

Software Engineering

Personal Health Monitor

Group 4

Chenfan Xiao, Chengyao Wen, Jianing Xu, Xinyu Li, Yuwei Jiang

**Content**

1. [**Customer Statement of Requirements**](#_gjdgxs) **4**
2. [Problem Statement](#_30j0zll) 4

[b. System Statement](#_1fob9te) 4

c. [Glossary of Terms](#_3znysh7) 6

**2.** [**System Requirements**](#_2et92p0) **7**

[a. Functional Requirements](#_tyjcwt) 7

b. [Non-functional Requirements](#_3dy6vkm) 9

c. [User Stories](#_1t3h5sf) 9

**3.** [**Functional Requirements Specification**](#_4d34og8) **11**

1. [Stakeholders](#_2s8eyo1) 11

b. [Actors and Goals](#_17dp8vu) 11

[c. Use cases](#_3rdcrjn) 11

d.  [System Sequence Diagram](#_26in1rg) [1](#_26in1rg)6

[**4.**](#_lnxbz9)**Class Diagram and Interface Specification 22**

a.Class Diagram 22

[b.](#_44sinio) Data Types and Operation Signature 23

c.Traceability Matrix 28

d.User Interface 29

e.User Effort Estimation 33

[**5.**](#_z337ya) **Functional Unit Tests 34**

a.Class Tests 34

b.Functional Unit Tests 35

**6.System Architecture and design 37**

a.Architectural Styles 37

[b.](#_44sinio)Identifying Subsystems 38

c.Mapping Subsystems to Hardware 39

d.Persistent Data Storage 39

e.Network Protocols 40

f. Global control Flow 41

g. Hardware Requirement 41

**7. Domain Analysis 41**

a.Concept Definitions 42

[b.](#_44sinio)Association Definitions 43

c.Attribute Definition 44

d.Traceability Matrix 45

**8. Algorithms and Data Structures 46**

a.Algorithms [15](#_35nkun2)

[b.](#_44sinio)Data Structure

**9. Individual Members’ Contribution**

**10. References**

## 

1.Customer Statement of Requirements

1. Problem Statement

Personal health monitoring can be useful for individuals: providing them with professional and highly-customized food and sports suggestions, motivating them to keep active and have a good habit. Also it gives a lot of potential important information by analyzing the health data collected. It can be used by public health specialists to provide more effective policies in a big way and by manufactures to make products that could really help.

From the customers’ views, the system can help them track and store their activity data and recommend activities or diet for them. They can also share their data with friends once they log in. They will also be able to read analysis and find the most important attributes to their personal health.

As a customer, I do care about if my daily data, including activity data and diet data, are stored to the server correctly. I also want to see a plot showing the change of my personal data. What else is I may want to know what kind of diet plan should I have according to my activity data and personal goals. Plus, I will care about how much activity my friends do.

As an analyst, I want to know if certain variable has anything to do with another one. I also want to know if users from different regions or have different activity data will have any change in their general information in the future. I do not and can not know their sensitive information individually.

Based on the analysis above, we will have these problems as follows:

Problem 1: Collecting data from wearable devices or health applications with small time interval

Problem 2: Users information or data should be kept safe

Problem 3: Users will be given diet / activity recommends according to their personal data

Problem 4: Users will interact with their friends

Problem 5: Users will request visualized plot for their own data

Problem 6: Analyzer will apply certain algorithms on the data collected

### 

### b. System Statement

In our project, we will design a system to keep track of users’ health data and provide them with valuable suggestions to achieve their goal. In the background we will design a powerful analyzing system to give feedback about any difference so that several main factors resulting in the difference can be found. Typical suggestions for users including food diet based on their active status. One of the information by analyzing them is whether or what kind of wearable devices can really help. Generally speaking we are going to design a system which will be useful for both users and the ones not using it. Figure 1 shows the overview of our designed system.

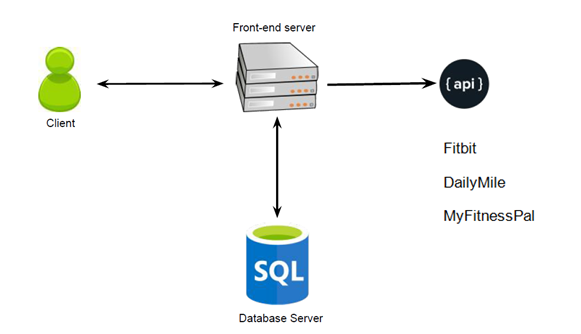


Figure 1 System Architecture

The system will collect the data from wearable health devices - fitbit - using the API provided by the manufacture. Also it will be collecting data from integrated health interface like Apple Health. The data will be uploaded or synced to the server and shared to the client’s friends if permitted. The clients will get visualized graphs of their progress and target daily, weekly and monthly.

We are going to imply APIs from the internet as the specialist to give recommendation to the users. We are still considering using single or multiple API to generate recommendations, because sometimes, there is no definite solution to the healthy diet, not every API would get identical result. So what we are trying to do is to add a function in our system: the system would keep a value to represent the weight of each API we utilized, and the users are presented with different recommendations generated by different API. As a result, certain API would get a increased weight when chosen by the users, and the most weighted API would be labeled as ‘Preferred’ which would make it easier for the user to decide among various options.

The fitbit API will mainly provide the following data: Activity & Exercise, Body & Weight, Friends, Heart Rate and Sleep information. It can be used to track users personal activities and body information. Data will be stored into our databased for further analysis. For recommendation, we will use the API from DailyMile, which will provide us the following functions: Track running route and GPS information and Social functions. With the API from MyFitnessPal, we will be able to access over 20k kind of foods’ information, which can be used for our customized food recommendation system. When the APIs have overlaps, we will use the system above to decide which one is most recommended to clients.

It will be possible for our users to share pictures with others who are not their friends in real life and can share their working out data like steps, running time or active hours with their real friends. Because according to our research, it works better if strangers share pix of themselves while acquaintances share data with each other. This will contribute them to use our system more often and keep track of their health information and be of great help for them to achieve their goal. Also it can provide us more accurate data for foods and sports suggestions as well as background health information search for specialists.

We intend to use PHP and MySQL for our website and we may also develop an Android app to collect users’ health data more efficiently.

### Glossary of Terms

Activity: what kind of activities we can monitor, including but not limited to, running, walking, hiking, swimming.

Dailymile: Track running route and GPS information and Social functions.

Fitbit :provide the following data: Activity & Exercise, Body & Weight, Friends, Heart Rate and Sleep information.

MyFitnessPal: be able to access over 20k kind of foods’ information, which can be used for our customized food recommendation system.

Diet: what kind of diet customer can get. For example low-calories food for customer expecting losing weight or high-protein food for those wanting to gain muscle.

Customer/Clients: synonym here, referring to personnel who upload their activity data via wearable devices or applications from their phones.

Analyst/Analyzer or Specialist: referring to those who have access to anonymous data collected by system and run filters or clusters to make some conclusion from it.

## System Requirements

### a. Functional Requirements

|  |  |  |
| --- | --- | --- |
| Identifier | Priority | Description |
| Req-1 | 3 | The system should attain raw data from the Internet and save it in a  local/online database. |
| Req-2 | 5 | The system should process raw data from the local/online database  and obtain the required analysis. |
| Req-3 | 2 | The system should allow users to register and provide personalized services. |
| Req-4 | 4 | The system should show the recommendation to the user about the healthy diet. |
| Req-5 | 2 | The system should obtain information from other social service platform. |
| Req-6 | 1 | The system should allow analyze give suggestions about how helpful it is of certain behavior which is commonly believed as helpful. |
| Req-7 | 3 | The system should allow analyst provide analysis of health conditions of users from different areas. |
| Req-8 | 2 | The system should show the rank of a certain user about his health among his or her friends as well as among the entire users. |
| Req-9 | 2 | The system should enable the user to customize his or her diet plan. |
| Req-10 | 1 | The system should provide user’s friend health goal with permission. |
| Req-12 | 2 | The system shall allow users to invite friends to the system by  sending emails. |
| Req-13 | 3 | The system shall allow users to share information of their exercise  activities to social networking websites such as Facebook. |
| Req-14 | 3 | The system should show the rank of user in his or her area(can be state,city or even around the house). |
| Req-15 | 5 | The system should allow analyst provide analysis about the major contribution to health. |
| Req-16 | 4 | The system should provide the graph which satisfy the analyst requirement to the analyst. |
| Req-17 | 4 | The system could classify the data by the index(like region, health index) which analyst has input. |
| Req-18 | 3 | The system should output the table that contains the data that analyst need. |

### Non-functional Requirements

|  |  |
| --- | --- |
| Req-1 | The database should keep all the information rather than user’s device. Direct modification from user should be forbidden |
| Req-2 | The system should be simple and easy to use. The webpage should be easy for the user to get the information they want |
| Req-3 | The system should be able to use when related API is updating |
| Req-4 | The response time should be quick |
| Req-5 | The recover time from failure should not be too long |
| Req-6 | The system should enable multiple users to use without overload |
| Req-7 | Minimum maintenance is required, at least once a week |

### User Stories

|  |  |
| --- | --- |
| Identifier | User Stories |
| ST\_1 | As a User, I can get a look at the average ratings(0-10 pts) and reviews of the software. |
| ST\_2 | As a User, I can get a look at the amount of the registered users. |
| ST\_3 | As a User, I can read the detail instruction of the software. |
| ST\_4 | As a Registered user, I can write reviews and rating. |
| ST\_5 | As a Registered user, I can contact with the Software maintainer. |
| ST\_6 | As a Registered user, I can get a look at the rank among all the Registered users( among Registered users in my State / among my friends), and get my ranking(%). |
| ST\_7 | As a Registered user, I can add friends(or follow other users) and press “like” |
| ST\_8 | As a Registered user, I can get a look at friends’ exercise data (daily/weekly) and get comparison data. |
| ST\_9 | As a Registered user, I can check my healthy condition (healthy or not). |
| ST\_10 | As a Registered user, I can set(change) my health goal. |
| ST\_11 | As a Registered user, I can get a recommended exercise plan (based on current health condition and goal). |
| ST\_12 | As a Registered user, I can get a recommended wearable equipment. |
| ST\_13 | As a Registered user, I can check my daily (weekly average) exercise data. |
| ST\_14 | As a Registered user, I can get a costumed meal plan (based on current health condition, daily exercise data and goal). |
| ST\_15 | AS a Registered user, I can get a look at my daily health data analysis, (get a score based on the time-distribution and duration of exercise) |
| ST\_16 | As a Registered user, I can share information of my exercise activities to social networking websites such as Facebook. |
| ST\_17 | As a Software maintainer, I can get feedback from users. |
| ST\_19 | As a Software maintainer, I can updates database and modify UI. |
| ST\_20 | As an Analyst, I can analyze users’ activity data, general information |
| ST\_21 | As an Analyst, I can filter users’ activity data, general information |
| ST\_22 | As an Analyst, I can plot graph as my demand to analyze if two factors are related |

## Functional Requirements Specification

### Stakeholders

Users, analyzers

### Actors and Goals

Wearable devices, health apps: initiating the system with data collected from users

system maintainers, frontend and backend system: participating the system to generalize the data and keep it functional

### c. Use cases

Casual description

The summary use cases are as follows:

UC-1 : The user get access to the website either from mobile device or PC , and get the health information provided by the fitbit APF. The information are collected by a wearable device on the users.

