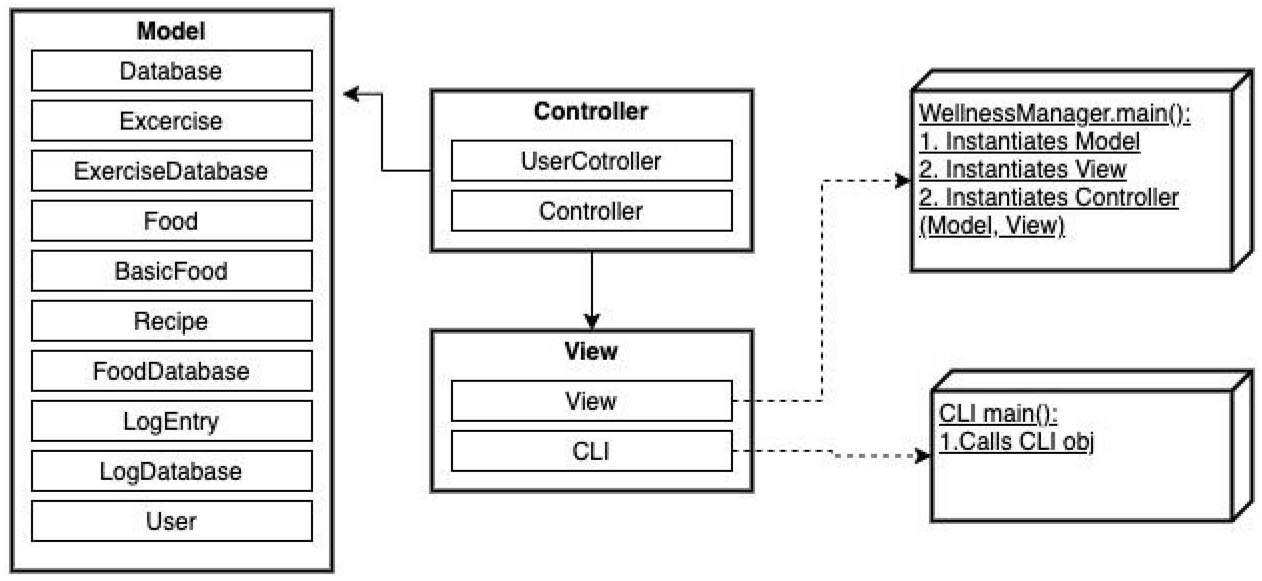
Design Overview

Team B has been working on developing an application of the project using architecture pattern and composite pattern. We have currently used the Model/View/Controller as our architecture pattern. The goal is to demonstrate the user's perspective to the audience. Basically, the user wants to keep track of its calories, proteins, etc. As the user keeps on tracking its health, the controller manages the ingredients of both the basic foods and recipes, and has access to the log entries as well.

Since we have discussed after the presentation,we were able to build a clear vision by using the GUI to display the output for the user to control and log the entries or readings of the user's data of consumption. Members of Team B were able to divide the tasks on who's responsible for each part. Once in a while, we often have meetings to talk about the improvements made, such as GUI output, new databases, and etc.. However, this version mainly focuses on the improvements and modification for a better user experience than the previous work. Team B is taking each step to work on the improvements of the project and making sure that it functions properly as a way to meet the expectations.

Team B has focused on using the two main patterns, such as MVC and composite pattern in order to create and reach the limits of what's required in our product. MVC is an architecture pattern that separates our applications' concerns of a data model, presentations designs, and control information. This pattern helps us organize where the classes belong to, and uses the idea to demonstrate the blueprint of the product. Along with the MVC pattern, we also used the composite pattern, which composes objects together. The composite pattern is what holds the control of the information, for example, food, and recipes. We often use the MVC pattern in our application to ensure that there is a balance between cohesion/coupling. We want to bring perfection and accurate visuals into this product to match the same vision as the clients explanation.

Subsystem Structure



Subsystems

Subsystem: Model

Class: Database	
Responsibilities	Basic interface object that allows for later creation specific data handlers for Food, Exercise, and logs.
Collaborators (uses)	- java.io.IOException : Handle input/output errors

Class: Exercise	
Responsibilities	Stores the information for an Exercise instance, Name and calories burnt.
Collaborators (uses)	

Class: ExerciseDatabase	
Responsibilities	Handles reading and saving the data from and to the files where the information is being stored. It only handles information regarding the exercise data.
Collaborators (uses)	- Implements Database - Uses Exercise data structure - java.io.*: Handles the input/output exceptions. Also responsible for writing to file.

