Health Monitoring Analytics

Prepared for： Personal Health Monitoring

Prepared by: Yuanxi Li, Xiaoran Fan, Lun Li, Jingsong Yuan, Tian Xie

Dec. 13, 2014

Contents

0. a. Individual Contribution Breakdown

Task	Yuanxi	Tian	Jingsong	Lun	Xiaoran	Total
Summary Of Changes (5points)	20%	20%	20%	20%	20%	100%
Sec.1: Customer Statement of Requirements (6 points)	20%	20%	20%	20%	20%	100%
Sec.2 Glossary of Terms (4 points)	20%	20%	20%	20%	20%	100%
Sec.3 System Requirements (6 points)	20%	20%	20%	20%	20%	100%
Sec.4 Functional Requirements Specification (30 points)	20%	20%	20%	20%	20%	100%
Sec.5 Effort Estimation (4 points)	20%	20%	20%	20%	20%	100%
Sec.6 Domain Analysis (25 points)	20%	20%	20%	20%	20%	100%
Sec.7a: Interaction Diagrams (30 points)	20%	20%	20%	20%	20%	100%
Sec.7b: Design Patterns (10 points)	20%	20%	20%	20%	20%	100%
Sec.8a Class Diagram and Interface Specification (10 points)	20%	20%	20%	20%	20%	100%
Sec.8b OCL Contract Specification (10 points)	20%	20%	20%	20%	20%	100%
Sec.9: System Architecture and System Design (15 points)	20%	20%	20%	20%	20%	100%
Sec.10: Algorithms and Data Structures (4 points)	20%	20%	20%	20%	20%	100%
Sec.11: User Interface Design and Implementation (11 points)	20%	20%	20%	20%	20%	100%
Sec.12: Design of Tests (12 points)	20%	20%	20%	20%	20%	100%
Sec.13: History of Work, Current Status and Future Work (5 points)	20%	20%	20%	20%	20%	100%
Sec.14: References (5 points)	20%	20%	20%	20%	20%	100%
PROJECT MANAGEMENT(17point	20%	20%	20%	20%	20%	100%
TOTALS (points)	40	40	40	40	40	200

Table 1: Contribution Breakdown Table

0.b. Summary of changes

We tried to give the user a questionnaire and collect the data the user inserted, and then we generate health index. But later we realized that we require the user to insert too much information, which will ruin the user interface experience. So we decided to change our main function in order to improve the user experience. And the most significant change is that we retrieve user's data from Fitbit instead of user input the data. And we also removed the user log in function because we don't need that anymore. And here is the changes listed below:

1. In the Customer Statement of Requirements session in report 1, we changed some of the glossary of term since we changed the main function.
2. In the System Requirements session in report 1, we changed most of the content. We removed the User Login and Register and all the functions for registered users, and put Fitbit function into this session. And the on-screen requirements, we use the latest screenshot of webpages.
3. In the Functional Requirements Specification session in report 1, because of the change of main functions, so we changed most of the Use Cases and System Sequence Diagram.
4. In the User Interface Specification session in report 1, we decide to use the latest screenshot of webpages of our project instead of the old ones. And the effort estimation and description are based on the new project.
5. In the Domain Analysis session in report 1, we also changed most of the domain models and concepts.
6. In the Interaction Diagram session in report 2, we made a big change because the structure of our project is changed.
7. In the Class Diagram and Interface Specification session in report 2, we changed the most part based on the domain analysis and added OCL Contract.
8. In the System Architecture and System Design session in report 2, almost all the things are changed based on the class diagrams.
9. In the Algorithm and Data Structures session in report 2, we hardly changed anything because our main algorithm is still FA. And we use that to calculate the Health Index.
10. In the Interface Design and Implementation session in report 2, we still use the screen shots of our latest website, and add some descriptions to it.
11. In the Design of Tests session in report 2, we changed the most of it based on the contents before.

1.Customer Statement of Requirements (CSR)

a. Problem Statement

Health is the most significant part in our life. As a Chinese proverb said "health is pre-condition of all work" and without a healthy body, we can't do anything we desire. With the development of science and technology, our lifestyles become more wonderful and colorful in both physical and spiritual levels. Undoubtedly, the categories of exercise are becoming more and more abundant under this circumstance and people are more accessible to these exercises. Some kinds of exercise like physical fitness, swimming, jogging and hiking gain a huge popularity over the world.

It is our belief that the most important factors to a happy and healthy life is physical exercise and a balanced diet. We can work our fingers to the bone and party to our limits but it isn't possible to live the most balanced life without first balancing our health. Some of you out there struggle with a variety of conditions related to your health: type 2 diabetes, high blood pressure, high cholesterol, et cetera. It would be much easier to manage these illnesses or even eradicate some of them if you could work up the will to exercise. It is often too late for many of you to realize that there's something wrong until your body hits a point of peak physical distress; communicating through you with the use of pain. **Pain is an undesired signal because:**

1.You do not enjoy pain.
2.It arises in moments of severe distress; i.e. when something has gone wrong enough to the point of medical attention.
3.It often comes too late when you'd want to anticipate the problems pain is alerting you to in advance.

Without enough physical exercise, we are more likely to expose to severe diseases. However, modern people can't keep the pace of the social development, which leaves them very unhealthy lifestyle. In America, people suffer from a high rate of diabetes and hypertensions because of their unhealthy daily habits. For example, staffs in many companies spend a whole day in front of their desks and eat food with high calories for their meals. They won't even move around unless they have something emergency. But this is just one reason for adults especially those who have work. For others, some may have pains or injuries in the past and lack of mobility, some may complain because the bad weather condition keeps them from doing outside activities, some may not have adequate motivation and think it has less reward.

We here at our organization understand the various problems related with exercise. **In a poll of our customers, the most likely reasons why you don't exercise can be found on this list:**

1.Haven't developed the habits
2.No motivation
3.Too overwhelming
4.Poor diet
5.Current physical condition
6.No access
7.Lack of results

8.No time No energy
9.Competing interests

Many developing and developed countries have launched some policies that all the citizens especially those in schools should take some time (at least half an hour) everyday to do physical exercise to strength their bodies, which help them better concentrate on their work. The city governors also give residents more space for physical exercise by exploiting free public sporting fields and green roads (for cycling and running).

In real life, physical exercise has lots of positive effects. They can
·maintain our fitness and body weight
·build our bone structure, muscle strength and mobility
·relax ourselves and get rid of the daily pressure
·improve our immune system and reduce the risk of diseases
·help prevent depression and promote our self-esteem

As a result, users might want to monitor their health condition and get some advice on his or her health condition via our software.
We conclude what features our users might want to have in our software as follow:

0. A user want to know if his or her lifestyle such as diet or exercise or sleeping hours are healthy or not.
1. A user want to monitor his or her health condition all the time.
2. A user want to know if he or she is healthy or not.
3. A user want to know how to improve his or her health condition if he or she is currently not that healthy.
4. A user want to know how to maintain his or her health condition if he or she is currently healthy.
5. A user want to know if he or she continues his or her life habit then what health condition will be in the near future.
6. A user want to know if he or she follow our advice then what health condition will be in the future.
7. A user want to know the health condition of a certain area. (The average and the optimal level)
8. A user wants to feedback his or her ideals or bugs to the software developer or the maintainer.
9. A user want to contact with other users who using our software.
10. A software maintainer want to updates the database and modify the UI of the software.
11. A software maintainer wants to get feedbacks from his or her users.

So how can we work together, you, the consumer, and us, the company, to break down these barriers to entry for a healthier life? We have heard and understand your complaints, and we'll take this opportunity now to explain how we think we can help you get in shape!

The operating principle is this: man is a social creature; it does not do well on its own and needs community support to survive and thrive. This logic can be applied to you, the consumer, as well; you are more likely to perform a task if you know you will have a compatriot "along for the ride". As such, we know you can be motivated to exercise by seeing people you know working-out (and subsequently joining them) or by meeting more people in your community who exercise as well,

transforming a previously solo venture into a social activity.

To deal with this issue, I think we should build a website can get users' health condition from other sites such as FitBit . In addition, we should also build a mobile application so people can immediately get the data of his exercise and know well about their health conditions.

b. Background on Health Monitor Analysis

1. Healthcare Hashtag Project

Figure 1-1 The front page of the Healthcare Hashtag Project

The stated goal of this project is to "... make the use of Twitter more accessible for providers and the healthcare community as a whole." How it works is that users are able to use a search bar to scan the web for relevant hashtag data. The web would then extract data from Twitter and display it in an analyzed, digestible form for the user. Users can it to search for specific topics in natural language or by a specific hashtag. People can find where the healthcare conversations are taking place, discover who to follow within your specialty or disease, and find the best from conferences in real-time or in archive. Unfortunately, this is more focused on an academic/research type of community. It does not leverage the power of twitter to focus on the individual; it is used to globalize and clarify health care specific topics (diseases, et al).

2. HMS Health Monitoring Systems

The EpiCenter system is capable of analyzing healthcare data for the purpose of detecting anomalies suggestive of public health threats, such as disease outbreaks and bioterrorism. Users can find reported data based on the location, settings and other options. However, the system is not able to reflect the actual condition of the community, which we know is changing in real time, and is not specific; this is a macro scale application of analyzing the health of a community through viral outbreaks and is meant for hospitals and not individuals.

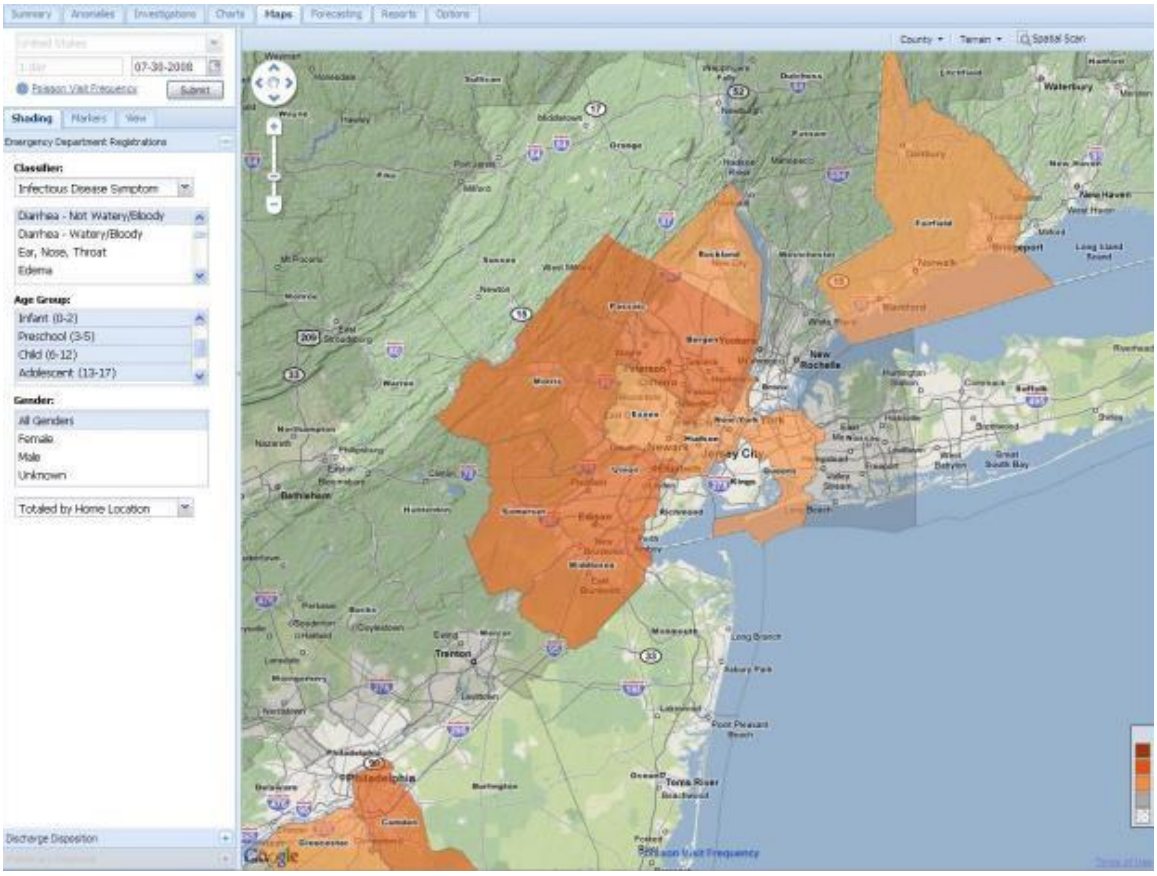


Figure 1-2 Heat Map

These are the only two products we could find being offered to consumers and neither of them, even combined, fully implement what we plan to do here. So we can conclude, with some aplomb, that this approach is highly innovative.

2. Glossary of Terms

Dashboard: Dashboard is name for progress report user health condition. Dashboard is displayed on a web page that is linked to a database which allows the report to be instantly updated by user's request.

Gauge: A gauge can show user's health Index range which requested by user.

Chart Line: A chart line that illustrated the health condition or health index of the user. Chart line can show user's monthly or yearly health condition directly.

Standard Deviation：In statistics and probability theory, the standard deviation (SD) measures the amount of variation or dispersion from the average.
Database: Data about the user's attributes maintained as a structured set

FA(Factor Analysis):We will research the inner relationships between variables such as people's working hours or the food preferences or their exercise types&times we collected to find out the basic structure of those data. Then we assume a few(less than what we collected) new variables to

show the basic data structure. Those assumed variables could show the majority information of variables we collected before. Those assumed variables can not be observed or collected so they are called Factors. The Factor Analysis is also a method of reduce dimensionality.

Health Index: a comprehensive index that indicating an individual's health. The outcome comes from many factors such as the sleeping hours or working hours or comparison with optimal health condition index.

Health Advice: Some number and statement that we will give our registered users after comparing his or her health index with the optimal health index.

3. System Requirements

a. Enumerated Functional Requirements

ID	Priority weight	Requirement
REQ-1a	5	System should retrieve data from Fitbit
REQ-1b	4	System should retrieve data from database
REQ-2	4	System should store data into database
REQ-3	5	System should display distribution via graphs and charts
REQ-4	5	System should give the index of the user
REQ-5	4	System should give the rank of index of different users
REQ-6	4	System should run the algorithm executable file
REQ-7	3	System should have a instruction to tell the user how to open the access to the Fitbit
REQ-8	2	System should let the user to contact with the developer
REQ-9	3	System should give the statistics of a certain region
REQ-10	2	System should let the user search the area
REQ-11	3	System should give advice to the user about daily activities

Table 3-1 Functional Requirements

b. Enumerated Nonfunctional Requirements

Identifier	**Requirement**	**PW**
REQ-12	The system should require minimum maintenance, at most once per week.	5
REQ-13	The system should remain functioning in the event of an update to Fitbit API.	4
REQ-14	The software shall present the graph and words in a neat and tidy website.	3

REQ-15	The system shall have high security. For example, only administrator could get access into Web Manage Page. And Users information will not be divulged.	3
REQ-16	The system shall respond rapidly to the users' operation. For example, users don't need to wait for too much time after clicking or selecting a button.	2
REQ-17	The system or server shall be easy to recover when they are broken.	3
REQ-18	The system shall allow many people to use at the same time. It will not be crashed just because of overloaded.	2

Table 3-2 Non-functional Requirements

c. On-screen Appearance Requirement

1. Welcome page
REQ-19 Welcome page
This is sort of the pre-website page. It displays information about what the interior of the website contains. And some buttons would be on the screen somewhere that the user can press to enter the website and access to the functions.

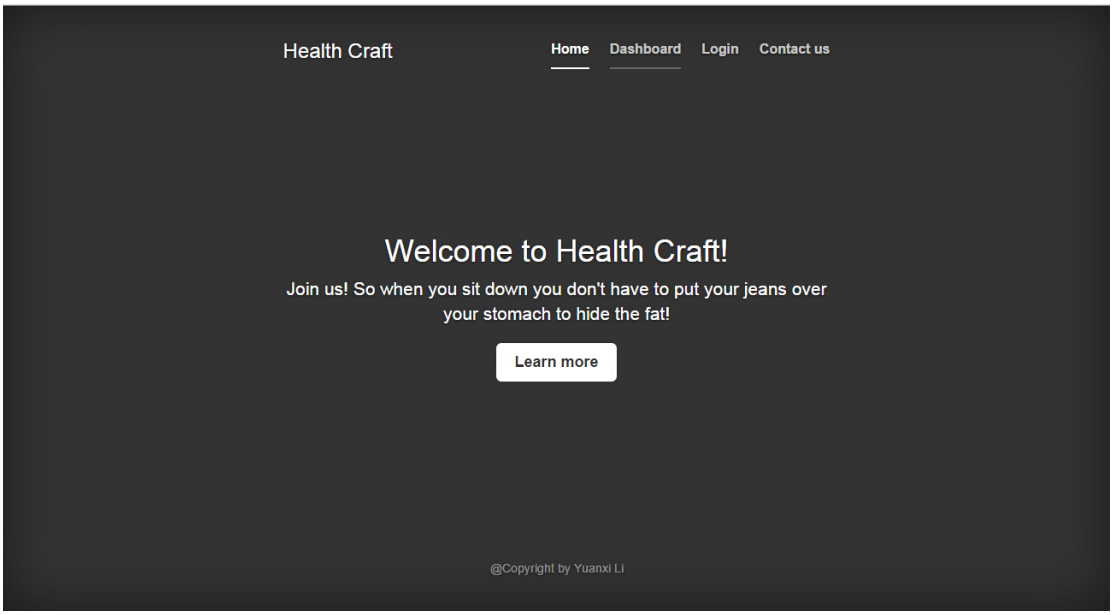


Figure 3-1 Welcome Page

2. Health Index
REQ-20 Health Index
The health index is based on the user's data from Fitbit. And it will tell the user how is the user doing lately. And there are many graphs and charts compared with different data to show a better view.

REQ-20(1) Tutorial

Tutorial of how to open access to the Fitbit if the user doesn't insert the CodeID and open the access.

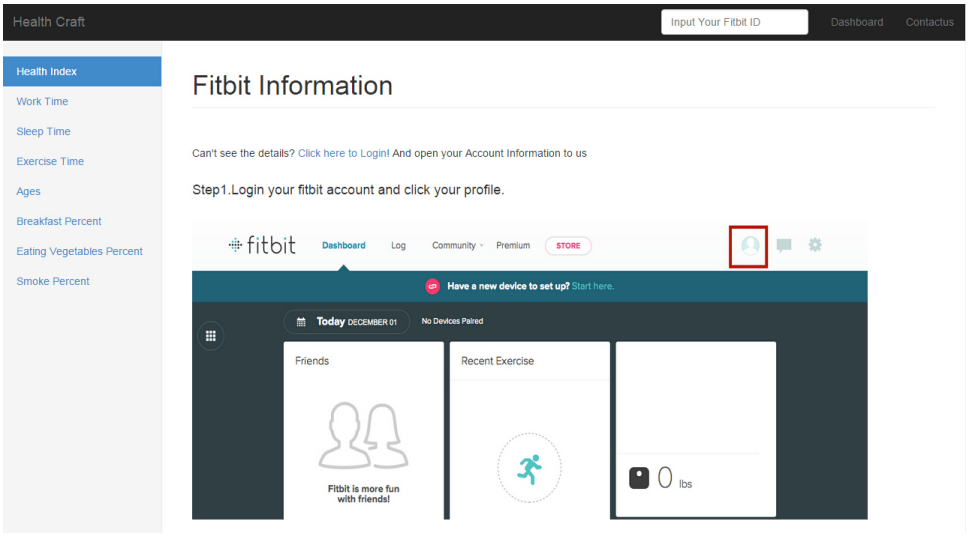


Figure 3-2 Tutorial(1)

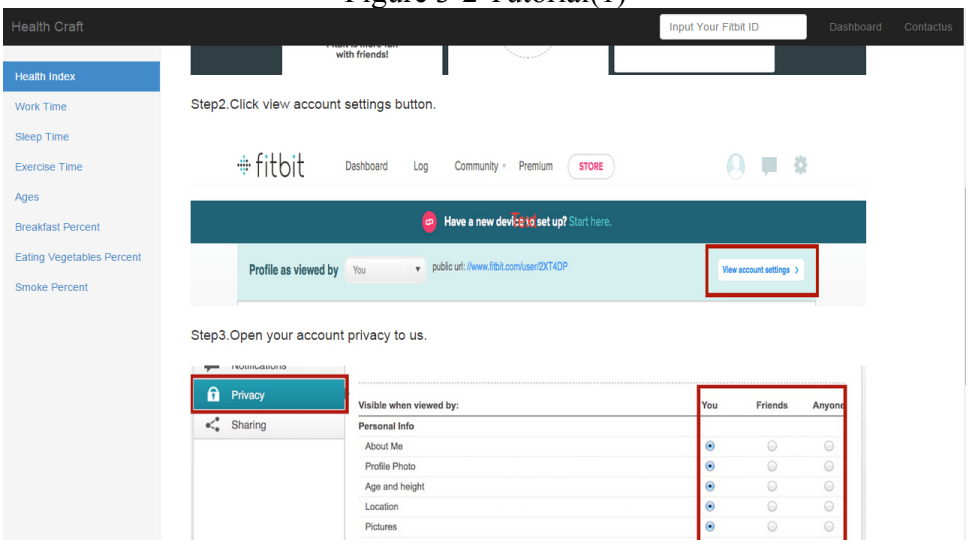


Figure 3-3 Tutorial(2)

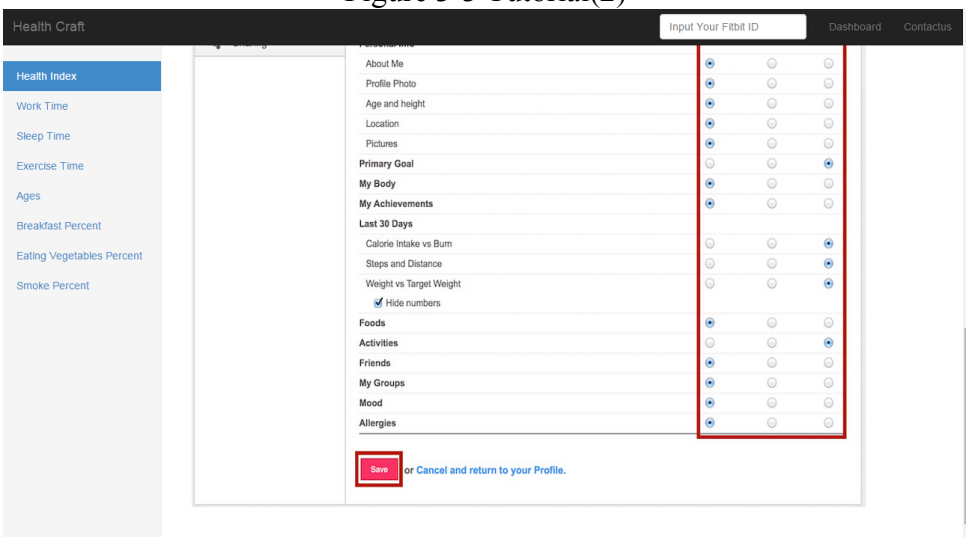


Figure 3-4 Tutorial(3)

REQ-20(2) Data from fitbit

After the user inserted the CodeID and opened the access to the Fitbit, the website can show the Fitbit data in a whole month including “Active minutes”, “Calories”, “Steps”, “Distance”, “Sedentary” and “Sleep”.

