Nutrition App CRC Cards

`User` Class (Entity)	
Responsibilities	Collaborators
• Represents a user interacting with the app. • Contains key attributes, including: – `username` – `passwordHash` – `weight` – `height` – `activityLevel` – `age` – `goals`	• `Stat` • `FoodItem` References `User` to keep track of their periodic intake. A `User` instance can be used to access all instances of `Stat` associated with it, as well as all `FoodItem` instances associated with them.

`Stat` Class (Use Case)	
Responsibilities	Collaborators
Stores statistics for the user. Has attributes: • `caloricIntake`: the total calories for the day. • `user`: The `User` instance the stat is linked to. • `startDate`: when this stat starts tracking. • `foodItems`: A list of `FoodItem` • `endDate`: when this stat stops tracking.	• `FoodItem` References a list of `FoodItem` to keep track of what is being consumed throughout that time period. This will also contribute to the `Stat` that are provided to track progression.

`FoodItem` Class (Entity)	
Responsibilities	Collaborators
The `FoodItem` class represents a consumed food item of an associated user. The `FoodItem`. Attributes: • `name`: Name of food item • `type`: Food type (fruit, vegetable, meat, drink) • `amount` (in grams)	• `Nutrient` Keeps track of the name, type, amount and associated nutrients (an instance of `Nutrient`) from each food item.

`FoodItems` Class (Entity)	
Responsibilities	Collaborators
The `FoodItems` class represents the known food items in our system. Attributes: • `FoodItem Instances`: Contains instances of the known food items in our program.	• `Food Item` Keeps track and contains the `Food Item`'s in our program.

`Nutrient` Class (Use Case)	
Responsibilities	Collaborators
The `Nutrient` class makes proper comparisons in terms of nutritional values in addition to keeping track of the totality of nutrient intake. Attributes: • `carbohydrates` (amount of carbohydrates) • `proteins` (amount of proteins) • `fats` (amount of fats)	• `FoodItem` Incorporates the different macronutrients that each `FoodItem` instance contains.

`Goal` Class (Use Case)	
Responsibilities	Collaborators
The user will set a goal of their body mass with specified duration/deadline. The class has `User`'s goal body mass and date they want to reach that goal. • `userWeight` • `goalWeight` • `dueDate` • `achieved`	• `User` Uses the user's goal body mass and date they want to achieve their goal, to keep track of progress.

`LoginSystem` Class (Use Case)	
Responsibilities	Collaborators
Stores the details of every user that is using the app and enables different users to use the app on the same device. It helps performing tasks like registering a user, logging in a user, checking if a user exists and checking a user's password. Attributes: • `users`: list of `user` objects Methods: • `registerUser`: registers a user into the system • `logInUser` allows a user to log into their account • `userExists`: checks if the user with given username exists in our system • `getUser`: retrieves the user's information from our system	• `User` Uses the user's username and password to perform functions helping the user log into their account.

`MealGenerator` Class (Use Case)	
Responsibilities	Collaborators
Constructors with no input or a meal type (Breakfast, Lunch, Dinner, Snack), Calories Goal, and Dietary Information (vegan, vegetarian, etc). Attributes: • `mealType` (either by input or time of day) • `caloriesAim` (either by calories left today or input) • `dietaryInformation` (input or NONE) Methods: `requestMeal`: takes in all 3 attributes, returns a string form of a meal with a little additional information	• `Food Item` • `Food Items` • `User` Uses the `user`'s information to generate customized meals according to their weight goals. These meals are formed using `Food Item`'s from the `Food Items` class or other sources.

`Progress` Class (Use Case)	
Responsibilities	Collaborators
Stores and update user's current weight, BMI and nutritional levels. Allow users to compare their current weight with their goal. Attributes: • `username`: Name of the User • `currentWeight`: current weight of the User • `currentHeight`: current height of the user • `currentBMI`: current BMI of the user • `nutrientLevel`: stores the level of various nutrients the user has based on their food intake. • `goal`:goal set by the user Methods: • `changeParameters`: updates the status of the user	• `Goal`

App Class (Controller)	
Responsibilities	Collaborators
controls the program itself. Designed to be extended by a User Interface class. Attributes: • User: current user that is logged in • system: database storing other accounts Methods: • login: Login a user into the app • getUserUsername: get the username of the current user • register: register a new user to the app • getHash: get the SHA256 hash of a String	• User Interface class

Main Class	
Responsibilities	Collaborators
Runs the program. Creates the command line environment, prompts users for information and allows users to interact and use the programs functionality. It launches the consoleApp UI and runs the program.	• consoleApp Uses the consoleApp UI to launch the application and interact with the user.

Utils Class	
Responsibilities	Collaborators
Implements methods that are used in order to get input from the console. This class is used in the Login System and in other places where user interaction and input is required. Methods: • getString: Gets an inputted string from the user. • getInt: Gets an integer input from the user. • getDouble: Gets an inputted double from the user.	• LoginSystem • ConsoleApp • User The methods of this class are used in LoginSystem and ConsoleApp to take in input from the user. This includes when a user is trying to sign in with their username and password, or when they're inputting important information about themselves (eg. height).

`ConsoleApp` Class (User Interface)	
Responsibilities	Collaborators
Runs the program through the console by asking for user input with Scanner objects. Methods: • `signInUser`: Welcome a new /existing user to the app • `showDashboard`: Perform tasks now that the user is logged in • `run`	• `App`