Personal Health Companion

Course: 16:332:567 Software Engineering

PROJECT REPORT 3

RUTGERS UNIVERSITY- FALL 2015

September 30, 2015

GROUP-3

https://github.com/shikha1990/HealthMonitoringAnalytics_Group3

Authored by:

AKSHITA AMBATI

HARIKA MATTA

JIANYU ZHANG

RUIQI LIN

SHIKHA KAKAR

YUEYANG CHEN

Table of Contents

Summary of Changes

Changes in REQs:

1. Deletion on REQ-10a, REQ-10b: The requirement is concerning about community based leaderboard. Our pattern of collecting user data is country-wise and state-wise which will not be able to support community accuracy.

2. Deletion on REQ-15, REQ-16: This requirement is basically for convenience of user who will use the websites for a long time. After reconsidering the work it takes and users real need for a website like this, we decided to remove them.

Changes in UCs:

1. Deletion of UC6: As the sequence result of deletion of REQ10, the UC6 is deleted.

2. Deletion of UC8: As the sequence result of deletion of REQ15 and REQ16, the UC8 is deleted.

3. Add a new UC13: The comparison based on geographic information. It is basically derived from UC3 with a little difference, and it is a new feature of our analysis.

4. Revise the UC3: We are able to show weekly tweets number of certain theme.

1. Customer Statement of Requirements

We will develop a product by improvising the features of the previous year's Health Monitoring projects. We will discuss the need for a health monitoring system and also discuss about our proposed solution which would be helpful to a wide range of users.

1.1 Problem Statement

1.1.1 Need for a Personal Health Monitoring System

Just one in 20 people worldwide (4.3%) had no health problems in 2013, with a third of the world's population (2.3 billion individuals) experiencing more than five ailments, according to a major new analysis from the Global Burden of Disease Study (GBD) 2013, published in The Lancet.[i] In other words, 95% of us in some extent have health problems attached to our body. Except for some people who have diagnostic diseases, others could be descripted as Suboptimal Health Status (SHS). Suboptimal health can be defined as a **state characterized by some disturbances in psychological behaviors or physical characteristics, or in some indices of medical examination, with no typical pathologic features,**[ii]which exactly characterize common situation that most of people are facing.

However, most of the people underestimate how severe this problem is. What they prefer most is a convenient and a leisure lifestyle. They can hardly realize whether their lifestyle is healthy or not until some diagnostic illness shows up. In some other cases, people know that they need exercise or that they need to follow good habits, but they pursue them irregularly and fail to put efforts on a daily basis. As a matter of fact, sub-health can pose a potential high risk to a variety of illnesses, and it will influence a person's life in every aspect. Some of them are chronic fatigue, distraction, memory deterioration and sleep disorder.

What was mentioned above are only some of the symptoms which precisely interpret the sub-health status. Essentially, the problem is caused by being accustomed to bad habits in the five aspects listed below:

1. **Long period of insufficient sleep**
   These days most people don't get enough sleep. People stay up all night to study, work, or have fun. Inadequate sleep has both short- and long-term consequences oh heath. Its effects can be seen in reduced efficiency and productivity, errors, and accidents.

2. **Irregular eating habits**
   Irregular eating patterns effects calorie burning, appetite, and hunger hormones in a human system. It can also create a health risk through a metabolic disturbance.

3. **Pressure from work or from academics**

Stress that continues for a long time can lead to a condition called distress which is a negative stress reaction. It can lead to many physical symptoms including headaches, stomach upset, high blood pressure, chest pain, and also sleeping problems.

4. **Lack of sport activities in the daily routine**
   Lack of exercise is the main cause of chronic diseases. Exercise plays a major role in protecting our health. Some physical activity is necessary to stimulate the body's own repair system.

5. **Smoking or Drinking alcohol**
   These activities cause multiple complications with the body that can range from mild to life-threatening. Smoking causes about 90% of lung cancers. Alcohol can affect the way the brain looks and works. Drinking and smoking too much can weaken your immune system, making your body a much easier target for disease.

According to the former professional study, eight causes of chronic disorders--mostly non-communicable diseases--affected more than 10% of the world population in 2013: cavities in permanent teeth (2.4 billion), tension-type headaches (1.6 billion), iron-deficiency anemia (1.2 billion), glucose-6-phosphate dehydrogenase deficiency trait (1.18 billion), age-related hearing loss (1.23 billion), genital herpes (1.12 billion), migraine (850 million), and ascariasis (800 million; giant intestinal roundworm).

If people want to improve those conditions, people should form complete and well-organized plans which can assist them in managing their daily schedule scientifically as well as efficiently. Besides, people also require motivation in order to maintain their interests on following the schedule. These are the reasons why our project would highly appeal to them.

1.1.2 Recommended Solution

We have discussed the importance and need of having a health monitoring system to make a change on personal habits so far. There are a lot of solutions which solve these problems. We have mentioned a few of them here. Motivation can help people to be more focused on their health. Groups can be formed and people can come with plans for combined activities.

1. For example, if a person is trying to get rid of his drinking and smoking habits, he will be more motivated to do so if he forms a group in which there are a lot of people working towards the same goal. People in the group who have overcome this problem also can advise the other people about how they have achieved their goal.

2. To motivate the people further, leaderboard competitions can be arranged and this would motivate people to contest for the top positions. A BBS can be built for the convenience for people who would like to share their experiences.

3. People who are trying to reduce their stress levels can get together with other people

who are also trying to do the same. Group activities can be organized for these people and psychiatrists can join the meetings to give some tips on how to overcome extreme stress levels.

4. People can be encouraged to write comments and suggest tips to their friends in their area or community.

5. Providing a one-stop information source makes people to have an easy access to the trending healthy food habits, tips for leading a stress-free lives, instead of them having to surf through multiple sources on the internet.

6. The Best way to motivate people is to statistically show all the data on the dashboard with the latest trends of, for example how the smoking/alcoholic beverages lead to how many health problems, and by doing something as simple as exercise can affect the overall health.

1.1.3. Project Description

The purpose of the project is to gather data from the web sources and analyzing the gathered data and reusing all the three projects from year 2014 and adding the features which are not covered in it, we will using machine learning for statistical analysis.

We will extract data from Twitter and analyze those tweets based on the list of hashtags we think are relevant. Then, we will classify the tweets and come up with trend charts. The details on what algorithms we will be using for data analysis, clustering and tweet analysis are mentioned in the next few sections of the project proposal. We will use the analysis as mentioned in the functional features and will display on the website as a trend or graph.

Our solution is to help people in three ways:

1. Out prior consideration is to help users realize that they might be in sub-health status. We will provide basic data in diagram to show this problem intuitively so that users will realize the severance of sub-health. We will provide an optional quick health investigation to find out the possibility of users in sub-health status as well.

2. We will provide some more careful and specific suggestions for the users who are willing to go deeper. We focus on population of particular geographic locations on the basis of their daily habits like:
   - **Sleeping habits**
   - **Sporting habits**
   - **Eating Habits like content of sugar, fat and carbohydrates (Junk food)**
   - **Smoking Habits**
   - **Alcohol Habits**

- **Stress-Pressure faced by the students, at workforce**

3. In addition to the purely data analysis and hollow suggestions, motivation can help people to be more focused on their health spontaneously. That is what we would like to provide.
   - People can be encouraged to write comments and suggest tips to their friends in their area or community. A BBS can be built for the convenience for people who would like to share their experiences. For example, People who are trying to reduce their stress levels can get together with other people who are also trying to do the same. Group activities can be organized for these people and psychiatrists can join the meetings to give some tips on how to overcome extreme stress levels.
   - To motivate the people further, leaderboard competitions can be arranged and this would motivate people to contest for the top positions. People can see the achievement from every significant step they make through the leaderboard.

1.1.3.1 Why Twitter?

The data can be collected from any of the social media websites like Facebook, twitter, Instagram etc. and can also be collected by the Sensors such as Fitbit etc. We are focusing on collecting the data from **Twitter**. Collecting enough data about people's health, fitness or exercise condition is the first and essential step of our project. There are two possible ways to help us handle this, professional sports websites and Social Networking Service (SNS). However, the data gets from professional sports websites can be limited for reasons as below:

1. Agreements from all manufacturers of body monitoring devices are required before collecting necessary information.
2. Device ownership is so coarse that may do not provide enough basic information, and statistics available information from device becomes even more of a privacy issue.
3. When relying on gym data, it is feasible to obtain such data only from local gyms, which can impact veracity and validity.

As a result, we choose SNS to implement data collection. And among variety of powerful SNSs such as Facebook, Google+, Twitter, etc., we decide to use Twitter. Comparing with other SNSs like Facebook, Twitter has an asymmetric network infrastructure of "friends" and "followers", which means people is able to scan tweets of every user even if they are not following each other, and this feature makes it more proper for us to get adequate and useful data. Besides, more than 100 million users who in 2012 posted 340 million tweets per day. And the number is still rising, because until May 2015, Twitter already has more than 500 million users, out of which more than 302 million are active users around the world. Therefore, we can get every information and data we want from Twitter. These are the reasons why we choose Twitter.

1.1.4. Vision

Our team has decided to build the following features for the product, after analyzing the work done by the groups of previous years'. We will be using some of the existing infrastructure for

data collection. On top of the existing features, we have decided to add enhancements to the data analysis by widening our scope of analysis and by taking other inputs into consideration for more meaningful results. Here are some features of our application:

1. The website will display the statistical trend showing the number of people who exercise in a given area, analysis will also include the percentage of people who exercise out of their areas population.
2. Data analysis will be done based on different age groups, gender and occupation and different demographic locations.
3. Data analysis on how often and how intensely people exercise will also be done.
4. Ranking of the popular exercises on the basis of how frequently, intensely, time duration and geo-location involved. The trend will also show different proportions of different exercises like running, walking, biking, hiking, swimming, yoga or cardio.
5. Trend showing for around which area people complain about health care/health issues. Ranking the areas on this criteria.
6. Correlation positive or negative between the groups that exercises and the group which constantly tweets about health and wellness. Checking correlation about people who are concerned for diet/foods on the basis of people who exercise, discuss about dieting and on the basis of gender.
7. Feeds or topics that appear common to people who exercise, watch on their diet and follow a healthy lifestyle.
8. In order to keep people motivated they will get healthy dietary suggestions, latest feed on exercise, they can keep notes and share it with friends and can put reminders for their exercise routine.
9. Ranking of the popular sports on the basis of how frequently, intensely, time duration and geo-location involved. The trend will also show different proportions of different sports.

We will be focusing our data analysis mainly on weekly trends. After tweet analysis and classification, we would like to do the following things:

1. **Weekly Diet Trends**: Classify tweets based on people who are following a healthy diet, people who are following unhealthy diet and people who are motivated to eat healthy diet and plot the weekly trends and histograms.
2. **Weekly Exercise Trends**: Classify tweets based on people who are doing exercise and going to gym, and people who need motivation to go to gym and exercise. The trend chart will be plotted.
3. **Weekly Alcohol Consumption Trends**: Classification of tweets will be done based on people who consume alcohol, people who want to quit alcohol and who need motivation.
4. **Weekly Smoking Trends**: Classify tweets based on people who smoke, people who want to quit smoking and need motivation.
5. **Weekly Sleep Pattern Trends**: Classify and analyze tweets based on people who have a sleeping problem, and people who are looking at medication/ other ways of resolving their sleep issues.

6. **Weekly Stress Level Trends**: Classify tweets based on people who are over-stressed and people who are looking at some medication/ and other ways of reducing their stress levels.

