



Progress Report




Brief Summary

Specifications, CRC Cards, scenario walk-through, and skeleton program

Specifications

In the Specification, we wrote:

- an explanation of what our program is doing and how the user will be interacting with the console
 - a clear instruction of what the user should expect when they run the program, such as what they will need to enter, and what the program will do with the User's inputs
 - what each functionality of our program does and outlines what input those functionalities need the user's to enter, as well as what they will output to the user
 - the databases we will be working with in this project, what we need them for, and the sources of these databases.
- 

CRC Model

Entity Classes

- **User:** a class containing all data relevant to the user
- **Food:** A class that represents a type of food object
- **Exercise:** A class that represents a workout move
- **Disease:** A class that represent a disease

Use Cases:

- **UserAnalyzer:** a parent class that manipulates different information of the user based on the functionality selected by the user.
 - Children Classes: BMIAnalyzer, EERAnalyzer, DiseaseAnalyzer
- **MealPlanGenerator:** generates a specific meal plan for the user according to the user's food preference and the user's energy requirement per day (measured in Calories)
- **CreateUser:** create a User object using the provided arguments passed in from RunCommand

CRC Model

Interface Adapter:

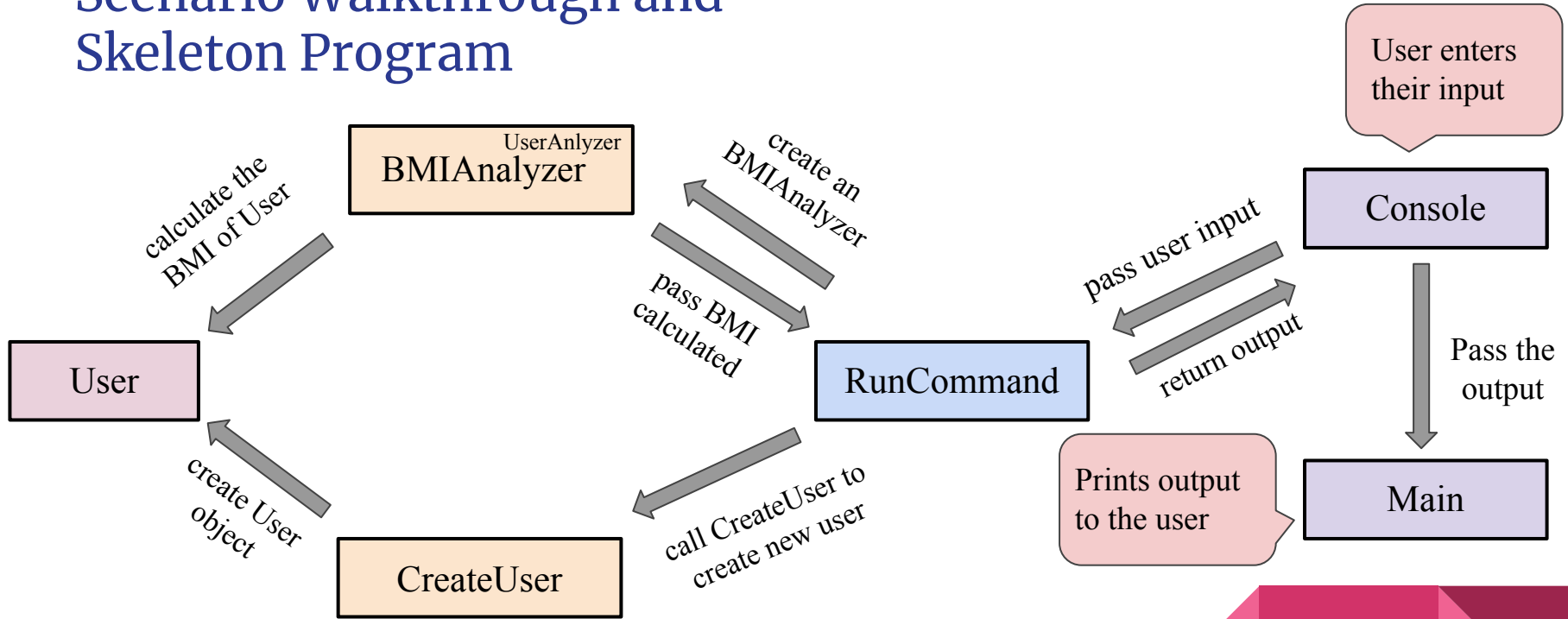
- **RunCommand:** The controller class
 - Takes in user's input and pass it to corresponding use cases
 - Outputs the returned String from the use cases to the Console

