

Software Engineering I 2016 Fall

Personal Health Monitor

Group 4

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

**Content**

**Part 1**

**1. System Sequence Diagram**

**2. Project Management**

**Part 2**

**1. Class Diagram and Interface Specification**

A. Class Diagram

B. Data Types and Operation Signature

C. Traceability Matrix:

**2. System Architecture and System Design**

**Part 3:**

**1.Algorithms and Data Structures**

A. Factor Analysis

B. Data Structure

**2. User Interface Design and Implementation**

A. Preliminary Design

B. User Effort Estimation

**3. Functional Unit Tests**

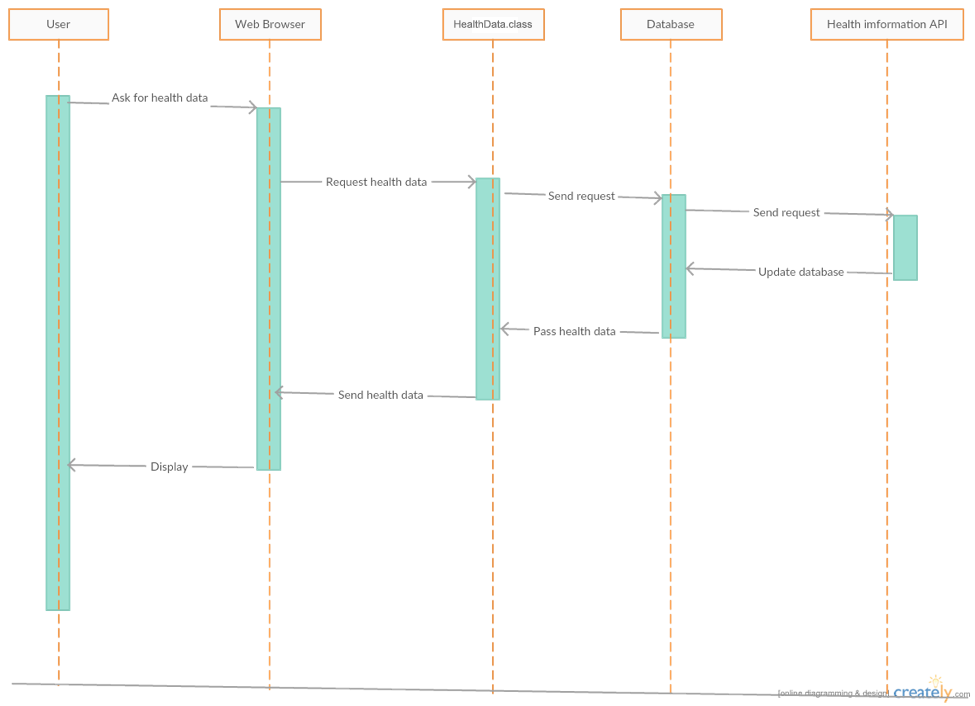
A. Class tests

B. Functional unit tests

**4. References**

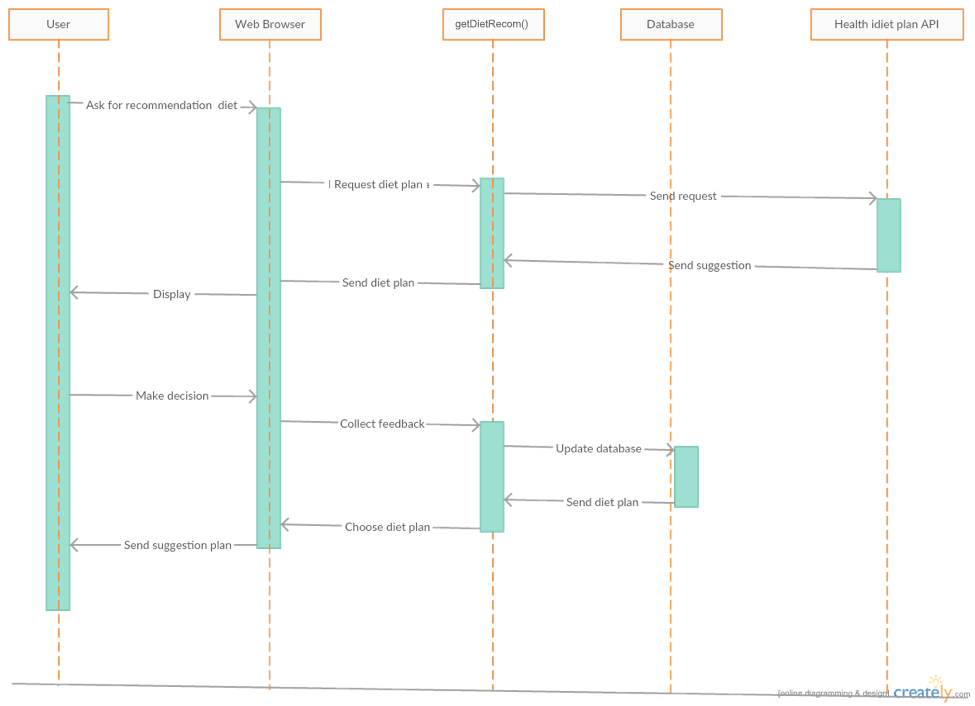
# Part 1

## 1. System Sequence Diagram



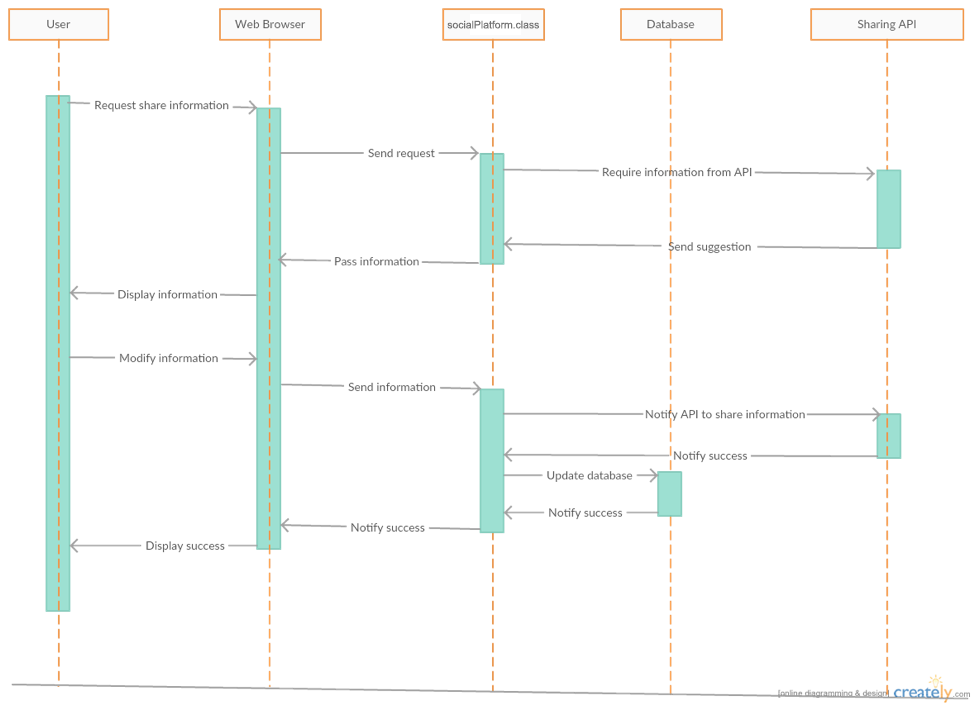
UC-1 Attain health data

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.



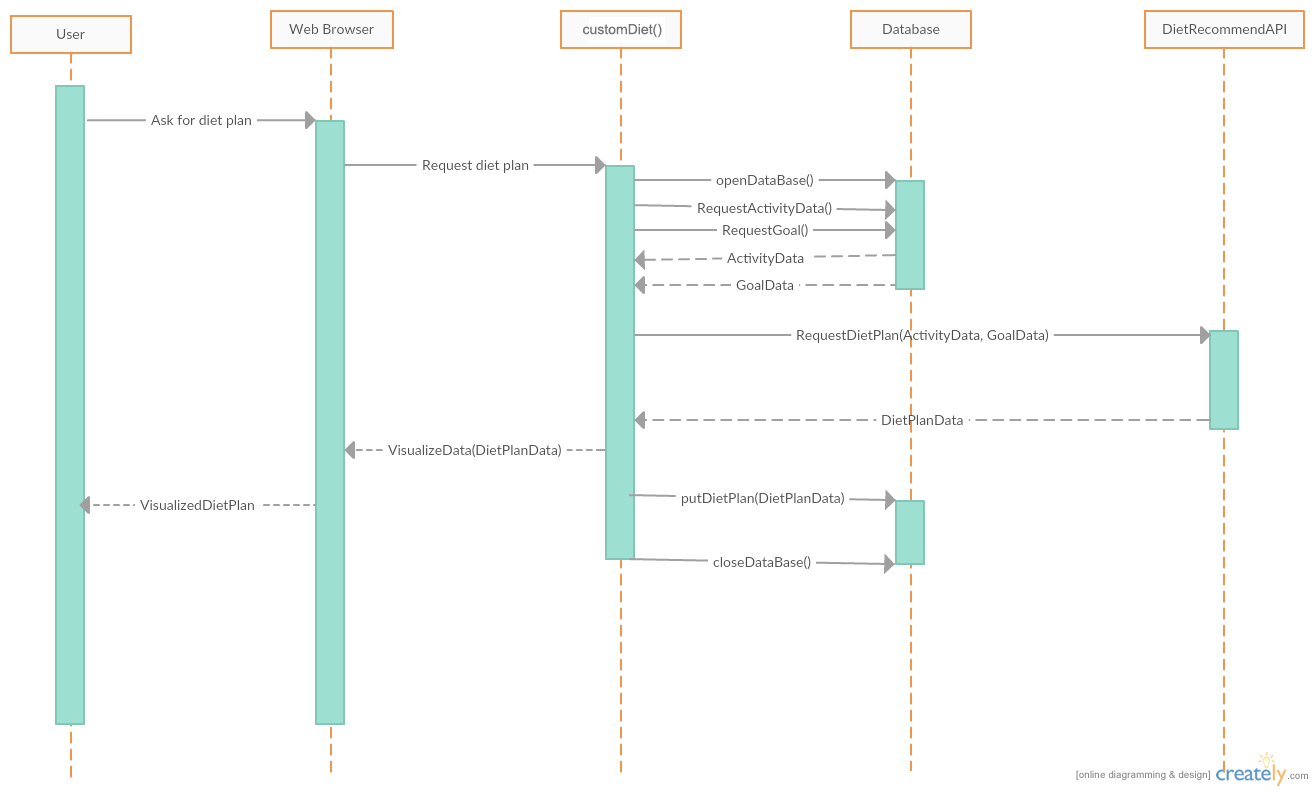
UC-3 Get Recommendation from the healthy diet

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.