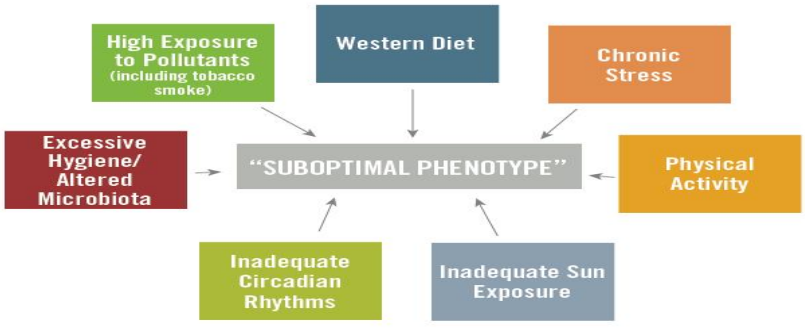
We will do all these trend analysis based on gender, geographic locations, etc. as mentioned earlier.

The earlier projects from last year have done some analysis of tweets based on the number of calories burnt to provide some visualizations. But, they haven't done it based on the geo-location or geo tags in Twitter. The main idea behind including analysis based on geo-location and geo-tags would be to determine the number of calories burnt for specific populations and reporting habits. Geo-location would be really useful in doing this analysis and the previous group haven't done their analysis based on this because only 1% of the tweets published on Twitter are geotagged. The only way to determine the tweet's location is by using the profile information. We will be using an API to extract the geo-locationfor improving the existing analyses.

Also, the previous projects haven't done much analysis on smoking and alcohol consumption trends. This is also something which we will be focusing on. We will also do data analysis on sleep patterns and stress levels.

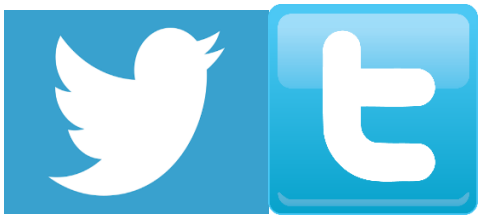
1.2. Glossary of Terms

1. Suboptimal health
   Suboptimal health can be defined as a state characterized by some disturbances in psychological behaviors or physical characteristics, or in some indices of medical examination, with no typical pathologic features,[iii] which exactly characterize common situation that most of people are facing.

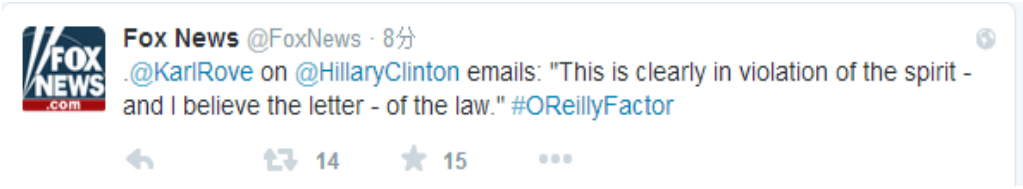


Carrera Bastos P, Fontes-Villalba M, Picazo O, Cordain L, 2014

2. Twitter
   Twitter is an online social networking service that enables users to send and read short 140-character messages called "tweets".



3. Hashtags
   A hashtag is a word or a phrase prefixed with the number sign ("#"). It is a form of metadata tag. Words or phrases in messages on micro blogging and social networking services such as Facebook, Google+, Instagram, Twitter, or VK may be tagged by entering "#" before them, either as they appear in a sentence or appended to it. The term "hashtag" can also refer to the hash symbol itself when used within the context of reciting a hashtag. (From Wikipedia)



4. Calories
   Calories are used as a judgement of how long or how much have you exercised. In this project, we are going to translate both the time and the sports type user have done to Calories.

5. Social Networking Service (SNS)
   A social networking service (also social networking site or SNS) is a platform to build social networks or social relations among people who share similar interests, activities, backgrounds or real-life connections. (From Wikipedia)

6. Application Programming Interface (API)

In computer programming, an application programming interface (API) is a set of routines, protocols, and tools for building software applications. An API expresses a software component in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. (From Wikipedia)

7. Server and database
   A server is a running instance of an application (Software) capable of accepting requests from the client and giving responses accordingly. A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring information. (From Wikipedia)

8. Yoga
   Yoga is the physical, mental and spiritual practices or disciplines that aim to transform the body and mind.
9. Automatic Tweet: A message sent using Twitter that has been created automatically by devices like mobile apps and wearable devices, not human beings. In our project, the automatic tweets always appear with the hash-tag “#LoseIt” and keywords “calorie, burn”.
10. BBS
    Bulletin board system, or BBS, is a computer server running custom software that allows users to connect to the system using a terminal program. Users can read news and bulletins, and exchanging messages with other users through public message boards.

2. System Requirements

As we discussed in Customer Statement of Requests and our analysis of data collecting, the reason we choose twitter for public information source is obvious now. The information amount is now adequate and the method for data sifting becomes the question.

First of all, the whole concept for our system is data mining. In another word, the question for the system is how to retrieve what users most concern about. On the next stage, the question is how we show our results of analyze to users in most intuitive way. In addition, there is also a concern about how users use our system and get relatively information they want. The following discussion will give more trivial and accurate requirements of our system.

2.1. Enumerated Functional Requirements

In this section, we discuss what exactly our system should do for users in its whole lifeline and split them into specific cases.

Identifier	Brief Description	PW
REQ1a	The system shall attain real time tweets from twitter using stream API	**5**
REQ1b	The system shall acquire the historical health-related tweets the users sent in a certain period. (e.g. in a month or two)	**5**
REQ1c	The system shall attain basic information from tweets of users which are health-related. (E.g. location, personal tweets numbers related, followers,	**5**
REQ2	The system shall be able to access geographic information associating with Google Map.	**4**
REQ3	The system shall have a database to store whole stream of data from twitter	**4**
REQ4a	The system shall filter out irrelevant tweets in database	**5**
REQ4b	The system shall classify relevant tweets data from database by different sets. (E.g. types of exercise, locations, genders, types of foods, etc.)	**5**
REQ4c	The system shall estimate twitter users workout calories by approximate calculation.	**3**
REQ5a	The system shall provide graph to display the exercising trends.	**3**
REQ5b	The system shall provide graph to display the smoking trends.	**3**
REQ5c	The system shall provide graph to display the alcohol consumption trends.	**3**
REQ5d	The system shall provide graph to display the dietary habit trends.	**3**
REQ6	The system shall provide approximate real time tweets rolling the screen.	**3**
REQ7	The system shall do the sentiment analysis by evaluating corresponding mood state the tweets show.	**2**
REQ8	The system shall provide graphs based on geographic information and show the dietary, smoking alcohol and exercise trends	**4**
REQ9	The system shall provide dynamic hashtag cloud to show tweets users are interested in.	**3**
~~**REQ10a**~~	~~The system shall provide an approximately real time leader board on community basis.~~	~~**2**~~
~~**REQ10b**~~	~~The system shall be able to show top 10 ranks for work out aggregated for 3days.~~	~~**2**~~

REQ11	The system shall have the ability to draw information of big events correlating the themes users care about from twitter and present on a calendar.	**2**
~~REQ12~~	~~The system shall provide a calculator for users to show their calories loss.~~	~~3~~
REQ13a	The system shall allow user to sign up and provide an alternative option of sign up using a third party API.	**5**
REQ13b	The system shall allow registered user to login.	**5**
REQ13c	The system shall allow guest users to visit.	**5**
REQ13d	The system shall allow login users to edit/delete their profile.	**5**
REQ13e	The system shall allow user to stay login status in a moment for convenience.	**1**
REQ14	The system shall define authorizations for different class of users. (e.g. Registered user, Guests, Administrators)	**5**
~~**REQ15**~~	~~The system shall allow user to invite their friends to join the community.~~	~~**2**~~
~~**REQ16**~~	~~The system shall provide a platform (like BBS) for users to share experiences and their comments as well as making friends.~~	~~**2**~~
REQ17a	The system shall provide users database to store user information and retrieve data from them.	**3**
REQ17b	The system shall separate users according to their public information like ages and form groups of common interest.	**3**
REQ18a	The system should be able to collect and save relevant comments.	**2**
REQ18b	The system should be able to analyze the comments and categorize them for correlation purposes.	**2**

In the above table, we list all functional requirements in three aspects as mentioned above. Some requirements divided into several parts with suffix like “a” or “b” are associated with each other.

REQ1 to REQ3 are requirements for data collections. In this part, the system needs to attain less more relevant information from tweets and associate the data with geographical information from google map.

REQ4 concerns all the data processing and analysis. The method we use for classification is a kind of learning machine and for data sifting part we use another algorithm for recognition of tweets’ language.

REQ5 to REQ12 are basically our form of data output to users. As mentioned in the table, the functionalities include graph display from analyzing big data(REQ5), real time relevant tweets displaying what people are talking about(REQ6), recommendations from estimating the average level(REQ7), trends for users based on exercise, smoking and dietary trends(REQ8), tag cloud to show up some hashtags which users can simply select as their wish and get relevant tweets(REQ9), leaderboard displaying ranks listing with various themes and it would be

additional to this system(REQ10), a calendar which provides information of big events abstracted from tweets' data (REQ11) and a calculator for calculating calories loss in convenience(REQ12).

The remaining (**REQ13 to REQ18**) represents the way we design for users to easily get the hang of our system as well as the interactions between system and users. What's needed to be note is **REQ14** which provide our design of authorization for different classes of users. Detailed speaking, as for guest users, they can get access to data analytic graphs and tag cloud for tweets; as for login users, they are allowed to get into BBS to share information and dig deeper in data mining results so as private recommendations. They are also permitted to build their own database and save their own workout with highly security; as for administers, they have the privilege such as locking users account which are under suspension and maintaining the running of system but no access to private database.

2.2. Enumerated Non-Functional Requirements

In this section, the following requirements are raised up basing on FURPS standard. Since functional requirements have been discussed on former section, we simply come up with few things about non-functional requirements applying the standard of FURPS.

Identifier	BriefDescription	PW
NF-REQ1	System must be easily run with few maintenance and quickly recovered if it is broke down for some reasons.	5
NF-REQ2	System must have a proper schedule for regular maintenance.	5
NF-REQ3	System should entirely protect privacy issues of users and anyone should never be allowed to access others detailed information.	5
NF-REQ4	System should be aesthetically designed in order to attract more users or stoppers-by.	5
NF-REQ5	The system server should have the ability to handle overload in site visits.	5
NF-REQ6	System should provide FAQ & efficient response for users (aka User-Friendly Design). In this part, users should be at ease when getting information and explanations about their concern.	5

In the above table, **NF-REQ1** interpret the reliability of system; **NF-REQ2** shows requirements in supportability;**NF-REQ3** is most important thing for users and it applies the standard of functionality.**NF-REQ4** concerns demand for usability; **NF-REQ5** is supportability;**NF-REQ6** represents the performance. The main idea in this section is to make users willing to stay with safety and comfort and what is worth noting is that all the rules should be applied to all stages of design and be pursued carefully.