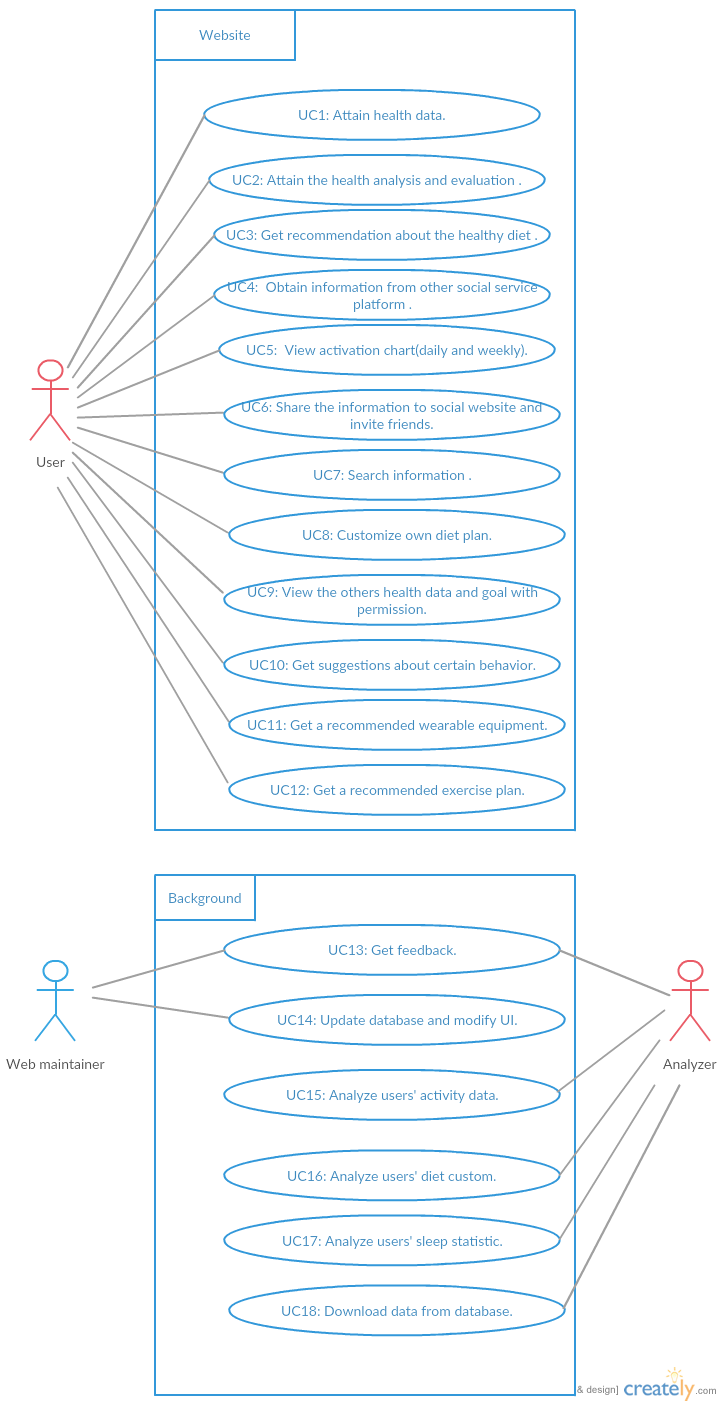
UC-3 : After log in, the user could get a random daily recommendation of food or chose to get a customized recommendation of diet through the external API.

UC-6 : Users could also share the health information through a social platform like facebook. The system provide such a platform for users to see other friends health or daily activities information.

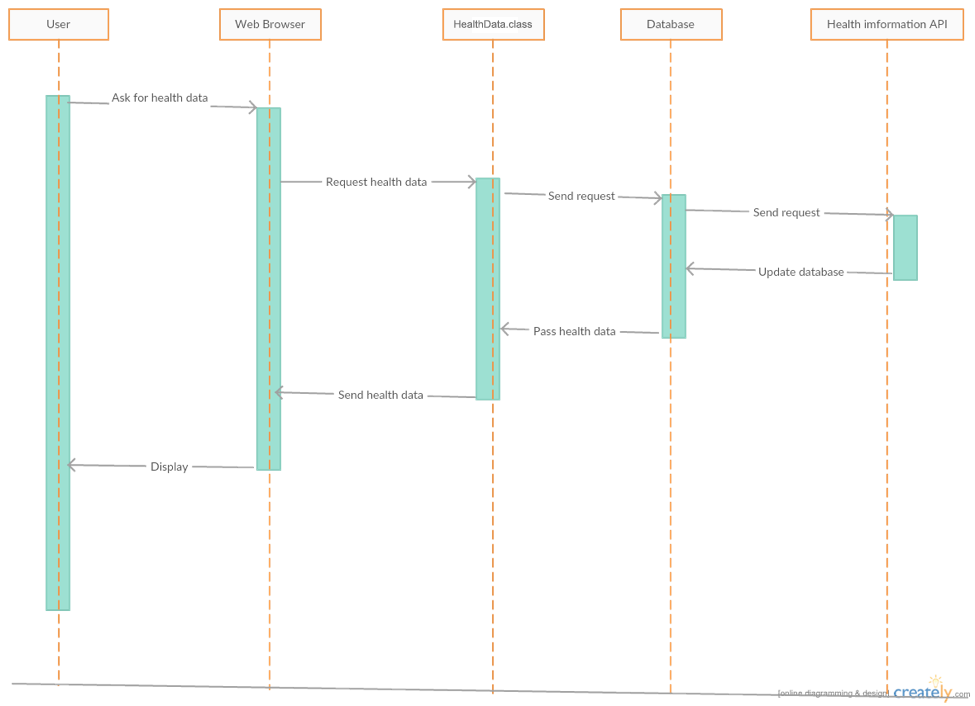
UC-10 : Once the recommended diet is provided from the website, the users have the capability to maintain or to modify the diet based on personal preference.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ST1 | ST2 | ST3 | ST4 | ST5 | ST6 | ST7 | ST8 | ST9 | ST10 | ST11 | ST12 | ST13 | ST14 | ST15 | ST16 |
| UC1 |  |  |  |  |  |  |  |  | X |  |  |  | X |  | X |  |
| UC2 |  |  |  |  |  |  |  |  | X |  | X | X |  |  |  |  |
| UC3 |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |
| UC4 |  |  |  |  |  |  | X | X |  |  |  |  |  |  |  |  |
| UC5 |  |  |  |  |  | X |  | X |  |  |  |  |  |  |  |  |
| UC6 |  |  |  |  |  |  |  | X |  |  |  |  |  |  |  | X |
| UC7 |  |  |  |  |  |  |  |  |  |  |  |  |  | X |  |  |
| UC8 |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |  |
| UC9 |  |  |  |  |  | X | X | X |  |  |  |  |  |  |  |  |
| UC10 |  |  |  |  |  |  |  |  |  |  |  |  |  |  | X |  |
| UC11 |  |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |
| UC12 |  |  |  |  |  |  |  |  |  |  | X |  |  |  |  |  |
| UC13 | X | X | X | X | X |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| Use Case UC-1 | Attain health data |
|
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_9, ST\_13, ST\_15  Users  Collecting the user’s data, analysis and evaluations  Database, health information API  user is logged in with proper authorization from the system  health information API is updated with user’s modification |
| Flow of events for main successful scenario | |
|  | 1. user successfully log in to the system 2. system requires information from health API 3. system display the required information 4. user reviews the information and modify the information 5. system notify the API and updates the information |
| Use Case UC-3 | Get Recommendation from the healthy diet |
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_11  Users  To hear suggestions from the system  diet recommendation API, Database  user is logged in with proper authorization from the system  Recommendations are kept in the system |
| Flow of events for main successful scenario | |
|  | 1. user successfully log in to the system   2 . user asks for recommendation  3. system requires diet API to provide recommendation  4. system display the recommendation  5. user determine whether to follow the suggestions  6 . system collect the reflection from the user  7. system determine whether to keep the recommendation based on the user’s decision |
| Use Case UC-6 | Share the information to social website and invited friends |
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_16  Users  To share the information to others  Sharing API, other platform, Database  user is logged in and the internet access is available  Information from others is kept in the system |
| Flow of events for main successful scenario | |
|  | 1. user successfully log in to the system 2. system requires information from health API 3. system display the required information 4. user reviews the information and modify the information 5. system notify the API and updates the information |
| Use Case UC-8 | Customize own diet plan |
|
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_10  Users  Modify the recommended diet plan  Database, diet recommend API  user has at least ask the system to recommend healthy diet  diet API is updated with user’s modification |
| Flow of events for main successful scenario | |
|  | 1. user successfully log in to the system 2. user require the recommendation diet from system 3. system display the required diet 4. user reviews the diet and modify the diet 5. system notify the API and updates the diet |
| Use Case UC-9 | View others health data and goal with permission |
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_6, ST\_7, ST\_8  Users  Compare with friends to share experience  Database, social communication API  both the user and user’s friend are using the system  the database would keep user’s friend’s health information |
| Flow of events for main successful scenario | |
|  | 1. user successfully log in to the system 2. user send a request to his friend to get health information 3. user get permission 4. system attain required information through network connection 5. system display the attained information 6. database keep the health information of the user’s friend |
| Use Case UC-11 | Get a recommended wearable equipment |
| Related requirements :  Initiating actor:  Actor’s Goal:  Partitioning Actors:  Preconditions:  Postconditions: | ST\_12  Users  Get recommendation of wearable equipment  Database, wearable equipment recommend API  User logged in and database has health information about the user  wearable equipment API is updated, database keep the information |
|  | 1. user successfully log in to the system 2. user requires recommendation of wearable equipment 3. system look into the database 4. based on the information stored in the database, the recommendation API make suggestions 5. system display the recommendation 6. user determine whether to follow the suggestion |

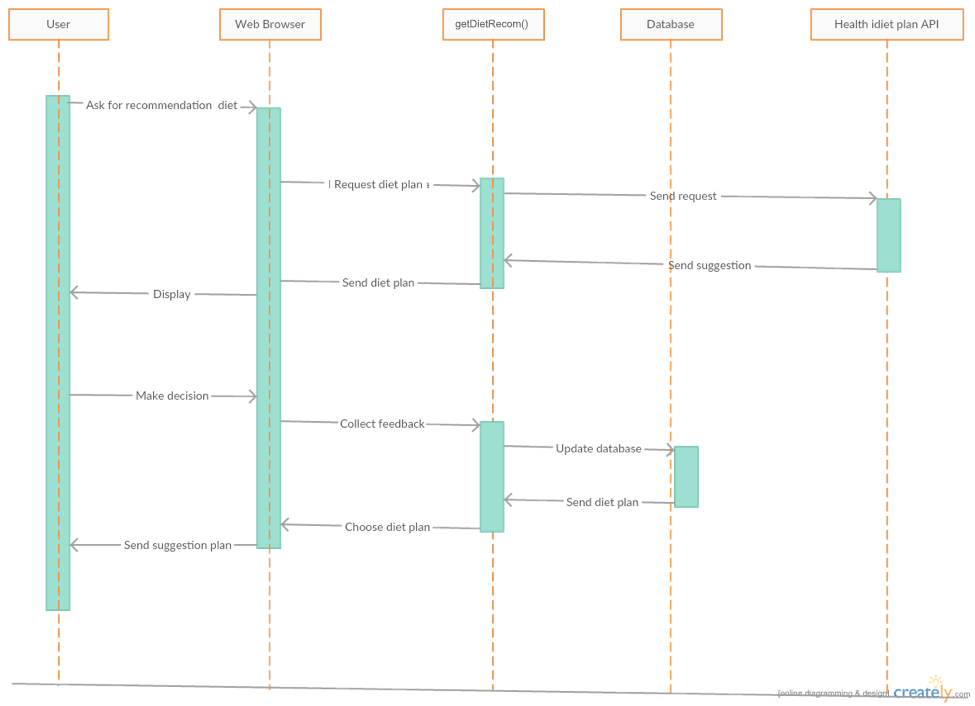


### d. System Sequence Diagram



UC-1 Attain health data

This diagram describe the use case 1. When user wants to attain health information, the web browser, in this case it is user page, sends the request to the HeathData class. The user page is the intrerface for user to interact with system. Then the class sends the request to the database, and database will get health information from the health data API. After update database, the HeathData class could query the health data from database and PageMaker class will update the user page. Finally, HealthData.display() send this to the user page and it will be displayed to the user.