Class: Food	
Responsibilities	Basic interface object that holds basic functionality of a food object. Accesors and mutators as well as calculation functionality and displaying functionality.
Collaborators (uses)	

Class: BasicFood	
Responsibilities	Stores the information of a BasicFood object: type, name, calories, fats, carbohydrates, protein. It also implements the accessor methods and the mutator methods for the BasicFood object data.
Collaborators (uses)	- Implements Food

Class: Recipe

Responsibilities	Stores the information of a Recipe object: type, name calories, fats, carbohydrates, proteins, and a hashmap with a list of ingredients of type Food mapped to Doubles to represent the quantity.
Collaborators (uses)	- implements Food, - Can contain BasicFood

Class: FoodDatabase	
Responsibilities	The main responsibility of the foodDatabase is to read and save in the data from the food csv files that stores the basic foods and the recipes that use those ingredients.
Collaborators (uses)	- implements Database - Uses these data structures: Food, BasicFood, Recipe - java.io.IOException : Handle input/output errors - java.util: is used to store the information in an orderly fashion

Class: LogEntry	
Responsibilities	Is responsible for calculating and storing the log information for the application. By using the information in the FoodDatabase and the ExcersiceDatabase LogEntry generates a LogEntry object that stores the information of date, foodsConsumed, ExercisesMade.
Collaborators (uses)	- FoodDatabase: uses information to output a LogEntry Obj - ExerciseDatabase: uses information to output a LogEntry Obj

Class: LogDatabase	
Responsibilities	Read and write to the log.csv file. It stores entries of LogEntry objects in a hashmap mapping String[DATE] to a LogEntry Object.
Collaborators (uses)	- Implements Database - Uses LogEntry data structure

Class: User	
Responsibilities	Stores a User object that contains the user information: first name, last name, weight.
Collaborators (uses)	

Subsystem: View

Class: CLI	
Responsibilities	Managing User input through the console, and sending commands respectively to the controller.
Collaborators (uses)	- UserController[Subsystem: Controller] - java.io.IOException : Handle input/output errors - java.util.Scanner : Handles User input in terminal

Class: View	
Responsibilities	Rendering the user interface and managing the user input and using the controller to respond with the correct functionality.
Collaborators (uses)	- Javax.swing : render UI - Controller[Subsystem: Controller]

Class: WellnessManager	
Responsibilities	Render the login window, checking if the user exists and passing on the used user to the GUI. Registering new users through the new user function of the GUI.
Collaborators (uses)	- Controller[Subsystem: Controller] - View

Subsystem: Controller

Class: UserController	
Responsibilities	Handles passing the data from the view to the model and vice versas. Adding, removing and updating methods are handled in this class for the logs, foods, and exercises exclusively for the CLI part of the project.
Collaborators (uses)	- FoodDatabase[Model] - LogDatabase[Model] - ExerciseDatabase[Model]

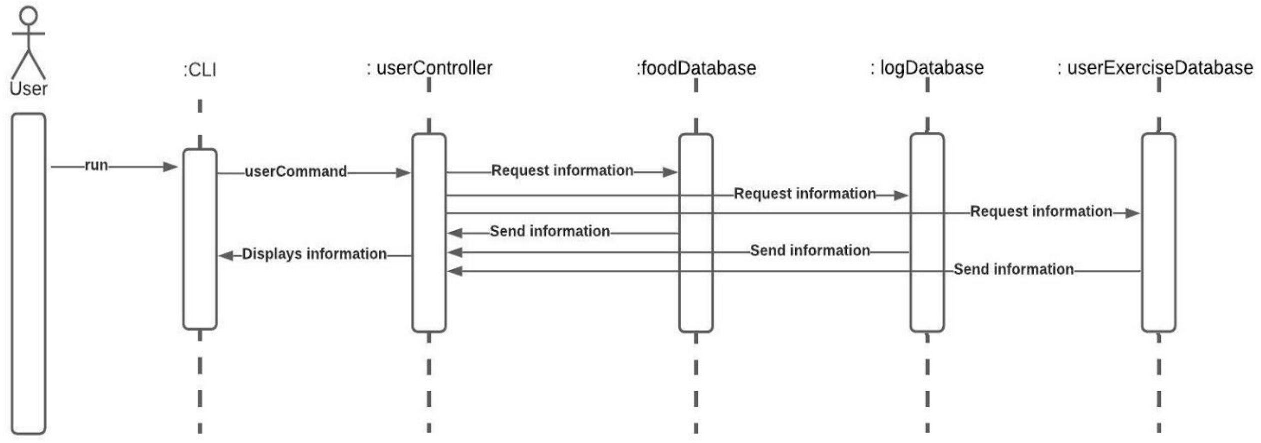
Class: Controller

Responsibilities	Handles passing the data from the view to the model and vice versas. Adding, removing and updating methods are handled in this class for the logs, foods, and exercises exclusively for the GUI part of the project.
Collaborators (uses)	- FoodDatabase[Model] - LogDatabase[Model] - ExerciseDatabase[Model]