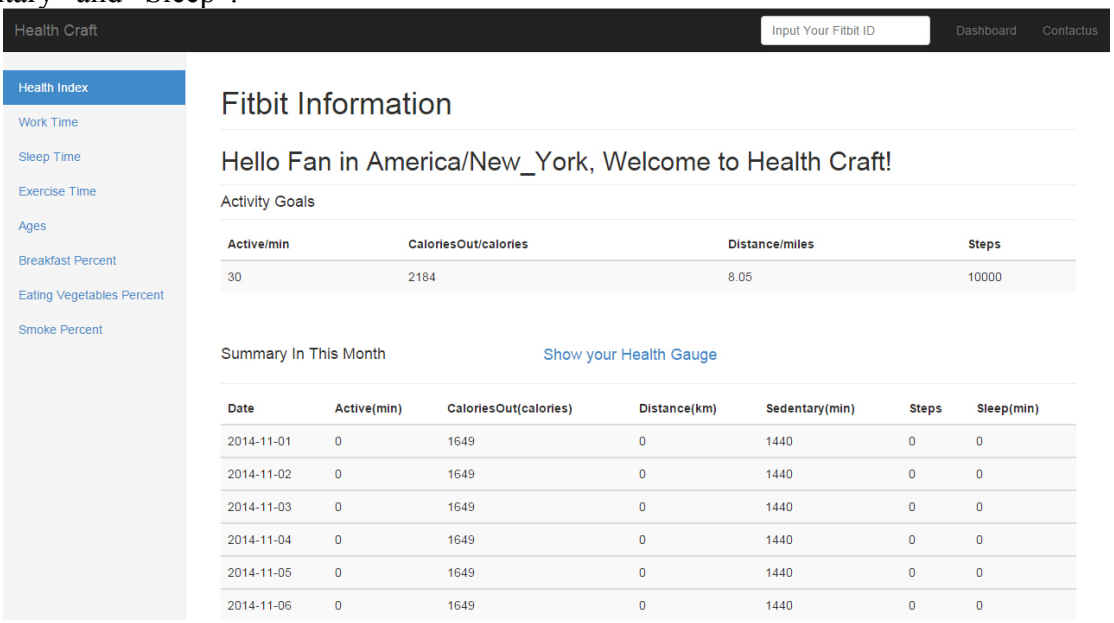


Figure 3-5 Data(1)

REQ-20(3) Gauge

When the user click the “Show your Health Gauge” button, the website will show a page of the gauge of the user’s health index in last month. The pointer will change day by day.

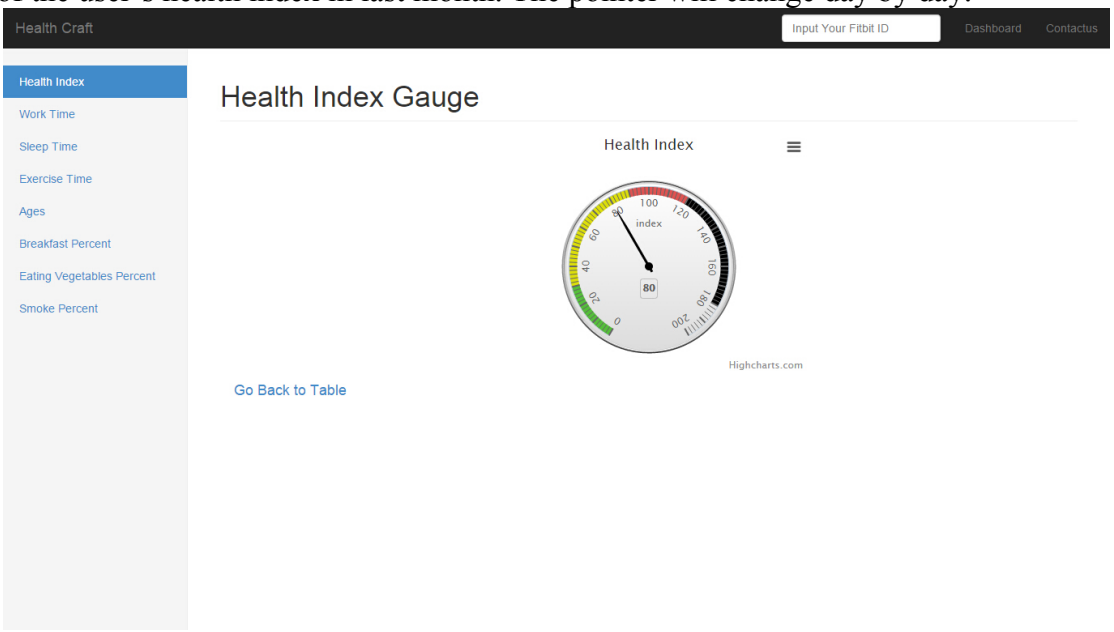


Figure 3-6 Data(2)

REQ-21 Data Graphs and Charts

At the bottom of the screen in Month data, there are four buttons.

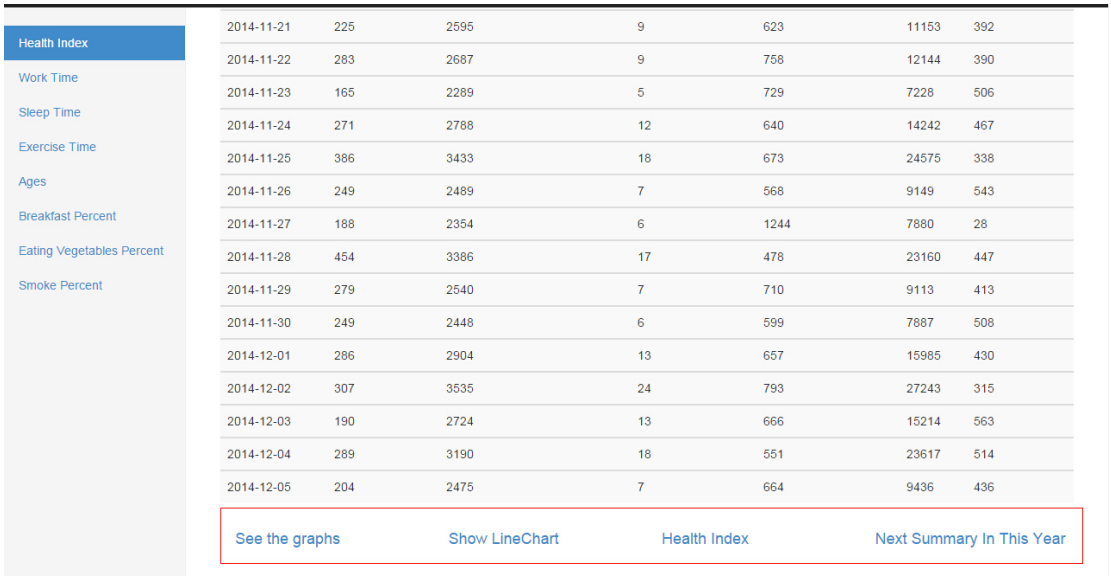


Figure 3-7 Data(3)

REQ-21(1)See the graphs

After the user click this button, then the website will show a page of histogram of all six factors. For example “Calories”. And the user can see other graphs by clicking the “Next Graph” button.

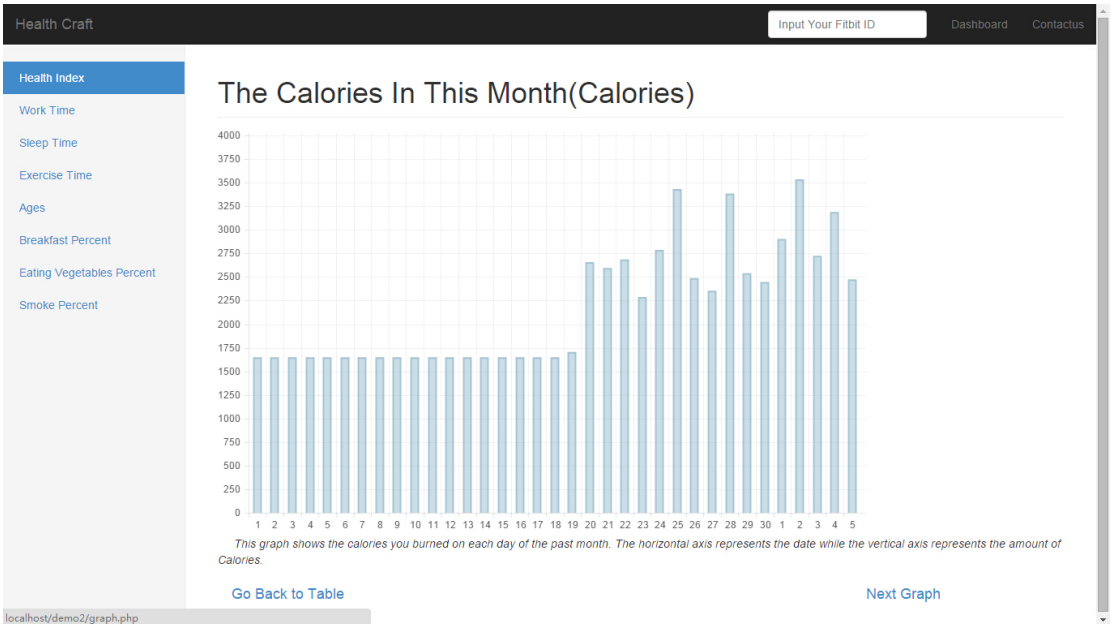


Figure 3-8 Graph(1)

REQ-21(2) Health Index Graph

After the user click the “Health Index” button, the website will show the Health Index of the user in this month.

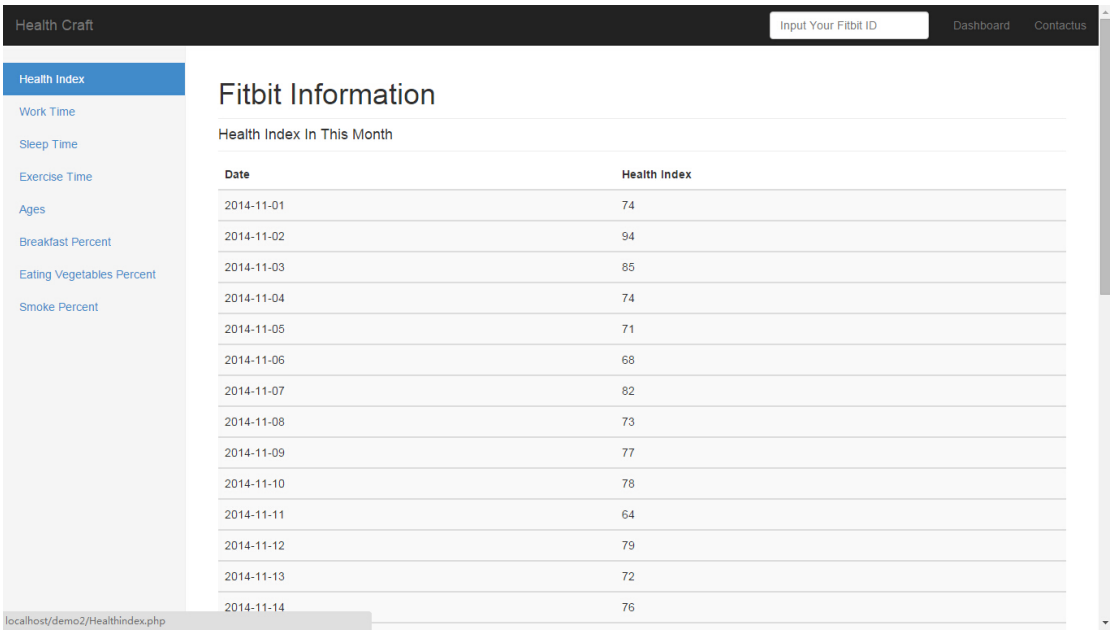


Figure 3-9 Graph(2)

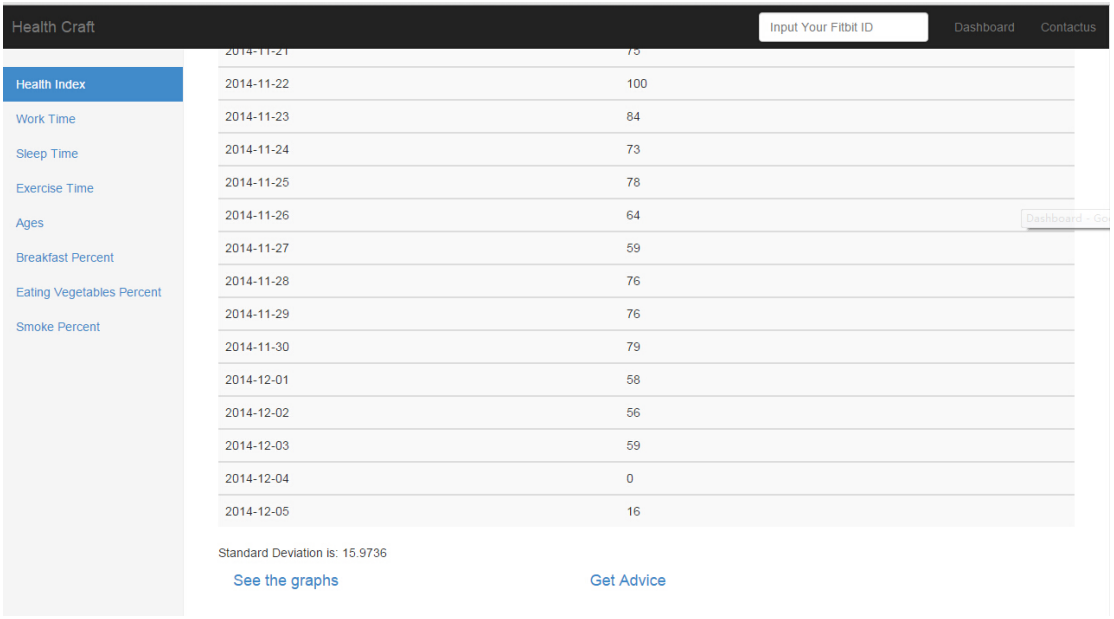


Figure 3-10 Graph(3)

And the user can click the button “See Graphs” below the Index. And the website will show a linechart of the index variation in this month.

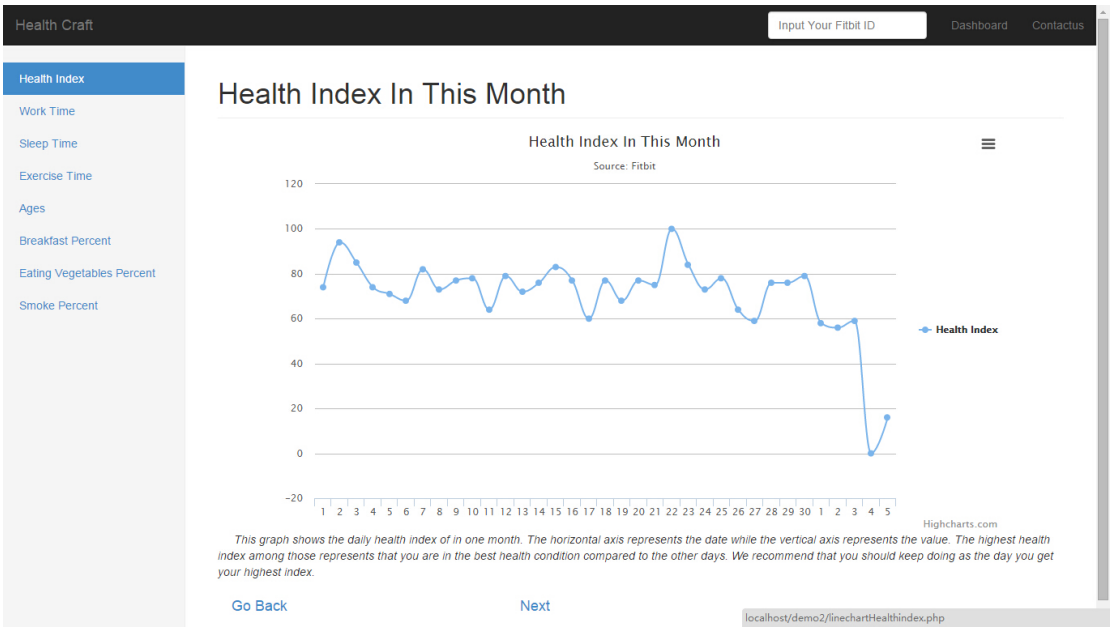


Figure 3-11 Graph(4)

After the user click the button “GetAdvice” below the Index. And the website will show a page of the user’s best activity situation and optimal activity management.

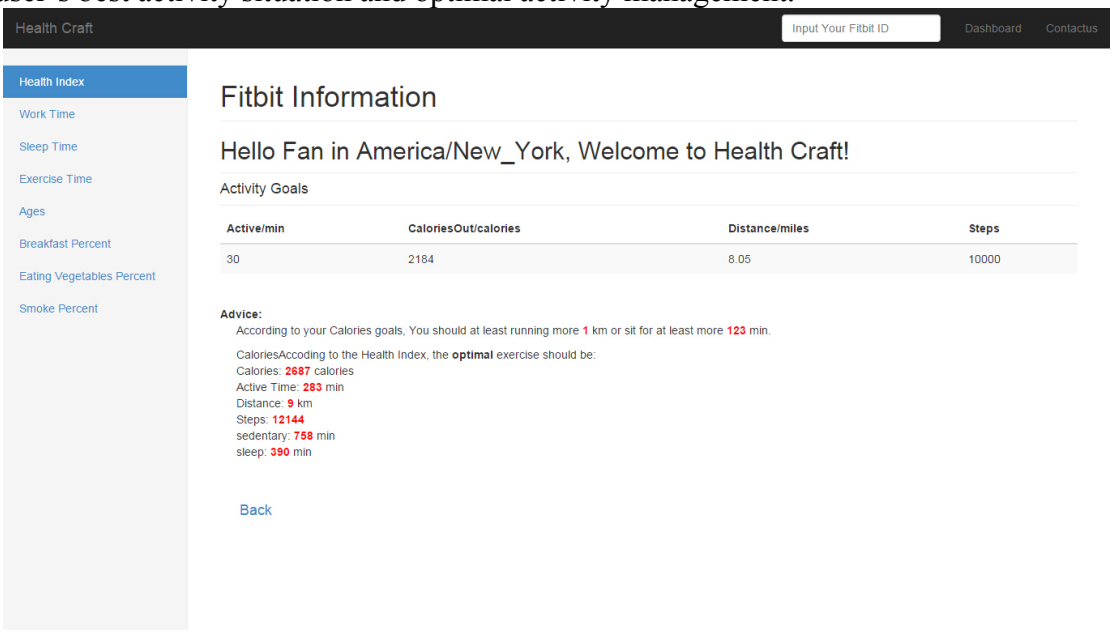


Figure 3-12 Graph(5)

REQ-21(3) LineChart

The Line chart is drawn based on the user’s data and health index. It will clearly reflect the user’s health condition and show the relationship between these factors and the health indexs. And after the user click the “show Linechart” button, the line chart page will show up.

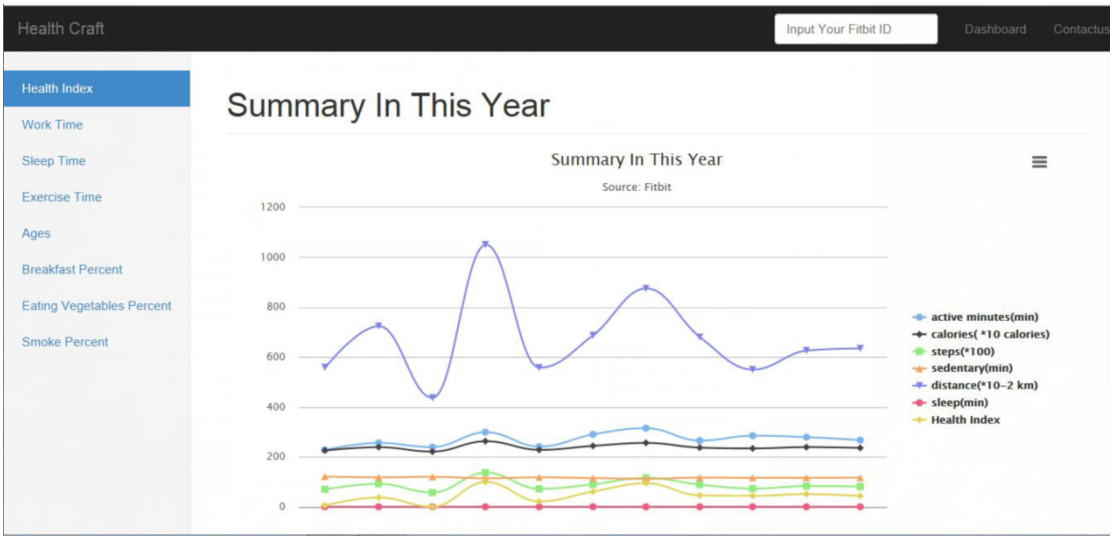


Figure 3-13 Graph(6)

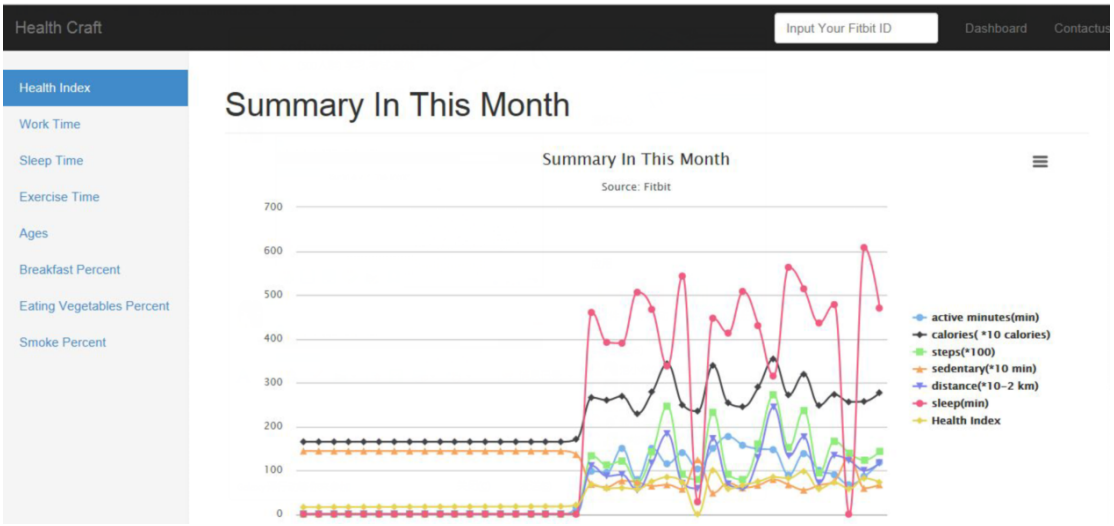


Figure 3-14 Graph(7)

REQ-21(4) Radar chart

The radar chart will show the user in these 5 different factors, which one the user did the best and which one did the worst in the past.