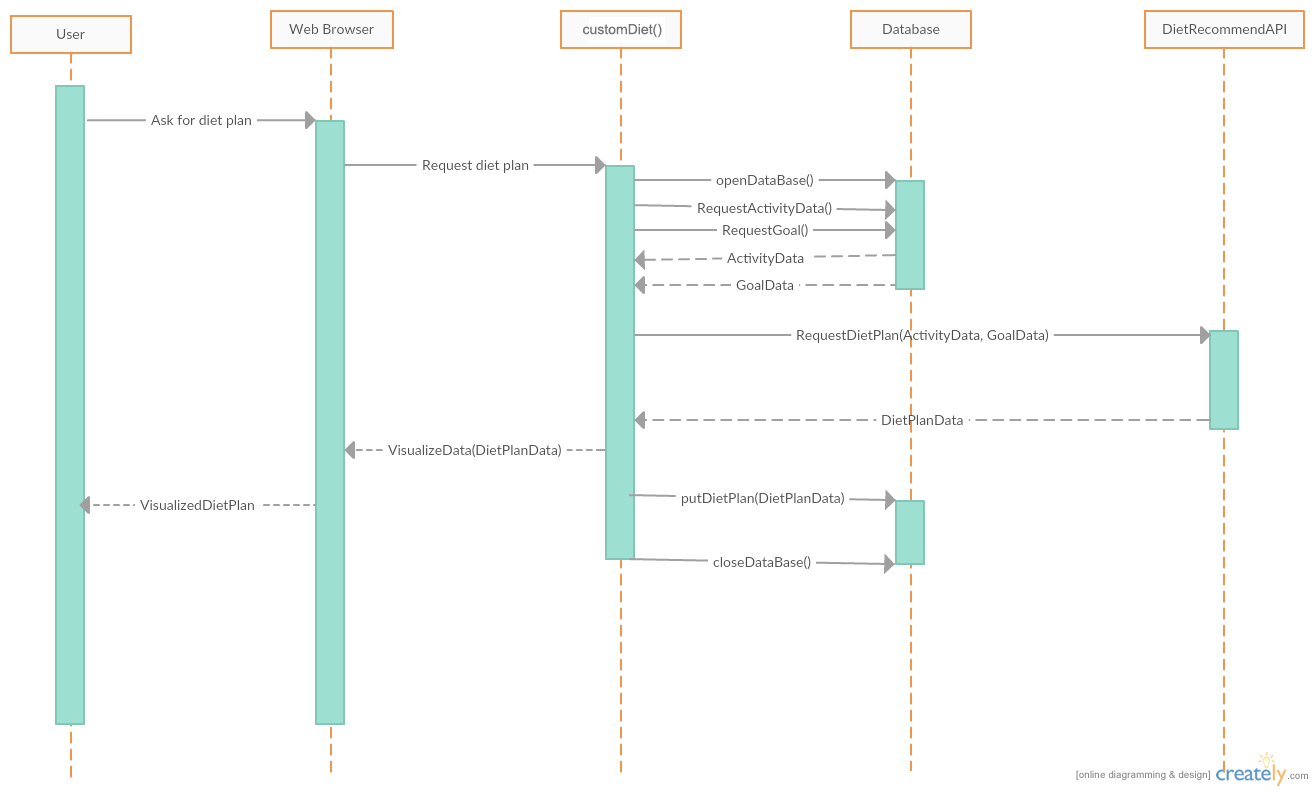
UC-3 Get Recommendation from the healthy diet

This diagram describe the use case 3. First, when user want to get some healthy diet suggestions from system, the web browser, in this case it is user page, will send this request to the getDietRecom() function. Then function will ask health diet API to get a recommended diet plan by calling NoticeDietReader, after this step, getDietRecom.display() function can show this plan on the user page with the noticepage maker. The getDietRecom.NoticeDietReader() can control the diet plan reader to read diet plan from external API. After the user decide whether to accept this suggestion, system can collect feedbacks of user’s decision and update the database. Finally, the system will show the last edition diet plan to the user page and store it into database by calling Database handler.



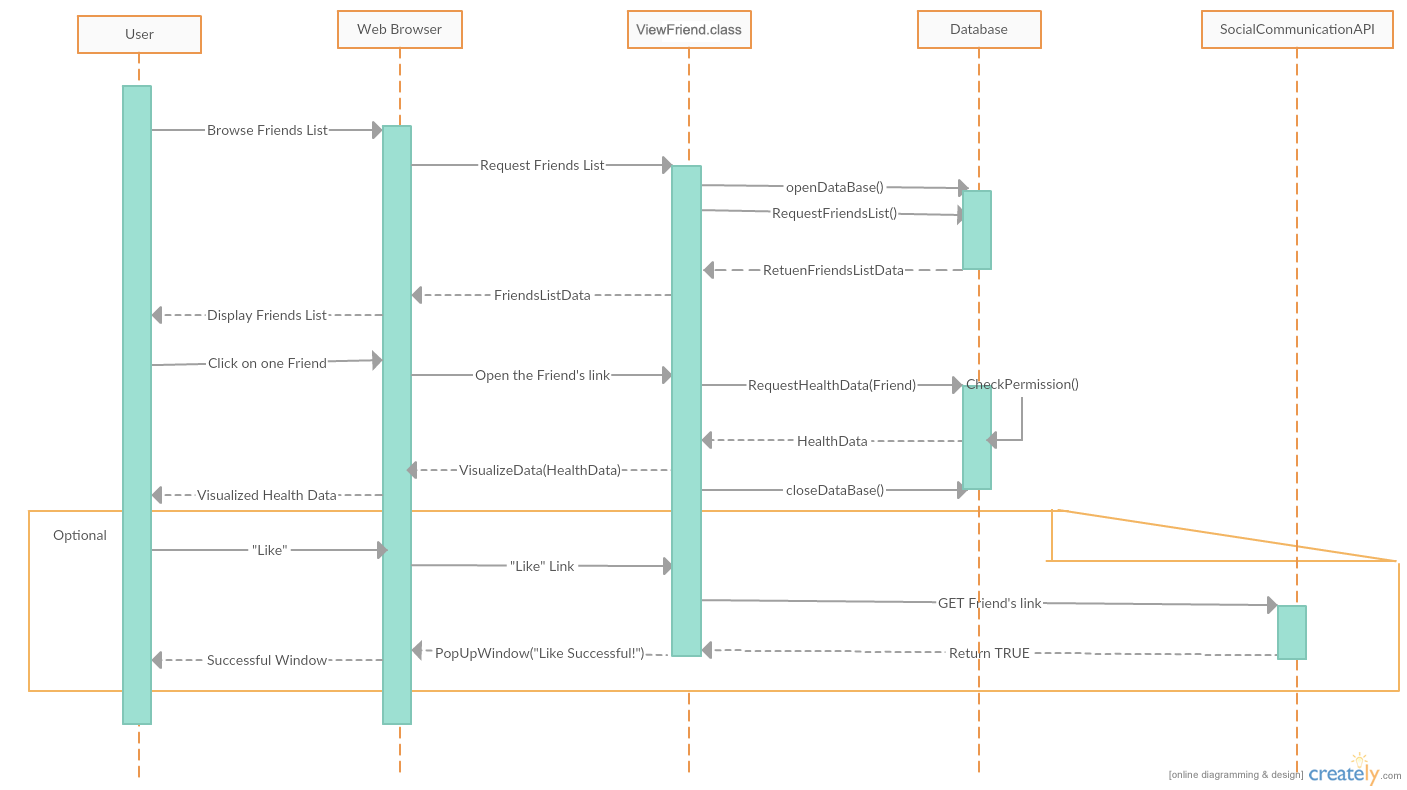
UC-6 Share the information to social website and invited friends

This diagram describe the use case 6. This use case introduces how to share some interesting informations by other external social website, when user wants to share some health data or some useful informations to their friends, the web browser, in this case is user page, will get this request and socialPlatform.request()send it to the function. socialPlatform class will send this requirement to the suitable API by using SN Connection and get the information from external API. The next step is system transport this message to the userpage and display to the user. After user modify the message and agree to share this message to others, the socialPlatform.NoticeSNConnection() which be called by controller will send the new message to the external API, and API will send them out. If The API successfully share this information, it will return a success signal. socialPlatform.display() will collect this signal and display to the user by updating user page. Finally, user will be notified that they have already shared the to his friends.



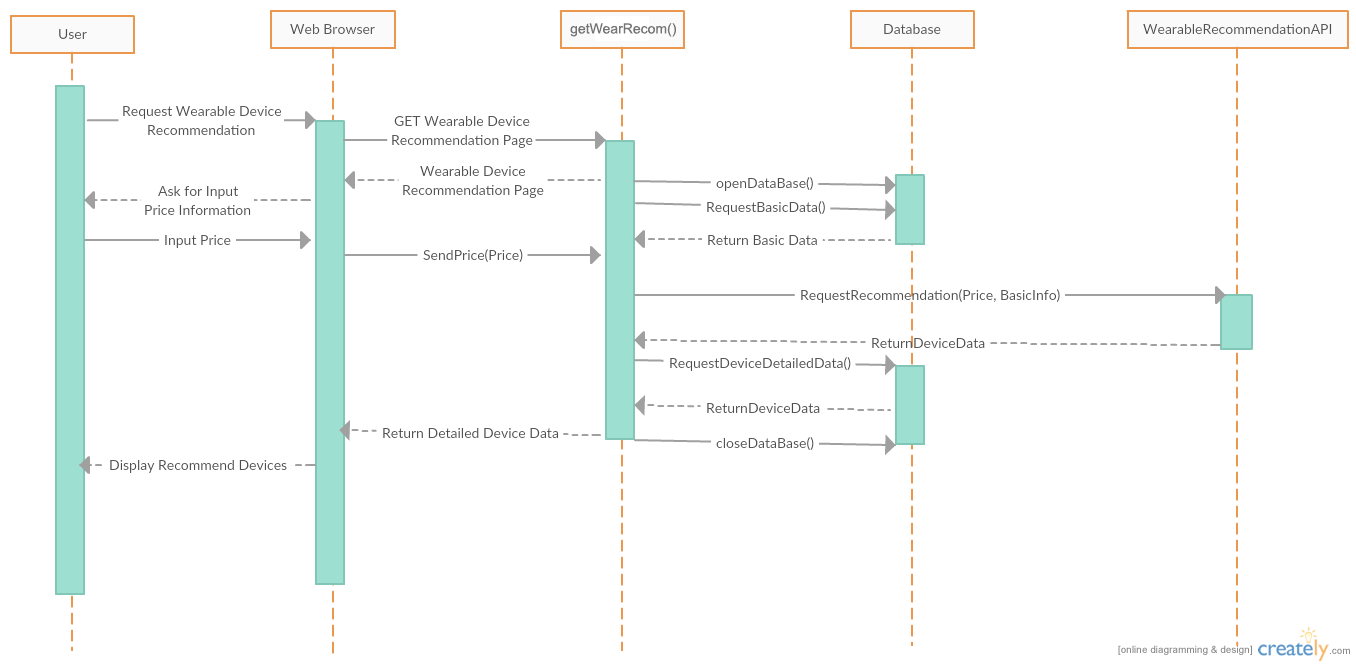
UC-8 Customize own diet plan

This diagram starts when user request for a diet plan. Then the browser send the request to System. The customDiet() class will query in Database and get the Health Information ( Activity Data ) and User’s goal Data. Based on the data, the class will send a request to our diet plan API with the customized data to request recommended diet plan. The difference between this class and getDietRecom() is this class will request Diet recommendation with customized user data instead of getting some general recommendation. After getting the diet plan, the customDiet() class will visualize the data with customerDiet.visualizer() class and then send to the browser to display to user. At the meanwhile, the Databse handler will store the diet plan data to the database in case the user would like to check in the future. The customerDiet() class will end when the data is stored in the database and it closes the database, and sends the data back to user, whichever is later.