2.3. On-Screen Appearance Requirements

One of the two subgroups worked on HTML and the other one worked on the Photoshop. For HTML screens we will be keeping them as the base and will add the Photoshop ideas in the existing code. Following are the screen of rough HTML:

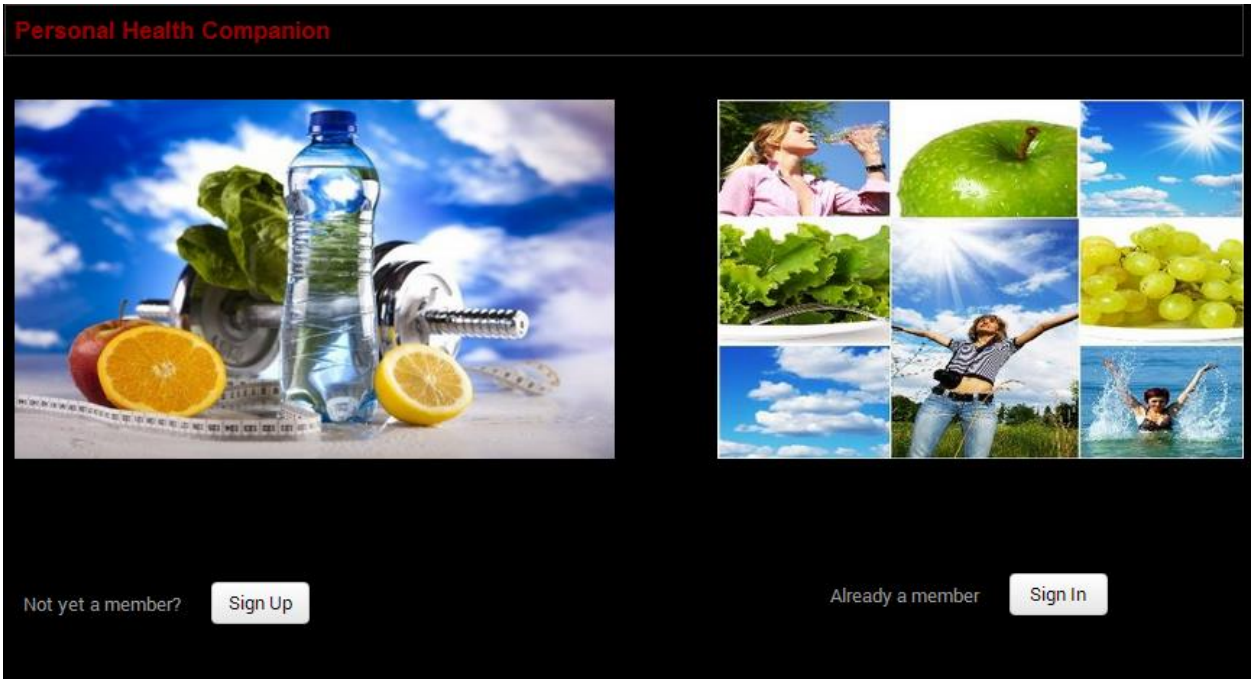


Figure 1

At the first page of our website, we want to give users a clean and intuitive view of what is this website for. We need to hide some information on the interactive layer. All kinds of features are in the icon on top and when user move the mouse to the icon it will show classifications. When users roll down the page, a big interactive tag cloud is designed and users can still point the mouse at each icon and the website can show a tiny window broadcasting relevant tweets filtered from database. Beyond all these, the information of website builders and terms of uses can be reached with relative links.

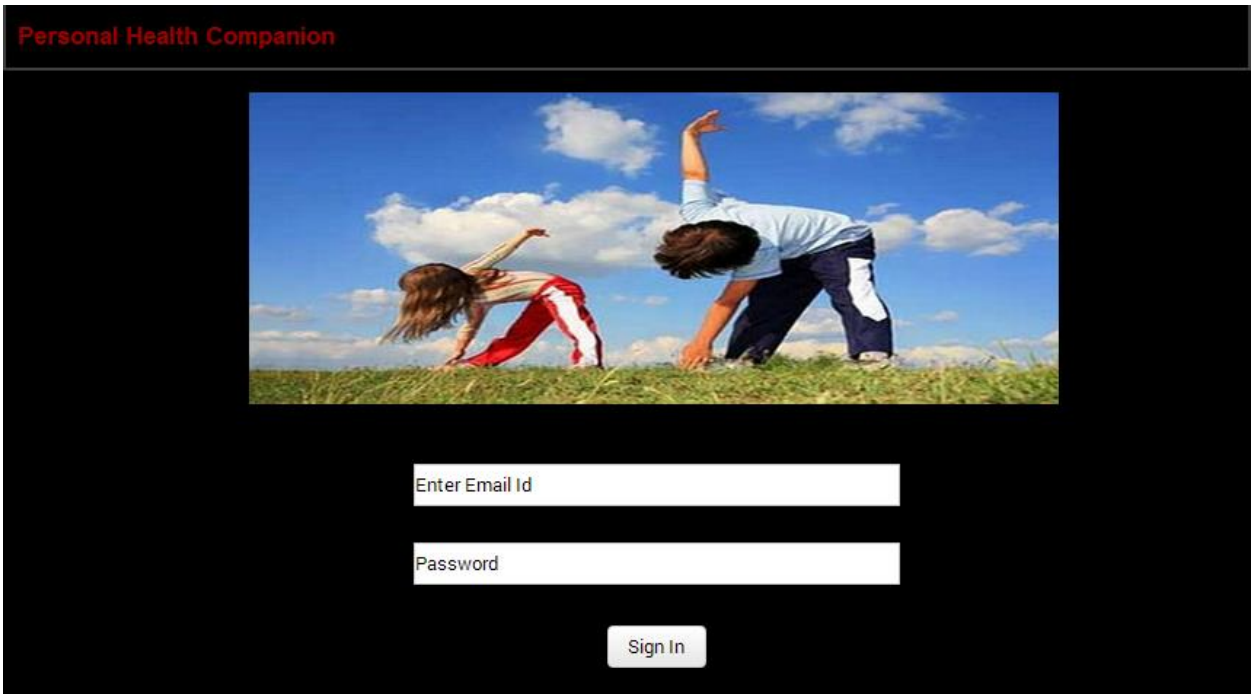


Figure2

The login page is also simple but tidy. One thing worthy of attention is that we provide third-party sign up with twitter accounts. If users do not want to go deeper, they can still find specific interests on the top line.

Figure3

When user get inside the specific feature, they can get relevant graphic data trends and the heat map linking to Google Map. Users will find out the sporting status around people in their community for example. They can also get recommendations from data analysis of system. In addition, the BBS service is also provide at any pages with health feature.

3. Functional Requirements Specification

3.1 Stakeholders

1. Academic Researchers& Health Organizations

   Define as third parties which might have interests on our data & analysis for academic use.

2. Individuals

   Define as customers, who want to use our system to get the information of their community or general health status of any aspects. They play the most important role in design of system.

3. Developers

   Define as the system designer, developer and administrator, who will only maintain the system on this stage and ensure the security level. Also, they might concern about the data analysis.

4. Enterprises

   Define as third parties which might gain information relating to health for business use. Some of them might use the system only to collect data while some others might use it to broadcasting their advertisement.

3.2 Actors & Goals

1. Users (Initiating Type)

   Goals: Viewing the information published on website & interacting with the system to get what they most concern about. Sharing their ideas and experiences and making friends.

2. Administrators (Initiating Type)

   Goals: Maintenance the system, protecting privacy issue, priority to collect data and access database, providing service to user.

3. Enterprises (Initiating Type)

   Goals: Collecting original data and analyzing. Advertising their business on the website or only investigating the primitive needs for market through data.

4. Databases (Initiating Type)

   Goals: Storing the information collected from SNS & Google Map, retrieving the information from it for relative use.

5. SNS (Participant)

   Goals: Providing source of data for all the analysis.

6. Google Map (Participant)

   Goals: Providing geographic information support for heat map.

3.3 Use Cases Casual Description

Use Case	Case Name	Description	Requirements
UC-1	Register	The system shall allow user to register, and create his profile	REQ13ab REQ14,REQ17a
UC-2	View/Edit Personal Profile	System shall display the personal profile for the logged in user.	REQ13c
UC-3	Show Graphs	System shall show users graphs of several aspects(e.g. heat map, tweet numbers in a period)	REQ2, REQ4ab, REQ5abcd,REQ8,REQ9, REQ10ab
UC-4	Display Calendar	System shall be able to support a calendar and allow uses to register for events.	REQ4ab, REQ11
UC-5	Show Calculator	System shall be able to support workout calculator on the home tab.	REQ4abc, REQ12
~~**UC-6**~~	~~Show Leaderboard~~	~~System shall calculate # of tweets collected per user and show the leaderboard in the UI~~	~~REQ4ab~~ ~~REQ10a,10b~~
UC-7	Show real-time Tweets	System shall be able to show real-time tweets with tags provided by user	REQ4ab, REQ6
~~**UC-8**~~	~~A BBS section~~	~~System shall provide users a platform for communication~~	~~REQ16~~

			~~REQ18ab~~
UC-9	Collect Twitter Information	System shall collect and store the tweets with a given region.	REQ1abc REQ3, REQ4ab
UC-10	Collect User Information	System shall collect User Information to perform draw the necessary conclusion.	REQ17ab
UC-11	Validate User Login Information	System shall validate when a user attempts to login.	REQ14
UC-12	Login	Once authenticated by system, User's status shall change.	REQ3
UC-13	Comparison based on geo Info	The system shall be able to show the comparison for different geographic information.	REQ2, REQ4ab, REQ5abcd,REQ8,REQ9, REQ10ab

According to the table, we mainly derive the use cases which are directly interacting with actors. These use cases might not be detailed enough, but they fully covered the features which we would like to implement.

3.4 Use Case Diagram

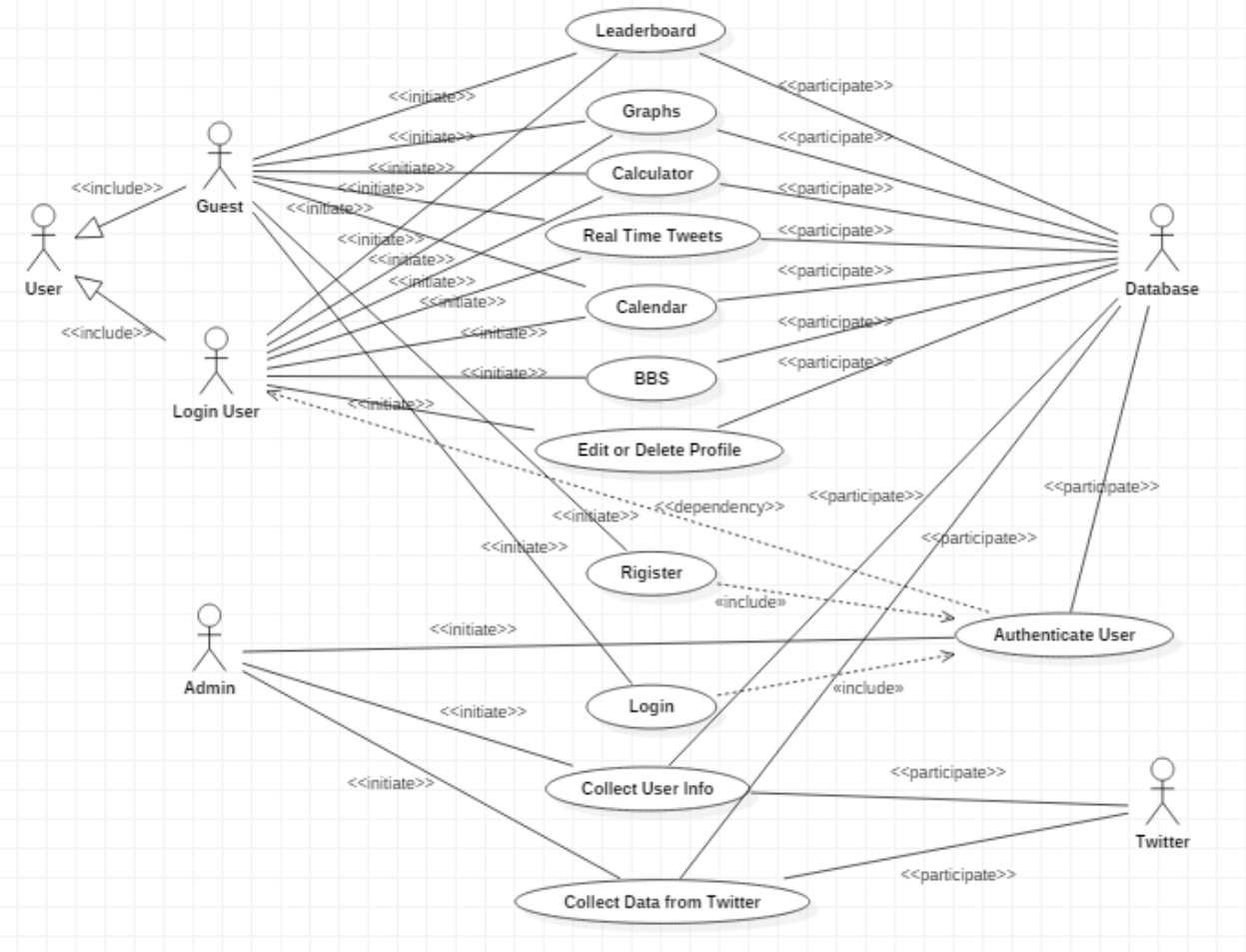


Figure 4

3.5 Traceability Matrix

REQs	PW	UC1	UC2	UC3	UC4	UC5	UC6	UC7	UC8	UC9	UC10	UC11	UC12
REQ1a	5									X			
REQ1b	5									X			
REQ1c	5									X			
REQ2	4			X									