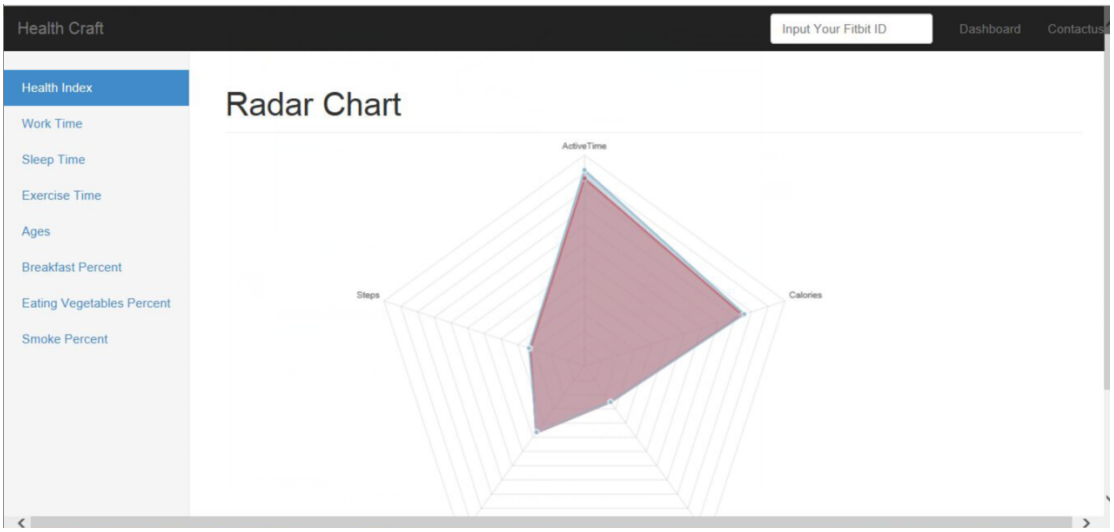


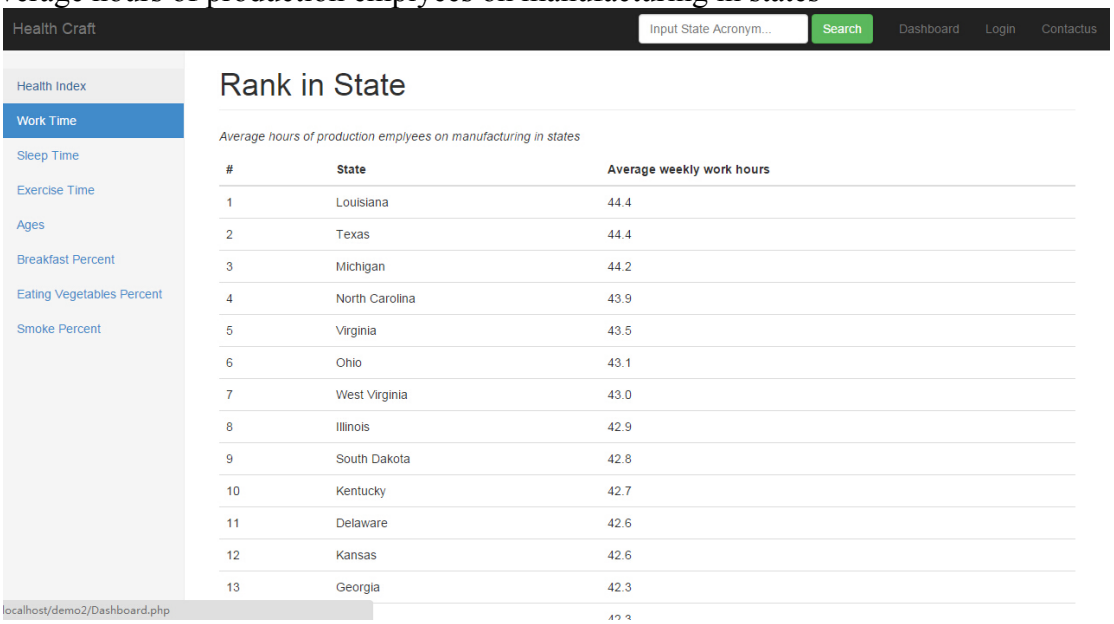
Figure 3-15 Graph(8)

REQ-22 Region statistics

The user can get the statistics in a certain state by choosing a tag on the dashboard

REQ-22(1) Work Time

Average hours of production emplyees on manufacturing in states



#	State	Average weekly work hours
1	Louisiana	44.4
2	Texas	44.4
3	Michigan	44.2
4	North Carolina	43.9
5	Virginia	43.5
6	Ohio	43.1
7	West Virginia	43.0
8	Illinois	42.9
9	South Dakota	42.8
10	Kentucky	42.7
11	Delaware	42.6
12	Kansas	42.6
13	Georgia	42.3

Figure 3-16 Region Statitics(1)

REQ-22(2) Sleep Hour

Age-adjusted percentage of adults who reported insufficient rest or sleep during the preceding 30 days

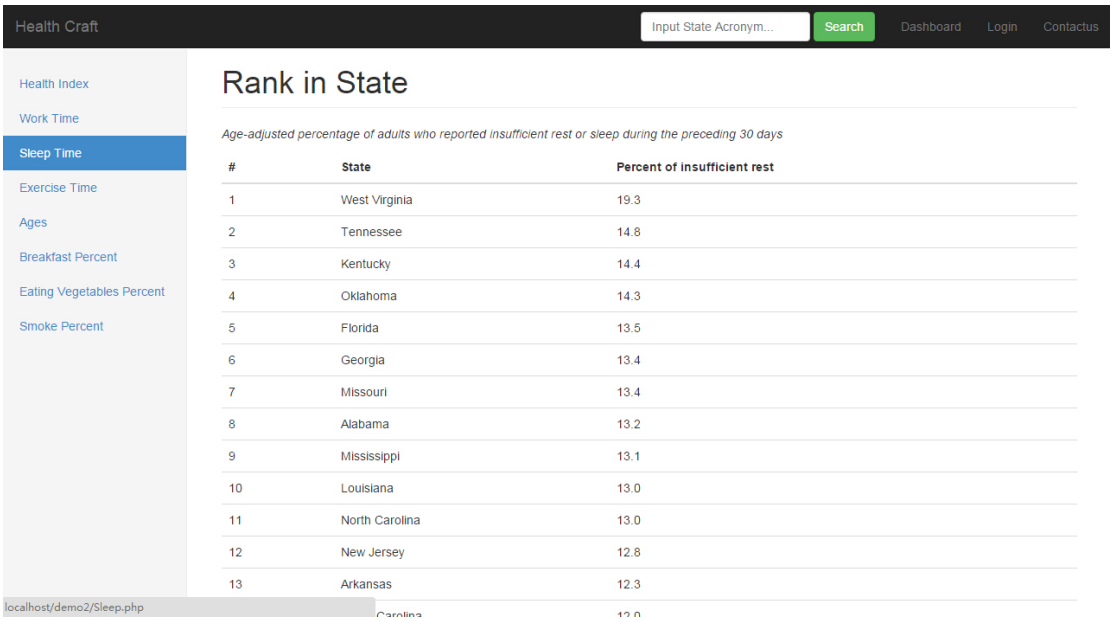


Figure 3-17 Region Statistics (2)

REQ-22(3) Exercise time

The percentage of weekly exercising at least 30 minutes

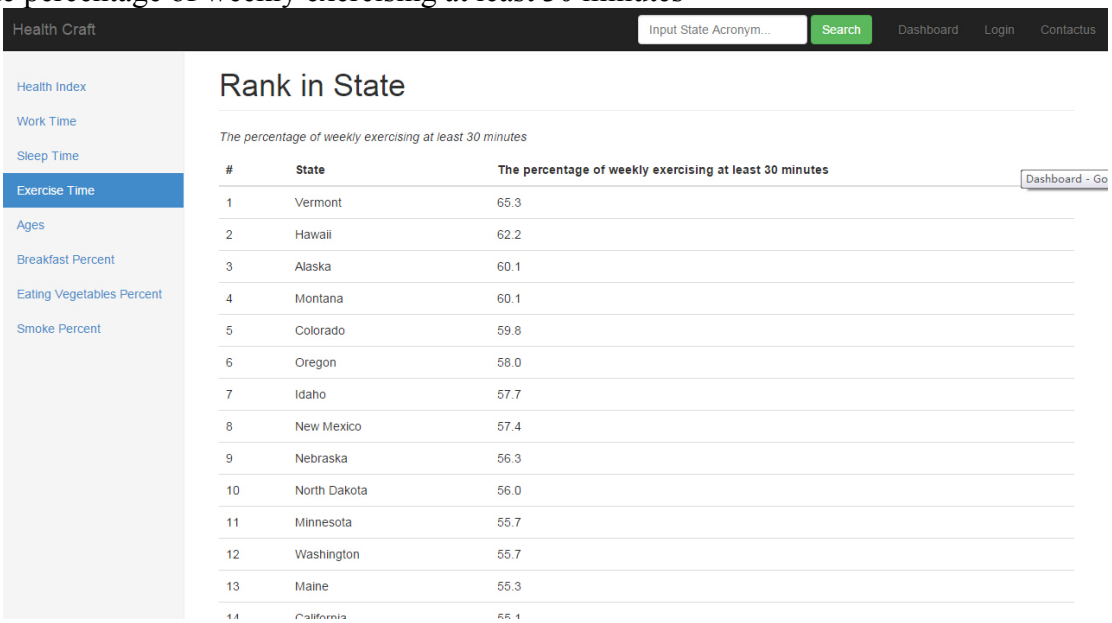


Figure 3-18 Region Statistics (3)

REQ-22(4) Ages

Life expectancy at birth and by race/ethnicity in every state where the population of that racial or ethnic group is sufficiently large for robust estimates.

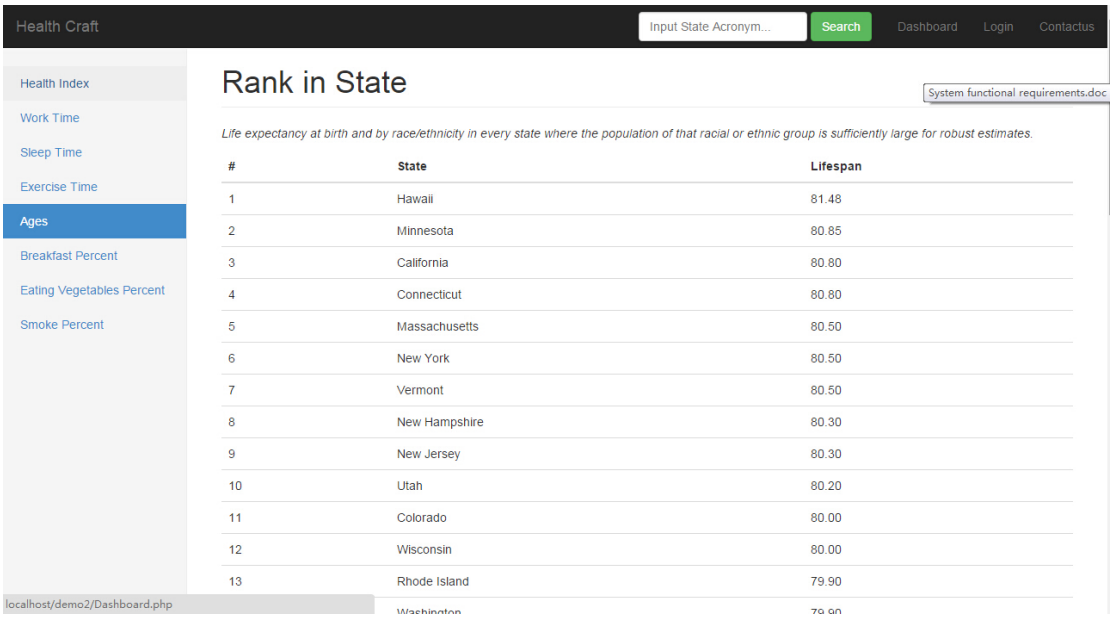


Figure 3-19 Region Statistics (4)

REQ-22(5)Breakfast Percentage

The percentage of adults who eat breakfast

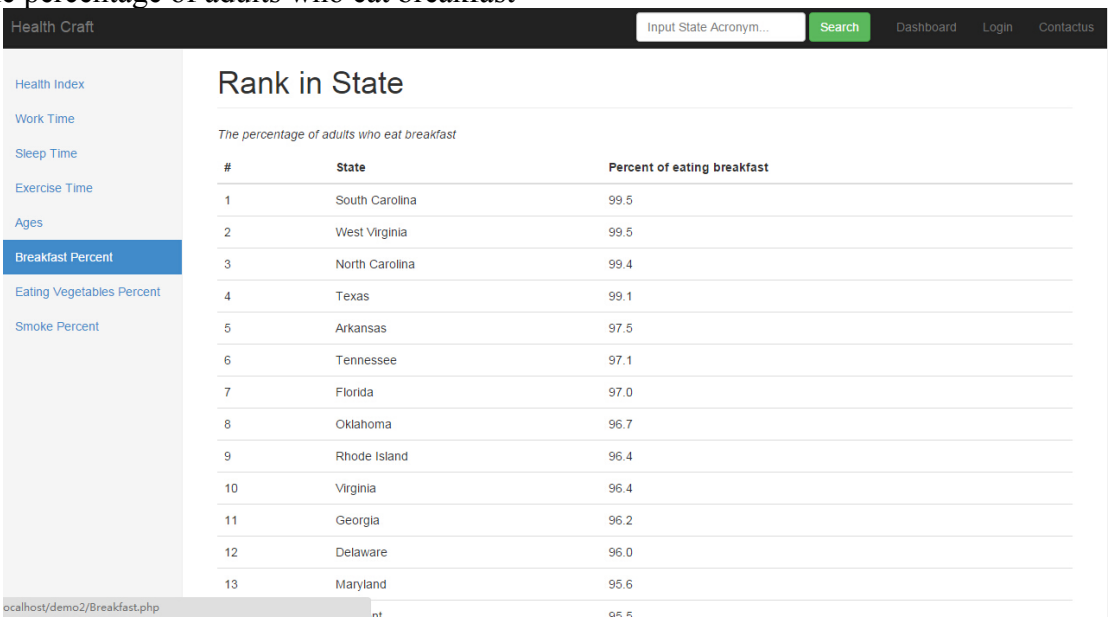


Figure 3-20 Region Statistics (5)

REQ-22(6) Eating Vegetables Percentage

The percentage of adults who eat Vegetables

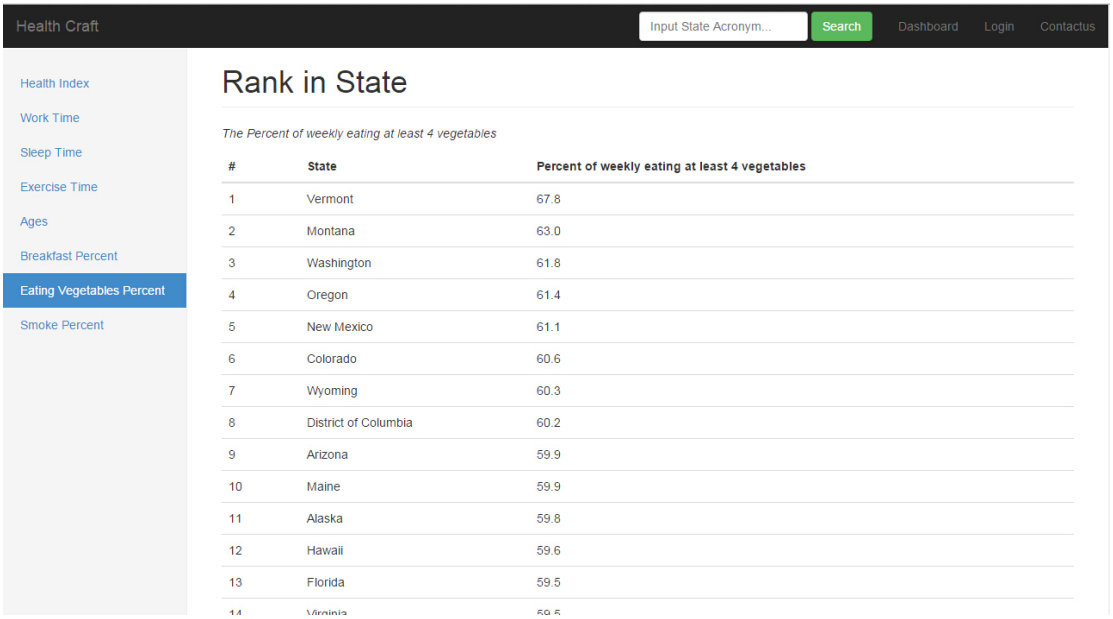


Figure 3-21 Region Statistics (6)

REQ-22(7) Smoke Percentage

Adults who are current smokers

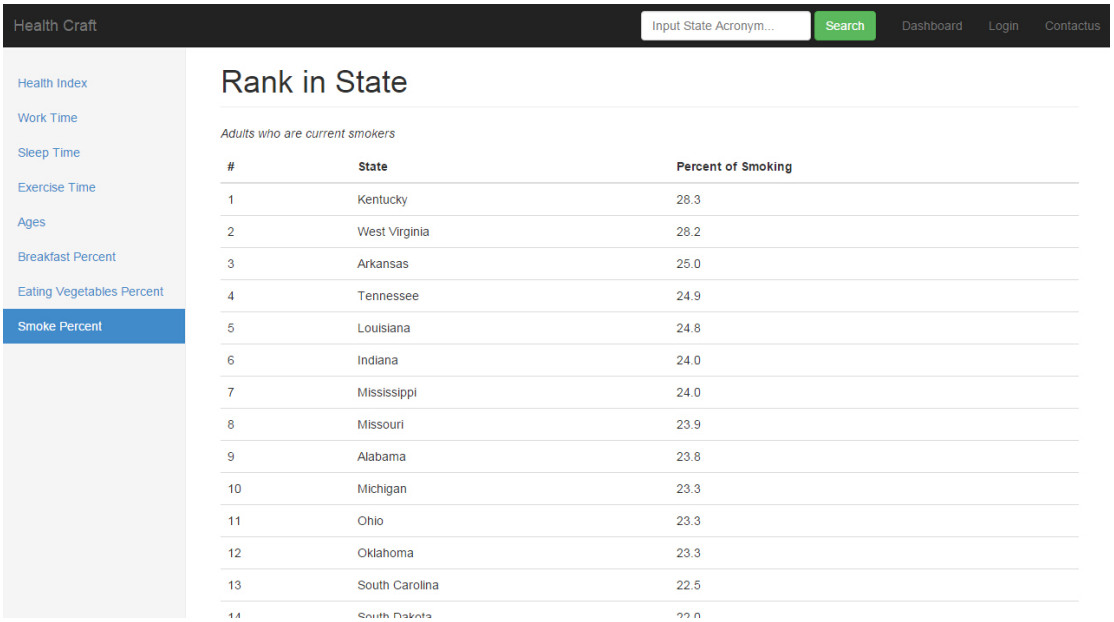


Figure 3-22 Region Statistics (7)

4. Functional Requirements Specification

a. Stakeholders

Governors: They would like to use this software to know the healthy distribution of every state

people in the entire nation, and give some proper advice.

People who want to keep health: They could use this software to know their health condition, and get the easy way to keep healthy.

Food Company and Exercise Equipment Company: They are interested in our Software, because we could promote and advertise their products in giving advice to users.

Academic Researchers: They would like to use these data like working hours, exercise time and people every diet to process their researches.

b. Actors and Goals

Actor	Goals	Use case
Visitor, Database	To look up the health information of a certain state that user insert.	SearchInfo (UC-9)
Visitor, Processor	To view line chart of users health condition according to the fitbit database.	Viewlinechart(UC-3)
Visitor, Processor	To view line chart of standard deviation according to the fitbit database.	GetStandarddeviationlinechart(UC-4)
Visitor, Processor	To view line chart of average health index	Getaveragehealthindexlinechart(UC-5)
Visitor, Processor	To view the health index guage.	Viewhealthindexgauge (UC-6)
Visitor, Processor	To get an approximate evaluation of calories consumed everyday by entering the diet.	CaloriesAnalysis(UC-8)
Visitor, Database	To get an evaluation of personal health condition by entering diet, work, exercise and rest information.	GetHealthIndexDashboard(UC-1)
Visitor, Database	To view users activity condition, like sleep time, active time, calories consuming	ViewActivityConditionDashboard(UC-2)
Visitor, Processor	To get an advice based on the health condition evaluation.	GetAdvice(UC-7)

Table 4-1 Actors and Goals

c. Use Cases

i. Casual Descriptions

UC#1 Getting Health Index Dashboard
The user wants to see his or her health index which is based on monthly activity data and yearly activity data which are analyzed by a scientific mathematic model. The health index will directly show how health the user is. The user will see the fluctuation of his or her health condition and how health he or she is by comparing his or her own monthly health index and other users health index.

UC#2 Viewing Activity Condition Dashboard
Users can view their activity condition per day or average activity condition per month in a dashboard. Our website dashboard provides users with those data in 6 or 5 parts(some users are lack of sleep based on their Fitbit devices): active minutes, steps, distance, sedentary calories burns, sleep.

UC#3 Viewing Line Chart
Users can view their activity data and health index in line charts which can directly shows to our users their health condition. Users can also know which activity factors influence more on their health(index) by comparing the lines.

UC#4 Getting Standard Deviation Line Chart
Users can view their Standard Deviation Line Charts of health index in past years. User can view their health condition's fluctuation directly by viewing the line. The most important function of standard deviation in health index is that this number can show the health condition is fluctuating or stabilizing live.

UC#5 Getting Average Health Index Line Chart
Users can get their average number of health index line chart. The line chart is drawn by the average health index in some months. The average heath index line chart shows he tendency of the user's health condition. Showing if he or she is more healthy or less healthy or keeping stable with time passing by.

UC#6 Viewing Health Index Gauge
Users can view their health index gauge in our website. The health index gauge shows the range of the user's health index. Users can monitoring their health index by viewing the gauge. It is funny and lively!

UC#7 Getting Advice
Users can get advice by our software. When user click get advice button we can provide users lots of information such as how long you should run or how many hours you should sleep per day to keep you healthy.

UC#8 Calories Analysis
Users can view how many calories they burn each day. And users can view the recommended calorie burn each day and the personal calorie burn goal set by users themselves on the line chart.

UC#9 Searching Info

Users can look up the health information of a certain state that user insert. This function provides our user with an access to view the health condition for group of people. It is an convenient function especially for health research people and government users.

ii. Use Case Diagram

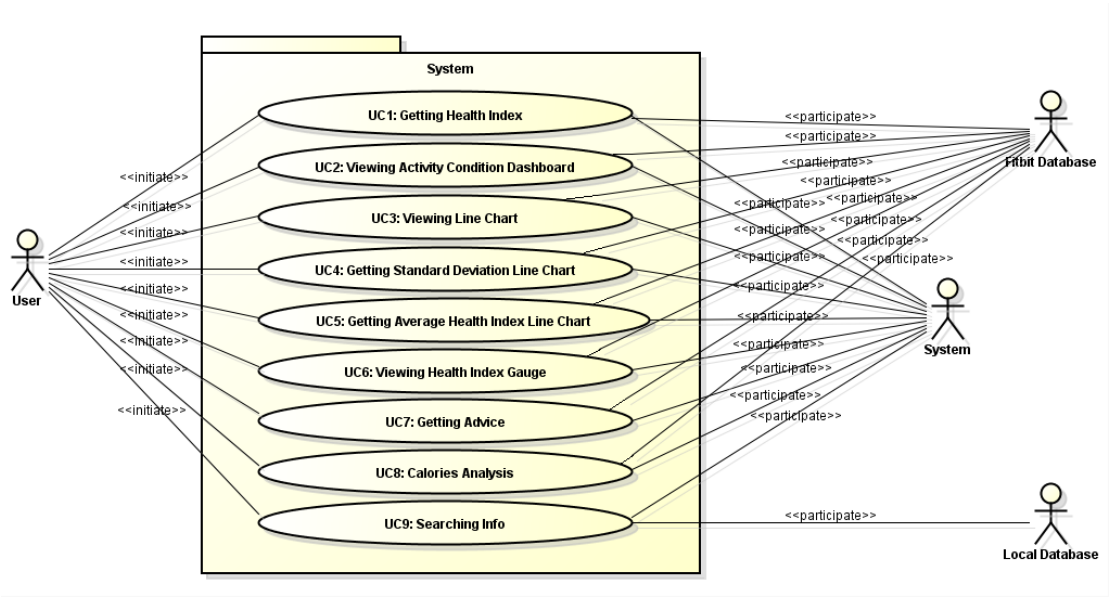


Figure 4-1 Use Case Diagram

The function of our system is based on calculating and displaying. Our data comes from Fitbit Database which record user's activity condition in several aspects. We will get some interested indexes by processing the data. UC1 is the home page of our website. UC2 is the most basic one which displays the collected data. UC3 to UC6 are different graphs whose data is derived from UC2. Deeper analysis will be given in UC7 and UC8. We will compare our data will the data from other sites and then give suggestions in written form. User can search information in specific area by UC9.

iii. Traceability Matrix

	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9
REQ-1a	✓	✓	✓	✓	✓	✓	✓	✓	
REQ-1b									✓
REQ-2		✓	✓	✓	✓		✓	✓	✓
REQ-3	✓	✓	✓	✓	✓	✓		✓	
REQ-4			✓	✓	✓				
REQ-5			✓						
REQ-6			✓	✓	✓		✓	✓	
REQ-7			✓						
REQ-8							✓	✓	
REQ-9									✓
REQ-10									✓

REQ-11							✓	✓	

Table 4-2 Traceability Matrix

iv. Fully Dressed Descriptions

Use Case UC-1:	Getting Health Index
Related Requirements:	
Initiating Actor:	Users
Actor's Goal:	To get users' health index which is based on monthly activity data and yearly activity data
Participating Actors:	System
Preconditions:	The website displays at the Dashboard Page.
Postcondition:	Users get their health index dashboard by month or day.
Failed End Condition:	Unable to show the health index please try again.