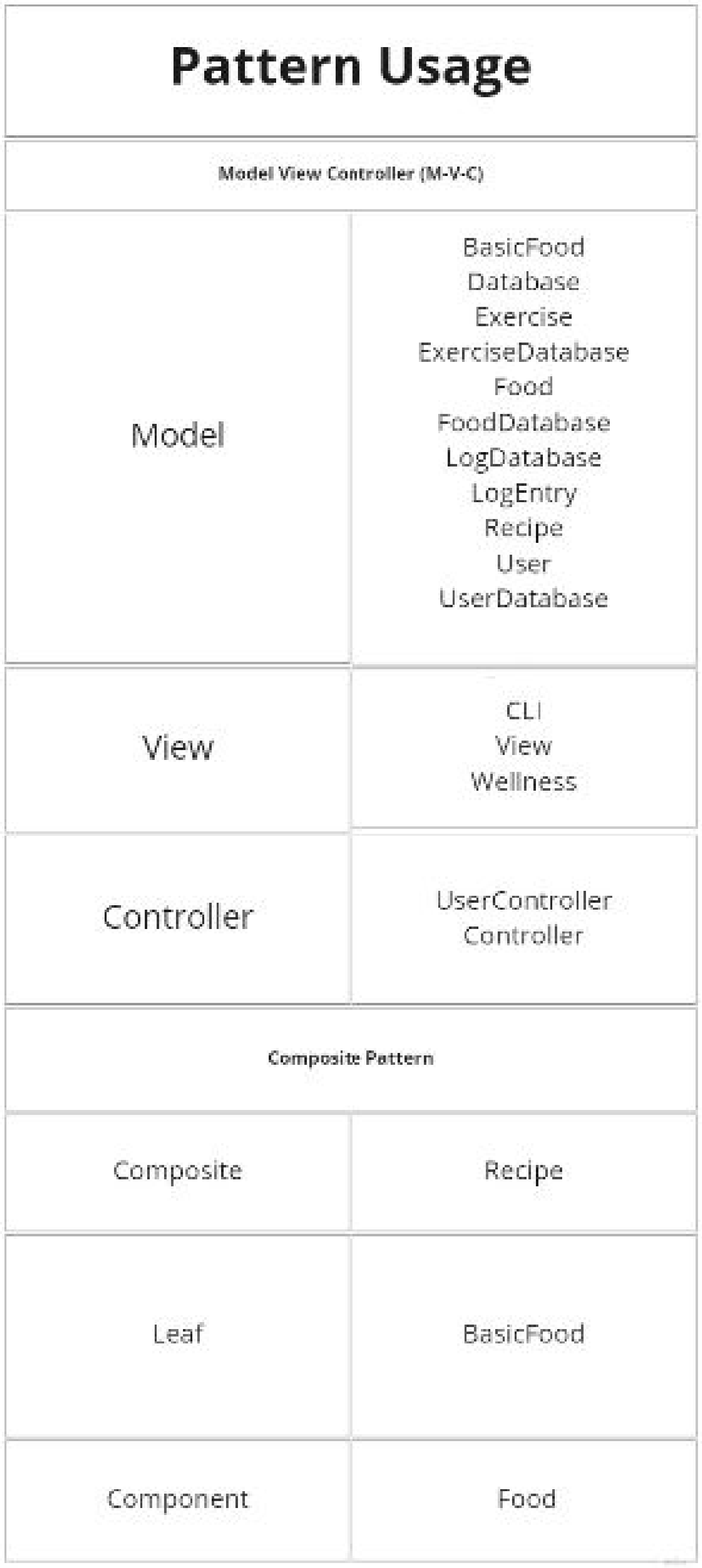
Sequence Diagrams

We will be creating an application that allows a user to register their own recipes, daily consumption of food and track their progress of their current diet. The user will be able to add their own basic foods and recipes by giving the program the required information. In addition to this, the user will also be allowed to register his daily food consumption and the applications will automatically keep track of the amount of calories, carbohydrates, fat and proteins that the user has consumed throughout the day.

The user runs the command line interface, which uses the method to call the userCommand from the userController. The userController makes the request to several databases, foodDatabases, log Databases, and userExereciseDatabase, depending on the user's selection of where they want to go. Once the user sends the request, the selected database will then return or send the information to the userController. The User Controller will display the information requested in the GUI.



Pattern Usage



Pattern Usage	
Model View Controller (M-V-C)	
Model	BasicFood Database Exercise ExerciseDatabase Food FoodDatabase LogDatabase LogEntry Recipe User UserDatabase
View	CLI View Wellness
Controller	UserController Controller
Composite Pattern	
Composite	Recipe
Leaf	BasicFood
Component	Food