UC-9 View others health data and goal with permission

When the user enter into the friends page, which is the start of the System, the ViewFriend class would request Friends List from Database, then it would send the data to web browser to display. Then, the user can choose one friend, the web browser will use the link of the friend and send it to ViewFriend class. The class will request the Health Data from database. The database handler will first check if the user has the permission to read the health data, if so it will return the data. The System will then use VisualizeData function to send it to user’s browser for displaying. If the user chooses to “Like” the friend’s data, the system will send the “like” request through API, if successful, the API will return true to the ViewFriend class. And the class will use the browser to pop-up a window indicating the “Like” operation is successful. The class will end after this.



UC-11 Get a recommended wearable equipment

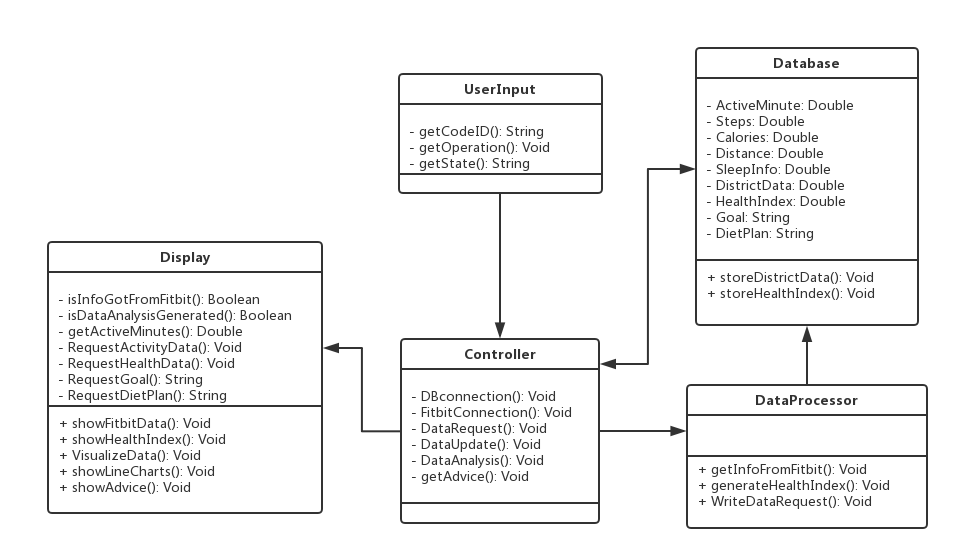
The System will start when the user enters into the wearable device recommendation page. The getWearRecom() function will first request the basic data of the user, at the same time it will also ask the user to input the desired price of the wearable devices. Then the function will bundle the price and the basic data of the user and send it to the Wearable Device Recommendation API. The API will send back a list of devices to the function. The function will then send the devices with the details information stored in the Database to the User. The getWearRecom() function will close the database after this. Then the function will end.

## 

## 4. Class Diagram and Interface Specification

### 

### Class Diagram



Description:

(1)UserInput:

Invariants: Operations the user do or text the user insert

Precondition: User has a keystroke or click

Postcondition: Return a value which means the function the user want to call

(2)Display

Invariants: Code ID the user inserted

Precondition: Data are retrieved from Fitbit and generated

Postcondition: Show webpages to the user

(3)Controller

Invariants: Actions like retrieving data from Fitbit, generating health index.

Precondition: A request from other classes

Postcondition: A request to another class or return a result

(4)Database

Invariants: The user’s data from Fitbit and the health index, and also the district data such as

smoking and exercise condition.

Precondition: Access to the Fitbit and update the data

Postcondition: Created the data of Fitbit and updated the data

(5)DataProcessor

Invariants: Actions to analyze the data such as calculating health index and deviation of different

health index.

Precondition: Data retrieved from Fitbit

Postcondition: Health index and other analysis created into the database

### 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.
* RequestActivityData(): Void
  + getSteps(), getCalories() and getDistance() will be called.
* 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.
* RequestDietPlan(): String
  + Show the user information about the diet plan, including everyday exercise plan, meal plan and so on.
* RequestGoal(): String
  + String variable corresponding to the user’s goal information based on database.
* RequestHealthData(): Void
  + getSleepInfo(), getDistance(), getSteps() and ShowHealthIndex() will be called.
* 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.
* VisualizeData(): 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 |

Description:

Controller:

Control the check to validate data  
Control the page maker to generate page to display data  
Control the SN connection to connect social network and get information from it  
Control the advice reader to read advice and store in database  
Control the diet plan reader to read diet plan and store in database  
  
Checker:

Validate data Check whether the data is legal

DB Connection

Access to Database and execute relative request.

InfoStorage

Store the data about health information.

Communicator

Render search request to Database.

TextReade:

Allow actor inputs data and put the external data into system

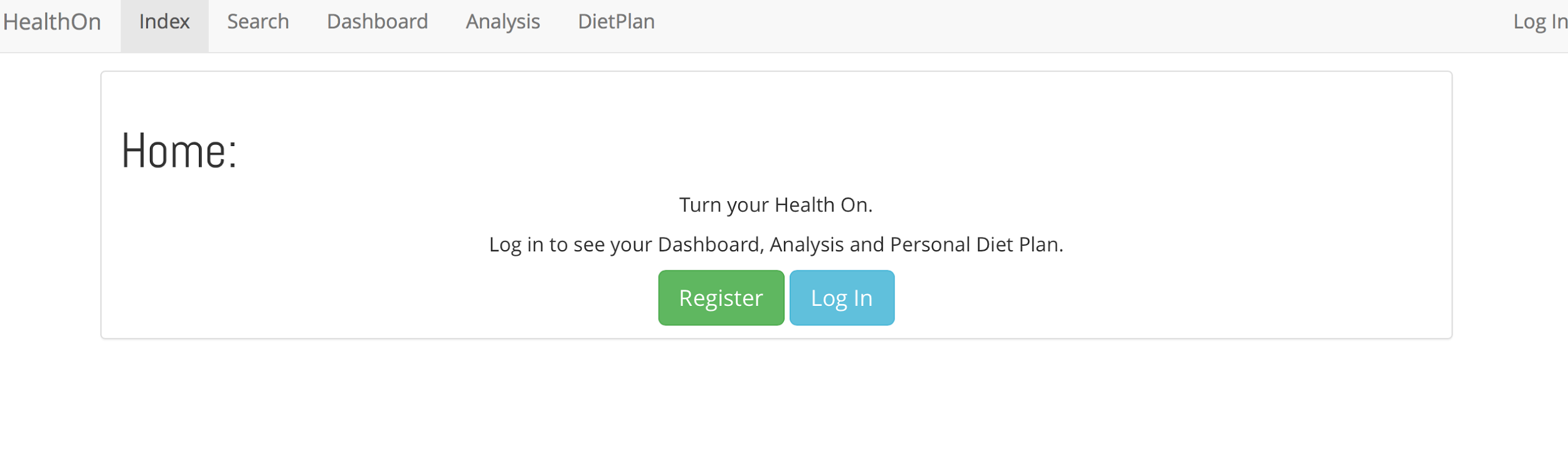
HealthAnalysis:

HealthAnalysis receives the text and calculate.

### d. User Interface

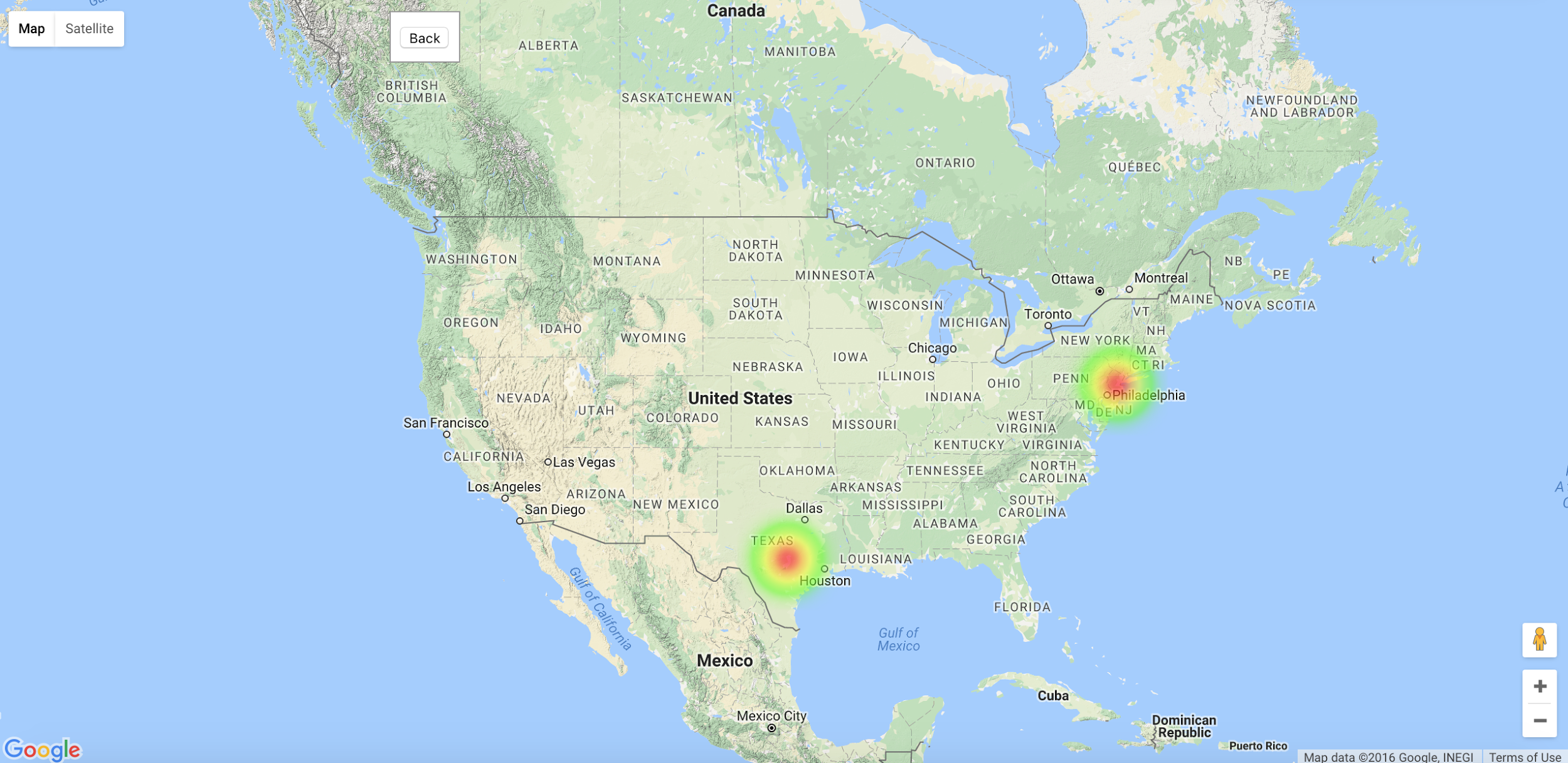
**(1)For users:**

This is the access page of the website. For a new user, he/she can click sign up to create an account. For a registered user or data analyst, he/she can input his/her username and password to get into the main page of the website.