Framework/Drivers:

- Basic Command Line interface:
 - **Console:** Outputs String prompts to the user and takes in user's input
- **API:** reads and processes data from our databases
 - FoodAPI
 - ExerciseAPI
 - DiseasesAPI



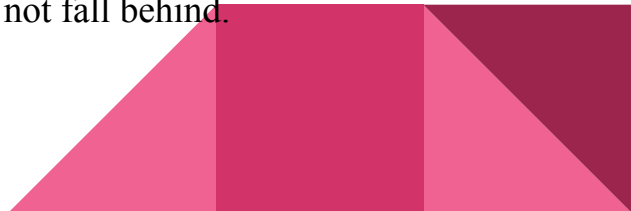
Scenario Walkthrough and Skeleton Program



Reflection

What we did well

- Collaboration
 - Divided tasks evenly between group members.
 - Good communication between members, with active responses from each group member.
 - Helped each other when members are encountering problems.
 - Adhered to the five principles of clean architecture, and our code shows a clear dependency hierarchy between the Entities, Use Cases, Interface Adapters, and Frameworks & Drivers. There are no signs of classes in lower layers trying to access a class in higher layers, or a class in higher layers
- Work progress
 - Created detailed outlines as best as possible.
 - Set up specific goals to accomplish day by day to ensure that we do not fall behind.



Reflection

What still needs Improvement?

- Some issues that still need to be resolved are within our code. Some of them may be violating one of the SOLID principles, which we are still working towards to improve.
- Making sure we have more collaborative time on code, rather than individually working on each component. We could also possibly do some explanations to each other, ensuring that everyone knows the basic concept of how each class is implemented.



Questions we are struggling with...

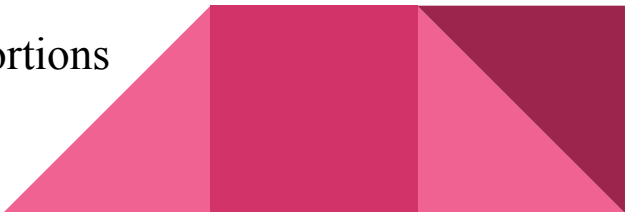
1. We currently have no Interfaces in our program, would this be a problem?
2. How are we going to implement the Presenter class? (See Progress Report for details...)
3. Some of our use cases calculate new data and put them into the User object. This means that some of the instance attributes inside the User object (such as the BMI Hashmap) would depend on the use cases. Would this be a violation to the clean architecture? If so, how can we change this? Can we keep this implementation if it's well justified?



What has each group member been working on?

Everyone worked on specifications/progress report collaboratively.

Classes, Written portion that individual members worked on:

- **David:** Console, RunCommand, First Build of a successful scenario walkthrough run
 - **Jenny:** API, Progress report presentation
 - **Winnie:** User, Disease, TestBMIAnalyzer
 - **Enid:** Exercise, Food, CreateUser, Console, Main, RunCommand
 - **Paul:** RunCommand
 - **Naomi:** BMIAnalyzer, README
 - **Yifan:** UserAnalyzer, Main, minor contributions in written portions
- 

What do each of us plan to work on next?

Our current plan is to divide up into two groups:

1. One group will be working on data processing and APIs, so the system can read in the three databases we have right now and create the corresponding entities.
2. The second group will be working on interfaces and the Presenter class, to make sure the structure of our code doesn't violate Clean Architecture.

Once this is done, we will split into four small groups where each group takes on an individual functionality of the program: Meal plan generator, exercise generator, Disease predictor, and EER calculator.