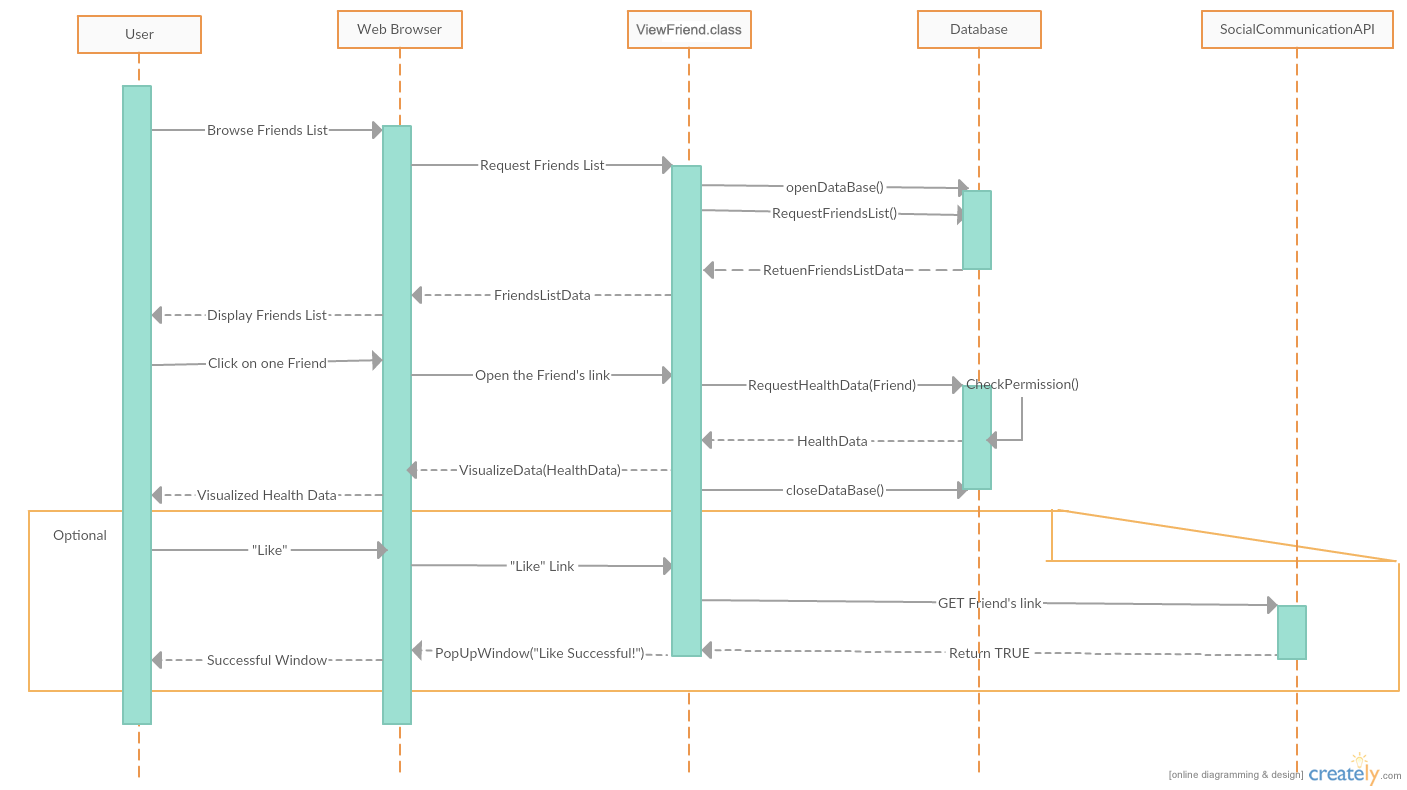
UC-6 Share the information to social website and invited friends

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.



