

B. Data Types and Operation Signature

(1)Display

The first class, Display, is based on the User Interface. And it is built to show results and messages that the system send to the user.

* IsInfoGotFromFitbit(): Boolean
  + Boolean variable corresponding to the status if the user’s information is retrieved from Fitbit
* IsDataAnalysisGenerated(): Boolean
  + Boolean variable corresponding to the status if the data analysis is generated.
* getActiveMinutes(): Double
  + Double variable corresponding to the user’s active minutes recorded on Fitbit device.
* getSteps(): Double
  + Double variable corresponding to the user’s steps recorded on Fitbit device.
* getCalories(): Double
  + Double variable corresponding to the user’s calories recorded on Fitbit device.
* getDistance(): Double
  + Double variable corresponding to the user’s distance recorded on Fitbit device.
* getSleepInfo(): Double
  + Double variable corresponding to the user’s sleep information recorded on Fitbit device.
* showFitbitData(): Void
  + show the user information retrieved from Fitbit, including steps, active minutes, distance, calories and sleep in a month or in a year.
* ShowHealthIndex():Void
  + show the user the health index generated from the Fitbit data. If in a month, then the health index will change day by day. If in a year, then the health index will based on the average data in one month, which means that the health index will change month by month. And there is a description of the health index to explain how this index represent and influence the user’s health.
* ShowGraphs(): Void
  + show the graphs of the user information retrieved from Fitbit, including steps, active minutes, distance, calories and sleep in a month or in a year. And in order to compare these data to make a clearer view of the user’s health, we can display these graphs in the same page or different pages. And there is a description for each factor to explain how this factor influence the user’s health.
* Showlinecharts(): Void
  + show the line charts of the user information retrieved from Fitbit, including steps, active minutes, distance, calories and sleep in a month or in a year. And in order to compare these data to make a clearer view of the user’s health, we can display these graphs in the same page or different pages. And there is a description for each factor to explain how this factor influence the user’s health. And the line charts can reflect the user’s health explicitly. l
* showAdvice(): Void
  + show the advice to the user based on the data retrieved from Fitbit. Basically, the advice is formed as a recommended activity situation, which tells the user how to improve the user’s health by improving some of the factors.

(2) User Input

This class is built to get the user’s input operation such as clicking button and inserting text. And the purpose of this class is to help implement the interaction between the website and the user.

* getCodeID(): String
  + String variable corresponding to the user’s input in the Code ID blank. The return value will be the CodeID to retrieve data from Fitbit for a certain user.
* getOperation(): Void
  + This method will be called when the user did any operation on screen, such as clicking the button. And it will tell the corresponding function.
* getState():String
  + String variable corresponding to the user’s inserted state, when the user wants to search data of a certain state.

(3)Controller

This class is in charge of control other classes to work. Many operations require several classes to work together. And the orders are passed through classes by the controller. And the functions are as followed:

* DBConnection(): Void
  + It establishes a connection to the database for either stored region data or Fitbit data.
* FitbitConnection(): Void
  + It establishes a connection to the Fitbit database for a certain user.
* DataRequest(): Void
  + This function serves as a general method to send the queries to the database and receive the response from the database. This function will be called by the Display and the DataProcessor class.

l

* DataUpdate(): Void
  + This function is used for manually updated the content in the database. Once the function is called, the current database will be erased and be ready for receiving the new data.
* DataAnalsis(): Void
  + This function is built to use the FA algorithm to analyze the data and generate the Health Index for the user. And it will be called when the data in retrieved from Fitbit and written into the database.
* GetAdvice(): Void
  + This function is built to let the user know how to get a optimal health condition. And the advice is generated by calculating the health index and choose the highest health index day as the optimal day, and user that day’s information as the optimal advice.

(4)DataBase

This class is built to store the data. And it defines some variables needed to be stored.

* ActiveMinutes: Double
  + Double variable corresponding to the user’s active minutes recorded on Fitbit device.
* Steps(): Double
  + Double variable corresponding to the user’s steps recorded on Fitbit device.
* Calories(): Double
  + Double variable corresponding to the user’s calories recorded on Fitbit device.
* Distance(): Double
  + Double variable corresponding to the user’s distance recorded on Fitbit device.
* SleepInfo(): Double
  + Double variable corresponding to the user’s sleep recorded on Fitbit device.
* HealthIndex(): Double
  + Double variable corresponding to the user’s health index generated from the user’s data.
* DistrictData(): String
  + Statistics of all the states in America about health.

* StoreDistrictData(): Void
  + This function would be called internally within any DataUpdate function and would update an associated value (e.g. districtData) within in the DataBase for that update.
* StoreHealthIndex(): Void
  + This function would be called internally within any calculate-HealthIndex function and would add an associated value (e.g. HealthIndex) within in the DB for that update.

(5)DataProcessor

This class is built to analyze the data retrieved from Fitbit and store the data into database. The functions are as followed:

* GetInfoFromFitbit(): Void
  + After Controller finished the Fitbit Connection, this method will be called to retrieve data from Fitbit.
* GenerateHealthIndex(): Void
  + This function will be called after the user chose a certain function, and need to show the health index. This function will generate the health index based on the user’s data.
* WriteDataRequest():Void
  + This function is built to request the database to store the Health Index of the user. And it will be called after the index are created for the user.

**C. Traceability Matrix:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Display | UserInput | Controller | DataBase | DataAnalysis |
| Interface | X |  |  |  |  |
| Controller |  | X | X |  |  |
| Checker |  | X | X | X |  |
| DBConnector |  |  | X |  |  |
| InfoStorage |  |  |  | X |  |
| Communicator |  |  | X | X |  |
| TextReader |  | X |  |  |  |
| HealthAnalysis |  |  | X | X | X |