Log in page

This is the heatmap function which reflects the activity times in different areas. The red mean more active and green means less active. As the picture below shows, the NJ areas is more active the TX area.



Heat map

This is the main page of the website. All the function can be access at this page.

(1)To click the dropdown menu to access system setting, personal profile and logout.

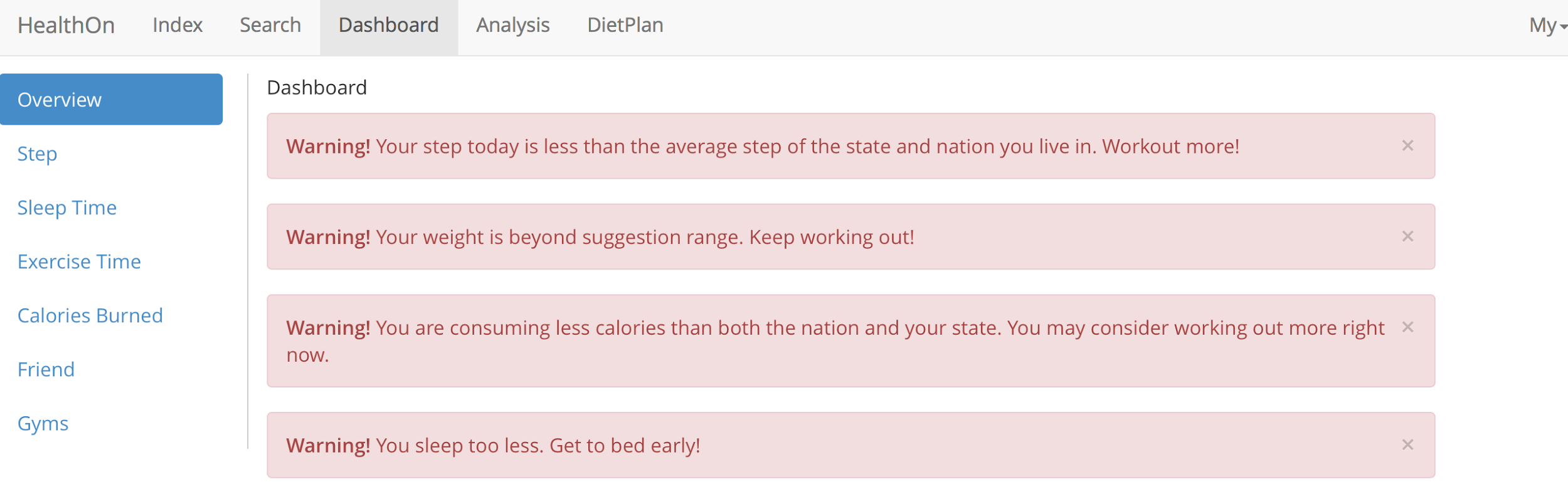
(2)To get the messages or news at the notification text window.

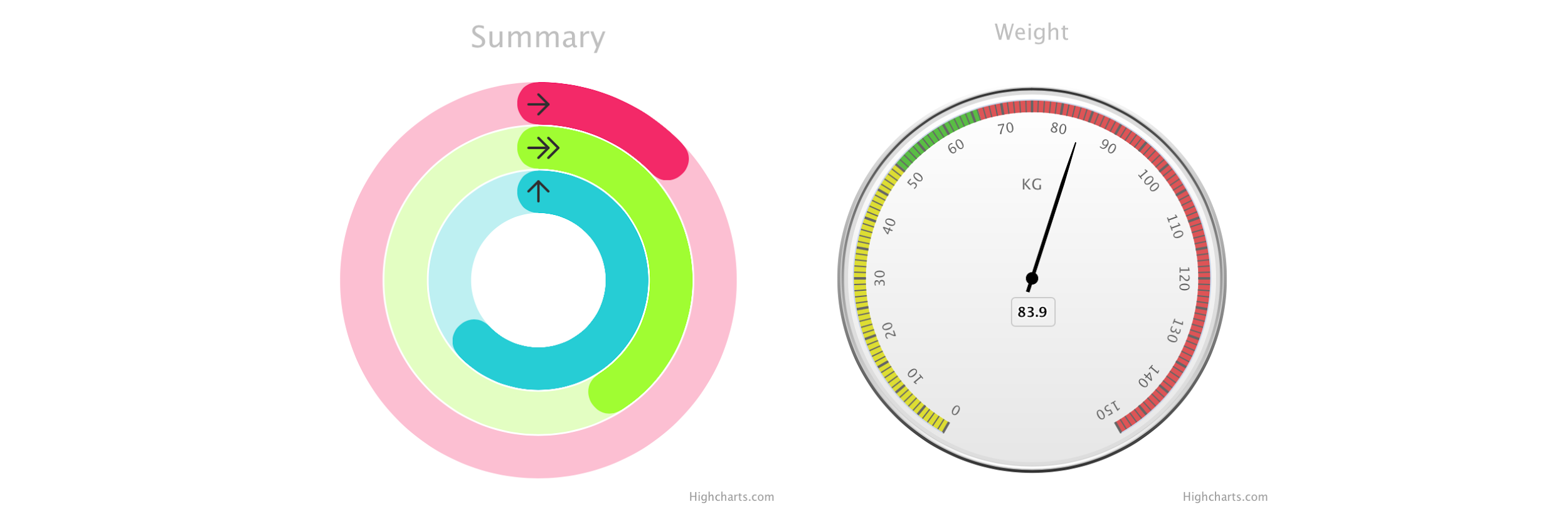
(3)To get an overview graph of daily data from the Daily data area.

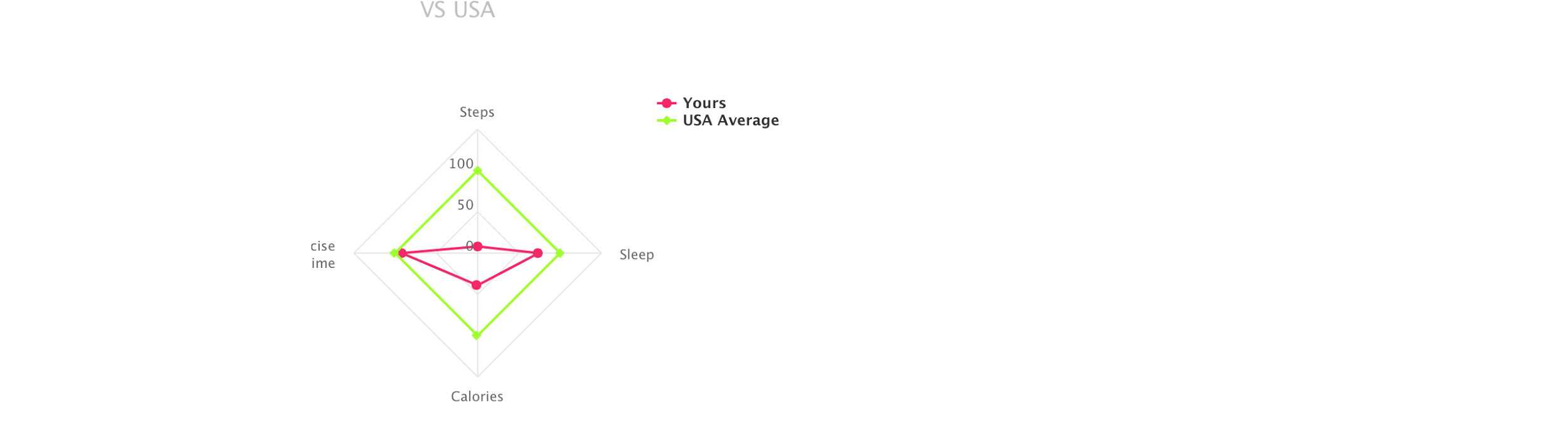
(4)To get the daily rank in the rank list, users can invite new user by clicking the “invite my friend”.

(5)To edit personal information (name, e-mail, health condition, goal) by clicking My profile.

(6)To get specific daily data, advice, recommend device and diet plan by clicking the Daily data area.









Main UI

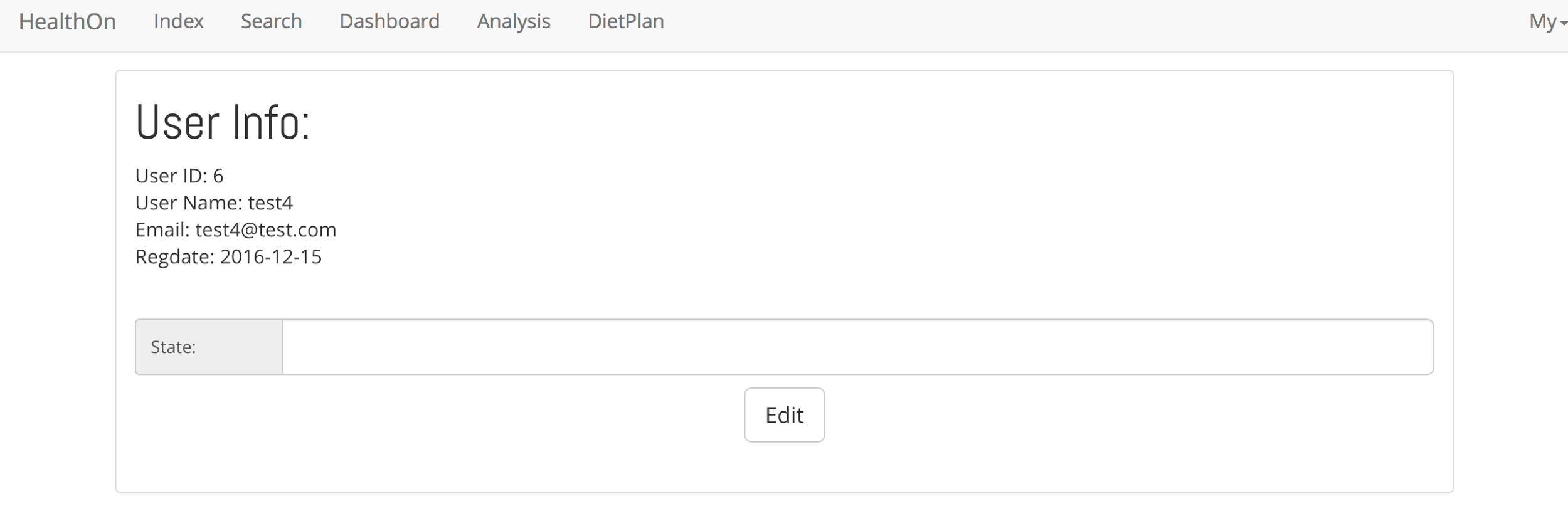
Personal profile

Personal profile provides the specific information of users.

(1)User can edit his/her logo, name, e-mail and more personal information.