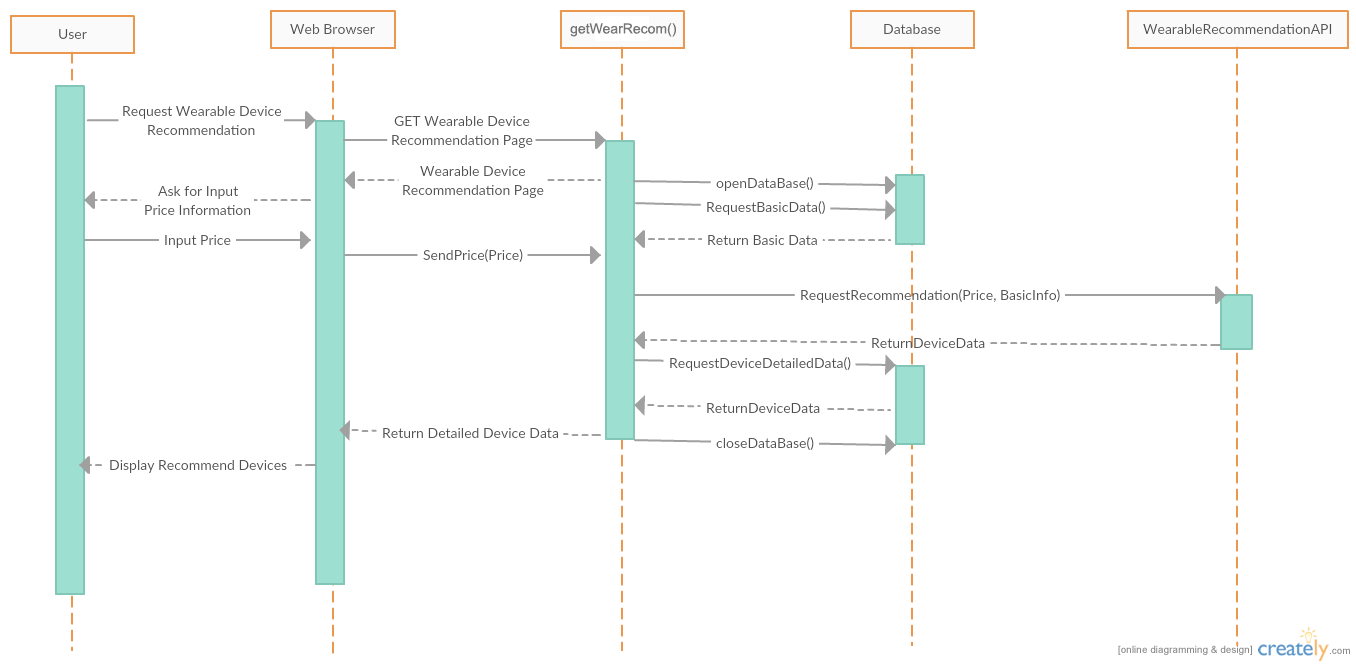
UC-8 Customize own diet plan

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

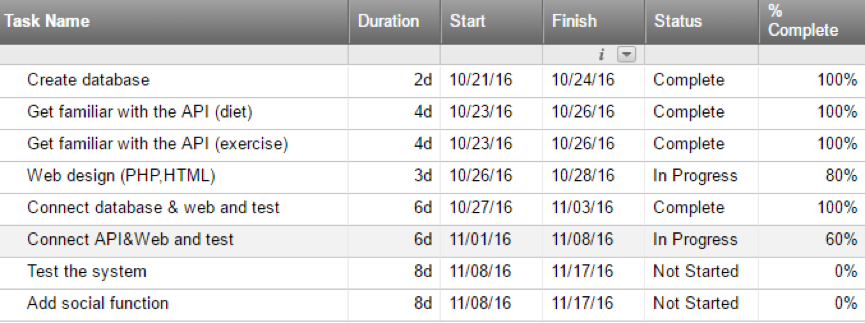
When the user enter into the friends page, which is the start of the System, the ViewFriend class would request Friends List from Database, then it would send the data to web browser to display. Then, the user can choose one friend, the web browser will use the link of the friend and send it to ViewFriend 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.



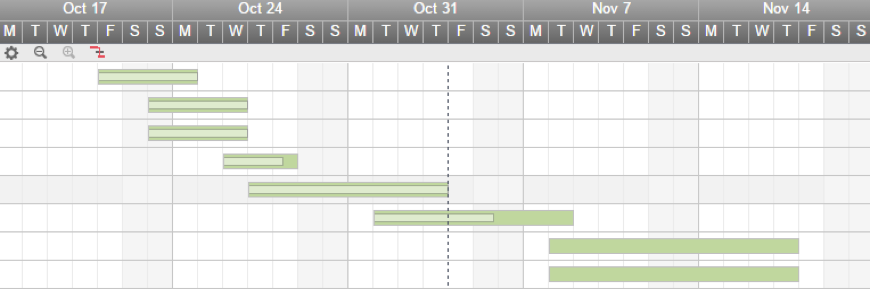
UC-11 Get a recommended wearable equipment

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.

## 2. Project Management



Project Breakdown



Project Schedule

**Since the project have not been finished yet and the workload will be adjusted accordingly, we cannot include our distribution matrix in this version of report now. But we have been trying to make sure every one of our group members contribute equally.**

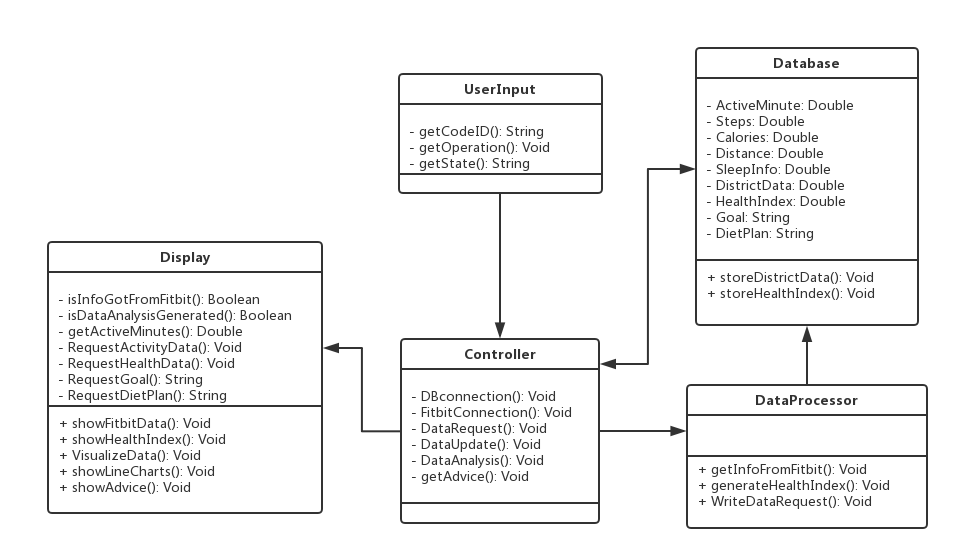
Our project basically followed the plan we made. We installed the MySql to store the information and connected it with the web through php. At the meanwhile, we found the 2 APIs to use. One is the API to give suggestions about the diet based on the input information about the users, the other one is to give suggestions about the exercises to the users based on input information. Implementing these 2 API tend to be harder than we expected since not many APIs are open source, we either need to contact the owner of the API or to find some other API.