REQ3	**4**									**X**		**X**
REQ4a	**5**		**X**	**X**	**X**	**X**	**X**			**X**		
REQ4b	**5**		**X**	**X**	**X**	**X**	**X**			**X**		
REQ4c	**3**				**X**							
REQ5a	**3**		**X**									
REQ5b	**3**		**X**									
REQ5c	**3**		**X**									
REQ5d	**3**		**X**									
REQ6	**3**						**X**					
REQ7	**2**											
REQ8	**4**		**X**									
REQ9	**3**		**X**									
~~**REQ10a**~~	**2**		**X**			**X**						
~~**REQ10b**~~	**2**		**X**			**X**						
REQ11	**2**			**X**								
REQ12	**3**				**X**							
REQ13a	**5**	**X**										

REQ13b	**5**	**X**											
REQ13c	**5**		**X**										
REQ13d	**5**		**X**										
REQ13e	**1**												
REQ14	**5**	**X**										**X**	
~~**REQ15**~~	**2**												
~~**REQ16**~~	**2**								**X**				
REQ17a	**3**	**X**									**X**		
REQ17b	**3**										**X**		
REQ18a	**2**								**X**				
REQ18b	**2**								**X**				
Max PW		**5**	**5**	**4**	**2**	**3**	**2**	**5**	**2**	**5**	**3**	**5**	**4**
Total PW		**18**	**10**	**37**	**12**	**16**	**14**	**13**	**6**	**29**	**6**	**5**	**4**

UC13: Almost the same as UC3, with Max PW: 4, Total PW: 37

As shown on the matrix, some requirements which are marked red are left over.

For **REQ7**, it is not the prior requirement. We define this one as an additional feature for the system and in the primitive traceability matrix it would not be an isolated use case.

For **REQ13e**, it is a low-priority requirement. Without it, the main functionalities would still work.

For **REQ15**, we see the importance of this feature, but we feel like it might be too general and we would defer it for later.

3.6 Use Case Fully Dressed Description

To further develop the detailed specification of use cases, we start by drafting usage scenarios.

We thought of two possible approaches to develop use cases. One is that we first start at the data collection and another is that we start at the user functionalities like register, login, etc.

The primary goal of the system is to display data we collected to the user intuitively, but the data analytic would be fundamental to all the features. Henceforth, we decided to begin with data collection and getting the data would be the first logical step to begin data analysis too. We start from Use Case 9.

USE CASE: UC9 **Collect Twitter Information**
Related Requirements: REQ-1abc, REQ-3, REQ-4ab **Initiating Actor:** Administrator **Actor's Goal:** To collect tweets from Twitter. **Participating Actors:** Database, Twitter. **Preconditions:** The number of API calls made should not have exceeded the limit and the administrator must be successfully authenticated by Twitter. **Success End Condition:** The tweets are successfully collected and stored in the database. **Failed End Condition:** The tweets are not collected from Twitter. **Extension Points: NA** **Flow of Events for Main Success Scenario:** →1. Administrator makes a request to the API to get tweets. Relevant hashtags are

also entered in order to get the related tweets.

←2.Twitter sends the tweets relative to the hashtags to the administrator which is stored in the database.

USE CASE: UC10	**Collect User Information**

Related Requirements: REQ-17a, REQ-17b

Initiating Actor: Administrator

Actor's Goal: To collect user information extracted from the user profile from Twitter using the corresponding user id and store it in the database.

Participating Actors: Database, Twitter.

Preconditions: Successful execution of UC9; Administrator should have a valid user ID for the profile to be retrieved.

Success End Condition: The user information is successfully collected and stored in the database.

Failed End Condition: The user information is not collected from Twitter.

Extension Points: NA

Flow of Events for Main Success Scenario:

→1. Administrator searches for the user id relative to the user profile to be retrieved from the tweets received from Twitter.

→2. Administrator sends an API request to Twitter to get the user profile. The user id

is also entered in this request.

←3.Twitter sends in the required user profiles, which are stored in the database.

The data collected through the above two use cases shall be further used in mathematical model section. The mathematical model section shall further identify the key attributes required to develop the necessary analytics. Now, we proceed with the use cases which are defined to display our data analysis intuitively.

USE CASE: UC3	**Show Graphs**

Related Requirements: REQ-4ab, REQ-5abcd, REQ-8, REQ-9, REQ-10ab

Initiating Actor: User

Actor's Goal: To view any graph of data.

Participating Actors: Database.

Preconditions: Data Connectivity and instantaneity.

Success End Condition: User can view any graph of data.

Failed End Condition: The graph is not visible because of un-availability of required workout data.

Extension Points: NA

Flow of Events for Main Success Scenario:

→1. User enters the website.

←2. System show all graphs of data to user.

→3. User selects location on heat map by inputting the name of location or pointing it on the map.

←4. System show a more specific data of chosen location to user.

USE CASE: UC4	**Display Calendar**

Related Requirements: REQ-4ab, REQ-11

Initiating Actor: User

Actor's Goal: To view any upcoming event and RSVP

Participating Actors: Database

Pre-Conditions: Data Connectivity

Success End Condition: User can view and RSVP for events.

Failed End Condition: NA

Flow of Events for Main Success Scenario:

Include: Login (UC12)

→1.User inputs events type he/she want to learn about.

← 2. System will request events information from Database.

←3. System will retrieve the events information according to interests and input from users, then display the events on the calendar.

→ 4. User can view the events and click on RSVP.

← 5. System will add the user name to the participants list in the database.

← 6. System will display confirmation message and mark the event as “Attending”.

USE CASE: UC7 **Display Real-Time Tweets**

Related Requirements: REQ-4ab, REQ-6

Initiating Actor: User

Actor’s Goal: To view real time tweets related to different tags

Participating Actors: Database

Pre-Conditions: Data Connectivity

Success End Condition: User can view interested tweets rolling the side screen.

Failed End Condition: NA

Flow of Events for Main Success Scenario:

→1.User inputs the tags or themes he/she want to see.

← 2. System will request relevant tweets information from Database.

←3. System will retrieve the tweets information according to the tags input from user and display the events on the side screen.

→ 4. User can view the tweets and link to twitter.

All these Use Cases above are concerning about how we collect data and display to users. Now we respect user to interact with our system. The interaction design can be found on some use cases listed above, i.e. BBS section has the option for login users to post some experiences, and UC4 allows login users to add their personal event to the calendar. All these interactions result in use cases designed only for users other than data. The following use cases are for users.

Use Case UC1:	**Register**

Related Requirements:REQ-13ab, REQ-14, REQ-17a

Initiating Actor: User

Actor's Goal: To register onto the system & create a profile on system.

Participating Actors: Database.

Preconditions: (a) User must be an unregistered user. (b) System displays the homepage.

Post conditions: User success to register, database store the profile

Failed End Condition: The user have registered, and fail to register.

Flow of Events for Main Success Scenario:

→1. User clicks "Sign Up" option on home page.

←2. System displays the profile page (user type (general user, governor)twitter ID, password, confirm password, gender….), and let user to fill in

→3. User type valid ID which has not registered in the system, and type same password twice, provide all mandatory information, and click bottom "save"

←4. System (a) prepares a database query to verify that ID has not registered in system; (b) signs to twitter to verify the twitter ID is a valid ID; (c) checks whether the two password are same; (d) check all mandatory information provided

←5. Database signs the ID has not registered in the system

←6. Twitter signs the twitter ID provided is valid

← 7. System (a) stores the profile in the Database； (b) prompts a message that " Registered successfully"

Flow of Events for Extensions (Alternate Scenarios)

3a. User type a valid ID which has registered in the system, and click "SAVE"

←1. System (a) detects error, (b) prompts a message "You have registered ", (c) directs to Homepage

3b. User type an invalid twitter ID, and click "SAVE"

←1. System (a) detects error, (b) prompts a message "Invalid twitter ID, please type again"

→2. User supplies a valid twitter ID

←3. Same as in Step 4

3c. User type different passwords, and click “SAVE”

←1. System (a) detects error, (b) prompts a message “Passwords do not match each other, please type again”

→2. User supplies same passwords twice

←3. Same as in Step 4

3d. User only provides part of mandatory information

←1. System (a) detects error, (b) prompts a message “You have to provide all the mandatory information, please type the rest.”

→2. User supplies all the mandatory information

← 3. Same as in Step 4

3e. User click cancel option

← 1. System (a) doesn’t store the profile; (b) redirects to Homepage

Use Case UC2:	**View/Edit Personal Profile**

Related Requirements: REQ-2c

Initiating Actor: Login User

Actor’s Goal: To edit his/her profile information.

Participating Actors: Database.

Preconditions: The system displays the profile page

Post conditions: User edit his profile successfully, Database updated user’s profile

Failed End Condition: profile failed to updated

Flow of Events for Main Success Scenario:

→ 1. User clicks “Edit” bottom

← 2. System display the profile page in editing condition, and let user change

→ 3. User type information that need be updated, and click “Save”

← 4. System (a) stores the updated profile to Database, (b) prompts a message ”updated successfully”

Flow of Events for Extensions (Alternate Scenarios)

3a. User type an invalid twitter ID, and click “SAVE”

← 1. System (a) detects error, (b) prompts a message “Invalid twitter ID, please type again”

→2. User supplies a valid twitter ID

←3. Same as in Step 4

3b. User type different passwords, and click “SAVE”

←1. system (a) detects error, (b) prompts a message “Passwords do not match each other, please type again”

→2. User supplies same passwords twice

←3. Same as in Step 4

3c. User click cancel option

← 1. System (a) doesn’t update the profile; (b) redirects to profile page in cannot-editing condition

Use Case UC2:	**Twitter Graphs**

Related Requirements: REQ-5a,REQ-5b, REQ-5c
Initiating Actor: User
Actor’s Goal: To achieve twitter histograms on the web page.
Participating Actors: Database.
Preconditions: The twitter histogram is available and is up to date.
Success End Condition: The twitter histogram is updated based on recent data and is available on website when a user logs in.
Failed End Condition: The histogram is not visible because of unavailability of required twitter data.
Extension Points: NA 3a. User type an invalid twitter ID, and click “SAVE”

← 1. System (a) detects error, (b) prompts a message “Invalid twitter ID, please type again”

→2. User supplies a valid twitter ID

←3. Same as in Step 4

3b. User type different passwords, and click “SAVE”

←1. system (a) detects error, (b) prompts a message “Passwords do not match each other, please type again”

→2. User supplies same passwords twice

←3. Same as in Step 4

3c. User click cancel option

← 1. System (a) doesn't update the profile; (b) redirects to profile page in cannot-editing condition

Flow of Events for Main Success Scenario:
→ 1. User clicks the kind of twitter histogram (exercise, smoke, dietary and alcohol).Similar analysis for colder and warmer places.
→ 2. System sends request about the twitter data of histogram to database.
← 3. Database checks and sends back the twitter data of histogram.
← 4. System calculates the twitter data of histogram.
← 5. System displays data on website.