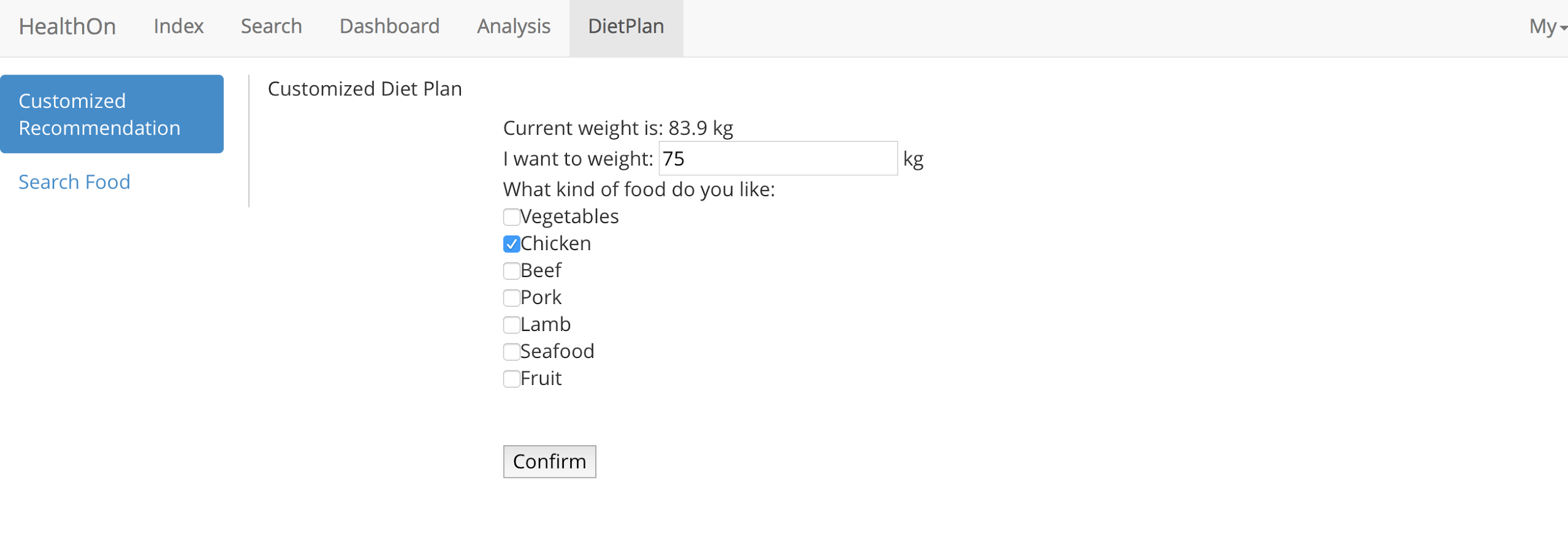
(2)Set personal goal.

(2)User can set his/her current health information to get health advice according to his/her daily data.

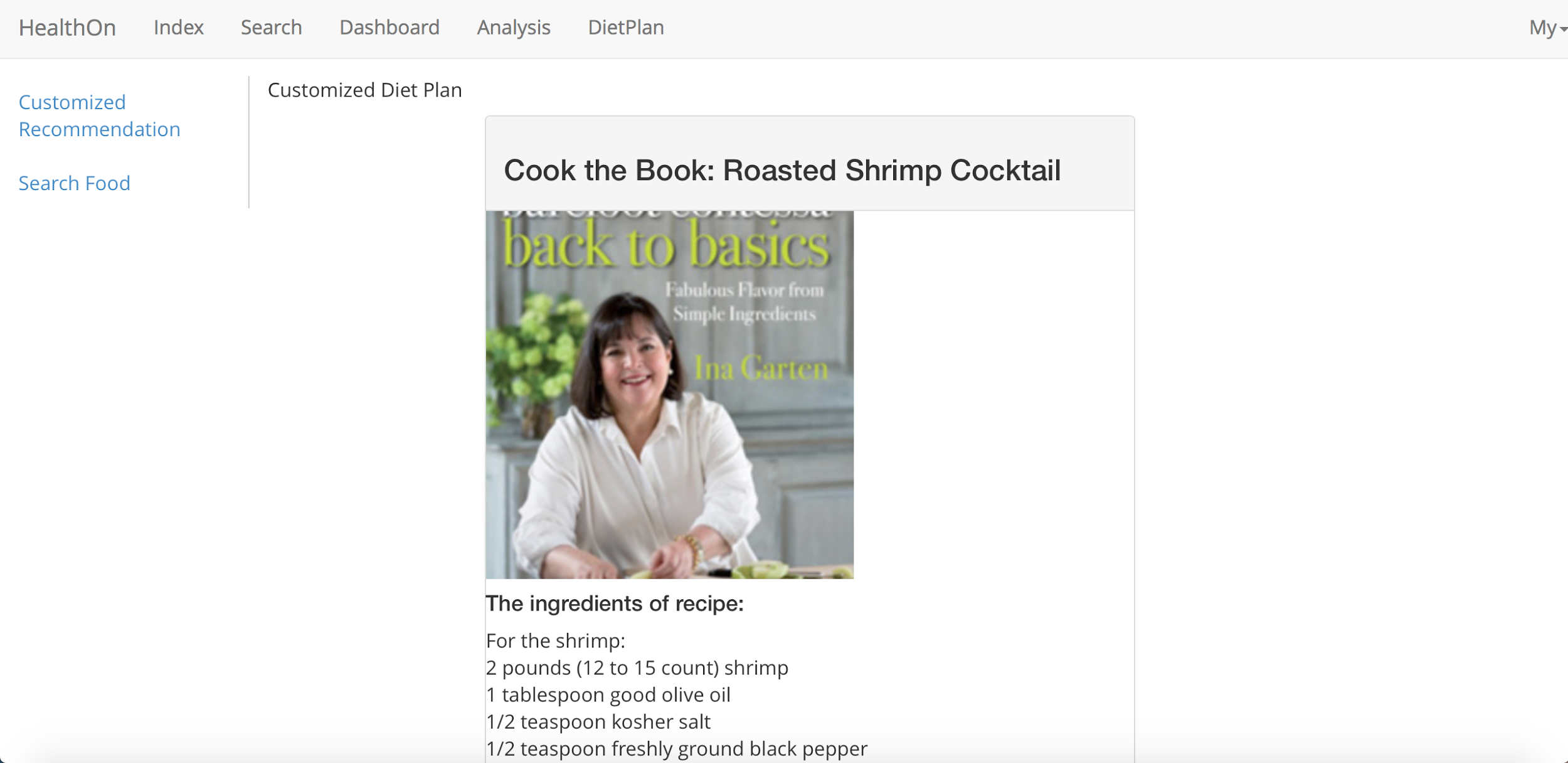


Personal Profiles

User can input the expected daily calories and get a recommend diet plan. User can also add or delete food from the plan.



Diet Goal



Diet Recommendation

**For Data Analysts:**

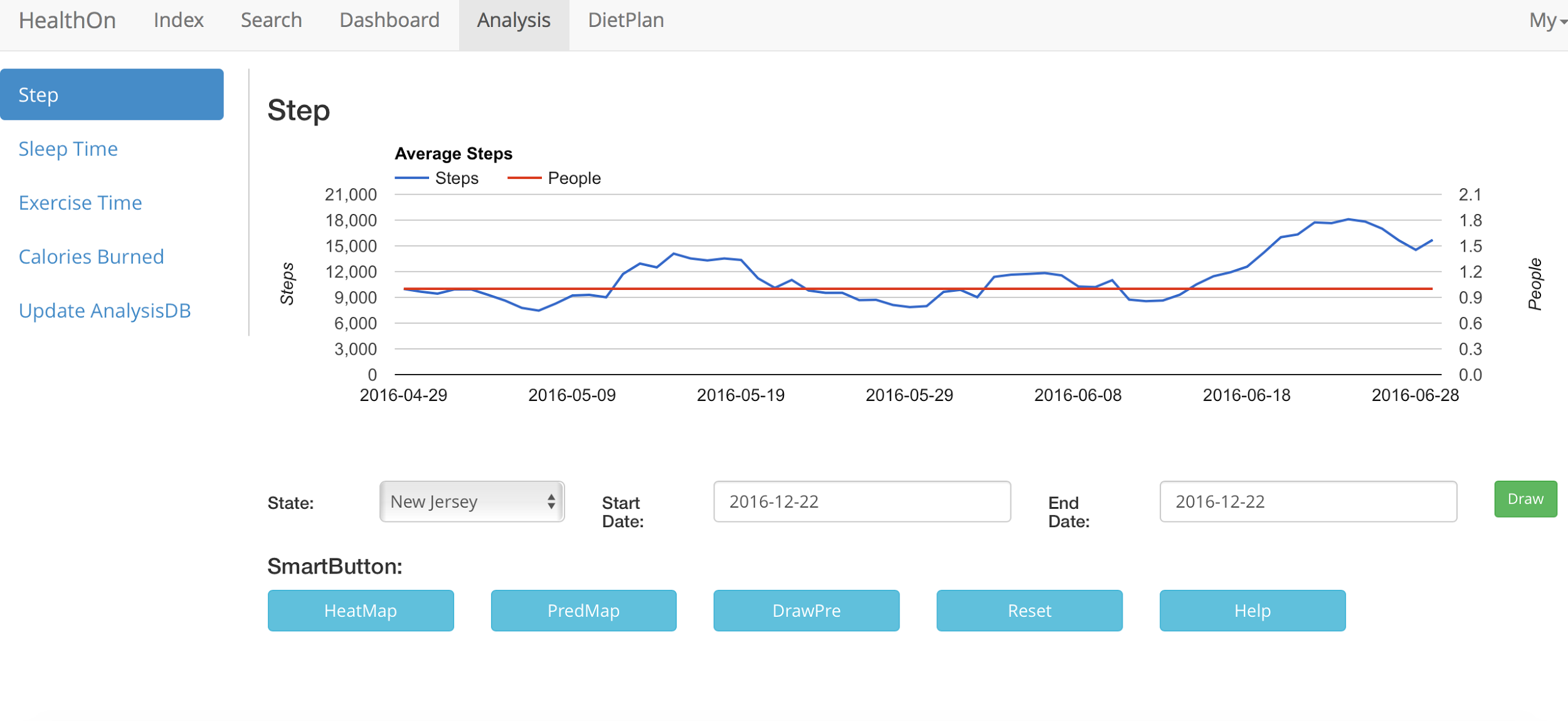
This page provides the specific advice from the data analysts for every part of the daily data of every user.

(1)Graph for goal accomplishment shows the credit the user gets for his/her goal according to his/her daily data.

(2)To click the dropdown list, user can get advice (including sleep, walking distance, food...) from the analyst .

(3)User can edit his/her custom diet plan by clicking the Get Diet Plan.

(4)User can get access to the page of recommended device according to his/her health information and daily data by clicking Get Device List.



Analysis

### e. User effort estimation

***For users:***

UC-1 Log in:

1. Click on sign on button.
2. Input username.
3. Input password.
4. Click sign in button.

UC-2 Invite friend:

1. Click on invite friends button.
2. Input your friend's email address.
3. Click on send button.

UC-3 Set Goals

1. Click on Set Goals button.
2. Select one item in the task list.
3. Set up the target.
4. Click on the Enter button.

UC-4 Share on social network

1. Click on the Share button in the dropdown list.
2. Input some description.
3. Click on the share button;

UC-5 Custom diet plan.

1. Click on diet plan button.
2. Input the expected calories.
3. Add or delete food in the list.

UC-6 Get advice

1. Get advice button of each health data in the dropdown list.

***For the data analyst:***

UC-7 View and get statistic charts

1. Select the item in the categories of data.
2. Input the filter to search.
3. Click on the enter button.
4. Click on the Save data button.