Since we have taken the Software Engineering 2 in the last semester, connecting the database and web is not difficult since we could use our experience in SE2, programing in php, MySql and html is almost completed, the only thing left is some appearance adjustments which could be done in near future. Compared with connection between web and database, the connection between web and API is much more difficult, the API we found is in C++ or Java, we need to convert it into php. Thus, there is still plenty of work to be done. Fortunately, we still have couple of days before the deadline we scheduled which is 11/08. After this process is finished, the majority of our project is completed and the remaining time could be used to test and improve our system. Some further functions might be added if possible.

# Part 2

## 1. Class Diagram and Interface Specification

### A. 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.

## 2. 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.

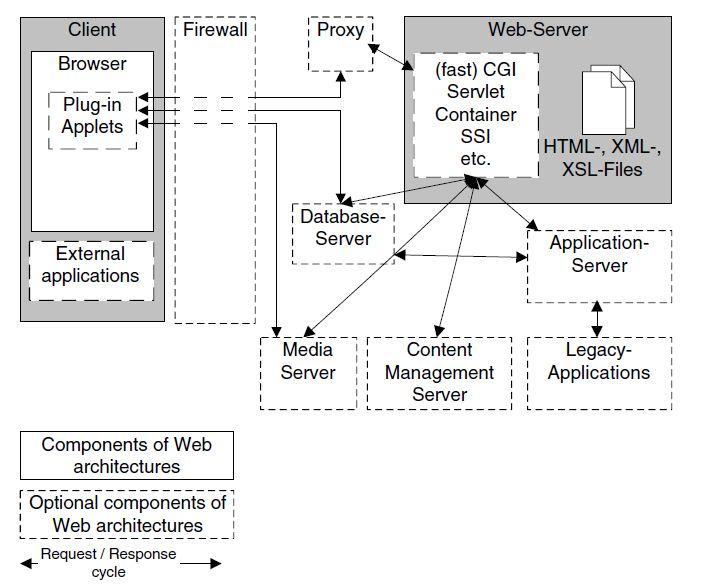
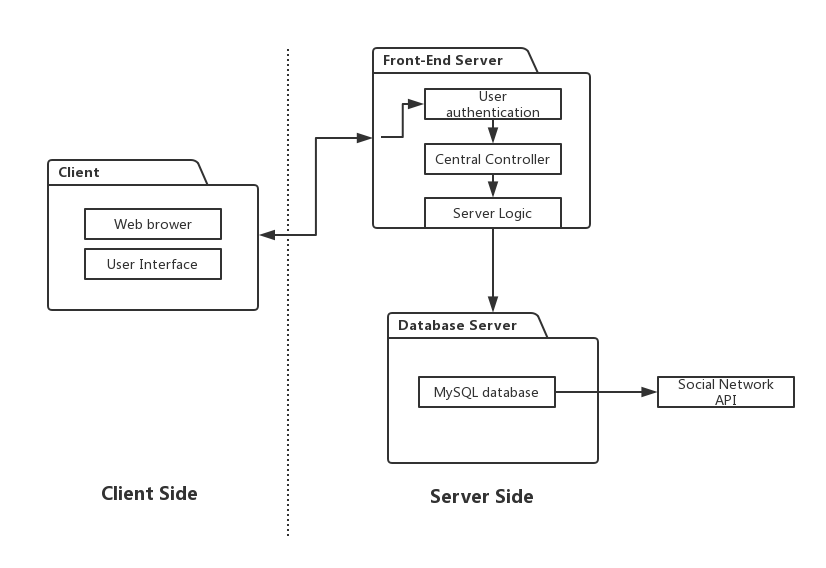


Figure 1 Client-Server Architecture Model

**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:

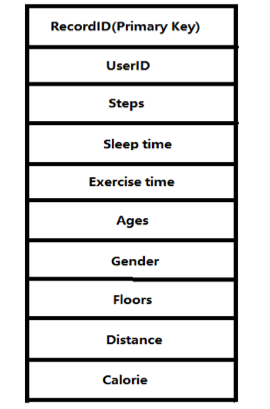


Figure 3 Health Data

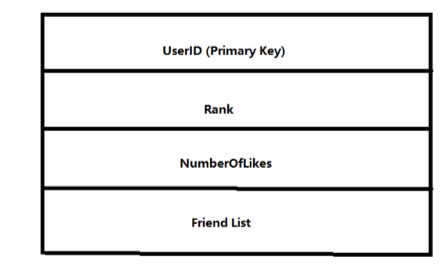


Figure 4 CustomerDietPlanData

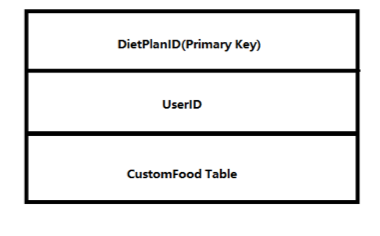


Figure 5 SocialData

**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.

# Part 3:

## 1.Algorithms and Data Structures

## a.Naive Bayesian

We use Naive Bayesian for diet recommendation. Naive Bayesian is an algorithm for classification, it can be modeled as:

Vmap=arg max P( Vj | a1,a2...an)

Where Vj belongs to V. Vmap is a predicted value based on the training example. A1,a2...an is the characteristic in the example. In our project, this model means: first we train a set of data to serve as an example, the characteristic in such example has like weight, height and so on. Then we compare the input data characteristic with the example to get a biggest value of Vmap, which is the most probable value of the diet plan that we would offer. For example, taking just one characteristic into consideration,if the weight is low, and the result would probably be the recommendation of some food with high calories.

b. SVM (Support Vector Machine)

We use the Nonlinear SVM Regression to do the 3-day prediction. In particular, we use the Polynomial formula which is:

*G*(*x*1,*x*2)=(1+*x*1′*x*2)*p*, where *p* is in the set {2,3,...}.