Flow of Events for Main Success Scenario:
-> 1.Users click the Health Index button.
<- 2.System calculates the user's health index based on users' activity data.
-> 3.Users wait.
<- 4.System displays users' health index in dashboard.

Flow of Events for Extensions:
4(a) No data in fitbit.com's API.
<- 1.System would display a error page, which shows Unable to show the health index please try again.

Table 4-2 UC-1

Use Case UC-2:	Viewing Activity Condition Dashboard
Related Requirements:	
Initiating Actor:	Users
Actor's Goal:	To get users' activity condition which are got from Fitbit portable devices
Participating Actors:	System
Preconditions:	The website displays at the start Page.
Postcondition:	Users get their activity data dashboard by month or day.
Failed End Condition:	Unable to show the activity dashboard please try again.

Flow of Events for Main Success Scenario: -> 1.Users click the Show My Dashboard button. <- 2.System is working with getting API from fitbit.com. -> 3.Users wait. <- 4.System displays users' activity condition into calories burns, steps, distance, sleeping, sedentary, active minutes in dashboard. Flow of Events for Extensions: 4(a) No data in fitbit.com <- 1.System would display a error page, which shows Unable to show the active data dashboard please try again.

Table 4-3 UC-2

Use Case UC-7:	Getting Advice
Related Requirements:	
Initiating Actor:	Users
Actor's Goal:	To provide user with advice by our software.
Participating Actors:	System
Preconditions:	The website displays at the dashboard page.
Postcondition:	Users get their health condition advice on the advice page.
Failed End Condition:	Unable to show the advice please try again.
Flow of Events for Main Success Scenario: -> 1.Users click the Get Advice button. <- 2.System is working with getting advice by our algorithm. -> 3.Users wait. <- 4.System displays our advice on how many minutes the user should active or how long the user should travel per day etc. Flow of Events for Extensions: 4(a) No data in fitbit.com <- 1.System would display a error page, which shows Unable to show the advice please try again.	

Table 4-4 UC-7

Use Case UC-9:	Searching Info
Related Requirements:	
Initiating Actor:	Users
Actor's Goal:	To search for the health condition information in a certain district
Participating Actors:	Database, System
Preconditions:	The website displays at the Start Page.
Postcondition:	Health condition in a certain state is shown on the screen.
Failed End Condition:	If the state is not existed in the entire nation, it will show Please Input Correct State Name, and then return to searching page.

Flow of Events for Main Success Scenario:
-> 1.User clicks the searching button to enter the searching page.
<- 2.System displays the searching page for user.
-> 3.User inputs the state names that he want to look for in the searching bar and then clicks the searching button.
<- 4.System searches for the health information at those states in its database
<- 5.System displays the people healthy condition level, working hours, exercise time on the screen.
-> 6.User get the information that he want, and then exit the software.

Flow of Events for Extensions:
4(a) At least one of those state names is not existed or wrong
<- 1.System would display a error page, which shows Please Input the Correct State Names, and then return to the searching page
5(a) User exports the detailed information to Excel before exiting
<- 1.User selects the information and clicks export button at the Results Page.
-> 2.System displays the saving page and requests user to choose the (a) saving route and (b) saving name.
<- 3.User fills the saving route and saving name at the bar.
-> 4.(a)System displays Successful Export Page, and then return 5. (b) The saving route or saving name is invalid, system displays a error page, which shows Please input valid saving route or saving name, and return 5(a).2.

Table 4-5 UC-9

d. System Sequence Diagrams

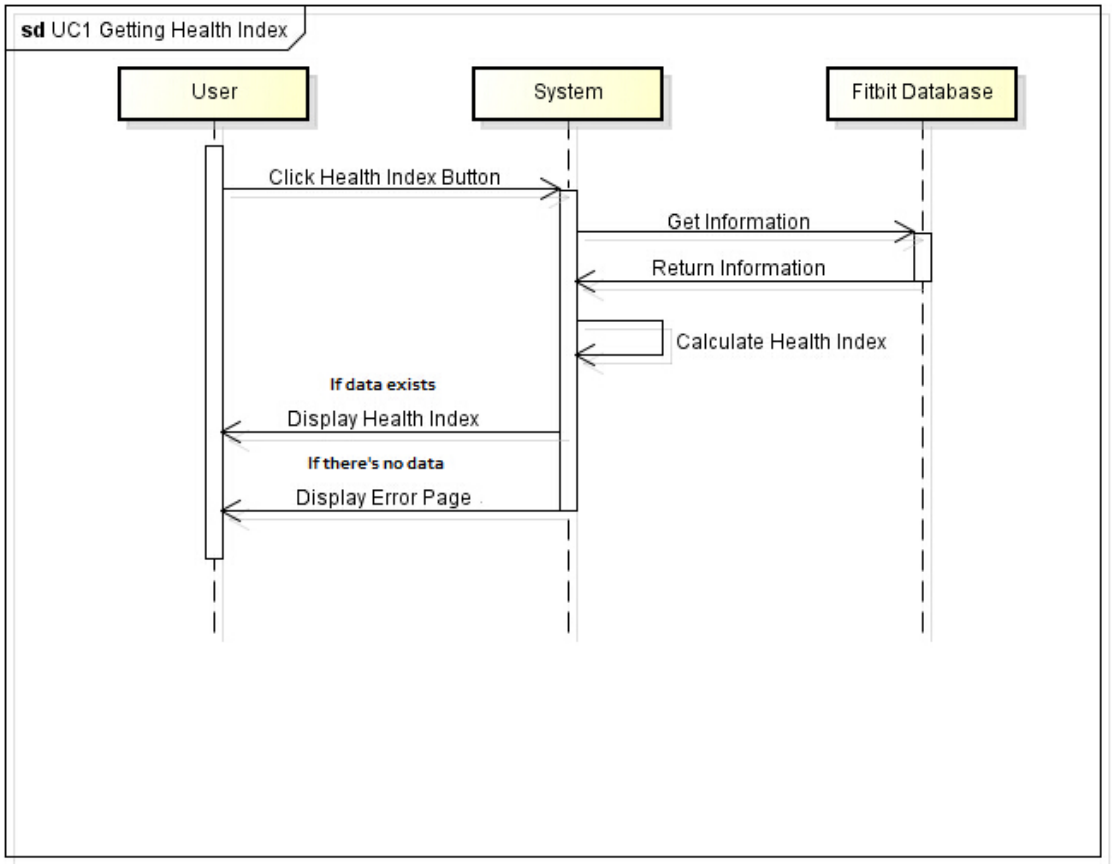


Figure 4-1 Sequence Diagram of Use Case 1

Figure 4-1 shows the sequence diagram of Use Case 1 which is “Getting Health Index”. It is used to get user’s health index (daily, monthly or yearly). First, the user clicks the “Health Index” button. Next, the “System” send request to “Fitbit Database” to get user’s activity information and “Fitbit Database” will send information to “System”. Then, “System” calculates the user’s health index based on the information from “Fitbit Database”. After that, “System” displays user’s health index to user in dashboard if there’s data in the database. Otherwise, “System” displays an error page and requires user to try again.

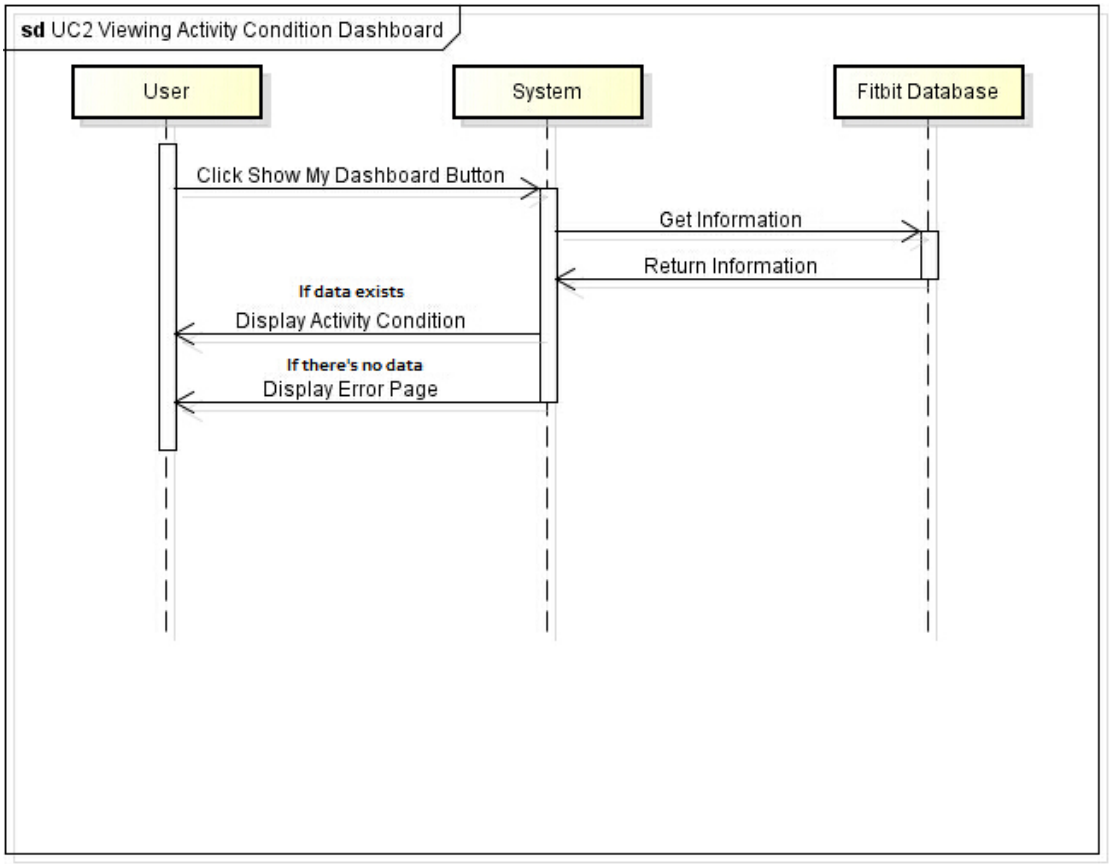


Figure 4-2 Sequence Diagram of Use Case 2

Figure 4-2 shows the sequence diagram of Use Case 2. It is used to get user's activity condition. First, the user clicks the "Show My Dashboard" button. Next, the "System" send request to "Fitbit Database" to get user's activity information and "Fitbit Database" will send information to "System". Then, "System" displays user's activity condition in several factors (calories burns, steps, distance, sleeping, sedentary, active minutes) to user in dashboard if there's data in the database. Otherwise, "System" displays an error page and requires user to try again.

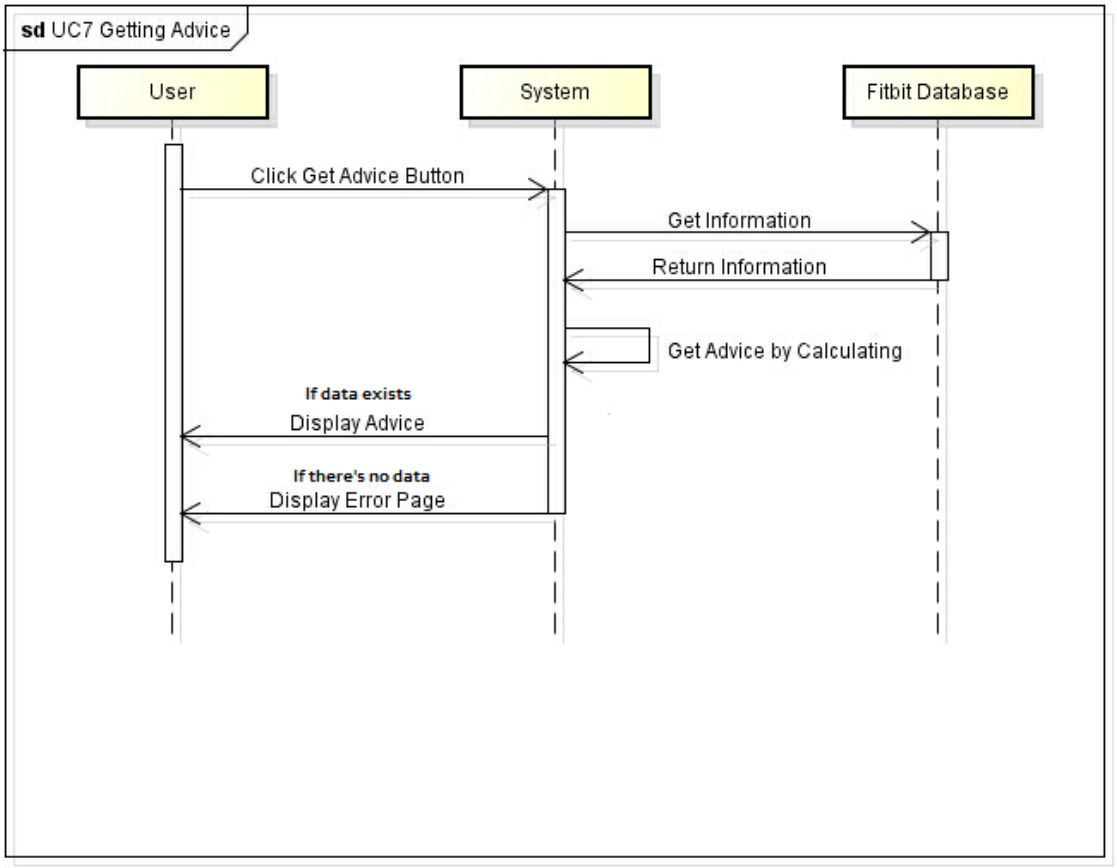


Figure 4-3 Sequence Diagram of Use Case 7

Figure 4-3 shows the sequence diagram of Use Case 7. It is used to provide user with advice. First, the user clicks the “Get Advice” button. Next, the “System” send request to “Fitbit Database” to get user’s activity information and “Fitbit Database” will send information to “System”. Then, “System” calculates some factors for the advice by our algorithm according to the information from “Fitbit Database”. After that, “System” displays the advice to user in dashboard if there’s data in the database. Otherwise, “System” displays an error page and requires user to try again.

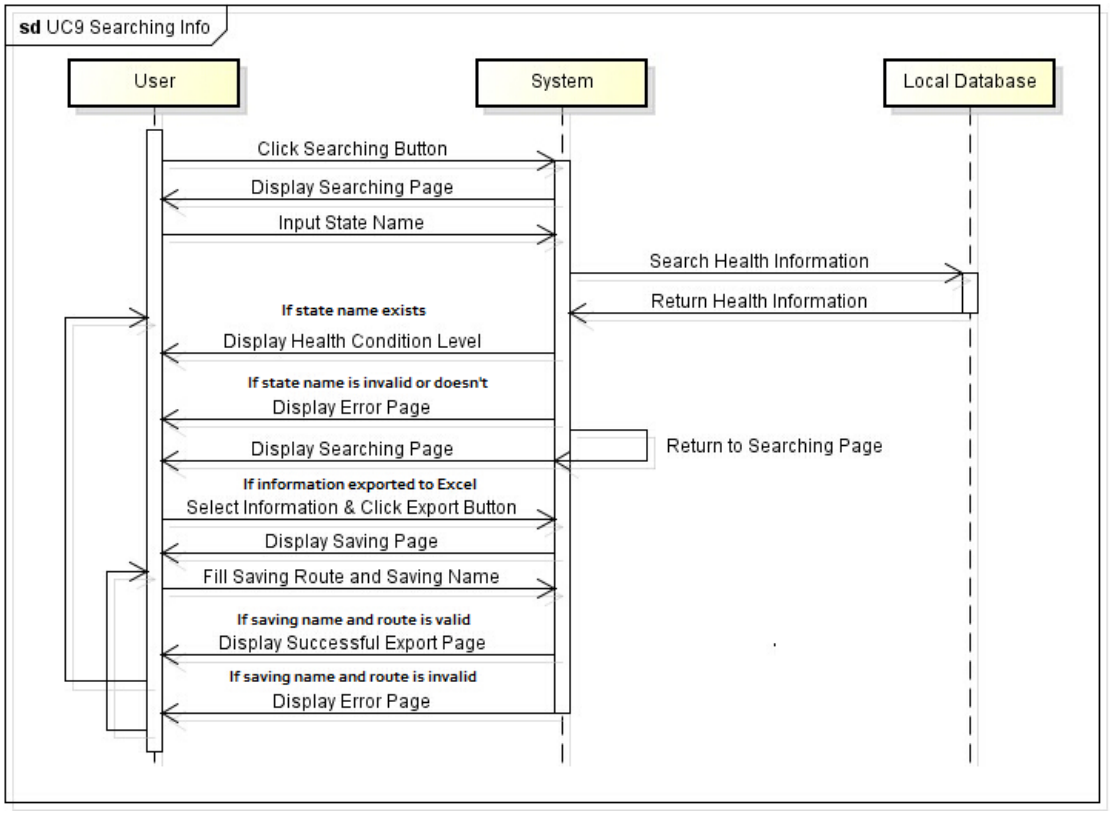


Figure 4-4 Sequence Diagram of Use Case 9

Figure 4-4 shows the sequence diagram of Use Case 9. It is used to search for the health condition information in a certain district. First, the user clicks the “Searching” button and “System” displays the searching page to user. User input the name of state which he is interested in. Next, “System” send request to search the health information from “Local Database” and “Local Database” will return the health information to “System”. If the state’s name is valid, “System” will display people’s health condition level on the screen and user will get wanted information. If the state’s name is invalid or doesn’t exist, “System” will display an error page which tells user to input correct name and return to the Searching page. After that, if user exports the detailed information to Excel by selecting the information and click “Export” button, “System” will display the saving page and request user to choose the saving route and saving name. If the saving route and saving name are valid, “System” will display Successful Export page and return to the page with health condition level. If they are invalid, “System” will display an error page and tell user to input valid saving route and saving name. It will return to the page for inputting saving name and saving route.

5. Effort Estimation using Use Case Points

a. Use Case Points

Projects with many complicated requirements take more effort to design and implement than projects with few simple requirements. In addition, the effort depends not only on inherent difficulty or complexity of the problem, but also on what tools the developers employ and how skilled the developers are. The formula for calculating UCP is composed of three variables:

1. Unadjusted Use Case Points (UUCP), which measures the complexity of the functional requirements.
2. The Technical Complexity Factor (TCF), which measures the complexity of the nonfunctional requirements
3. The Environment Complexity Factor (ECF), which assesses the development team's experience and their development environment.

UCP = UUCP × TCF × ECF

Actor Type	Description of how to recognize the actor type	Weight
Simple	The actor is another system which interacts with our system through a defined application programming interface (API)	1
Average	The actor is a person interacting through a text-based user interface.	2
Complex	The actor is a person interacting via a graphical user interface.	3

Table 5-1: Actor classification and associated weights

i. Functional Requirements

Unadjusted Use Case Points (UUCPs) are computed as a sum of these two components:

1. The Unadjusted Actor Weight (UAW), based on the combined complexity of all the actors in all the use cases.
2. The *Unadjusted Use Case Weight (UUCW)*, based on the total number of activities (or steps) contained in all the use case scenarios.

Actor name	Description of relevant characteristics	Complexity	Weight
Database	Fitbit is another system which interacts with our system through a defined API.	Simple	1
Processor	Processor is interacting through a text-based user interface(assuming that identification is through a keypad)	Average	2
Visitor	Visitor is interacting with the system via a graphical user interface (when managing user on the central computer).	Complex	3

Table 5-2 Actor classification for Health Monitoring Analytics

UAW=3 × Complex + 2 × Average + 1 × Simple= 3 × 1 + 2 × 1 + 1 × 1 =6.

Use case category	Description of how to recognize the use case category	Weight
Simple	Simple user interface. Up to one participating actor (plus initiating actor). Number of steps for the success scenario: ≤3. If presently available, its domain model includes ≤3 concepts.	5
Average	Moderate interface design. Two or more participating actors. Number of steps for the success scenario: 4 to 7. If presently available, its domain model includes between 5 and 10 concepts.	10
Complex	Complex user interface or processing. Three or more participating actors. Number of steps for the success scenario: 7. If available, its domain model includes≥10 concepts.	15

Table 5-3 Use case weights based on the number of transactions

The UUCW is calculated by tallying the use cases in each category, multiplying each count by its specified weighting factor (Table 5-3), and then adding the products.

Use case	Description	Category	Weight
UC-1	Complex user interface. 6 steps for the main success scenario. Two participating actors (Visitor, Database).	Average	10
UC-2	Complex user interface. 12 steps for the main success scenario. 2 participating actors. (Visitor, Database).	Complex	15
UC-3	Simple user interface. 4 steps for the main success scenario. 2 participating actors. (Visitor, Processor).	Simple	5
UC-4	Simple user interface. 3 steps for the main success scenario. 2 participating actors. (Visitor, Processor).	Simple	5
UC-5	Complex user interface. 7 steps for the main success scenario. 2 participating actors. (Visitor, Processor).	Average	10
UC-6	Simple user interface. 4 steps for the main success scenario. 2 participating actors. (Visitor, Processor).	Simple	5
UC-7	Complex user interface. 8 steps for the main success scenario. 2 participating actors. (Visitor, Processor).	Complex	15
UC-8	Complex user interface. 13 steps for the main success scenario. 3 participating actors. (Visitor,	Complex	15

	Processor, Database).		
UC-9	Complex user interface. 10 steps for the main success scenario. 2 participating actors. (Visitor, Database).	Complex	15

Table 5-4 Use case classification for the Health Monitoring Analytics

$$UUCW=3 \times Simple + 2 \times Average + 4 \times Complex = 3 \times 5 + 2 \times 10 + 4 \times 15 = 95$$

$$UUCP= UAW + UUCW = 6 + 95 = 101$$

ii. Nonfunctional Requirements

Technical factor	**Description**	**Weight**	**Perceived complexity**	**Factor**
T1	Distributed system (running on multiple machines)	2	3	6
T2	Performance objectives (are response time and throughput performance critical?)	1	3	3
T3	End-user efficiency	1	3	3
T4	Complex internal processing	1	2	2
T5	Reusable design or code	1	2	2
T7	Easy to use (including operations such as backup, startup, and recovery)	0.5	6	3
T8	Easy to change (to add new features or modify existing ones)	1	5	5
Technical Factor Total:				24

Table 5-5 Technical complexity factors and their weights

Constant-1 (C1)=0.6, Constant-2 (C2)=0.01.

$$TCF=0.6+(0.01 \times 24)=0.84$$

As a result, this will cause in a reduction of the UCP by 16%.

iii. Environmental Factors

Environmental Factor	**Description**	**Weight**	**Perceived Impact**	**Factor**
E1	Beginner familiarity with the	1.5	1	1.5

	UML- based development			
E2	Some familiarity with application problem	0.5	2	1
E3	Some knowledge of object-oriented approach	1	2	2
E4	Beginner lead analyst	0.5	1	0.5
E5	Highly motivated, but some team members occasionally slacking	1	4	4
E6	Stable requirements expected	1	5	5
E7	No part-time staff will be involved	-1	0	0
E8	Programming language of average difficulty will be used	-1	3	-3
Environmental Factor Total:				11

Table 5-6 Environmental Complexity factors

Constant-1 (C1)=1.4, Constant-2 (C2)=-0.03

$$ECF=1.4+(-0.03\times16)=1.07$$

As a result, this will cause in an increase of the UCP by 7%.

UCP=UUCP×TCF×ECF=101×0.84×1.07=90 use case points.

b. Deriving Project Duration from Use Case Points

Use case points (UCP) are a measure of software size. When we set Productivity Factor PF=28 per use case point, we can get our project Duration=UCP×PF, where PF=28 hours per use case point.

Duration=UCP×PF=90×28=2520 hours.

6. Domain Analysis

a. Domain Model

i. 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.