One thing needs to be noted is that, some of the use cases can split into more trivial detailed use cases. On the high-priority design, we only list the use cases which are more inclusive in order to show the general scenario of our project.

3.7 Sequence Diagram

In this part, we barely choose a few complicate & important Use Cases derived on fully dressed description to draw the sequence diagrams.

First, we consider the data collecting use cases. Collecting Data diagrams are derived as:

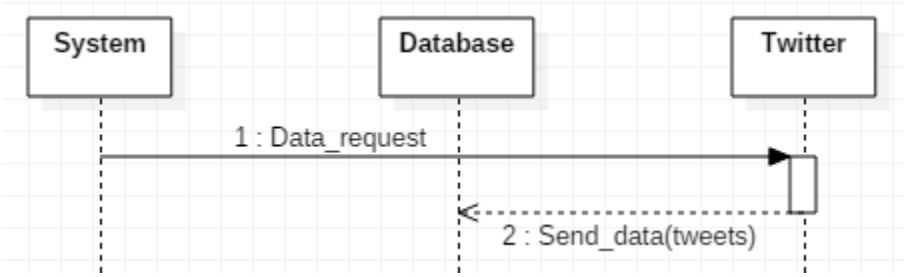


Figure 5: Sequence Diagram for UC9

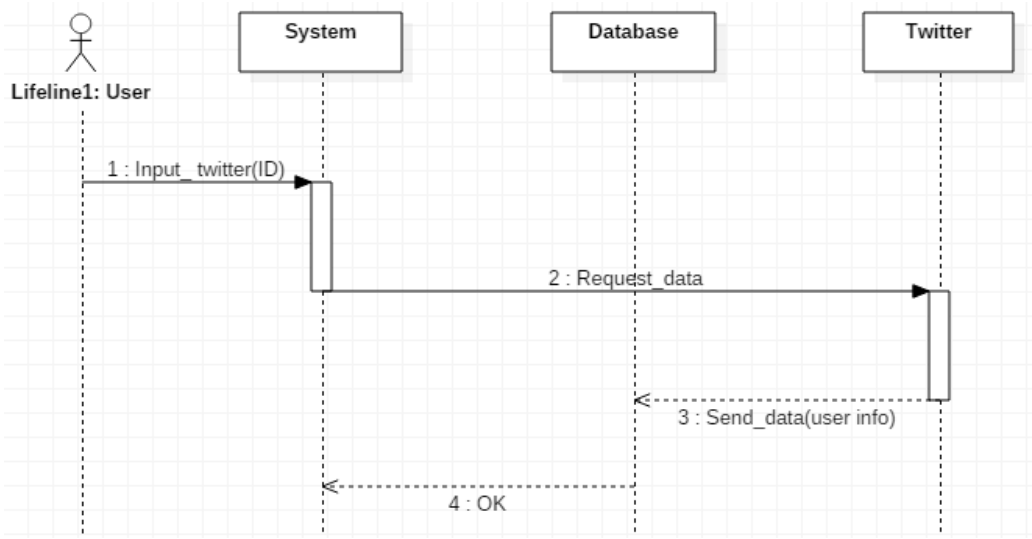


Figure 6: Sequence Diagram for UC10

Second, we choose some data displaying from UC3 to UC8, which have similarities in some extent in the information flows between objects. Thus, these use cases are displayed as:

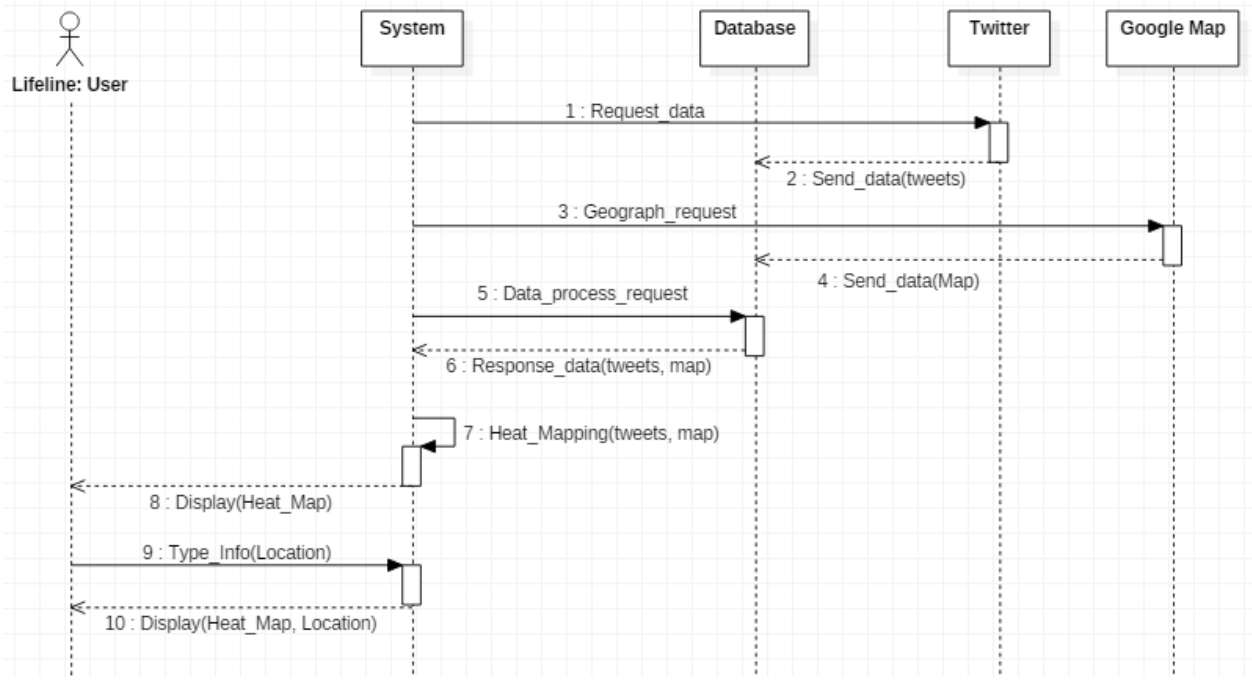


Figure 7: Sequence Diagram for UC3

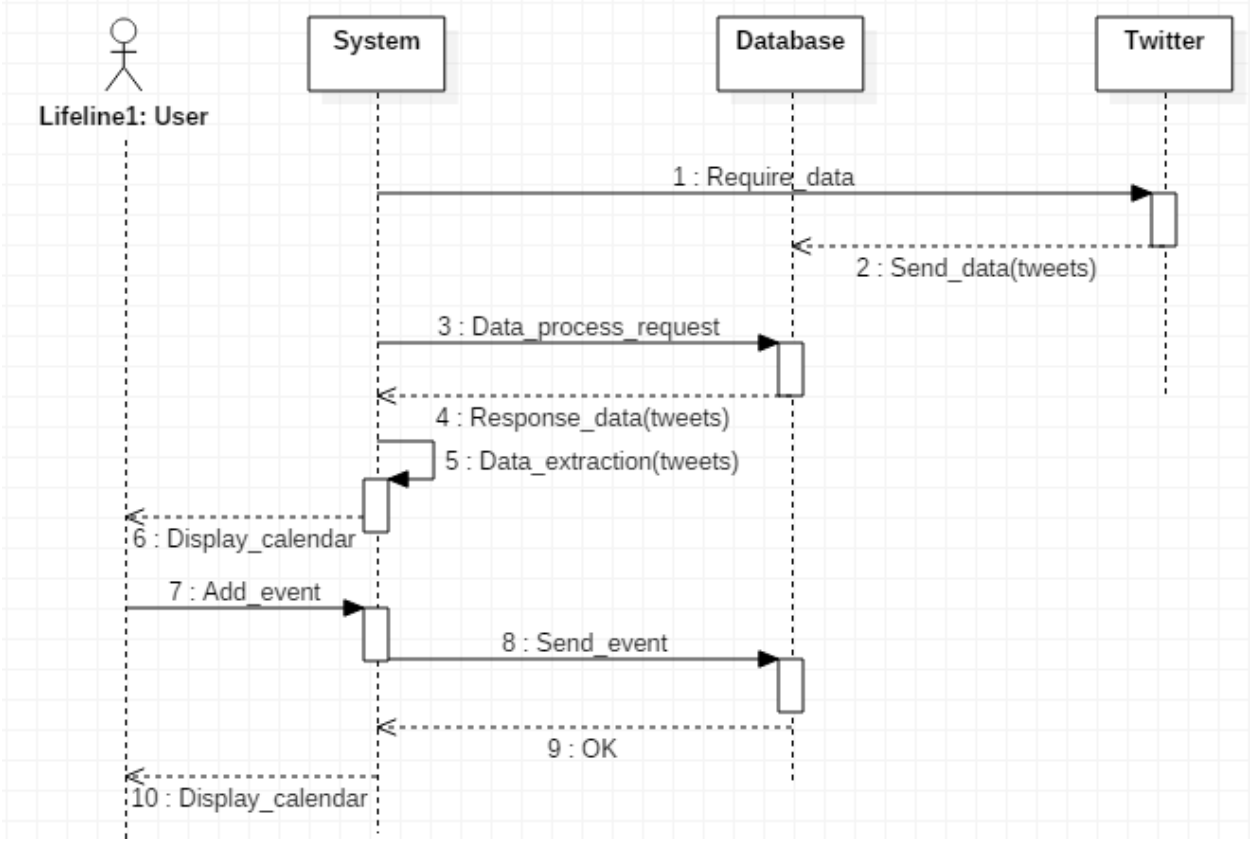


Figure 8: Sequence Diagram for UC4

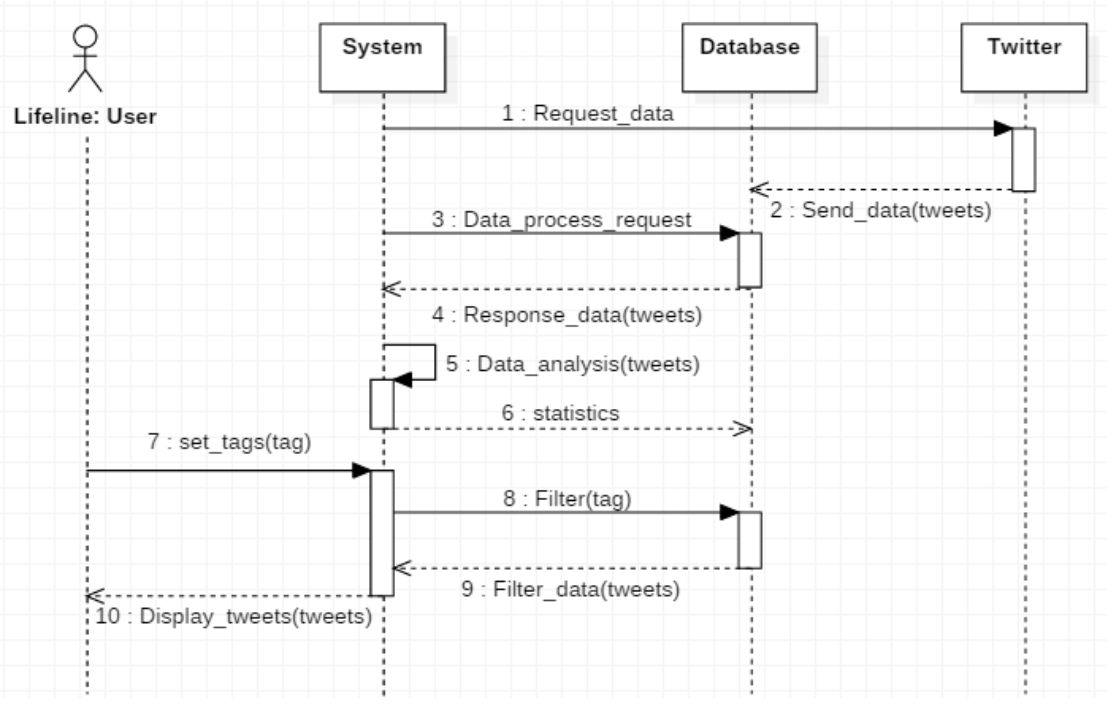


Figure 9: Sequence Diagram for UC7

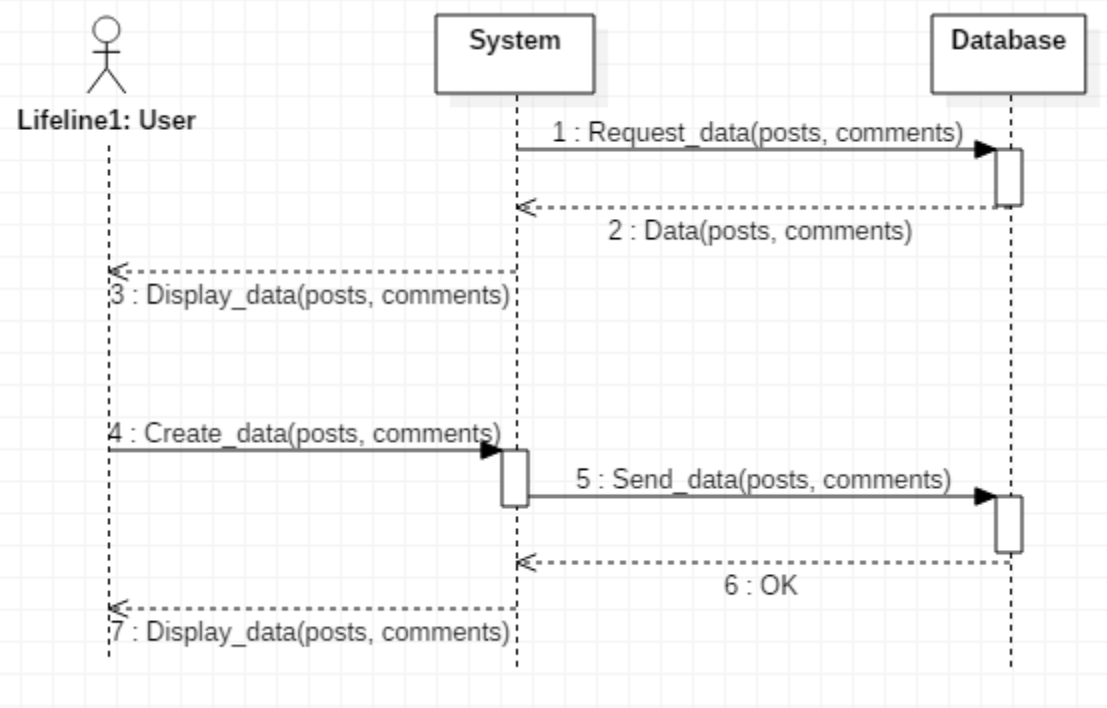


Figure 10: Sequence Diagram for UC8

Finally, we show the sequence diagram for Register and Profile Operations.

The diagrams are shown as:

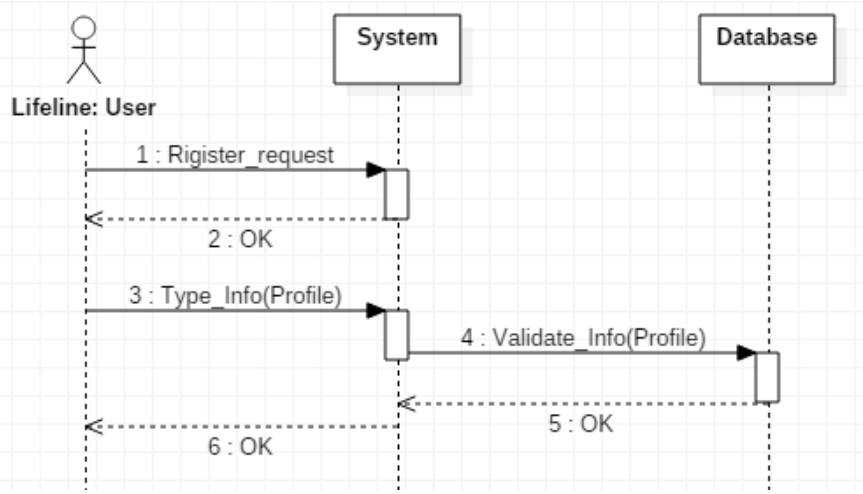


Figure 11: Sequence Diagram for UC1

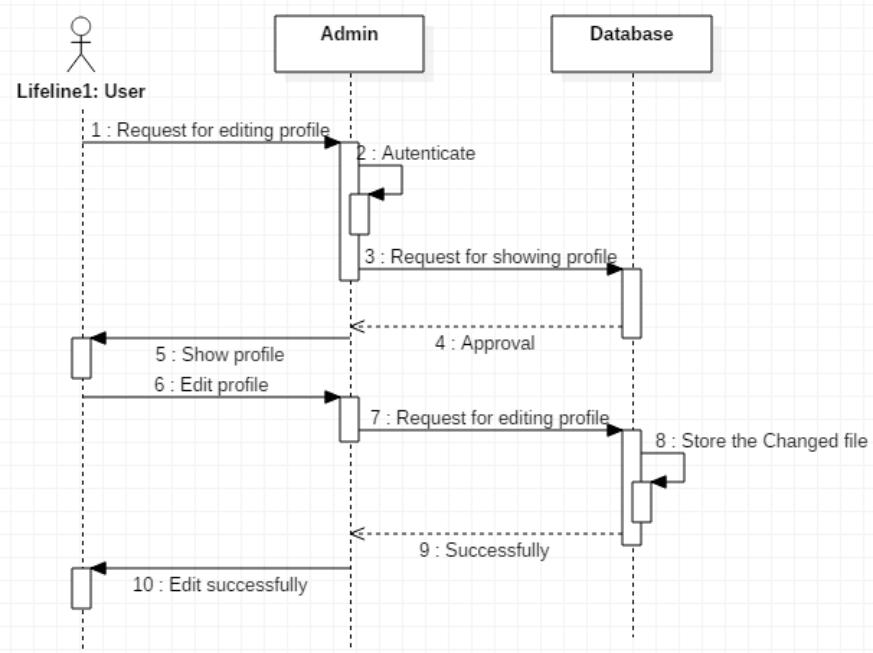


Figure 12: Sequence Diagram for UC2

4. User Interface Specification

The Fig 1-4HTML design mentioned on the On-Screen Section gives our initial sketch of the User Interface that was envisioned during the Customer Statement of Requirements Phase. After the development of the functional specifications, the requirements are better understood. Based on the detailed description of the Use Cases, we were able to ensure that the User Interface design resembles the customer requirements and reflects the details of use cases. The interface design may change once we start our developing.

Preliminary Design

- All the use cases except for UC-9 and UC-10 have User Interface Requirement. The below screenshots focuses on the use cases UC-3, UC-5, UC-6, UC-7 and UC-12. The below figures show how our website would look like. It has the login page and a home screen where the user can select their further categories.

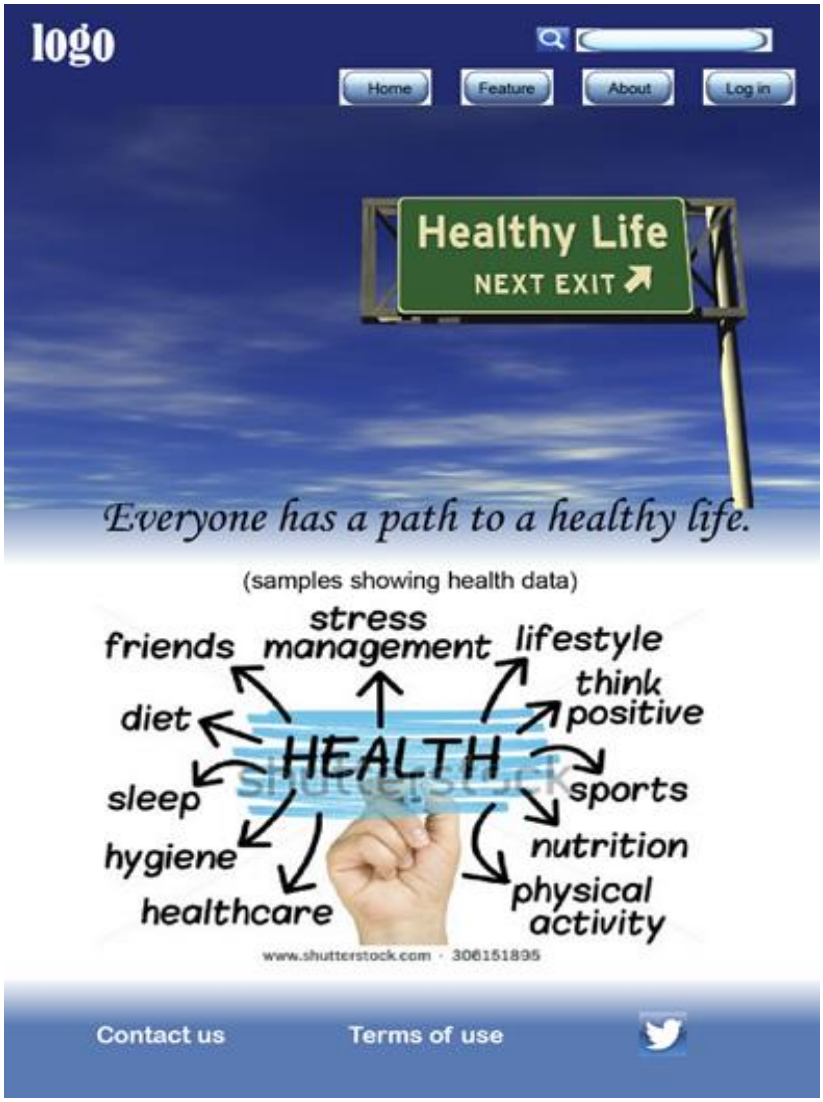


Figure 13: UI Design- Home Page

- When the user enters the home page, they can navigate to the User ID and password fields where they can fill their credentials. If the user is a new user, they can sign up by filling out all their details. A guest user can directly view the webpages without providing any login options. After filling the respective fields, the user will click on the Login button. After successful login, the user will see a tab screen, from where the user can click on the Profile tab to view their profile.