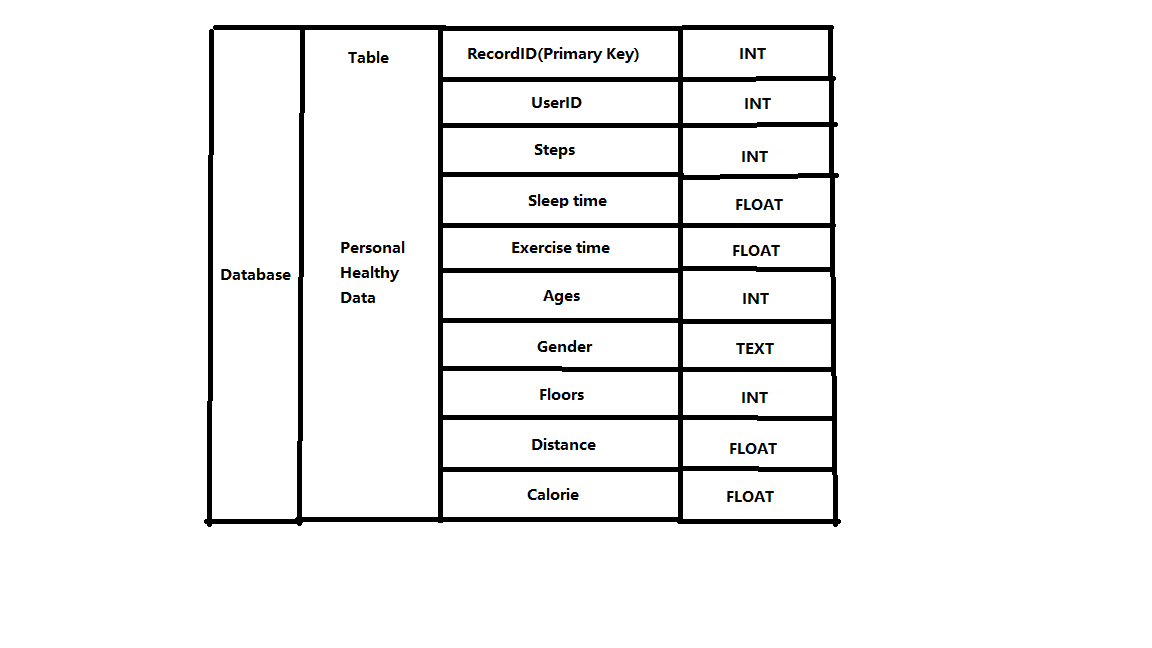
In SVM regression, the input x is first mapped onto a *m*-dimensional feature space using some fixed (nonlinear) mapping, and then a linear model is constructed in this feature space. Using mathematical notation, the linear model (in the feature space) f(x,w) is given by



where g(x),j=1,...,m denotes a set of nonlinear transformations, and *b* is the “bias” term. Often the data are assumed to be zero mean (this can be achieved by preprocessing), so the bias term is dropped.

