APPLICATION SKELETON

Name of front-end class	UI
Front-end class: instance variables	○ advice_format ○ repetitive_comment ○ educational_information ○ first_time_use ○ table_format
Front-end class: information the constructor would take as arguments	○ all methods would take user input, with that being sent backend as arguments ○ the pop_up_advice would take information from either goals_assessor, symptom_assessor, goal_tracker, or recommended_nutrition
Front-end class: Names of methods used	○ pop_up_advice ○ gerd_information ○ diet_symptoms_tracker ○ nutritional_planner ○ search_function ○ personal_info_collector ○ excersise_tracker
Name of back-end class	PresetInfo
Back-end class: instance variables	○ food ○ drinks ○ symptoms

	○ symptom_assistance ○ nutrition_advice ○ fitness_advice
Back-end class: information the constructor would take as arguments	'if' arguments dependent on triggers coming from VariableInfo would trigger sections of PresetInfo to display
Back-end class: names of methods used	
Name of additional back-end component class	SymptomVariables
Additional back-end component class: instance variables	○ Call object from PresetInfo ○ eaten_food ○ eaten_drinks ○ number_of_symptoms ○ severity_of_symptoms ○ possibly_bad ○ definitely_bad ○ good_foods ○ diagnosis_date
Additional back-end component class: information the constructor would take as arguments	
Additional back-end component class: names of methods used	
Name of additional back-end component class	DietFitnessVariables

Additional back-end component class: instance variables	○ goal_weight ○ current_weight ○ age ○ difficulty_scale ○ exercise _amount ○ peds (performance enhancing drugs, registers as true or false)
Additional back-end component class: information the constructor would take as arguments	
Additional back-end component class: names of methods used	
Name of additional back-end component class	Main
Additional back-end component class: instance variables	○ Call object from PresetInfo ○ Call object from DietFitnessVariables
Additional back-end component class: information the constructor would take as arguments	○ The collectors would take user input from the front end ○ The assessors would take info gathered from PresetInfo and DietFitnessVariables ○ Excessive_symptoms would take number_of_symptoms and severity_of_symptoms as arguments
Additional back-end component class: names of methods used	○ goals_collector ○ goals_assessor ○ symptom_collector

	○ symptom_assessor ○ bad_food_sorter ○ bad_drinks_sorter ○ good_food_sorter ○ good_drinks_sorter ○ goal_tracker ○ recommended_nutrition ○ excessive_symptoms

PROGRAM EXPLANATION

Each method explained:

- pop_up_advice:
    - This method will be one used repetitively throughout the front end of the program, taking information from either goals_assessor, symptom_assessor, goal_tracker, or recommended_nutrition as arguments that will help to select which piece of advice the user will be given.
    - It will then display the advice using the advice_format for presentation.
    - This method will require an if statement to be called, which will either be placed at the end of each time the program obtains added information from the user. To avoid code repetition, the if statement will be made into a method of its own and called rather than typed out each time.

- gerd_information:
    - This front-end function will interact with the user, displaying information about GERD, and taking inputs such as search options when the user decides to do so.

- diet_symptoms_tracker:
    - The diet_symptoms_tracker method will be the base for the diet and symptoms tracker tab, taking information from the user each week, sending notifications accordingly.
    - This method will utilise backend functions to store and assess information after each input and using pop_up_advice afterwards when deemed necessary.

- nutritional_planner:
    - This method is the user interface for the GERD-Based Nutritional Planner tab, displaying tables created for both individual meals and daily totals for the user to view via table_format, as well as helping the user to assess their ideal requirements via the back-end goals_collector, goals_assessor, exersise_tracker, goal_tracker, and recommended_nutrition.

- search_function:
    - The search function acts as an information collector, locating data from the educational_information list and displaying it back to the user.

- Personal_info_collector:
    - When both the Diet and Symptoms Tracker and the GERD-Based Nutritional Planner tabs are opened for the first time, the Personal_info_collector will run via their corresponding UI methods and will gather important data from the user such as diagnosis_date (for the SymptomVariables class), and necessary information to ender the DietFitnessVariables via goals_collector (the backend method). This can otherwise only be accessed via the profile settings.

- goals_collector:
    - The goals collector is the first line of input at recognition of goals (taken from the front-end function “personal_info_collector”). It is to be used for the transfer of goals-based data into the DietFitnessVariables class.

- goals_assessor:
    - The goals assessor works directly with the DietFitnessVariables class and is used to process the information stored within. The type of processing it will do is assessing the information inside the DietFitnessVariables class and using this to ascertain which information to pull from the fitness_advice list.

- symptom_collector:
    - The symptom_collector is the method that will be used to transfer symptoms ascertained through the front-end diet_symptoms_tracker, splitting it into the right categories, and adding it to the SymptomVariables class.

- symptom_assessor:
    - The symptom_assessor is what will be used to assess the information within the SymptomVariables class, determining whether or not it will need to display advice through symptom_assistance, and then being displayed to the user via pop_up_advice (which will be the formatter).

- bad_food_sorter:
    - As part of the assessment for the diet_symptoms_tracker class and relevant storage/backend classes, the bad_food_sorter will use an integer into dictionaries for each food type to measure the number of times a type of acidic food is consumed whilst the person is experiencing symptoms. Regular

experiences of symptoms when eating this food will add it to the possibly_bad category, and then further the definitely_bad list. The definitely_bad list will be hidden with mutators only allowing for the addition to, not the subtraction from this list.

- bad_drinks_sorter:
    - This will act almost identically to the bad_food_sorter, only for the drinks category.

- good_food_sorter:
    - Acting like the opposite of the bad_food_sorter, this method will take all acidic foods consumed during an extended period of the user stating they have not experienced any symptoms and add those foods to the good_foods list. Much like the definitely_bad list, the good_foods list will be hidden and use mutators to prevent back-end functions from accidentally removing items from this list and avoid system confusion over data analysis.

- good_drinks_sorter:
    - This will act almost identically to the good_food_sorter, only for the drinks category.

- goal_tracker:
    - The goal_tracker will be in charge of the collection of goal related data, working differently to the goals_assessor by means of readying read data and history, comparing what the users' goals are, and what their progress is looking like, ready to be either be displayed back to the user or used for the recommended_nutrition method, instead of the goals_assessor’s purpose, which is to ready advice based upon this type of data.

- recommended_nutrition:
    - Each person has different diet requirements to others, dependent upon their current body, age, exercise, goals etc. Therefore, this recommended_nutrition method will be the method that is programmed to gather all data from the DietFitnessVariables, with help from the goal_tracker, and excersise_tracker. The data collected will be used to calculate the persons recommended nutritional intake per day.

- excessive_symptoms:
    - This method will be triggered when a large quantity or severity of symptoms of symptoms have been detected. The excessive symptoms will first prepare warning message to the user to be displayed via pop_up_advice, and then go on to pull advice from the PresetInfo class.

- excersise_tracker:
    - The excersise_tracker will be the method in charge of a possible addition to this application (or likely a later update), which will be used to gather exercise habits from the user, and store the data as a displayable list (acting like the users personal exercise diary).