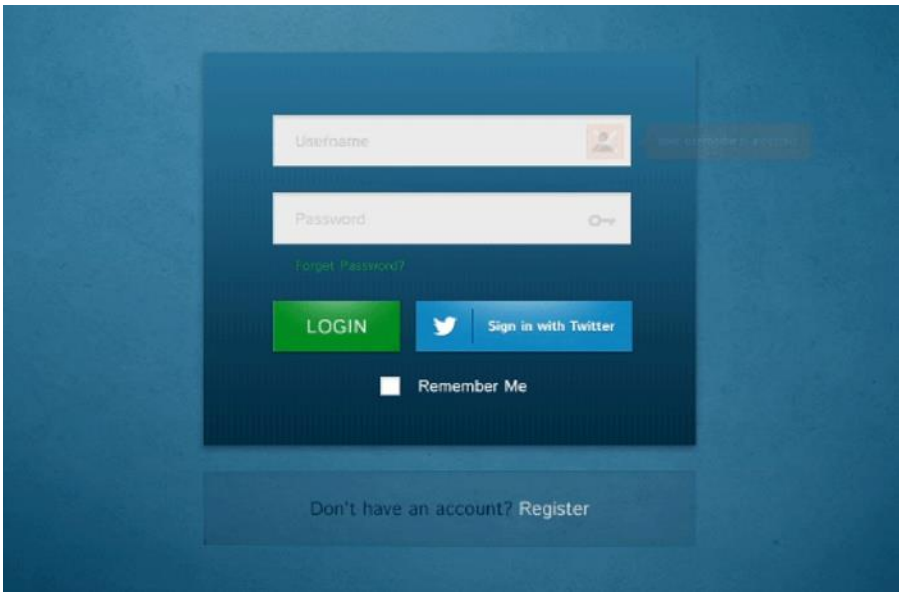


Figure 14: UI Design- Login

- For the first webpage, the visitor (user who has not registered) can browse both "Public Display", "Our Features" sections. And visitors will chose one feature which they are interested in and website will go to the following page:

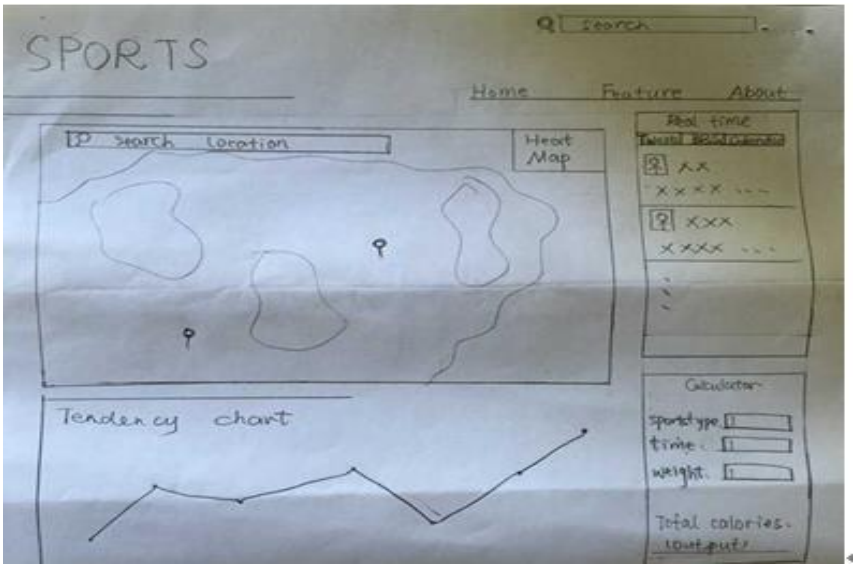


Figure 15: UI Design – Display

There will be several sections for the visitor:

1. The visitor will be able to see different heat maps varied by area, time and demography. He/she can select the display way such as in area, time, type or demography. The map also enables dragging and zooming in/out.

2. The visitor can see the leaderboard by selecting on the relevant button on the webpage, which includes ranking for different users varied by area, type and demography.
3. The visitor can view the ranks listed by the website from the data we have collected, and the rank will include several aspects to let the visitor to choose from.
4. The visitor can access to the BBS section, which they can see the experience sharing by other people, and after you login, you comment below and communicate with other people.

After the user login, there will be other sections where the registered users can see:

1. The member (user who has registered) can use these features stated above that all visitors can use.
2. The member can navigate to the calculator where they can enter their Age, Sex, Height, Weight and the type of workout they performed to calculate the amount of workout they have performed. Then the user clicks on calculate button. Based on all the factors mentioned, the calculator will then display the amount of calories based on the workout the user has performed.
3. The member can also check personal record of exercising in time varied by type.
4. The website will provide relatively professional personal advice to make the user's lifestyle healthier and efficient.

5. Effort Estimation Using Use Case Points

Here we use the following equation to calculate the Use Case Points (UCP).

UCP = UUCP* TCF *ECF

Where 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.

The table below shows the Unadjusted Actor Weight (UAW).

Unadjusted Actor Weight (UAW)

Table: Actor Classification Associated with Weights

Actor Name	Complexity	Weight
Users	Complex	3
Administrator	Complex	3
Database	Average	2
Enterprise	Simple	1
Twitter	Simple	1

UAW = 3 + 3 + 2 + 1 + 1 = 10

The UUCW is derived from the number of use cases in three categories: simple, average, and complex. We use the following table to record the classification of each user case.

Use Case	Case Name	Description	Category	Weight
UC-1	Register	The system shall allow user to register, and create his profile	Average	10
UC-2	View/Edit Personal Profile	System shall display the personal profile for the logged in user.	Average	10
UC-3	Show Graphs	System shall show users graphs of several aspects(e.g. heat map, tweet numbers in a period)	Complex	15
UC-4	Display Calendar	System shall be able to support a calendar and allow uses to register for events.	Complex	15
UC-5	Show Calculator	System shall be able to support workout calculator on the home tab.	Average	10
UC-7	Show real-time Tweets	System shall be able to show real-time tweets with tags provided by user	Average	10
UC-9	Collect Twitter Information	System shall collect and store the tweets with a given region.	Simple	5
UC-10	Collect User Information	System shall collect User Information to perform draw the necessary conclusion.	Simple	5
UC-11	Validate User Login Information	System shall validate when a user attempts to login.	Simple	5

UC-12	Login	Once authenticated by system, User's status shall change.	Average	10
UC-13	Comparison based on geo Info	The system shall be able to show the comparison for different geographic information.	Complex	15

Based on the table above, we can get UUCW = 110. Therefore, UUCP = 10 + 110 = 120.

Technical Complexity Factor (TCF)—Nonfunctional Requirements

Table: Technical complexity factors and their weights.

Technical factor	Description	Weight	Perceived Complexity	Calculated Factor
T1	Friendly interface to user.	1	2	2
T2	Processing is complex	2	3	6
T3	Users expect good performance	1	3	3
T4	Security is important	1	5	5
T5	Easy to change minimally as required	1	1	1
T6	No access for third parties	1	0	0
T7	Convenience and Ease to use	0.5	5	2.5
Technical Factor Total(TFT)				19.5

C1=0.6, C2=0.01, TCF = C1 + C2 × TFT = 0.6 + 0.01 × 19.5 = 0.795

Environment Complexity Factor (ECF)

The environmental factors measure the experience level of the people on the project and the stability of the project. For detail, please check the below table.

Table: Environmental complexity factors and their weights.

Environmental Factor	Description	Weight	Perceived Impact	Calculated Factor
E1	Beginner familiarity with UML	1.5	1	1.5
E2	Familiarity with application	0.5	2	1
E3	Knowledge of object-oriented approach	1	2	2
E4	Beginner lead analyst	0.5	1	0.5
E5	Highly motivated	1	3	3
E6	Stable requirements	2	5	10
E7	Average difficulty in programming	-1	3	-3
Environmental Factor Total:				15

Here we use the following formula to calculate ECF,

ECF = Constant-1 + Constant-2 × Environmental Factor Total

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

Given these data, the ECF = 1.4 + (- 0.03 × 15) = 0.95.

In the end, the final UCP = 120 × 0.795 × 0.95 = 90.63

Productivity factor is 28 hours per user case point. The effort estimation would be 2537.64

6. Domain Analysis

6.1 Domain Model

To start building the domain model, first step is to define the domain model concepts based on the requirements identified in fully dressed description of Use Cases. Table below lists all the requirements with the concerned concept. Each concept will be responsible for completing the responsibility either by doing it or by helping some other.

Concepts Definitions

Based on UC6 we can infer that **maintaining leader board** is one responsibility say **R1** that needs to be assigned, so we introduce a concept here **LBOrganizer** which will be responsible to carry out the analysis in order to assign ranks to the users and display the top 10 rankers on the leaderboard. In order to complete this responsibility, **LBOrganizer** has to do the workout calculations for every user. In order to satisfy this requirement, we introduce a new concept called **Calculator** which will calculate the workout done by each user by using a system defined mathematical model. **LBOrganizer** will be **Using Calculator [LBOrganizer *<uses>*Calculator]** and completes the assigned responsibility **R1.**

Keeping in view all Use Cases that were developed above and breaking down the system further into smaller parts we identified several concepts and their responsibilities that are listed in the below table.

R#	Responsibility	Type	Concept
R1	Display real-time Tweets	D	Webpage
R2	Display Leader Board	D	Webpage
R3	Update/maintain the Leader Board	D	LBOrganizer
R4	Compare workout among different users	D	LBOrganizer
R5	Show calculator on UI	D	Webpage
R6	On screen workout calculation	D	Calculator
R7	Display calculated workout	K	Controller
R8	Display calculated workout	D	Webpage
R9	Store the user workout information in database	D	Communicator
R10	Host a calendar with all upcoming event information	D	Calendar
R11	Display Calendar on the UI	D	Webpage
R12	Take user registration for upcoming events (RSVP for events)	D	Calendar
R13	Check whether user is already registered for the event.	D	Calendar
R14	Create User Profile	D	ProfileCreator
R15	Store User information in database	K	Communicator
R16	Edit/Update user profile	D	ProfileCreator

R17	**Authenticate user login**	**D**	**Authenticator**
R18	**Collect relevant tweets from twitter**	**D**	**TweetCollector**
R19	**Store collected tweets**	**K**	**Communicator**
R20	**Extract User information from twitter profile**	**D**	**TweetCollector**
R21	**Store user twitter profile information**	**K**	**Communicator**
R22	**Carry out data analysis on tweets and user profile**	**D**	**Analyzer**
R23	**Create the graph of collected data**	**D**	**PatternGenerator**
R24	**Display graph on related tab on UI**	**D**	**Webpage**

Further analysis if the model tells us that our system will have two way communications which are:
1) Requests from user & their response, 2) Requesting Information and updating information in the Database.
Thus in order to separate the responsibilities into 2 different concepts, we introduce **Controller** (For user related communication) & **Communicator** (For DB related communication). Controller would use Communicator in order to complete user query and in response communicator gives back the required information to Controller after querying the data base.

ProfileCreator has the functionality of registration of the new users and profile creation. It enables user registration and allows user to complete their profile. It also gives user the facility to update/modify their profile by allowing an edit option where they can modify their profile when they login. This profile information of the user is saved in separate tables in the data base allowing the data to be used in the future. It also displays the created profile on commands from the user.

Authenticator takes the responsibility of existing user login and authentication; it checks the database and authenticates the logging in user. It also checks the number of attempts made by user to login and counts number of unsuccessful attempts; it locks out the user if the number of unsuccessful attempts is more than 3. This concept basically provides an initial line of security to the system and its users.

Calculator is the only mathematical model in the system that provides input for multiple concepts in the form of **TotalWorkout** done by a user. This number basically helps in order to do the leader board and also serves as an input to the pattern generation module. It provides an onscreen workout calculation for user, where it accepts input from user and displays the workout amount done by the user. Based on the algorithms and assigned weightage for workouts it calculates the workout.

LBOrganizer maintain the leader board on the web page. In order to do so, it performs various steps such as comparison of the workout by different user and arranging users in descending order based on their workout. Top 10 users are selected for the period and their names are displayed on leaderboard. It checks the leaderboard and updates it according to the schedule. **Calendar** conceptualizes the onscreen event calendar which displays the upcoming events on the community page. It also allows the user to register for a particular event by storing RSVP in the data base for the event. In order to maintain a dynamic calendar, administrators are allowed to edit calendar by adding upcoming events to it, thereby making new events available for user registration.