### 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:



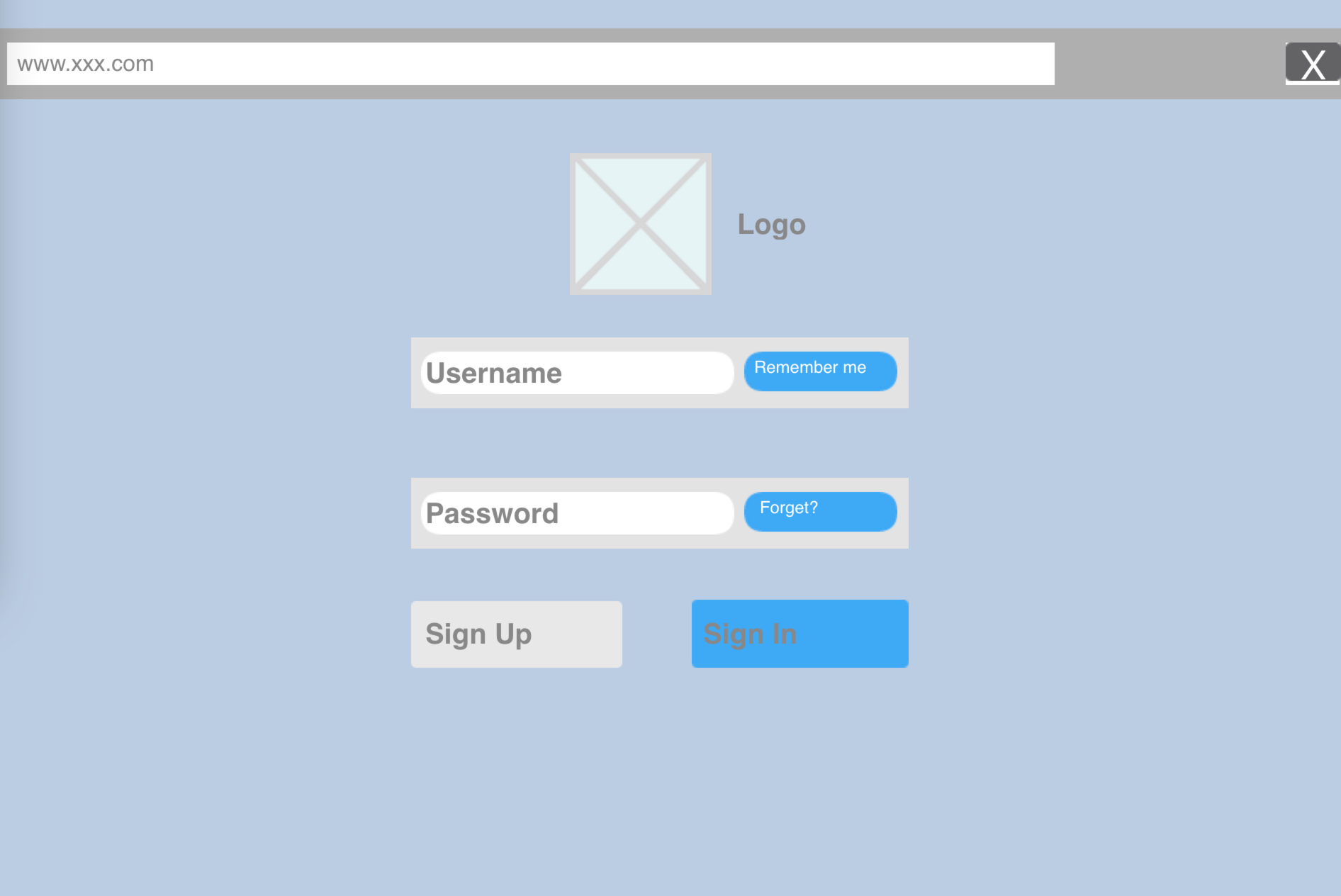
*Data structure of Healthy Data*

## 2. User Interface Design and Implementation

### A. Preliminary Design

1. Welcome page

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.



2. Main UI

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.

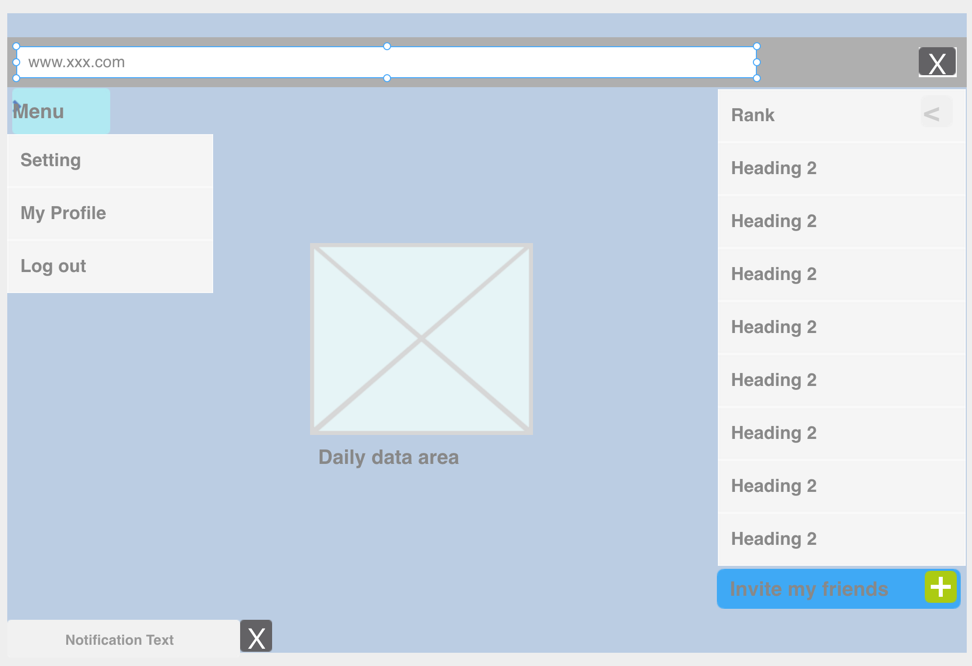
(2)To get the messages or news at the notification text window.

(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.



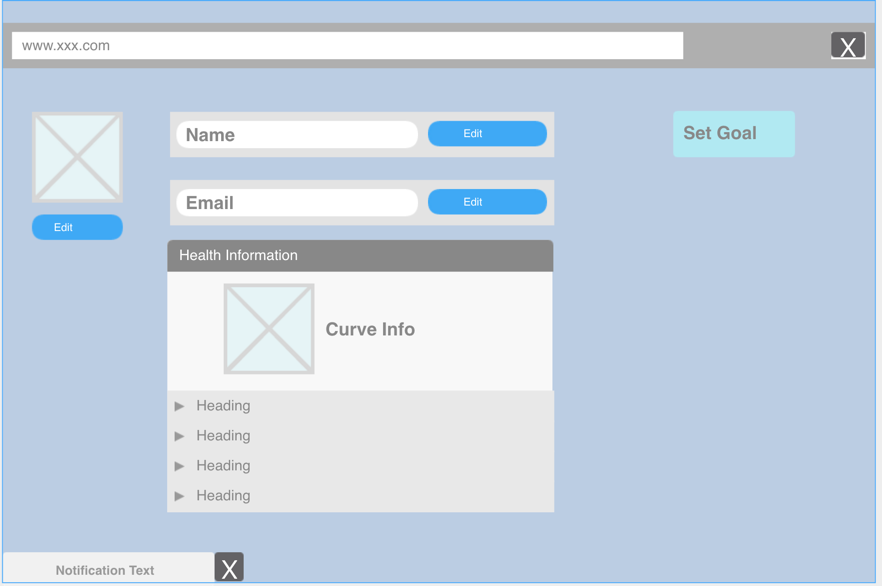
3. Personal profile

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.