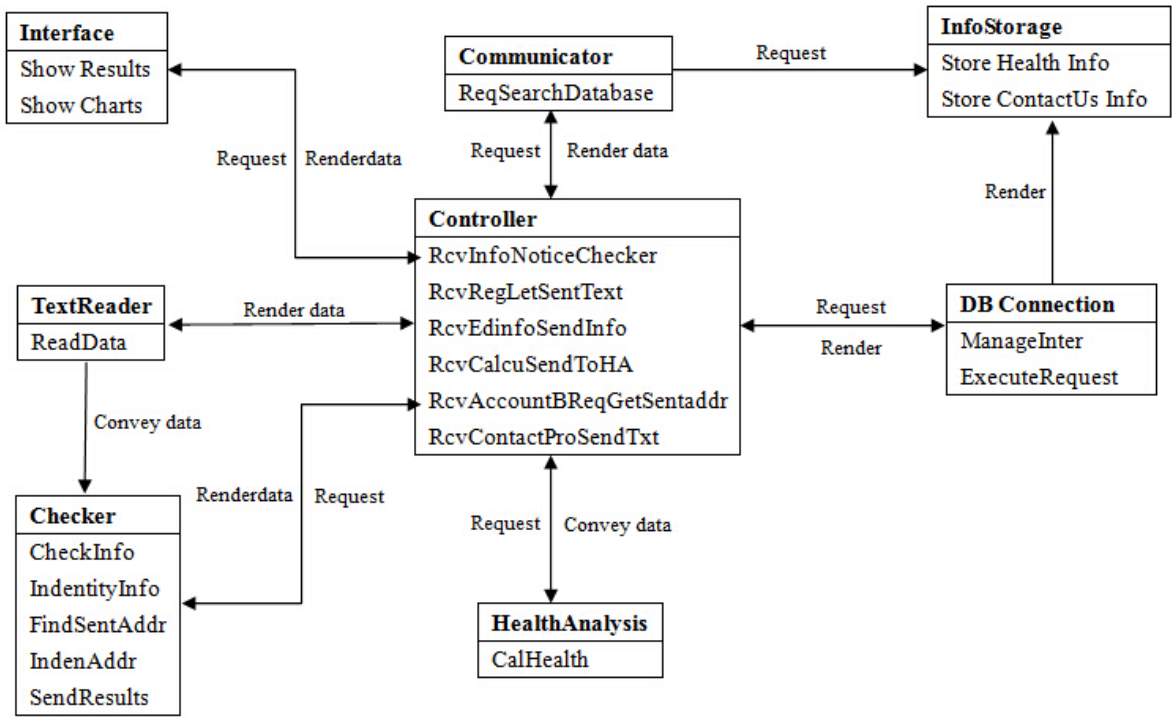


Figure 6-1 Domain Model

Responsibility	Type	Concept
R1: Read data that user types in.	K	TextReader
R2: Store the data about health information	K	InfoStorage
R3: Receives signals from TextReader to notice Checker to start to work	/	Controller
R4: Render search request to Database.(InfoStorage)	D	Communicator
R5: Check information between the InfoStorage and TextReader	D	Checker
R6: Display the searching results to interface(screen) for user from Controller	/	Interface
R7: Manage interactions with the Database(InfoStorage)	D	DB Connection
R8: Checker gets information from TextReader and check if	D	Checker

the information is valid		
R9: Display the result from the Controller	/	Interface
R10: Checker receives the username and requests Controller to use communicator for using datatbase(InfoStorage) to find data in database to compare.	D	Checker
R11: Controller receives the operation of "HealthIndex" and controls TextReader to get text and send it to the HealthAnalysis	D	Controller
R12: HealthAnalysis receives the text and calculate.	/	HealthAnalysis
R13:Controller receives the operation "Contact provider" and controls TextReader to get text and send to DB Connection to write into the InfoStorage.	D	Controller
R14: Access to Database and execute relative request.	D	DB Connection
R15: Checker sends its results to Controller	/	Checker
R16: Checker receives the Fitbit code ID and requests Controller to use communicator for using datatbase(InfoStorage) to find data in database to compare.	D	Checker

Table 6-1 Conception Definition

ii. 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
Communicator<->InfoStorage	Communicator renders requests to InfoStorage	Render requests
Checker<->InfoStorage	Check checks information from InfoStorage	Check
Checker<->TextReader	Checker checks information from InfoStorage	Check
Controller<->Interface	Controller renders its results and generate requests to Interface to display	Generate requests Convey data
DB Connection<->InfoStorage	DB Connection gets access into InfoStorage and saves data in InfoStorage	Save data
Controller<->DB Connection	Controller generates requests	Render requests

	to use DB Connection	
Controller<->Communicator	Controller requests communicator to search information in InforStorage	Generate requests
Controller<->HealthAnalysis	Controller requests HealthAnalysis to analyze data and HealthAnalysis returns results to Controller.	Generate requests Convey data

Table 6-2 Association Definiton

iii. Attribute Definition

Responsibility	**Attribute**	**Concept**
R1: Read data that user types in.	ReadData	TextReader
R2: Store the data about health information.	StoreHealthInfo	InfoStorage
R3: Receives information from TextReader and notice Checker to start to work.	ReceiveInfoNoticeChecker	Controller
R11: Controller receives the operation of "HealthIndex" and controls TextReader to get text and send it to the HealthAnalysis.	ReceiveCalculateLetSendToHealthAnalysis	
R13: Controller receives the operation "Contact provider" and controls TextReader to get text and send to DB Connection to write into the InfoStorage.	ReceiveContactProviderLetSendText	
R4: Render search request to Database.	RequestToSearchDatabase	Communicator
R5: Check information from the InfoStorage.	CheckInfo	Checker
R8: Checker get information from TextReader and check if the information is valid.	IndentityInfo	
R10: Checker receives the Fitbit code ID and finds the address in InfoStorage and send the address to the DB Connection.	FindSentAddress	
R15: Checker sends its results to Controller to prepare	SendResults	
R6: Display the searching results to interface (screen) for user.	ShowMapResult	Interface
R9: Display the result from the Controller.	ShowResult	
R14: Access to Database and execute relative request.	ExecuteRequest	DB Connection

R7: Manage interactions with the Database.	ManageInteractions	
R12: HealthAnalysis receives the text and calculate.	CalculateHealthText	HealthAnalysis

Table 6-3 Attribute Definitions

iv. Traceability Matrix

Domain Model	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9
TextReader		✓							✓
InfoStorage	✓	✓	✓	✓	✓	✓	✓	✓	✓
Controller	✓	✓	✓	✓	✓	✓	✓	✓	✓
Communicator									✓
DBConnection		✓							✓
HealthAnalysis	✓	✓		✓	✓	✓		✓	✓
Checker		✓					✓		✓
Interface	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 6-4 Traceability Matrix

b. System Operation Contracts

System Operation Contracts for the operations of the fully-dressed user cases.

Getting Health Index
1. PRE-CONDITION- The website displays at the Dashboard Page
2. POST CONDITION- Users get their health index dashboard by month or day.

Viewing Activity Condition Dashboard
1. PRE-CONDITION- The website displays at the start Page
2. POST CONDITION- Users get their activity data dashboard by month or day

Getting Advice
1. PRE-CONDITION- The website displays at the dashboard page
2. POST CONDITION- Users get their health condition advice on the advice page

Searching Info
1. PRE-CONDITION- The website displays at the Start Page
2. POST CONDITION- Health condition in a certain state is shown on the screen

7. Interaction Diagrams

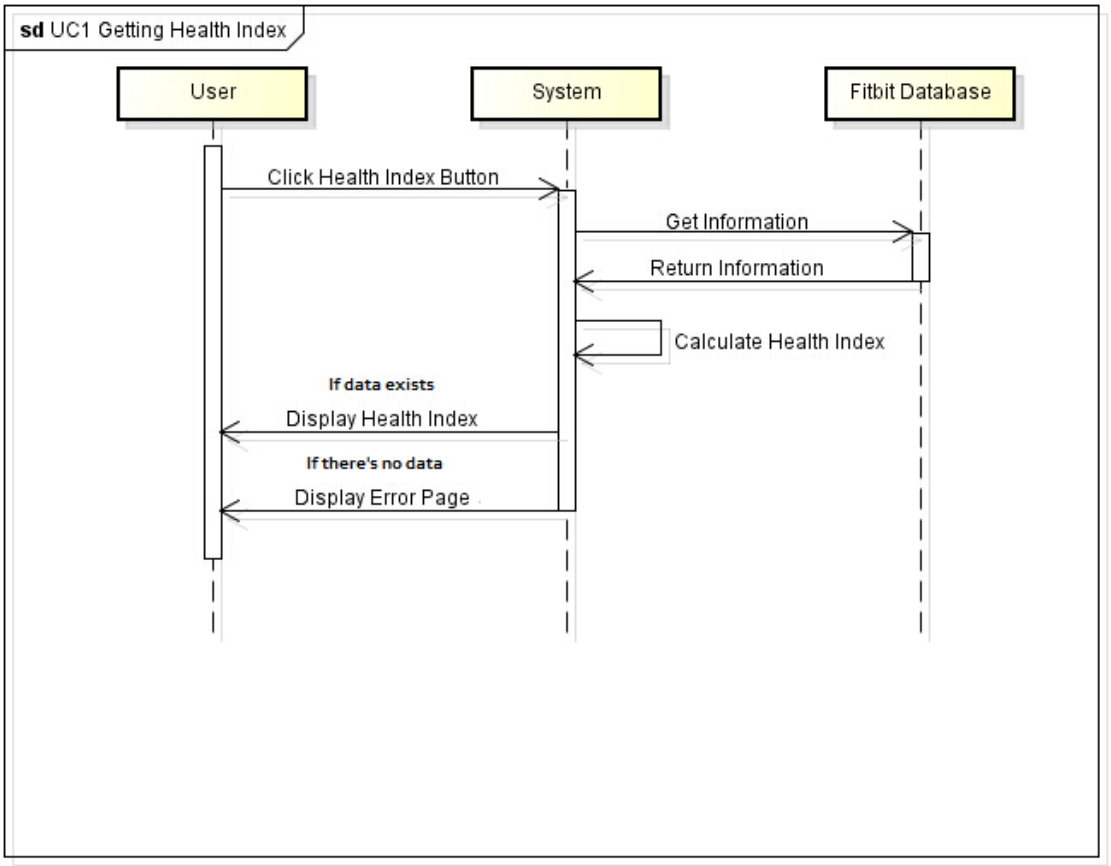


Figure 7-1 Sequence Diagram of Use Case 1

Figure 7-1 shows the sequence diagram of Use Case 1 which is “Getting Health Index”. It is used to get user’s health index (daily, monthly or yearly). First, the user clicks the “Health Index” button. Next, the “System” send request to “Fitbit Database” to get user’s activity information and “Fitbit Database” will send information to “System”. Then, “System” calculates the user’s health index based on the information from “Fitbit Database”. After that, “System” displays user’s health index to user in dashboard if there’s data in the database. Otherwise, “System” displays an error page and requires user to try again.

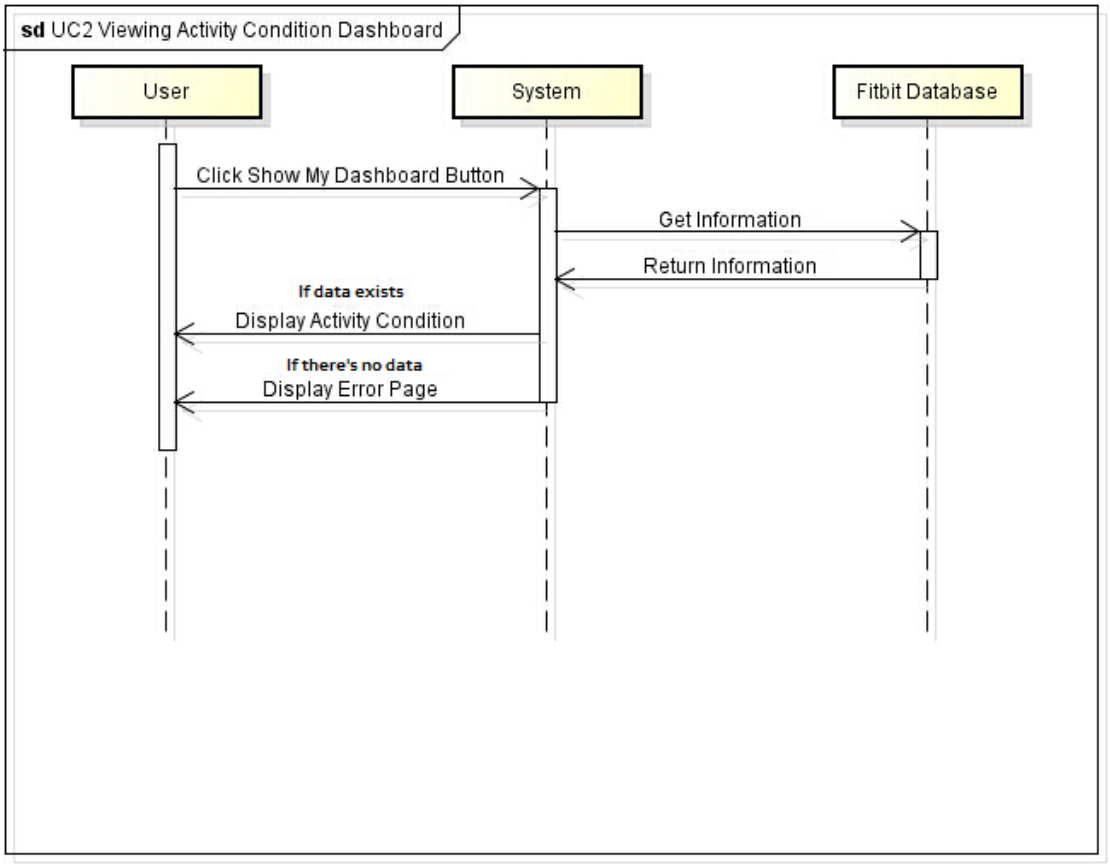


Figure 7-2 Sequence Diagram of Use Case 2

Figure 7-2 shows the sequence diagram of Use Case 2. It is used to get user's activity condition. First, the user clicks the "Show My Dashboard" button. Next, the "System" send request to "Fitbit Database" to get user's activity information and "Fitbit Database" will send information to "System". Then, "System" displays user's activity condition in several factors (calories burns, steps, distance, sleeping, sedentary, active minutes) to user in dashboard if there's data in the database. Otherwise, "System" displays an error page and requires user to try again.

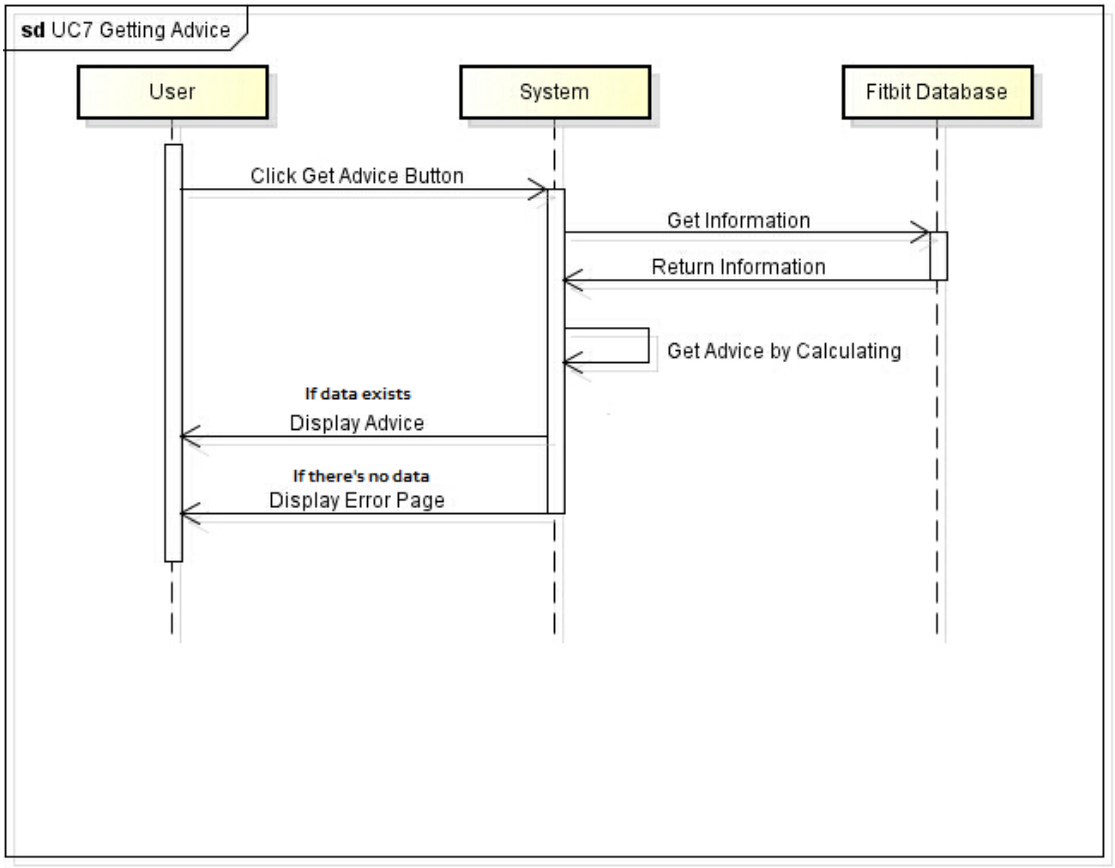


Figure 7-3 Sequence Diagram of Use Case 7

Figure 7-3 shows the sequence diagram of Use Case 7. It is used to provide user with advice. First, the user clicks the “Get Advice” button. Next, the “System” send request to “Fitbit Database” to get user’s activity information and “Fitbit Database” will send information to “System”. Then, “System” calculates some factors for the advice by our algorithm according to the information from “Fitbit Database”. After that, “System” displays the advice to user in dashboard if there’s data in the database. Otherwise, “System” displays an error page and requires user to try again.

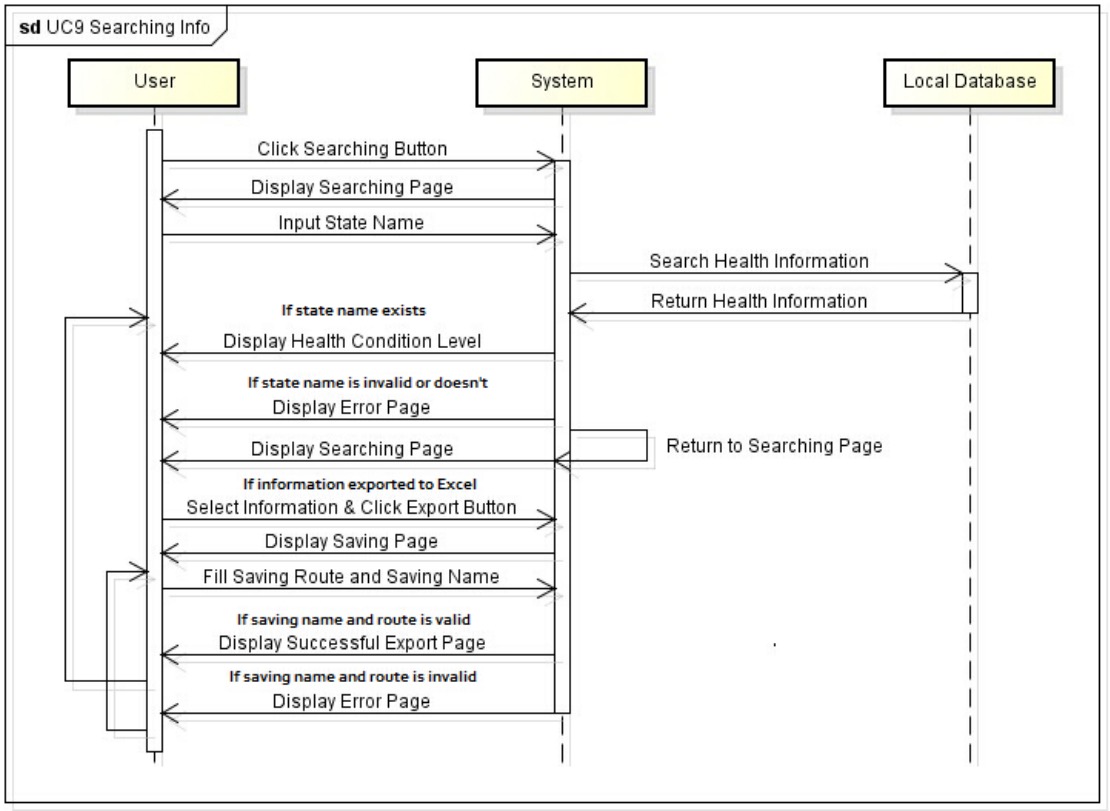


Figure 7-4 Sequence Diagram of Use Case 9

Figure 7-4 shows the sequence diagram of Use Case 9. It is used to search for the health condition information in a certain district. First, the user clicks the “Searching” button and “System” displays the searching page to user. User input the name of state which he is interested in. Next, “System” send request to search the health information from “Local Database” and “Local Database” will return the health information to “System”. If the state’s name is valid, “System” will display people’s health condition level on the screen and user will get wanted information. If the state’s name is invalid or doesn’t exist, “System” will display an error page which tells user to input correct name and return to the Searching page. After that, if user exports the detailed information to Excel by selecting the information and click “Export” button, “System” will display the saving page and request user to choose the saving route and saving name. If the saving route and saving name are valid, “System” will display Successful Export page and return to the page with health condition level. If they are invalid, “System” will display an error page and tell user to input valid saving route and saving name. It will return to the page for inputting saving name and saving route.

8. Class Diagram and Interface Specification

a. Class Diagram

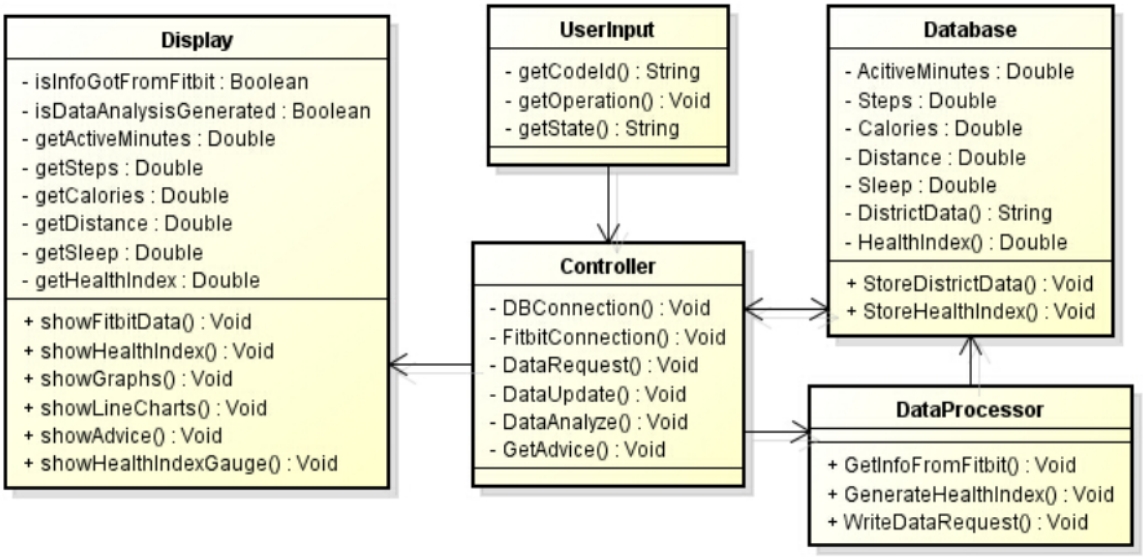


Figure 8-1 Class Diagram

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. The

- 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.

- getSleep():Double
  -Double variable corresponding to the user's sleep 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 heath index will based on the average data in one month, which means that the health index will change month by month. And there is a description of the health index to explain how this index represent and influence the user's health.