## 5. Functional Unit Tests

### a. Class tests

**Goal:** To check whether each class in the whole system can work separately, the aim is to ensure that all the basic functions can work correctly:

* The creation and accessibility of MySQL database
* The connection of MySQL database with data downloaded from the website and display the data on the web page
* Test whether the webpage can be shown correctly
* Test whether the user can register and log in
* Test the connection of social network and the return of information from it
* Test the user dashboard functions
* Test whether the analyst can view data correctly
* Test whether all graphs or Line Chart can be displayed correctly
* Test the getting advice function
* Test whether users can custom their diet plan

**The results:**

* The MySQL database is created successfully in our computer.
* The data can be stored in database and we can download or upload and modify these data from our remote devices.
* The UI display context correctly as we design it.
* Users can sign up and all the profiles registered can be saved in database.
* Connection of social network is successful and we can get the information we needed.
* The activity condition information can be displayed on the dashboard.
* Analysts can see given user’s healthy information and can give advice to the user.
* Different kinds of line charts can be shown based on user’s health condition.
* The request of getting advice can be processed correctly.
* The user custom diet plan can be loaded in webpage and can be modified by user.

### 

### b. Functional unit tests

**Goal:** To test the running condition of all the central parts of the application: the data analysis and interactions between different parts

1. Test ID: TC1\_Register

Unit to test: User Register

Assumption: The application is run on the user register screen, and is waiting for the user’s action.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/ Fail** | **Comment** |
| User’s information | Register successfully | Pass if the register is successful. | This is to make sure the user’s information is correct |
| Wrong information or information is incomplete | The error information | Pass if the system can detect the error |

1. Test ID:TC2\_Login

Unit to test: User Login.

Assumption: The application is run on the login screen, and is waiting for the user’ s action.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/ Fail** | **Comment** |
| Vail UserID and Password | User log in successfully | Pass when the user log in and show correct information | This is to make sure that user can access their profile successfully. |
| Incorrect userID or password | Error information | Pass when system shows the log in is rejected |

1. Test ID: TC3\_DataCollection

Unit to test: Healthy data collection unit

Assumption: The user has logged in and at the user main page, and is waiting for collecting user’s data.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/ Fail** | **Comment** |
| The user information | All the data related to user | Pass if we can get the data we wanted | This is to make sure we can get healthy data from Monitor device’s API |
| Invalid information | Error message | Pass if it displays the error info. |

1. Test ID: TC4\_Dataanalysis

Unit to test: Data analysis unit for analyst

Assumption: The user has requested data analysis and the analyst receives the request.

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/ Fail** | **Comment** |
| Start Data analysis | Show the user’s healthy data to analyst | Pass if the corresponding information is shown correctly | This is to make sure the analyst part is working well. |

1. Test ID:TC5\_ConnectSocialNetwork

Unit to test: Social Network connection unit

Assumption: The user has logged in and at the main page, and sends request to connect to social network

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Requirement** | **Expected Output** | **Pass/ Fail** | **Comment** |
| Start connection to social network | All the data related to user | Pass if we can get the data we wanted | This is to make sure that we can connect to social network to retrieve data and share on it. |

## 

## 6. System Architecture and System Design

### a) Architectural Styles

As a web application, our system is based on the Client/ Server model, which benefit the system for centralized data storage, and a scalable amount of users accessing and providing data. The clients of the system are different applications run on different user machines, which allow users to login the system, upload their monitoring data, and get or give the data analysis and interactive with social network. While on the other hand, the server which runs on a separate machine is in charge of the system database and all the backstage requests process. The server can be further partitioned into several basic functional units, which in charge of data management, data transit, and user application communication.

The reason why we choose Client-Server architecture is we must have powerful centralized processor to provide an always-on service. The application is always available, the data and functions can be accessed at anytime. Also, servers support remote access which enables users can access to application without physically being in front of the system. Client-Server also can provide a high security which ensures that only individuals with proper permissions access specific data and applications residing on the server.

By using this client/ server model, the system reduced the procedure done on the client side to a minimum degree, and leave all the process work to the server. In this way, we can provide a light user application, also a better coherence with all the user data. We could have a light front-end application for user to run fast on most device. But on the other hand, it requires a powerful server to deal with and store lots of data effectively. The centralized architecture make the server act as both the database and the central data processor in whole system.

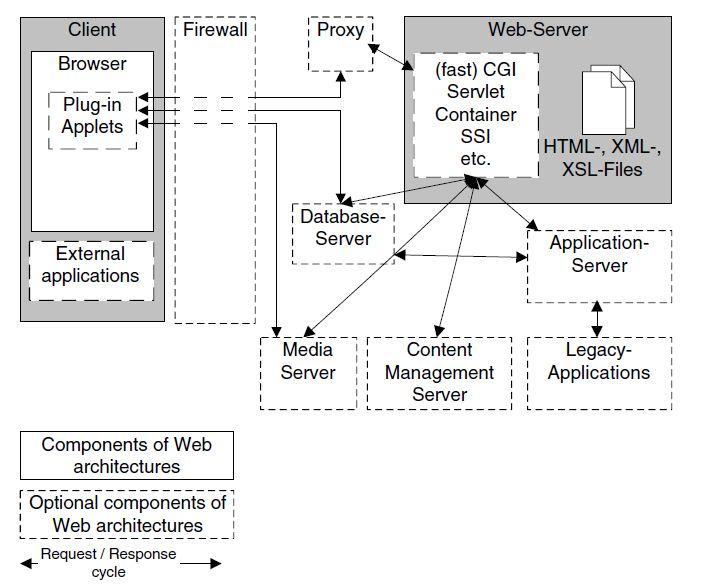


Figure 1 Client-Server Architecture Model

### **b)** **Identifying Subsystems**

We can basically divide the system into two subsystems: the Client side and the Server side. The Client subsystem is consisted of Web Brower and other User Interfaces, such PHP, HTML and XML files. As for Server side, there is a central controller unit to control the whole works inside server side. User authentication unit can validate user information and send user request to central controller. Then the controller forwards the request to Server Logic unit to parse and process it. After analyzing request, the central controller will connect to Database Server. The Database server package is responsible for data storage and providing external data such as Social Network information.

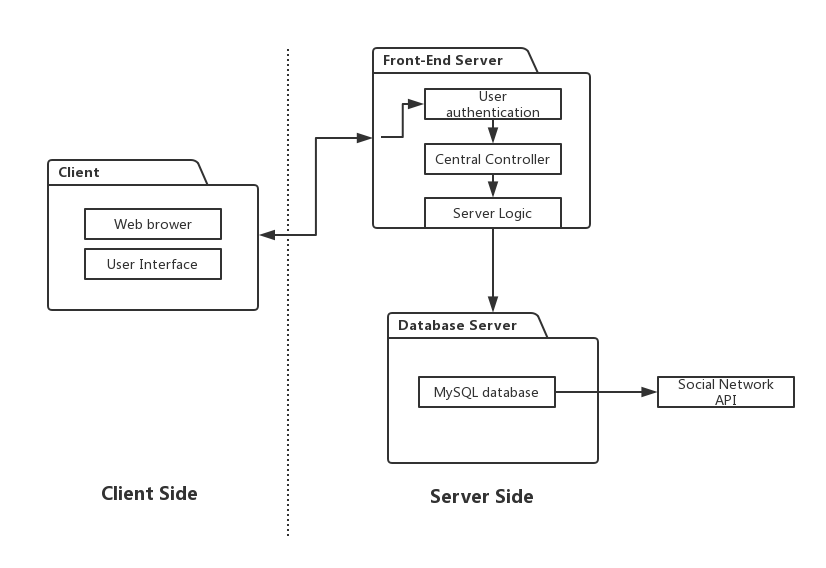


Figure 2 UML Package Diagram

### c) Mapping Subsystems to Hardware

As mentioned above, we can map the subsystems to the following hardware.

Client Subsystem: Allocated in user’s device, such as PC and intelligent mobile phone. The web browser can be run on different user machines.

Server Subsystem: Allocated in one powerful server. We will use a PC as server (For current stage), which includes Central Controller part and User authentication part.

Third party System: Apache/nginx for front-end application. MySQL Database for persistent data storage and Social network APIs, for example, Facebook API.

### d) Persistent Data Storage

We use MySQL database for the persistent data storage. MySQL Database is an open-source document database and one of the best Relational Database Management System in WEB application. It gives great support to the PHP. It has multiple storage engines, allowing one to choose the one that is most effective for each table in the application.

We will store health data, custom diet plan data and social information data in MySQL database. Different type of data has different schema. We use PHP to retrieve and save data in Server. The health data keeps healthy records which get from personal health monitor. It can be read both by user and analyst. The custom diet plan contains a food list which chosen by user. At last, the social network factors are saved in social information data, such as rank among all friends.

Below are schemas of persistent data:

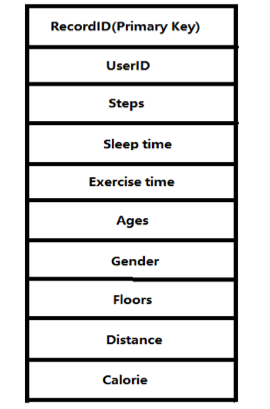


Figure 3 Health Data

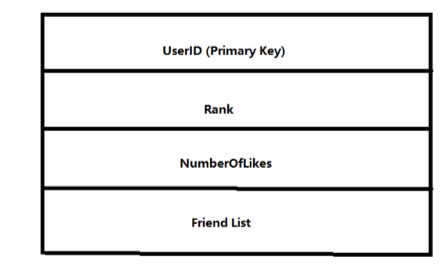


Figure 4 CustomerDietPlanData

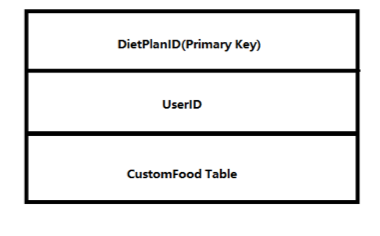


Figure 5 SocialData

### e) Network Protocols

There are many network protocols such as FTP (File Transfer Protocol), HTTP (Hyper Text transfer Protocol) and SSH (Secure Shell). In this system, a user uses the website to log into his/her personal account and access the software user interface. Since we have a web design, we need to use HTTP so users can navigate to the website from their personal computer.