4. Data Analysis

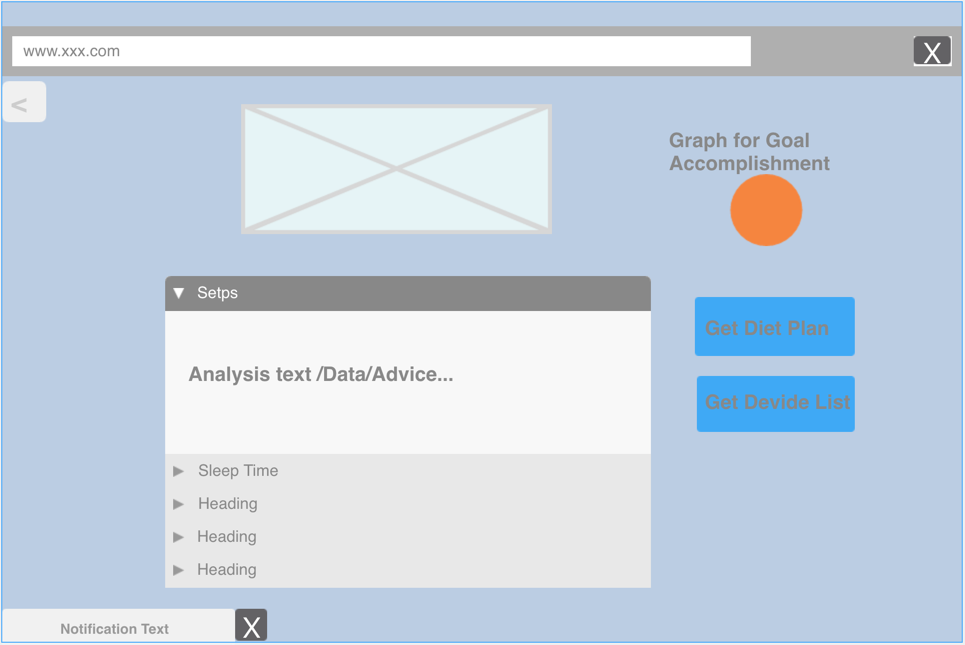
This page provides the specific advice from the data analysts for every part of the daily data of every user.

(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 .

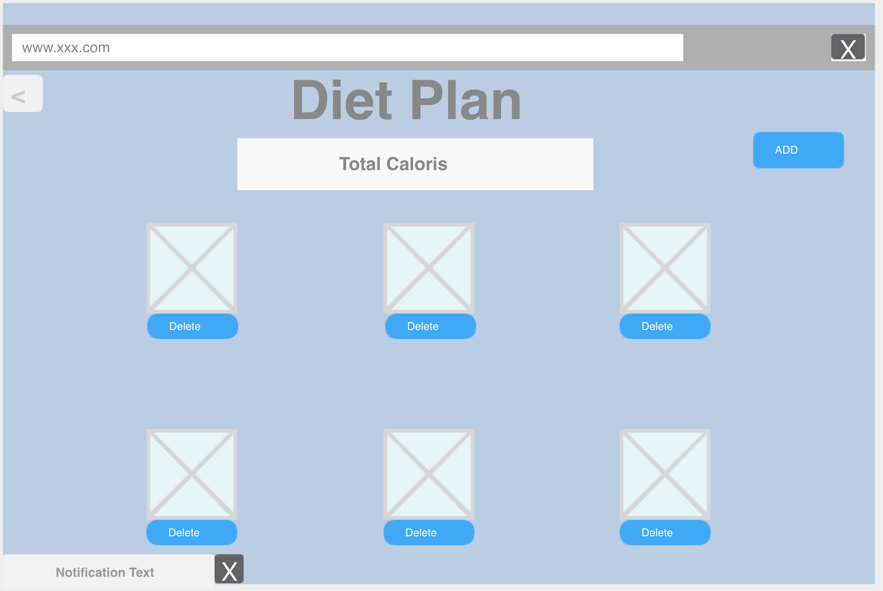
(3)User can edit his/her custom diet plan by clicking the Get Diet Plan.

(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.



5. Custom diet plan

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



### B. 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.

## 3. 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. |

## 4. References

1. For MySQL Database: http://php.net/manual/en/book.mysql.php
2. Especially for MySQL Handler: https://www.tutorialspoint.com/mysql/
3. For DietRecom() API: http://www.programmableweb.com/category/food/apis?category=20048
4. For MySQL Management: https://sourceforge.net/projects/phpmyadmin/
5. For Front-end Techniques: http://www.w3school.com.cn/html5/
6. For Front-end APIs: http://www.webdesignerdepot.com/2011/07/40-useful-apis-for-web-designers-and-developers/
7. For UI Design: http://www.gregreda.com/2015/02/15/web-scraping-finding-the-api/
8. For XML Reference: http://itransact.com/support/toolkit/xml-connection/api/