|  |  |
| --- | --- |
| **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 * excersize\_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 |

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 place at the end of each time the program obtains added information from the user, or 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, exersize\_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 backed 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 excersize\_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 a warning message to the user to be displayed via pop\_up\_advice, and then go on to pull numerous pieces of advice from the PresetInfo class.

* excersize\_tracker:

- The excersize\_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)

References:

https://rmit.instructure.com/courses/84922/files/20718751/download