- ShowGraphs():Void
  -show the graphs of the user information retrieved from Fitbit, including steps, active minutes, distance, calories and sleep in a month or in a year. And in order to compare these data to make a clearer view of the user's health, we can display these graphs in the same page or different pages. And there is a description for each factor to explain how this factor influence the user's health.

- Showlinecharts():Void
  --show the line charts of the user information retrieved from Fitbit, including steps, active minutes, distance, calories and sleep in a month or in a year. And in order to compare these data to make a clearer view of the user's health, we can display these graphs in the same page or different pages. And there is a description for each factor to explain how this factor influence the user's health. And the line charts can reflect the user's health explicitly.

- 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.

- showHealthIndexGauge():Void
  -show the gauge of the user's health index. The pointer of the gauge will change as the health index day by day.

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.

- 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.

- DataAnalyze():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.

- Sleep():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 calculateHealthIndex 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 FitbitConnecion, 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:

Classes / Domain Concept	Display	UserInput	Controller	DataBase	DataAnalyze
Interface	✓				
Controller		✓	✓		

Checker		✓	✓	✓	
DBConnector			✓		
InfoStorage				✓	
Communicator			✓	✓	
TextReader		✓			
HealthAnalysis			✓	✓	✓

Table 8-1 Traceability Matrix

d. Object Constraint Language(OCL) Contracts

1. Display

Display	
invariants	Code ID the user inserted
precondition	Data are retrieved from Fitbit and generated.
postcondition	Show webpages to the user

Table 8-2 Display OCL Contracts

2. Controller

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

Table 8-3 Controller OCL Contracts

3. UserInput

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

Table 8-4 UserInput OCL Contracts

4. Database

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

Table 8-5 Database OCL Contracts

5. DataProcessor

DataProcessor	
invariants	Actions to analyze the data such as calculating health index and deviation of different health index.
precondition	Data retrieved from Fitbit
postcondition	Heath index and other analysis created into the database

Table 8-6 DataProcessor OCL Contracts

9. System Architecture and System Design

a. Architectural Styles

Our system has a client/server architectural model. The client and server are communicated through network. Our system has one centralize server to keep data consistent. Many clients can use the same serve to access the same data at the same time.

The server will do most of the data storage and data calculations. It will have a database to store all the information and use algorithms to calculate all the desired data required by the clients.

The client communicates with server effortlessly and displays the data stored on the server efficiently. The communication is done with XML. The client sends a GET or POST request to the server, and then the server acknowledges it and response with the appropriate data imbedded.

By using this architecture, we can have a light front--end application with easy ability to update and runs fast on most hardware. It requires a powerful back--end server with large storage and fast computing power to handle a lot of data and calculate the required data effectively.

b. Identifying Subsystems

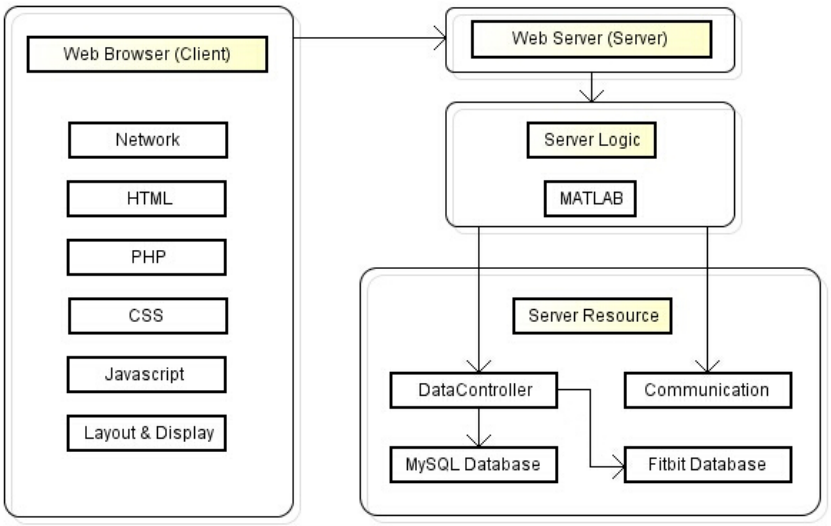


Figure9-1 UML Package Diagram

In this system, there are two subsystems. One is the client side - the web browser, the other side is server - web server. The client package mainly refers to the website framework, and it contains the building structure of the user interface. For the server side, when the user uses sends a request to the web server, the server would response through processing it in its logic package. The logic unit would analyze it and further pass it to the server resource package. This package is responsible for retrieving data from local MySQL Database, Fitbit Database and internal communication.

c.Mapping Subsystems to Hardware

As shown in Figure 3.1, the subsystem can be mapped into the following hardware component easily. The Web Browser package are allocated in client's PC, and the user would use this to access the application user interface. The Web Server as well as Server Logic package is allocated in the server. They would use this to appropriately analyze and process the user requests. The Server Resource package is allocated in the central server.

d. Persistent Data Storage

We choose MySQL Database to store our data. 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.
Large volume of raw data searched and gathered from website will stored in MySQL Database. Our system will use PHP to analysis the data and store in different collections (tables). Other parts of system can retrieve data freely by their need. The persistent data objects are shown in the schema below:
HealthData will store different kinds of statistic data from website.

Healthdata	
PK	Field
*	Worktime
	Sleep Time
	Exercise Time
	Ages
	Breakfast%
	Vegetable%
	Smoke%
	HealthIndex
	Calorie Table

Figure9-2 Database Schema

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.

f. Global Control Flow

This is an event-driven system. The user use our system by clicking and typing on the web page.
The users can get the information they want once they click or type on our user interface which means that a related page will show up for the users.

g. Hardware Requirements

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.

10. Algorithms and Data Structures

i. Factor Analysis

1. Why we use Factor Analysis

What has contributed to this continued increase in the use of factor analysis in the health sciences in particular? Several possibilities come to mind:

1. Increased researcher interest in the complex organizational structure of various health-related constructs

2. Recent developments in the use of confirmatory factor analysis and structural equation modeling

3. Greater sophistication concerning statistics on the part of some health care researchers from all disciplines and levels of expertise

4. Increased availability of inexpensive but powerful personal computers, which can undertake analyses quickly and inexpensively

5.Availability of increasingly user-friendly statistical computer packages

Factor analysis provides us with the means to undertake a structural analysis of that problem. In the Introduction, for example, we identified a construct we wished to explore in greater depth: the concerns of individuals who are considering undergoing genetic testing for cancer. Our interim goal is to develop an instrument of data collection that would adequately measure and reflect the structure of this construct. Ultimately, we would like to formulate programs of intervention that would address the specific concerns of individuals who make up this population.

Making Sense of Factor Analysis: The Use of Factor Analysis for Instrument Development in Health Care Research ----By Marjorie A. Pett, Nancy R. Lackey, John J. Sullivan

http://books.google.com/books?hl=en&lr=&id=9kB5jE2IjS4C&oi=fnd&pg=PR9&dq=Making+Sense+of+Factor+Analysis&ots=yXp82yowf5&sig=aVg-Xq7FkDmUQIaWi48Azkrt4_Q

2. The Implementing Steps for Factor Analysis:

In order that the variables be on equal footing, they are standardized:

$$z_{ai} = \frac{x_{ai} - \mu_a}{\sigma_a}$$

where the sample mean is:

$$\mu_a = \frac{1}{N}\sum_i x_{ai}$$

and the sample variance is given by:

$$\sigma^2 = \frac{1}{N}\sum_i (x_{ai} - \mu_a)^2$$

The factor analysis model for this particular sample is then:

$$z_{1,i} = l_{1,1}F_{1,i} + l_{1,2}F_{2,i} + \varepsilon_{1,i}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$z_{10,i} = l_{10,1}F_{10,i} + l_{10,2}F_{11,i} + \varepsilon_{10,i}$$

or, more succinctly:

$$z_{ai} = \sum\nolimits_p l_{ap}F_{pi} + \varepsilon_{ai}$$

where

- $F_{1,i}$ is the *i*th student's "verbal intelligence",
- $F_{2,i}$ i is the *i*th student's "mathematical intelligence",
- l_{ap} are the factor loadings for the *a*th subject, for $p = 1, 2$.

In matrix notation, we have

$$Z = LF + \varepsilon$$

Observe that by doubling the scale on which "verbal intelligence"—the first component in each column of *F*—is measured, and simultaneously halving the factor loadings for verbal intelligence makes no difference to the model. Thus, no generality is lost by assuming that the standard deviation of verbal intelligence is 1. Likewise for mathematical intelligence. Moreover, for similar reasons, no generality is lost by assuming the two factors are uncorrelated with each other. In other words:

$$\sum F_{pi}F_{qi} = \delta_{pq}$$

where δ_{pq} is the Kronecker delta (0 when $p \neq q$ and 1 when $p = q$).The errors are assumed to be independent of the factors:

$$\sum F_{pi}\varepsilon_{ai} = 0$$

Note that, since any rotation of a solution is also a solution, this makes interpreting the factors difficult. See disadvantages below. In this particular example, if we do not know beforehand that the two types of intelligence are uncorrelated, then we cannot interpret the two factors as the two different types of intelligence. Even if they are uncorrelated, we cannot tell which factor corresponds to verbal intelligence and which corresponds to mathematical intelligence without an outside argument.

The values of the loadings *L*, the averages μ, and the variances of the "errors" ε must be estimated given the observed data *X* and *F* (the assumption about the levels of the factors is fixed for a given *F*). The "fundamental theorem" may be derived from the above conditions:

$$\sum z_{ai}z_{bi} = \sum l_{ap}l_{bp} + \sum \varepsilon_{ai}\,\varepsilon_{bi}$$

The term on the left is just the correlation matrix of the observed data, and its N_a diagonal elements will be 1's. The last term on the right will be a diagonal matrix with terms less than unity. The first term on the right is the "reduced correlation matrix" and will be equal to the correlation matrix except for its diagonal values which will be less than unity. These diagonal elements of the reduced correlation matrix are called "communalities":

$$h_a^2 = 1 - \varphi_a = \sum_p l_{ap} l_{ap}$$

The sample data z_{ai} will not, of course, exactly obey the fundamental equation given above due to sampling errors, inadequacy of the model, etc. The goal of any analysis of the above model is to find the factors F_{pi} and loadings l_{ap}, which, in some sense, give a "best fit" to the data. In factor analysis, the best fit is defined as the minimum of the mean square error in the off-diagonal residuals of the correlation matrix:

$$\varepsilon^2 = \sum_{ab, a \neq b} [\sum_i z_{ai} z_{bi} - \sum_p l_{ap} l_{bp}]^2$$

This is equivalent to minimizing the off-diagonal components of the error covariance which, in the model equations have expected values of zero. This is to be contrasted with principal component analysis which seeks to minimize the mean square error of all residuals.Before the advent of high speed computers, considerable effort was devoted to finding approximate solutions to the problem, particularly in estimating the communalities by other means, which then simplifies the problem considerably by yielding a known reduced correlation matrix. This was then used to estimate the factors and the loadings. With the advent of high-speed computers, the minimization problem can be solved quickly and directly, and the communalities are calculated in the process, rather than being needed beforehand. The MinRes algorithm is particularly suited to this problem, but is hardly the only means of finding an exact solution.

Quoted from Wikipedia： http://en.wikipedia.org/wiki/Factor_analysis

We will research the inner relationships between variables we collected to find out the basic structure of those data. Then we assume a few(less than what we collected) new variables to show the basic data structure. Those assumed variables could show the majority information of variables we collected before. Those assumed variables can not be observed or collected so they are called Factors. The Factor Analysis is also a method of reduce dimensionality.

The raw data are the same as PCA.

The matlab code(data name is saved as sssgs in txt):

```
clc,clear
load ssgs.txt
n=size(ssgs,1);
x=ssgs(:,[1:7]);
x=zscore(x);
r=corrcoef(x)
[vec1,val,con1]=pcacov(r)
f1=repmat(sign(sum(vec1)),size(vec1,1),1);
vec2=vec1.*f1;
f2=repmat(sqrt(val)',size(vec2,1),1);
a=vec2.*f2
num=input('please choose the number of factors£º');
am=a(:,[1:num]);
[bm,t]=rotatefactors(am,'method', 'varimax')
bt=[bm,a(:,[num+1:end])];
con2=sum(bt.^2)
check=[con1,con2'/sum(con2)*100]
rate=con2(1:num)/sum(con2)
```

```
coef=inv(r)*bm
score=x*coef
weight=rate/sum(rate)
Tscore=score*weight'
[STscore,ind]=sort(Tscore,'descend')
display=[score(ind,:)';STscore';ind']
```

3. Data Structure

Our data are mostly collected from statistic website and government website. Our previous version asks our users input some data. But now we decide to use FITBIT and hyperlink user data from the website.

The screen shot shows all types of data we need for calculating health index.

Figure 10-1

All data are string type.

11.User Interface Design and Implementation

a. Preliminary Design

1 . Welcome page

This is sort of the pre-website page. It displays information about what the interior of the website contains. And some buttons would be on the screen somewhere that the user can press to enter the website and access to the functions.

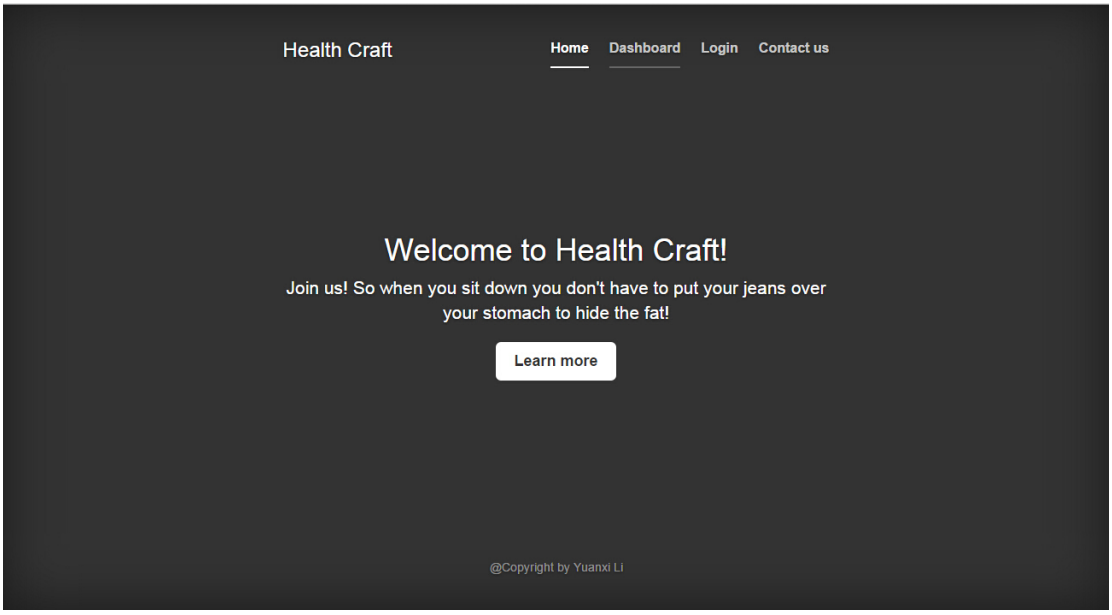


Figure 11-1 Welcome Page

3. Health Index

(1) Tutorial of how to open access to the Fitbit if the user doesn't insert the CodeID and open the access.

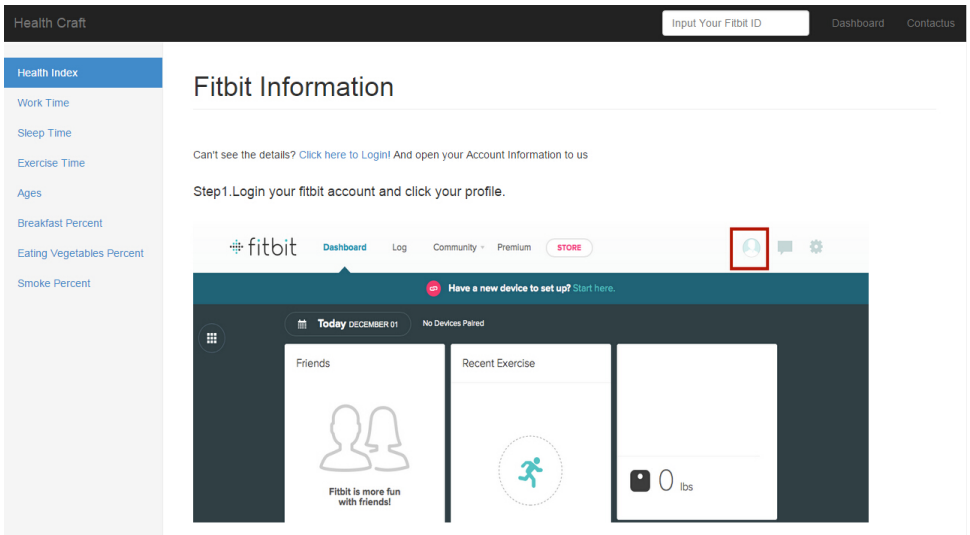


Figure 11-2 Health Index (1)

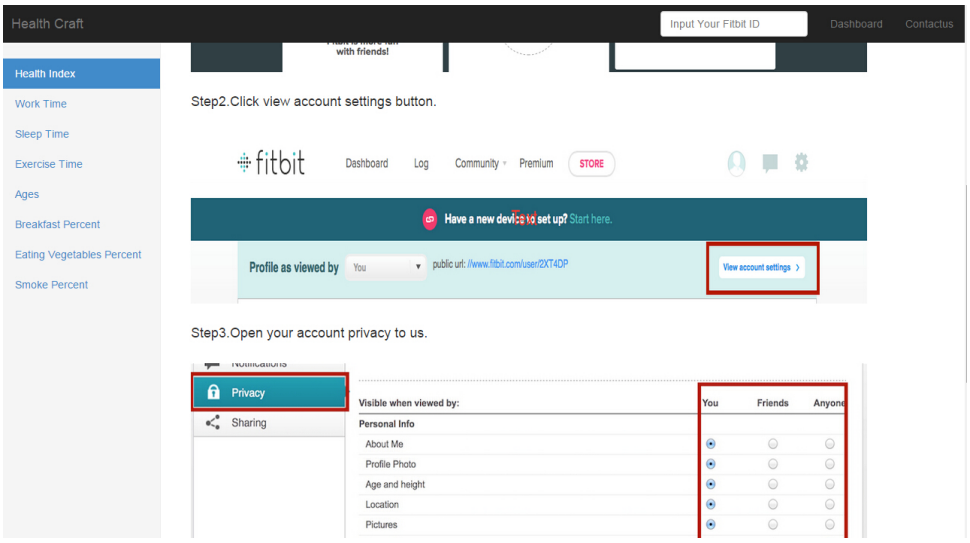


Figure 11-3 Health Index (2)

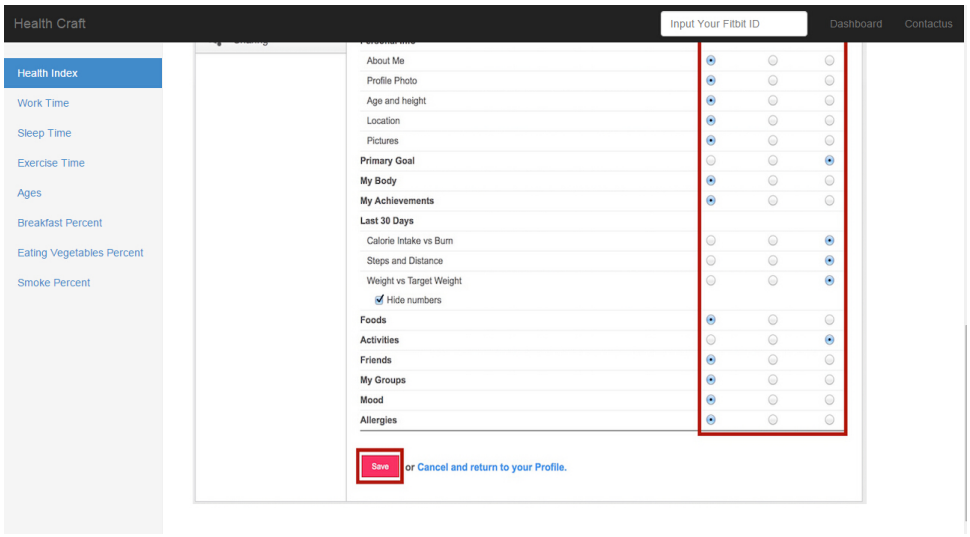


Figure 11-4 Health Index (3)

(2) Data from fitbit

After the user inserted the CodeID and opened the access to the Fitbit, the website can show the Fitbit data in a whole month including “Active minutes”, “Calories”, “Steps”, “Distance”, “Sedentary” and “Sleep”.

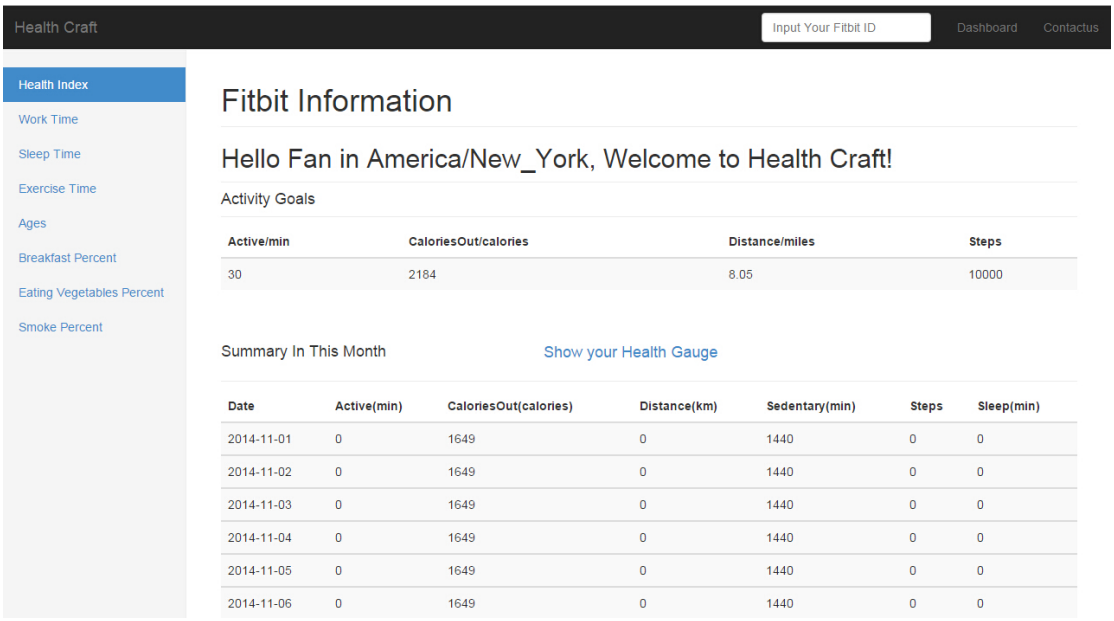


Figure 11-5 Data (1)

And when the user click the "Show your Health Gauge" button, the website will show a page of the gauge of the user's health index in last month. The pointer will change day by day.