Also, we will use SQL through PHP to exchange data from database

### f) Global Control Flow

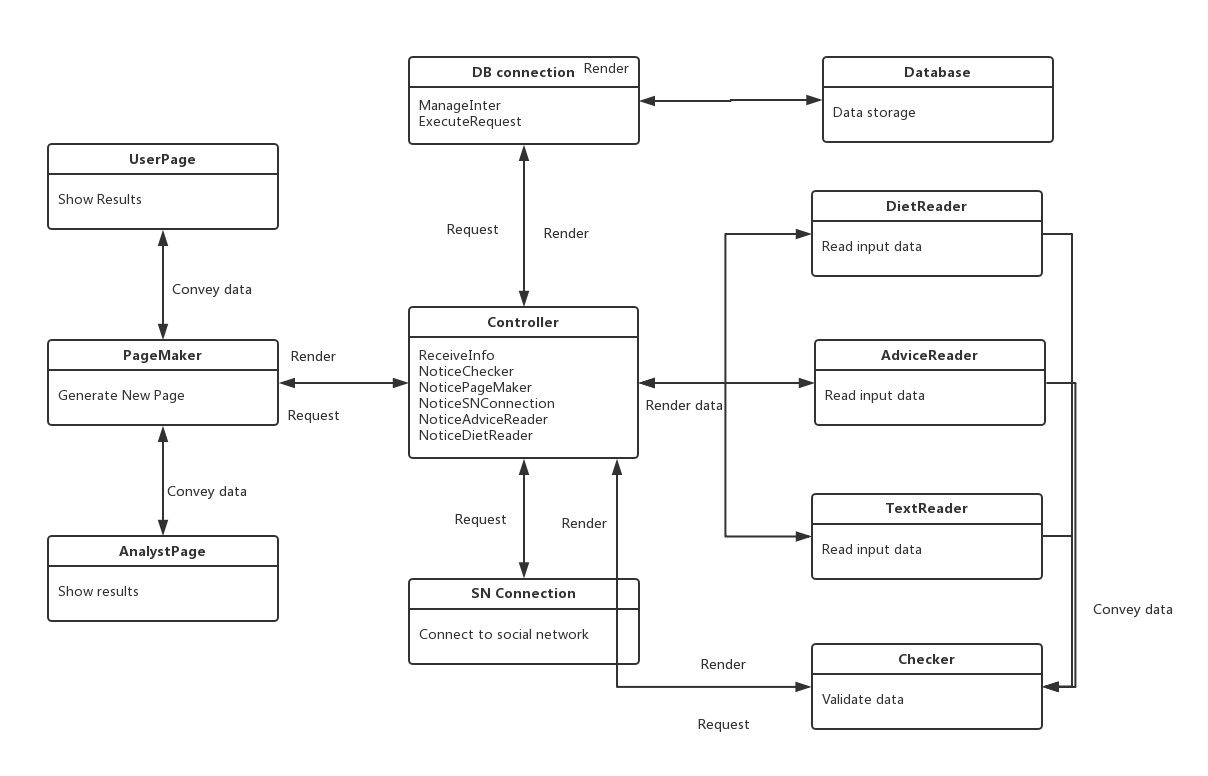
This system is an event-driven, which means that the system must respond to different events at any time. When user log in, he/she can send request by clicking or typing at any time, and the server will respond to his/her request, accept or decline, and display corresponding page to user.

### g) Hardware Requirement

We require Fitbit health monitor to get healthy data. The operating system of our system can be Window XP, Windows 7, Windows 8 or Mac. The hard drive storage needed should bigger than 3 Gbytes. The backend will based on the PHP using the web server xampp and Matlab, which make our computer as both the client and server. Fitbit Database and MySQL Database are the databases for our system.

## 

## 7. Domain Analysis



### **a. Concept Definitions**

To analyze the domain model, we first derive the domain model concepts and corresponding responsibilities from the formerly defined system use cases. Table lists all the domain model concepts and corresponding responsibilities

|  |  |  |
| --- | --- | --- |
| Responsibility Description | Type | Concept Name |
| R1:Store the data about healthy information | K | Database |
| R2:Read user’s custom diet plan | D | DietReader |
| R3:Read data that user types in | D | TextReader |
| R4:Container for user’s custom diet plan | K | DietPlan |
| R5:Controller receives the operation “Custom Diet Plan” and controls DietReader to get DietPlan and send to DB Connection to write into the Database. | D | Controller |
| R6: HTML document that shows the user’s healthy data | K | UserPage |
| R7:HTML document that shows the user’s healthy data such as number of steps , sleeping time and activity hours to analyst. | K | AnalystPage |
| R8: Prepare a database query that best matches the actor’s input and retrieve the records from the database | D | DB connection |
| R9:Connect to Social Network and retrieve information | D | SN connection |
| R10: Render the retrieved records into an HTML document for sending to actor’s Web browser for display. | D | Page Maker |
| R11: Controller receives the operation “Share on Facebook” and controls SN connection to connect to social network and retrieve information | D | Controller |
| R12: Controller receives the operation “Get advice” and controls DB connection to get Advice from Database | D | Controller |
| R13:Container for the advice that analyst gives | K | Advice |
| R14:Checker receives the username and requests Controller to use DB connection for using datatbase(InfoStorage) to find data in database to compare. | D | Checker |
| R15:Read advice from analysts | K | AdviceReader |
| R16: Controller receives the operation “Input advice” and controls AdviceReader to get Advice and send to DB Connection to write into the Database. | D | Controller |
| R17: Checker gets information from TextReader and check if the information is valid | D | Checker |

### **b. Association definitions**

Some of the concepts defined above as domain concepts have to work in certain patterns to finish some target requirements. Table below gives the corresponding association definitions based on the defined domain concepts.

|  |  |  |
| --- | --- | --- |
| Concept Pair | Association Description | Association Name |
| Controller<->Checker | Controller calls checker to check if information is valid and Checker returns results to Controller or checker requests to use database and controller returns results | Generate requests  Convey data |
| TextReader<->Controller | TextReader sends signals to Controller or Controller sends signals to TextReader to receive data | Generate Requests |
| Controller<->Page Maker | Controller renders its results and generate requests to Page Maker to display | Generate requests  Convey data |
| Checker<->TextReader | Checker checks information from TextReader | Check |
| Checker<->Database | Check checks information from Database | Check |
| DB Connection<->Database | DB Connection gets access into Database and saves data in Database | Save data |
| Controller<->DB Connection | Controller generates requests to use DB connection | Render requests |
| Controller<->Advice Reader | Controller calls Advice Reader to read Advice from analyst | Generate requests  Convey data |
| Checker<->DietReader | Checker checks information from DietReader | Check |
| Controller<->SN Connection | Controller generates requests to use SN connection | Render requests  Convey data |

### **c. Attribute Definition**

Among the defined concepts, some concepts share the same attribute, and only different from each other as they have different operands. These concepts are listed in below.

|  |  |  |
| --- | --- | --- |
| Concept | Attributes | Attribute Description |
| TextReader | Read input data | Allow actor inputs data and put the external data into system |
| DietReader |
| AdviceReader |
| Database | Data storage | Store all the data that the system needs |
| Controller | ReceiveInfo  NoticeChecker | Control the check to validate data |
| NoticePageMaker | Control the page maker to generate page to display data |
| NoticeSNConnection | Control the SN connection to connect social network and get information from it |
| NoticeAdviceReader | Control the advice reader to read advice and store in database |
| NoticeDietReader | Control the diet plan reader to read diet plan and store in database |
| UserPage | Display Result | Interface for user and analyst to interact with system |
| AnalystPage |
| Checker | Validate data | Check whether the data is legal |
| PageMaker | Generate New Page | Generate Userpage or Analystpage to show the result or get input |
| SN Connection | Connect to Social Network | Connect to social network to share information or retrieve information |

### **d. Traceability Matrix**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Domain Model | UC-1 | UC-3 | UC-6 | UC-8 | UC-9 | UC-11 |
| Database | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| DietReader |  |  |  | ✓ |  |  |
| TextReader | ✓ |  |  |  |  |  |
| AdviceReader |  | ✓ |  |  |  | ✓ |
| Controller | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| UserPage | ✓ |  |  |  | ✓ | ✓ |
| AnalystPage |  | ✓ |  |  |  | ✓ |
| PageMaker | ✓ | ✓ |  |  | ✓ | ✓ |
| Checker | ✓ | ✓ |  | ✓ |  |  |
| DBConnection | ✓ | ✓ | ✓ | ✓ | ✓ |  |
| SNConnection |  |  | ✓ |  | ✓ |  |

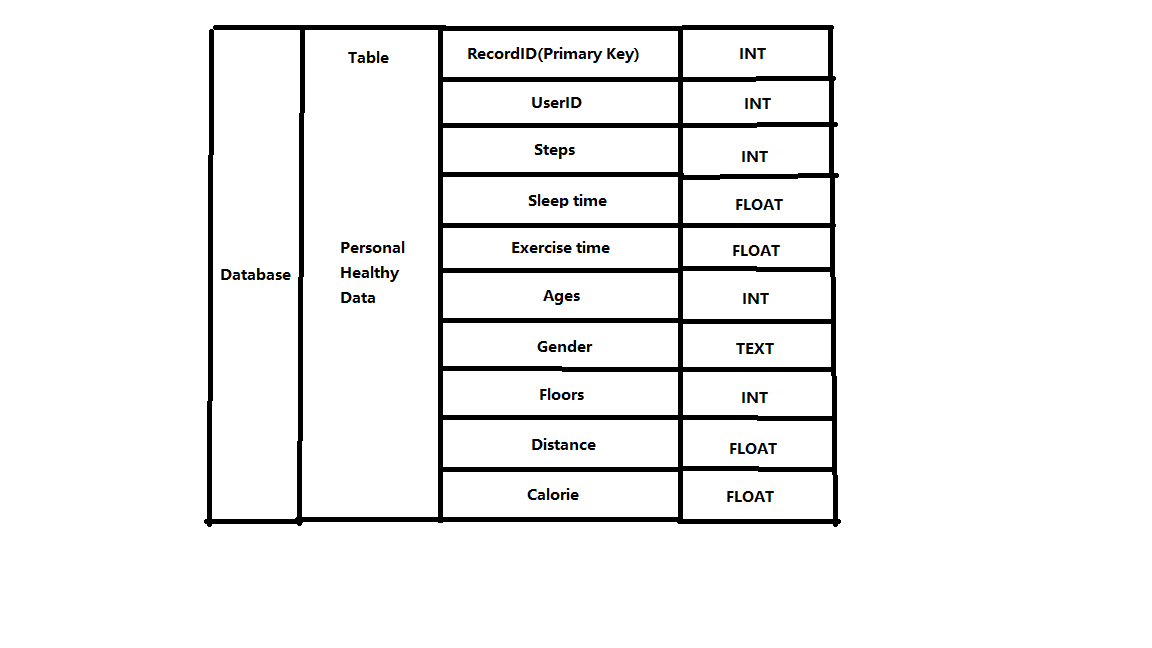
## 8. Algorithms and Data Structures

### a. Algorithms

#### (1)

### b. Data Structure

Our data are collected from Fitbit interface and healthy data website. Moreover, we must collect data from personal social networks. All these data will be stored in MySQL database. Here are some data diagrams to describe data structure:



*Data structure of Healthy Data*

## 9. Individual members’ contribution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Contribution | Yuwei Jiang | Xingyu Li | Chenfan Xiao | Jianing Xu | Chengyao Wen |
| User Interface |  |  |  | 50% | 50% |
| System Architecture | 60% |  | 40% |  |  |
| Database Architecture | 40% | 60% |  |  |  |
| User Management | 60% |  |  |  | 40% |
| Social Network API(construction, test) |  | 50% | 50% |  |  |
| Fitbit API (construction, test) | 60% |  | 40％ |  |  |
| DietPlan API (construction, test) |  | 40% | 60% |  |  |
| AWS Construction | 60% | 40% |  |  |  |
| Documentary Work |  |  |  | 60％ | 40% |
| Project Management |  |  |  | 40% | 60% |
| Peer Review | 20% | 20% | 20% | 20% | 20% |
| Subsystem Integration |  |  |  | 50% | 50% |
| System Test & Debugging |  |  |  | 50% | 50% |

## 10. References

1. Database(MySQL): http://php.net/manual/en/book.mysql.php
2. Database(MySQL): https://www.tutorialspoint.com/mysql/
3. DietPlan API: http://diet-agents.sourceforge.net/SoftwareDownload.html
4. Food API: http://www.programmableweb.com/category/food/apis?category=20048
5. MySQL Databases Management: https://sourceforge.net/projects/phpmyadmin/
6. MySQL Databases Management: https://www.phpmyadmin.net/
7. MySQL management tool: https://github.com/phpmyadmin/
8. HTML: http://www.w3school.com.cn/html5/
9. APIhttp://www.webdesignerdepot.com/2011/07/40-useful-apis-for-web-designers-and-developers/
10. API: http://www.gregreda.com/2015/02/15/web-scraping-finding-the-api/
11. XML API: http://itransact.com/support/toolkit/xml-connection/api/