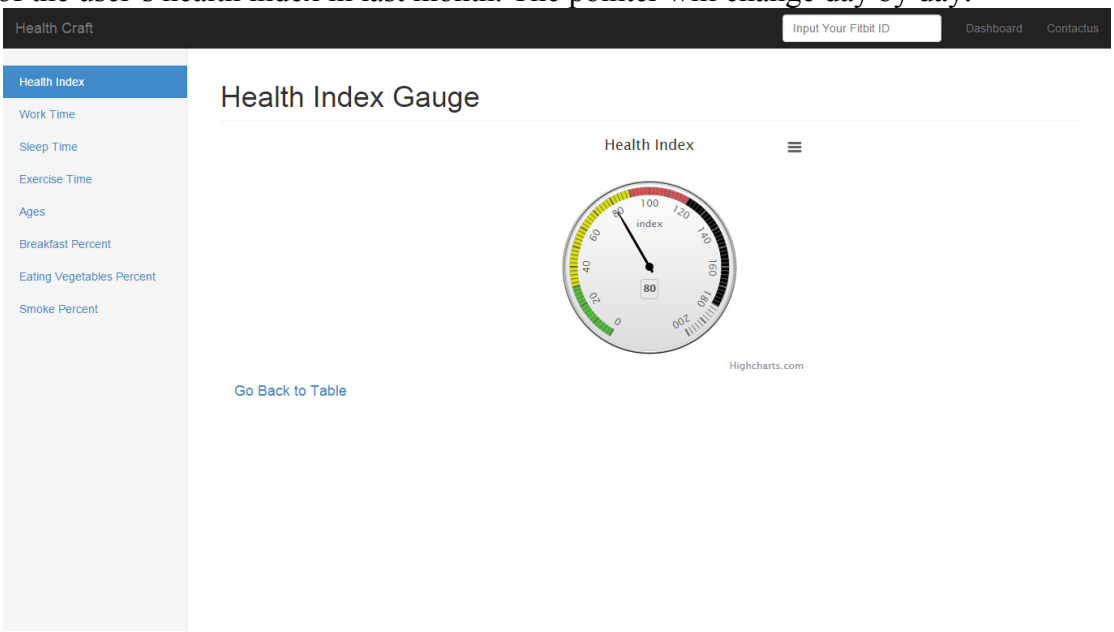


Figure 11-6 Data (2)

(3) Data analysis

At the bottom of the screen in Month data, there are four buttons.

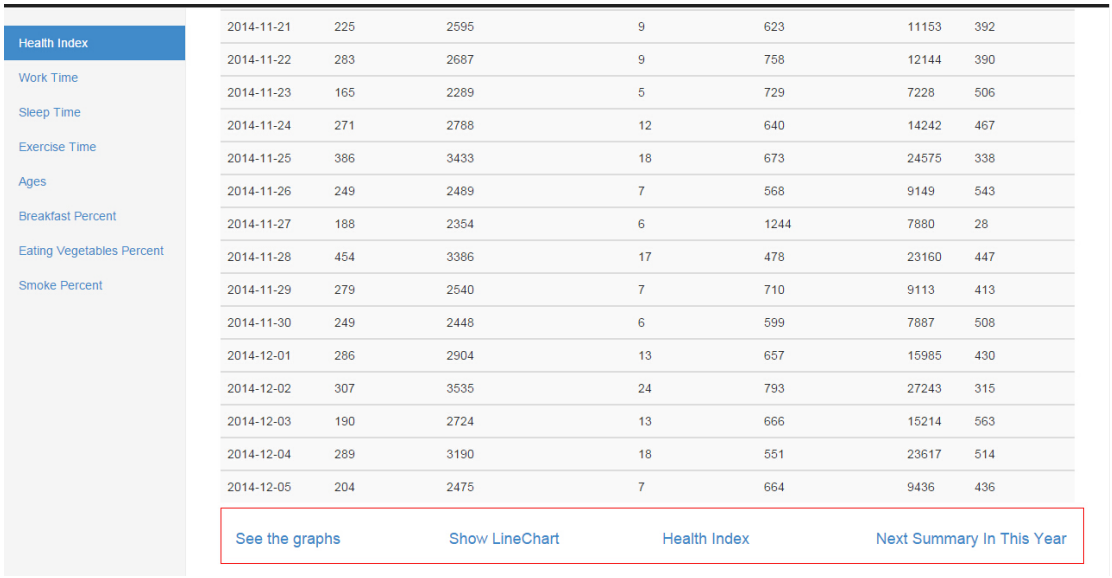


Figure 11-7 Data (3)

(1) See the graphs

After the user click this button, then the website will show a page of histogram of all six factors. For example “Calories”. And the user can see other graphs by clicking the “Next Graph” button.

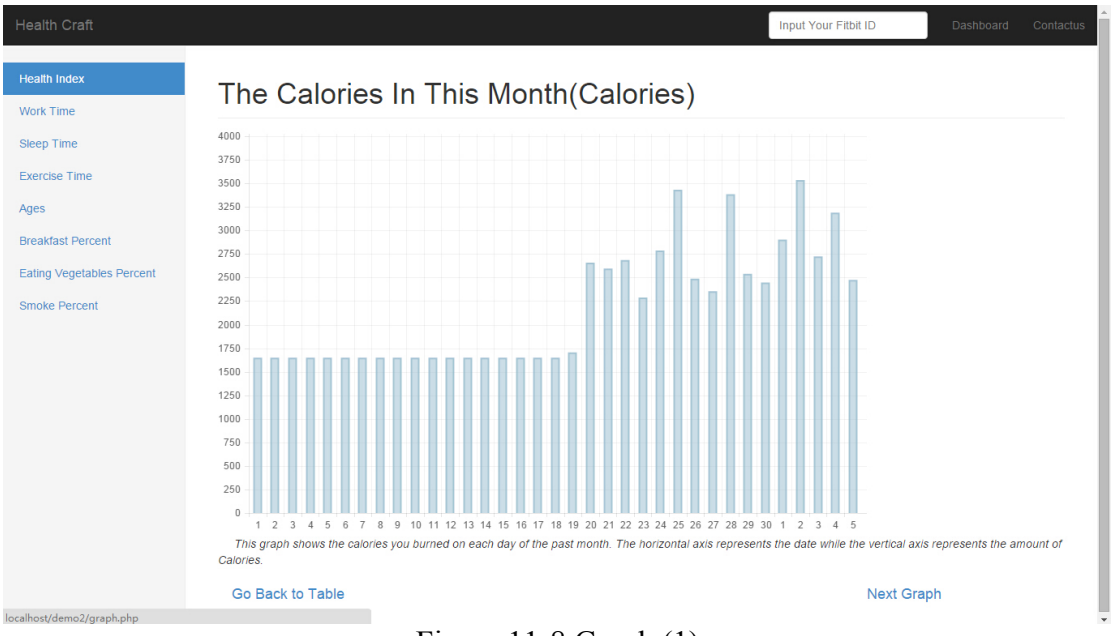


Figure 11-8 Graph (1)

(2) Health Index

After the user click the “Health Index” button, the website will show the Health Index of the user in this month.

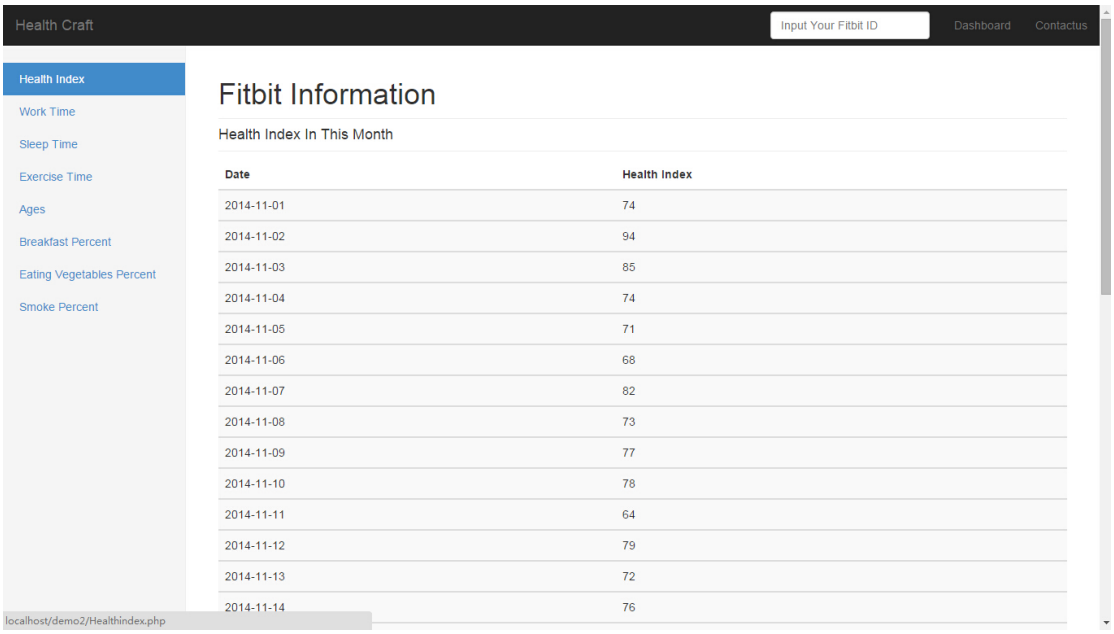


Figure 11-9 Graph (2)

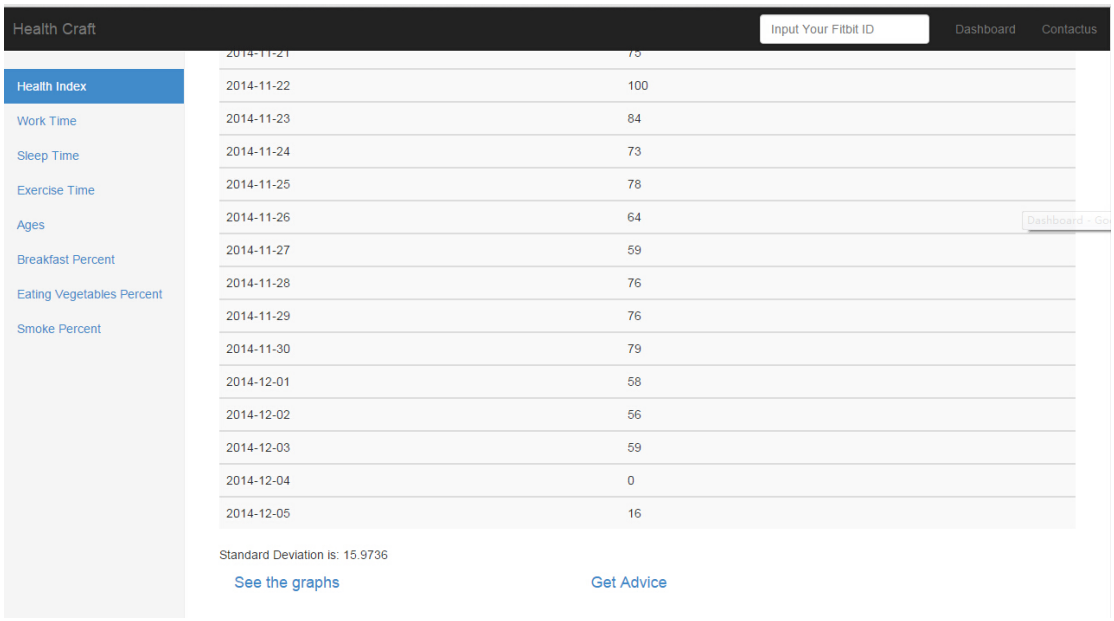


Figure 11-10 Graph (3)

And the user can click the button “See Graphs” below the Index. And the website will show a linechart of the index variation in this month.

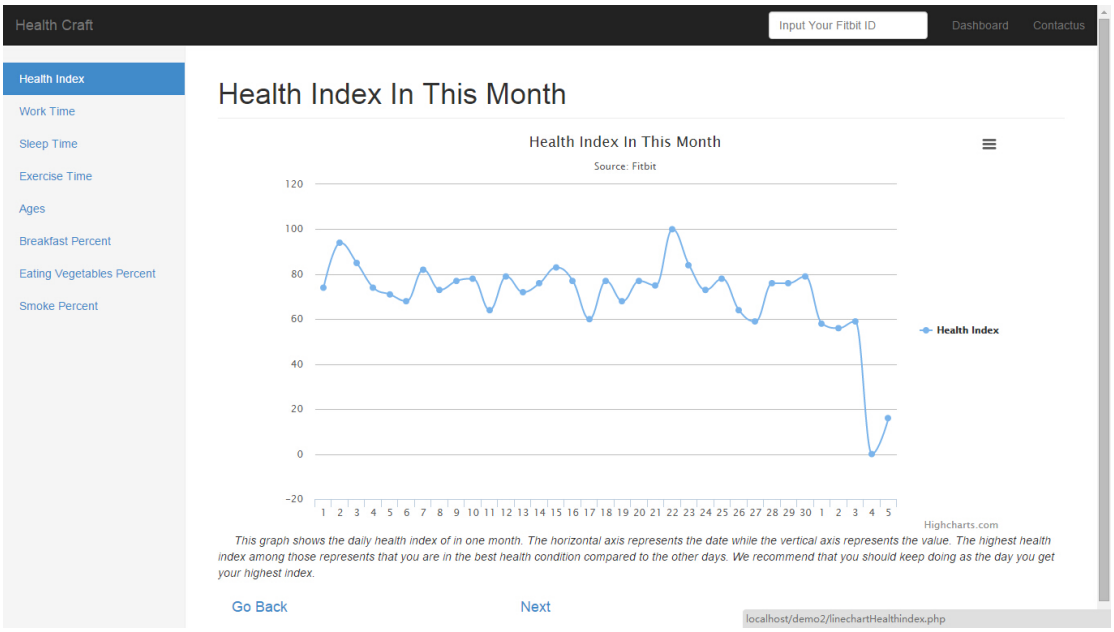


Figure 11-11 Graph (4)

After the user click the button “GetAdvice” below the Index. And the website will show a page of the user’s best activity situation and optimal activity management.

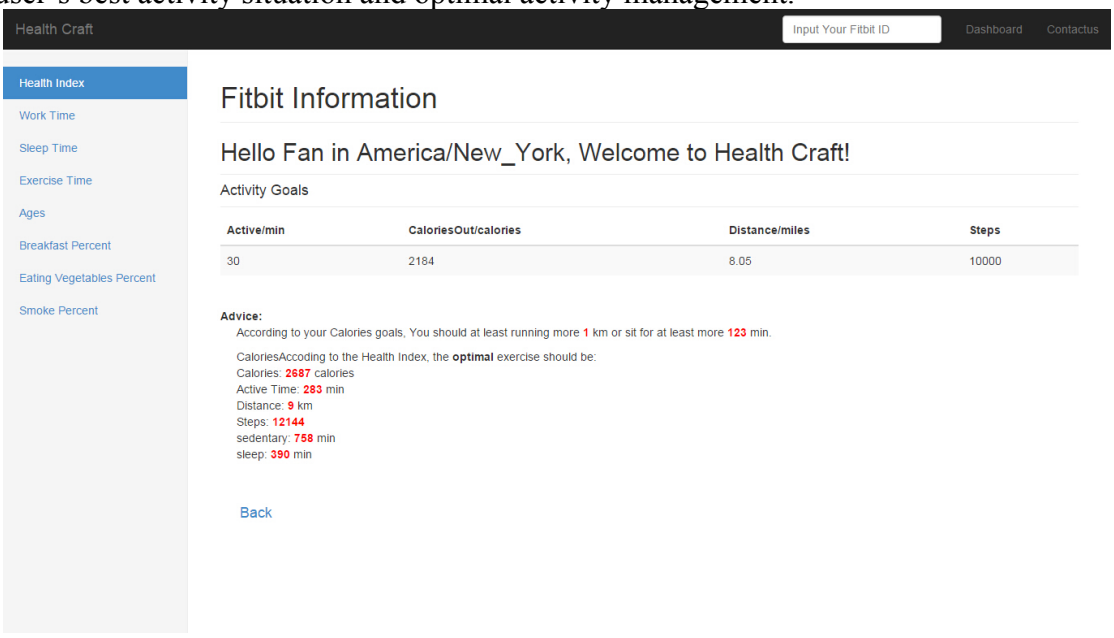


Figure 11-12 Graph (5)

(3) LineChart

The Line chart is drawn based on the user’s data and health index. It will clearly reflect the user’s health condition and show the relationship between these factors and the health indexs. And after the user click the “show Linechart” button, the line chart page will show up.

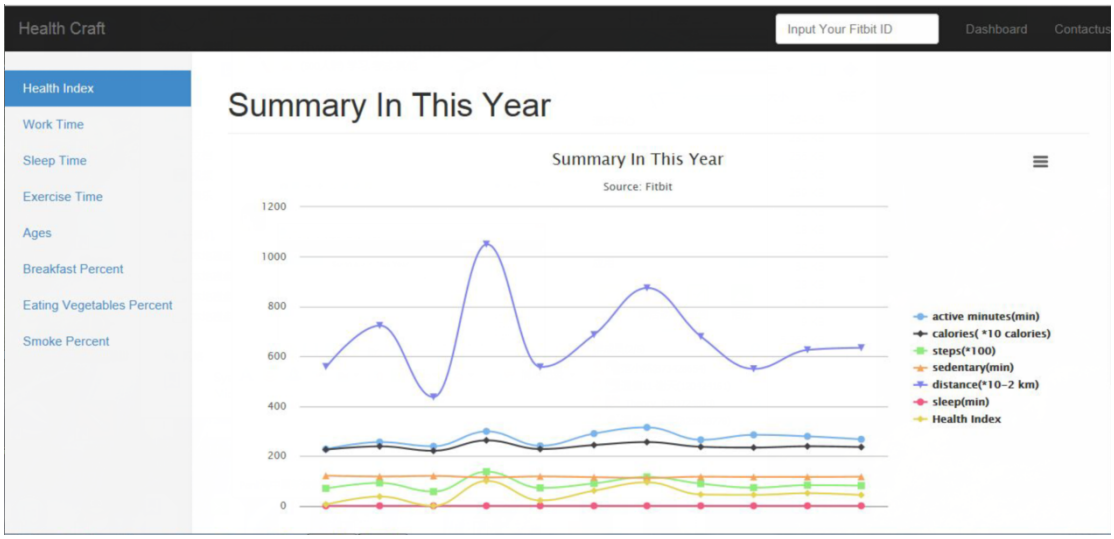


Figure 11-13 Graph (6)

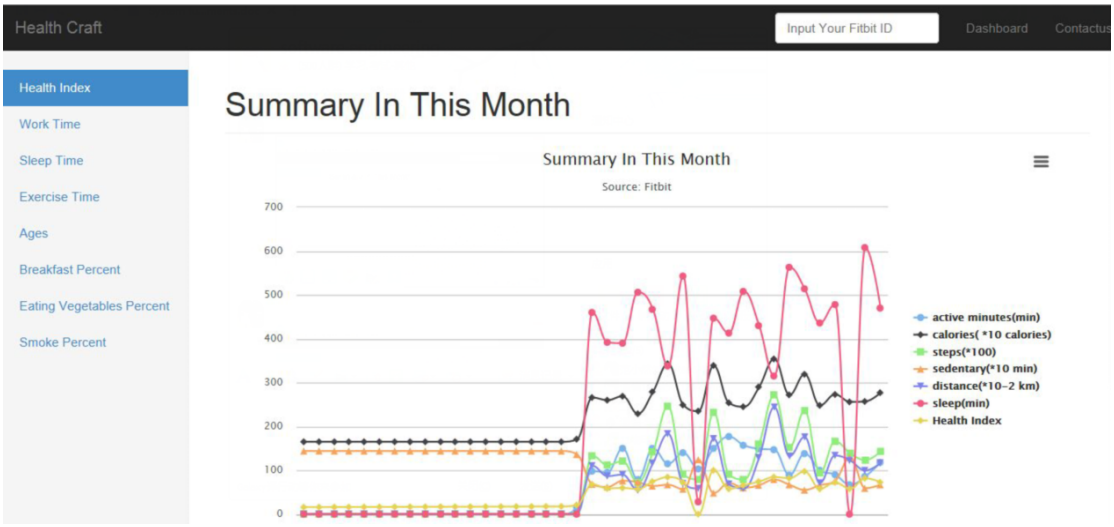


Figure 11-14 Graph (7)

(4) Radar chart

The radar chart will show the user in these 5 different factors, which one the user did the best and which one did the worst in the past.

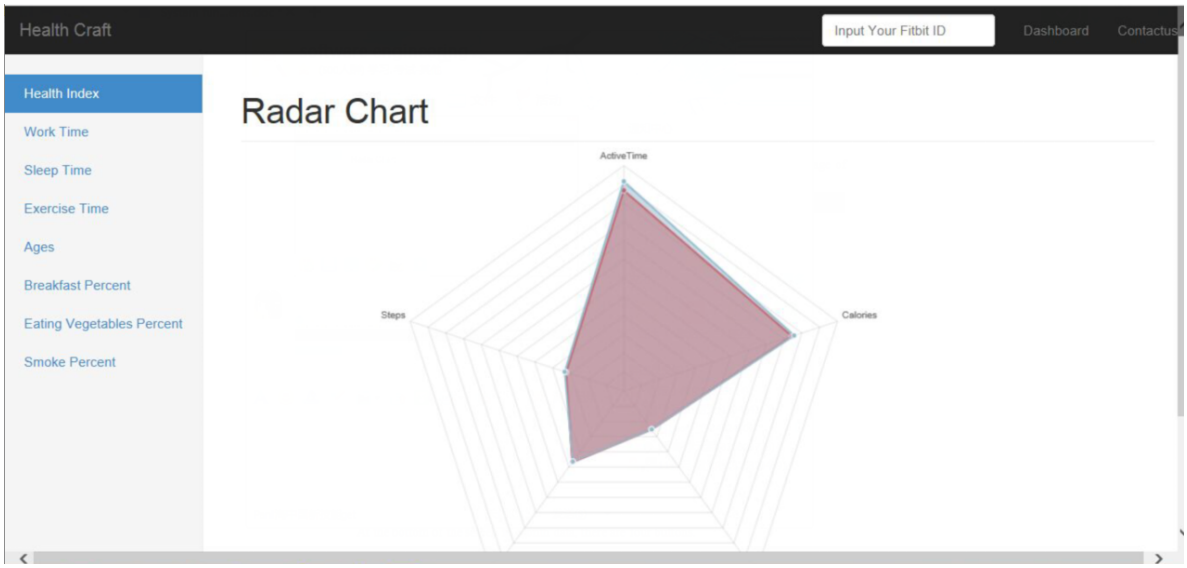


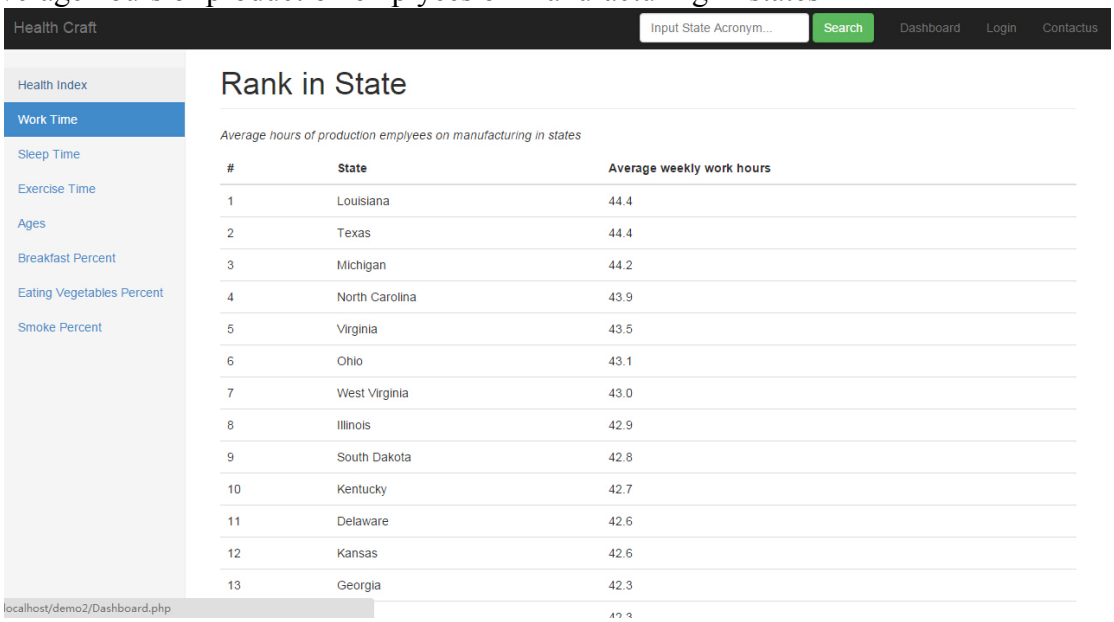
Figure 11-15 Graph (8)

4. Region statistics

The user can get the statistics in a certain state by choosing a tag on the dashboard

(1)Work Time

Average hours of production emplyees on manufacturing in states



#	State	Average weekly work hours
1	Louisiana	44.4
2	Texas	44.4
3	Michigan	44.2
4	North Carolina	43.9
5	Virginia	43.5
6	Ohio	43.1
7	West Virginia	43.0
8	Illinois	42.9
9	South Dakota	42.8
10	Kentucky	42.7
11	Delaware	42.6
12	Kansas	42.6
13	Georgia	42.3

Figure 11-16 Region Data (1)

(2)Sleep Hour

Age-adjusted percentage of adults who reported insufficient rest or sleep during the preceding 30 days

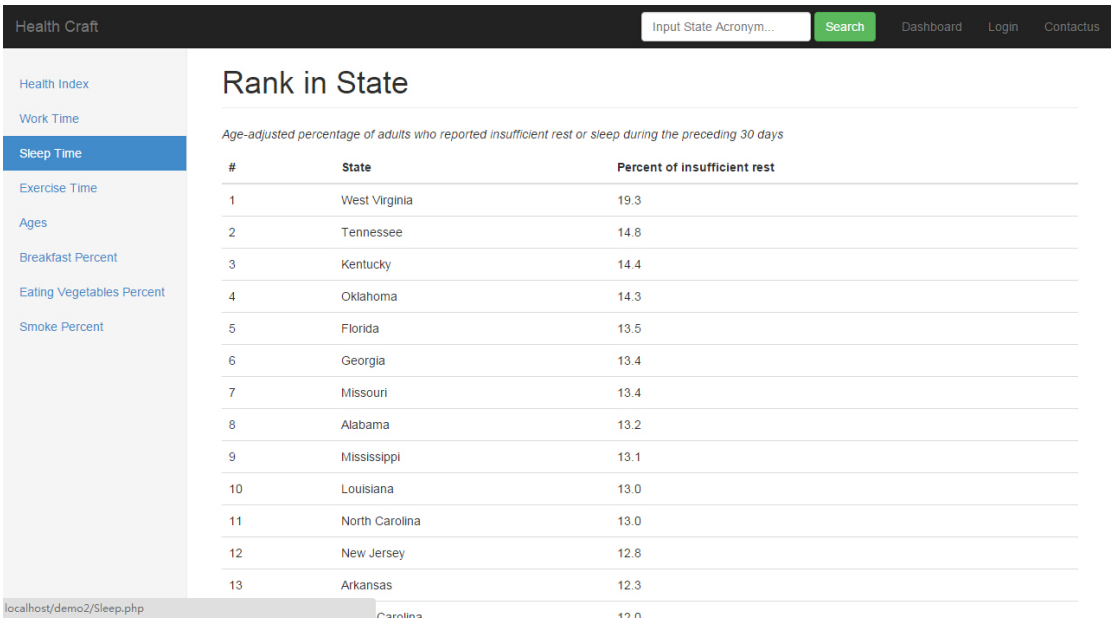


Figure 11-17 Region Data (2)

(3)Exercise time

The percentage of weekly exercising at least 30 minutes

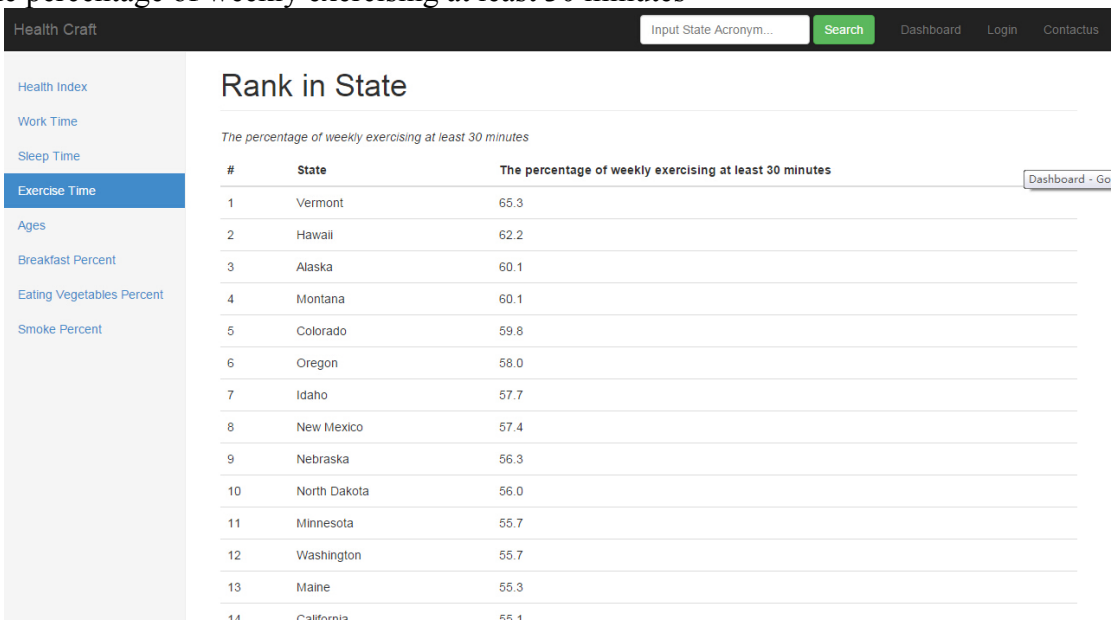


Figure 11-18 Region Data (3)

(4)Ages

Life expectancy at birth and by race/ethnicity in every state where the population of that racial or ethnic group is sufficiently large for robust estimates.

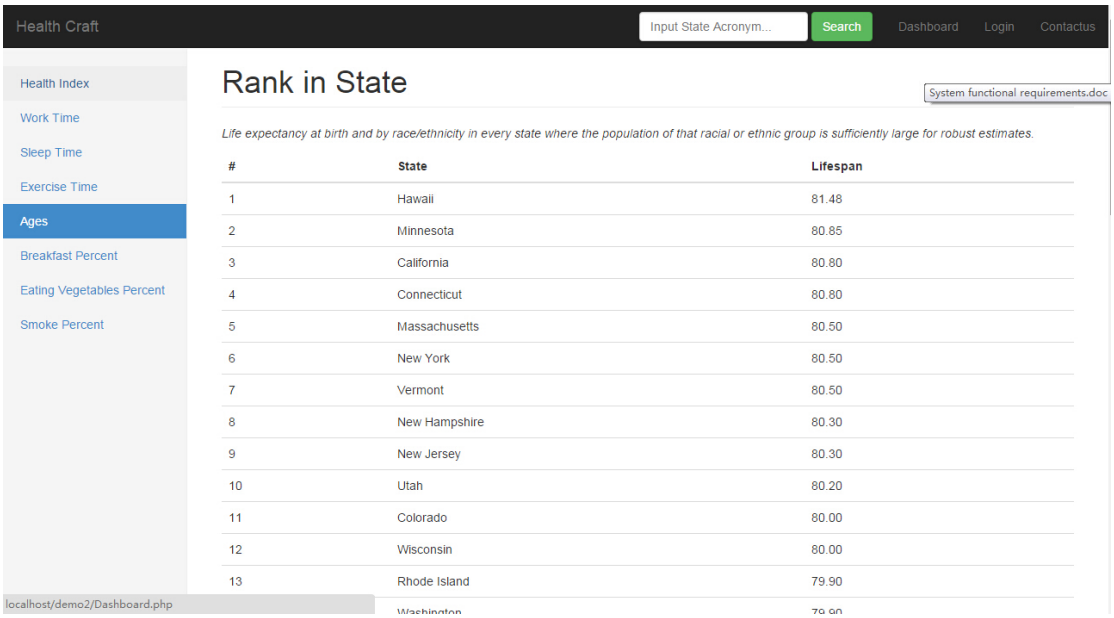


Figure 11-19 Region Data (3)

(5)Breakfast Percentage
The percentage of adults who eat breakfast

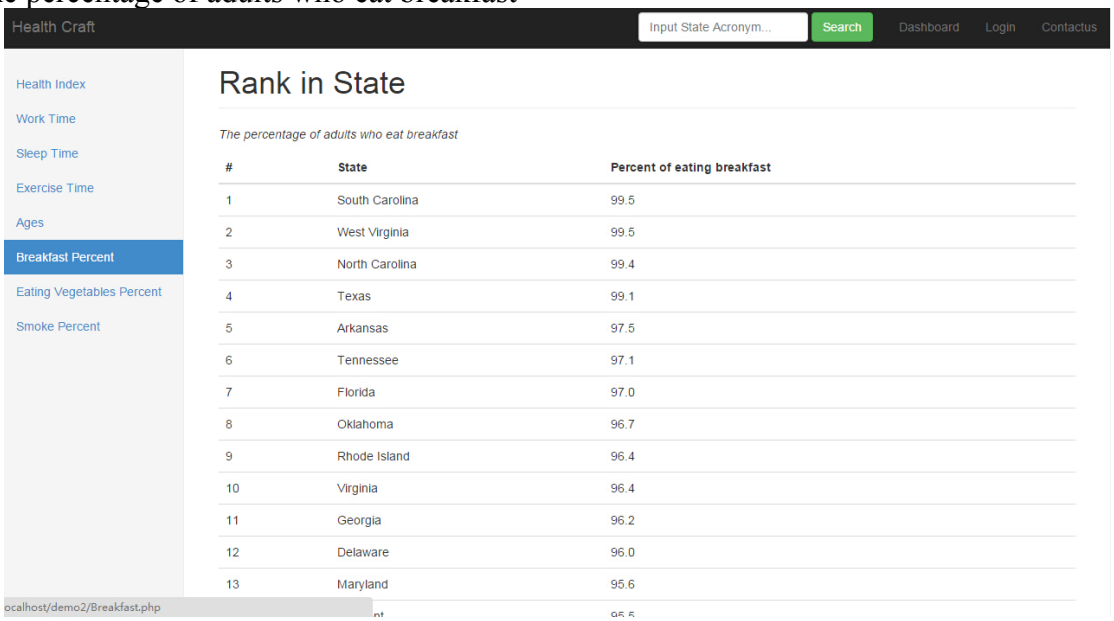


Figure 11-20 Region Data (4)

(6)Eating Vegetables Percentage
The percentage of adults who eat Vegetables

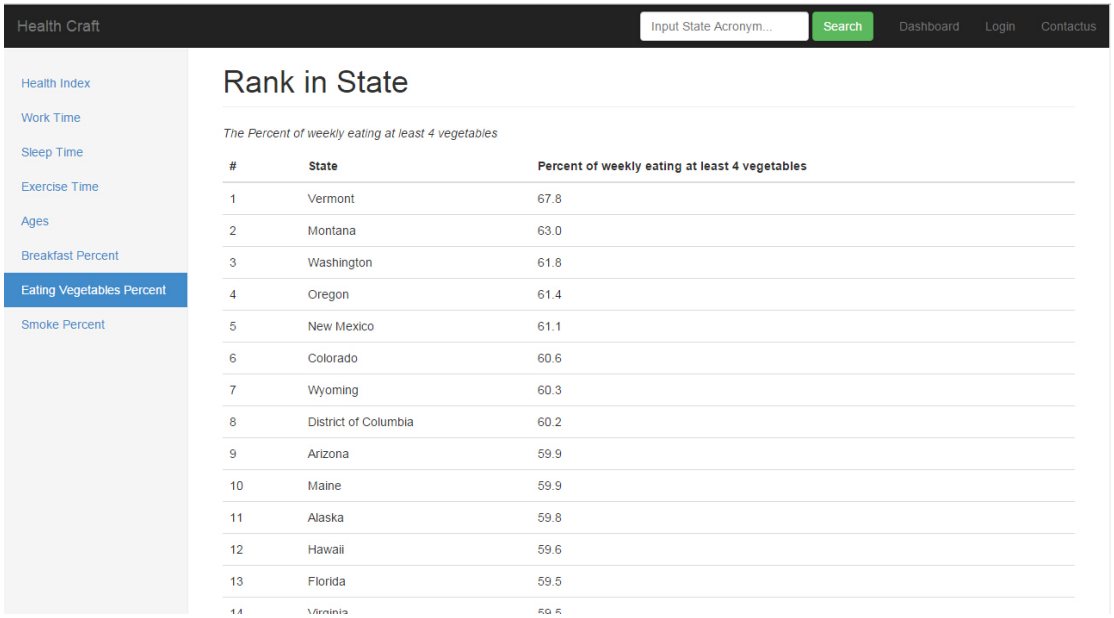


Figure 11-21 Region Data (5)

(7)Smoke Percentage
Adults who are current smokers

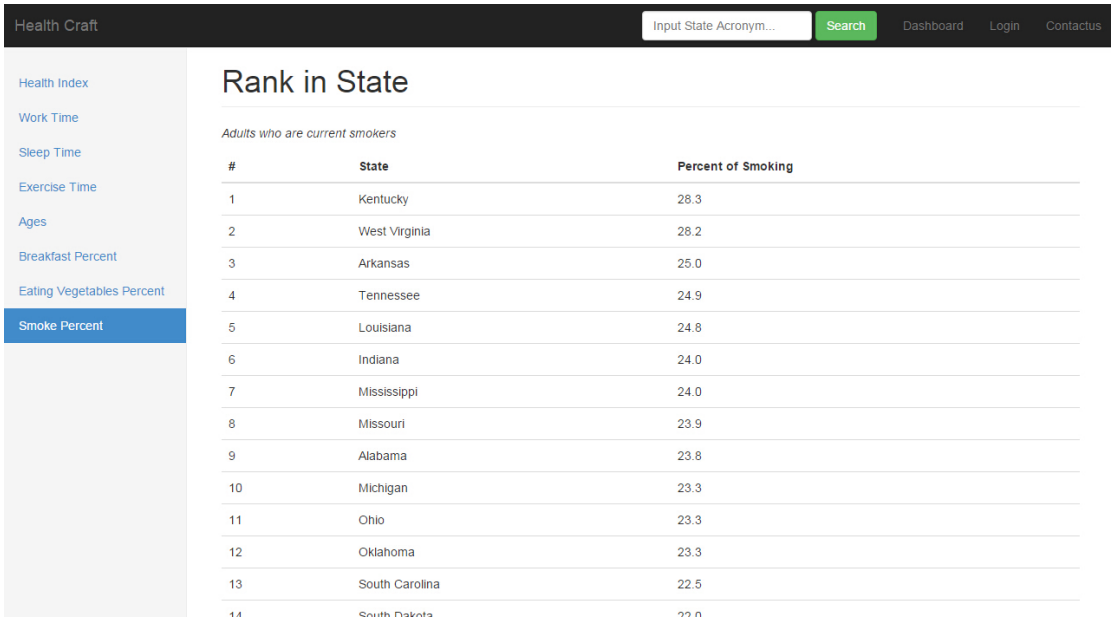


Figure 11-22 Region Data (6)

b. User Effort Estimation

UC-1 Getting Health Index Dashboard
Total 2 clicks and 1 keystroke

1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Health Index” Button.

UC-2 Viewing Activity Condition Dashboard
Total 1 click and 1 keystroke
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER

UC-3 Viewing Line Chart
Total 2 clicks and 1 keystroke
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Show LineChart” Button.

UC-4 Getting Standard Deviation Line Chart
Total 3 clicks and 2 keystrokes
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Next Summary in this Year” button.
5. Click the “Show Deviation” button.

UC-5 Getting Average Health Index Line Chart
Total 4 clicks and 2 keystrokes
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Next Summary in this Year” button.
5. Click the “Health Index” button.
6. Click the “Show LineChart” button.

UC-6 Viewing the Health Index Gauge
Total 2 clicks and 2 keystrokes
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Show Gauge” button.

UC-7 Getting Advice
Total 3 clicks and 2 keystrokes
1. Click the “Dashboard” button on the Home Page.
2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Health Index” button.
5. Click the “Get Advice” button.

UC-8 Calories Analysis
Total 3 clicks and 2 keystrokes
1. Click the “Dashboard” button on the Home Page.

2. Follow the instruction to open the access Fitbit and find the Code ID.
3. Insert the Code ID in the blank of the Dashboard, and press ENTER
4. Click the “Next Summary in this Year” button.
5. Click the “Calories Analyze” button.

UC-9 Searching Info
Total 2 clicks
1. Click the “Dashboard” button on the Home Page.
2. Click the different tag on the Dashboard.

12. Design of Tests

a. Class Tests

Goal: To test the basic function of the whole product, we plan to test the functions includes:

- 1. The establishment of MySQL database.
- 2. The connection of MySQL database with data downloaded from the website and display the data on the web page
- 3. Test the basic view of the web page
- 4. Test the activity condition dashboard function
- 5. Test the Line chart function
- 6. Test the Health Index Gauge function
- 7. Test the health index function
- 8. Test the getting advice function
- 9. Test the search health condition information function
- 10.Test the Calories Analysis function

The results:

- ✓ 1. The MySQL database created successfully in our computer
- ✓ 2. The data gathered from the website can be successfully store in MySQL database and display successfully in our webpage
- ✓ 3. The UI of the webpage show the way we design it
- ✓ 4. The activity condition information can be displayed on the screen
- ✓ 5. Different kinds of line charts can be shown based on user’s health condition
- ✓ 6. The health index gauge can be shown to the user
- ✓ 7. The health index can be calculated and displayed on the screen
- ✓ 8. The advice about user’s health information can be calculated and displayed on the screen
- ✓ 9. The health condition information in an area can be presented on

the screen

- ✓ 10.The user can view the recommended calorie burn each day and the personal calorie burn goal set by users themselves on the line chart

b. Functional Unit Tests

Below are the designs of the test cases as per required by our project. Every test case is listed the diagram

1. Test ID: TC1_The MySQL database establishment
2.

Input Requirement	Expected Output	Pass/Fail	Comments
The data	The Same data	Pass if the same data shown in our terminal	This test is to make sure that our MySQL database has already created

Table 12-1 Test Results

3. Test ID: TC2_The Display of the data from websit

13. History of Work, Current Status, and Future Work

a. History of work, Current status

1. Although there is a great change in our project after demo 1,we have successfully implemented most use cases from the report 3. Merging the Contributions from Individual Team Members One of the first problems we encountered. Merging the contributions from individual team members was really not an easy thing to tackle. One of the first problems we encountered was that when we had done our specified parts, everyone had their own part on a separate Word document. Almost everyone had a different format for their document. We spent a lot of time to uniform the format, the title, the figures and even the references. At last, we finally accomplished all this challenge and make the format uniformed.

2. Through the project, our team conquered several key issues in our way to progress. The first problem occurred when we start to crawling data from FitBit. We can only retrieve 30 requests using rest API per hour. After checking the streaming API, we figure out that we can use streaming data to continuously extract data from Fitbit. At last we can get enough data for our demo and analyze.

3. Another task is to build the DataBase and implement the connection between the data crawling program and the database. However, it is easy to store data in local database. We have to achieve the connection that data collection part can directly store data into database. So that two parts can work separately in two different computers. After we search the web and get the function name, the db object will be a connection to a DB server for the specified database . We just need to put the database's server IP into the function as the argument.

4. How to connect the database with webpage is also a challenge for us in the first stage. Thanks for XAMPP document, which provide detailed information for how we can get the database object and how to operate the database and how to get the 105

5. It is difficult to analyze data and how to synthesise variety of data we get from FitBit API into an index that can represent the user's health condition. At first we have tried principle component analysis to synthesise data into an index. It works but the result is not so good for PCA cut many useful data off, and we can only get limited data

from FitBit. After modification, we implement another mathematic method Factor Analysis to synthesise data. It works quiet well now, and the result is very convincible.

6. It is not easy to explain why our mathematic model is fit for this case ie.for health monitoring. We refer to many paper and books and finally find a book which tells us how to use factor analysis into health analysis. It is perfect match for our case.

7. It is not easy to code Factor Analysis program. Though there is the function in SPSS, we can't link SPSS function into PHP which build our whole project. So we used matlab code factor analysis ourselves and created exe link to PHP program.

b. Future work

We have finished personal health index function. We can tell our users how healthy they are by comparing data themselves. And our future work is to build a database to store the users' FitBit information into our database. When a user wants to know how healthy he or she is. He can know his or her health index in the group by comparing his or her health condition in implementing factor analysis.

We don't complete this function for this function need a database that can store many users FitBit information and we don't have many "users" trough out our project. Though they are many users in FitBit, it asks users to open their privacies and users' ID.

We only have 2 "users" now. Because our algorithm is based on big data analysis so we need a bigger group of users to make this function more convincing.

This function is easy and effective once we get more users in our project. The code and algorithm are almost the same. And we built database in our previous work. So we have experience in building database. The rest of work is not so tough and just a matter of resources.

14. Reference

1."Use case", Wikipedia
http://en.wikipedia.org/wiki/Use_case'
2."System requirements", Wikipedia
http://en.wikipedia.org/wiki/System_requirements
3."User interface specification", Wikipedia
http://en.wikipedia.org/wiki/User_interface_specification
4."Physical exercise", Wikipedia
http://en.wikipedia.org/wiki/Physical_exercise
5."Software Engineering book", Ivan Marsic
http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
6. Software Engineering Project: Health Monitoring Analytics
http://www.ece.rutgers.edu/ marsic/books/SE/projects/HealthMonitor/analytics.html
7."Workout with Friends -- Health Monitoring for Fitness Applications"
http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2012-g1-report3.pdf
8."A review of studies on expert estimation of software development effort"
http://www.idi.ntnu.no/grupper/su/publ/ebse/RK15-reviewexpertestim-jorgensen-jss04.pdf
9. Copy of Web iRis: Task - All-Easy 2
http://creately.com/diagram/example/goc8uhkx/Copy+of+Web+iRis
10. Monitoring the health of web page analytics code,
http://www.google.com/patents/US20110035486
11."Share Dialog", Facebook
https://developers.facebook.com/docs/plugins/share-button
12."Twitter Data Analytics", Shamanth Kumar, Fred Morstatter, and Huan Liu
http://tweettracker.fulton.asu.edu/tda
13. Software Engineering Project: Health Monitoring Analytics
http://www.ece.rutgers.edu/marsic/books/SE/projects/HealthMonitor/analytics.html
14.Kumar, Shamanth, Morstatter, Fred, and Huan Liu. Twitter Data Analytics. Springer,2013
http://tweettracker.fulton.asu.edu/tda/
15. Kenneth. M. Anderson, Aaron Schram, Design and Implementation of a Data Analytics Infrastructure in Support of Crisis Information Research (NIER Track)"
http://epic.cs.colorado.edu/wp-content/uploads/icse2011.pdf
16. Where DO Twitter F-Bombs Come From? Heat Map Shows Rudest Places
http://mashable.com/2012/08/22/twitter-rudeness-heat-map/
17. Healthcare SocialMedia Analytics,

http://www.symplur.com/healthcare- social-mediaanalytics/
18. Monitoring the health of web page analytics code, http://www.google.com/patents/US20110035486
19."Use case", Wikipedia http://en.wikipedia.org/wiki/Use_case'
20."System requirements", Wikipedia http://en.wikipedia.org/wiki/System_requirements
21."User interface specification", Wikipedia http://en.wikipedia.org/wiki/User_interface_specification
22."Physical exercise", Wikipedia http://en.wikipedia.org/wiki/Physical_exercise
23."Software Engineering book", Ivan Marsic http://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf
24. Software Engineering Project: Health Monitoring Analytics http://www.ece.rutgers.edu/marsic/books/SE/projects/HealthMonitor/analytics.html
25."Workout with Friends -- Health Monitoring for Fitness Applications" http://www.ece.rutgers.edu/~marsic/books/SE/projects/HealthMonitor/2012-g1-report3.pdf
26."A review of studies on expert estimation of software development effort" http://www.idi.ntnu.no/grupper/su/publ/ebse/RK15-reviewexpertestim-jorgensen-jss04.pdf
27. Copy of Web iRis: Task - All-Easy 2 http://creately.com/diagram/example/goc8uhkx/Copy+of+Web+iRis
28. Monitoring the health of web page analytics code, http://www.google.com/patents/US20110035486
29."Share Dialog", Facebook https://developers.facebook.com/docs/plugins/share-button
30."Twitter Data Analytics", Shamanth Kumar, Fred Morstatter, and Huan Liu http://tweettracker.fulton.asu.edu/tda