PatternGenerator is the concept of graphical representation of various upcoming trends in Exercise, diet and Smoking & alcohol consumption. The main idea behind graphical representation is to consider the large data of the users which are available on twitter and examine the proclivity of users. Showing these trends with a help of Histograms will make them easy to analyze and comparison the large volume of data. The **PatternGenerator** gets the rationalized and analyzed data from controller and will convert that data into histograms which are displayed on the webpage.

TweetCollector does the tweet collection to collect the data. It uses the Twitter Streaming API for collection of tweets. Tweets are collected based on Hashtags in the tweets of the user and if the required hashtag is available in the tweet then that tweet is considered to be relevant and is stored into the database. This block also gives facility to narrow down the tweet collection for a particular region, hence making our prediction for a region more accurate.

Analyzer is the most critical concept of the system as it carries the responsibility of the data analysis and data segregation. In order to perform different analysis these collected tweets requires some **data analysis**, these tweets are subjected to the analyzer to get more meaningful information. It tries to build some sense by considering the tweet history of the same user and

reads the hashtags, combining the past and the present tags it is capable of predicting the lifestyle and hence global trend.

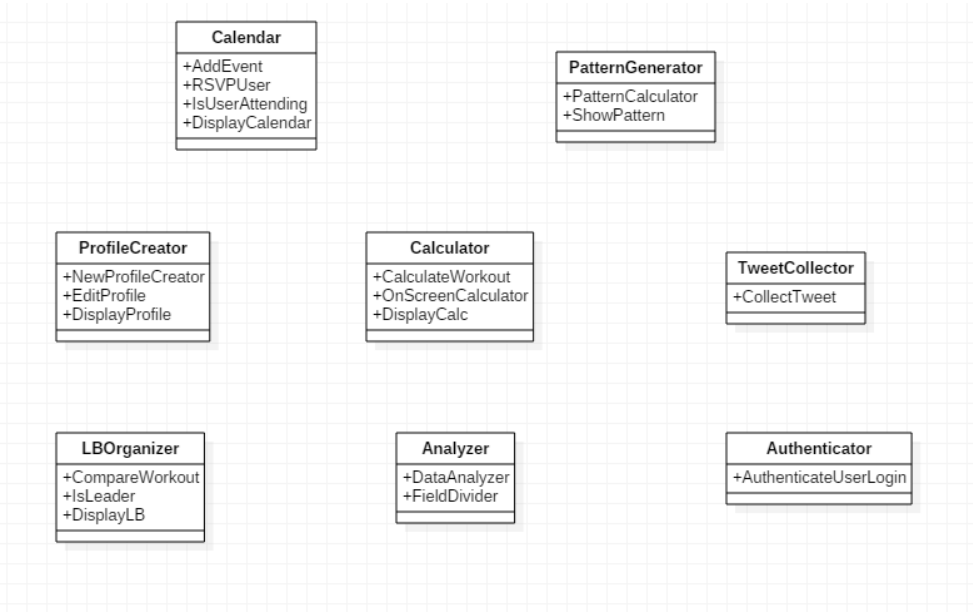


Figure 16: Conceptual Diagram for Domain Model

Attribute Definitions

Attributes, Concepts and their responsibilities are given in the table below. Concepts have different attributes to perform the function and complete the task they are responsible for. Concepts in our model are more likely to have a **Logical Analysis** attribute and **decision maker** attribute, also supported by **display** attribute at time when required. The Mathematical calculation if required by any concept, the attribute is outsourced to calculator concept.

IsUserAttending is required in order to check the reservations in the data base for every user and update the calendar on user option accordingly.

RSVPUser attribute enables the user to register for a particular event. It will update the data base for that event with the users who have registered to attend that particular event.

AddEvent will be used by the administrators to maintain the calendar and adding the new events. Even the description can also be added in the calendar.

CompareWorkout will compare the total workout in a period of last 3 days by different users and use the mathematical model to arrange the users in leader board by using the attribute **IsLeader**.

OnScreenCalculator is a facility that allows the users to calculate their work out without tweeting, an on screen appearance of calculator is available on webpage. By entering the type of

the workout the user did and the time they spent can easily get the number of calories burnt and the work out details that they have performed. Different workouts are assigned different weightages. Based on the weightages assigned to different workout styles and calculates according.

Tweetslist is required to show the real-time tweets collected by administrators. User can click an interesting tweet and enter the homepage of this person and follow him.

All of the above attributes support user in some or the other way, but to have a user base we need to have a Login and Registration functionality. To carry out these responsibilities, our model introduces the following

NewProfileCreator attribute gives scope of user registration in the system. It allows new user to register to the system. It asks user about the important information that are updated in profile, it also allows user to choose their username and password. All these information are stored in database. In future if user wants to edit some of his profile fields, system allows user to perform this task, system uses **EditProfile** attribute to edit the profile of an existing user.

DisplayProfile serves the purpose of viewing the profile. If the logged in user is an "Administrator", system allows him to navigate through the data base and view the entire user profiles stored in it and if the logged in user is "User" it shows only personal user profile.
Data analytics is the backbone of the whole system. Our system is a speculative solution to the trends that are most likely to prevail in near future. To serve this purpose our system needs to have advanced data analysis logic. The data collected from twitter in the form of tweets needs to be read in analytical manner and draw more conclusions and meaning from those tweets. To serve this purpose we have **DataAnalyzer** in our system, this attribute is capable of reading the tweets in different manner and draw logical conclusions. Taking an example, a tweet by user *"acb: **#Exercise #Alone #terrified**"*, looking at the tweet single hashtag at a time will not make any sense or there is a high priority that the conclusion drawn by looking at one hashtag only will give you the wrong prediction (The user actually intends to say that he is not working out as he is alone and terrified to work out alone whereas we may predict it only by reading #Exercise that the user is doing exercise). We not only misread the prediction but also predict wrong trend. In order to overcome such issue we may use the **DataAnalyzer** which reads series of hashtags at a time and thus draw some meaning full conclusions. We can use the history of particular user who is very frequent tweeter, to store the history we need to have the user name and other field from the tweet in the data base.

FieldDivider is the responsible attribute for dividing different field of a tweet and store in database. After all the logical data analysis is performed, we use **PatternCalculator** attribute

which calculates a histogram showing the pattern and trend. These histograms are divided into 3 parts based on type: 1) Exercise Histogram, 2) Diet Histogram & 3) Smoking and Alcohol

Consumption trend. These histograms serve the purpose of prediction and are available for users on the respective tabs of our community website.

Responsibility	Attribute	Concept
R1:Display real-time tweets	Tweetslist	Webpage
R2: Compare workout among different users	CompareWorkout	LBOrganizer
R3: Update/maintain the leader board.	IsLeader	LBOrganizer
R6: On screen workout calculation	OnScreenCalculator	Calculator
R10: Host a calendar with all upcoming event information.	AddEvent	Calendar
R12: Take user registration for upcoming events (RSVP for events)	RSVPUser	Calendar
R13: Check whether user is already registered for the event.	IsUserAttending	Calendar
R14: Create User Profile	NewProfileCreator	ProfileCreator
R16: Edit/Update user profile	EditProfile	ProfileCreator
R17: Authenticate user login	IsUserCorrect	Authenticator
R18: Collect relevant tweets from twitter	CollectTweet	TweetCollector
R20: Extract User information from twitter profile	FeildDivider	TweetCollector
R22: Carry out data analysis on tweets and user profile	DataAnalyzer	Analyzer
R23: Create the graph of collecting data	PatternCalculator	PatternGenerator

Domain Diagram:

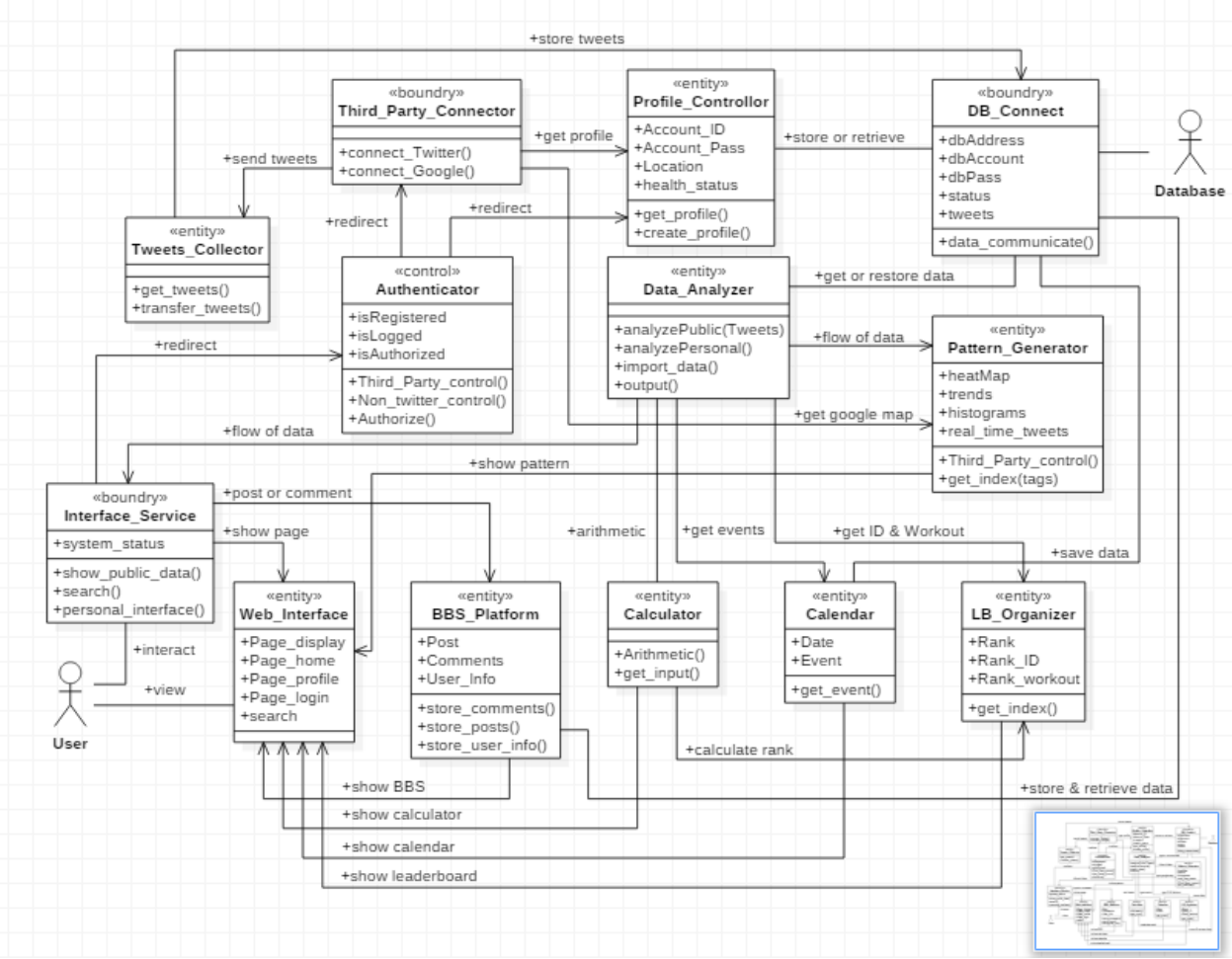


Figure 17: Domain Model Diagram

Traceability Matrix:

Use Case	PW	LBOrganizer	Calendar	Analyzer	TweetCollector	Authenticator	PatternGenerator	ProfileCreator	Calculator
		Domain Concepts							
UC1	**5**					X		X	
UC2	**5**					X		X	

UC3	**4**				**X**		**X**				**X**			
UC4	**2**			**X**										
UC5	**3**													**X**
UC6	**2**	**X**			**X**		**X**							
UC7	**5**				**X**		**X**							
UC8	**2**										**X**			
UC9	**5**				**X**		**X**							
UC10	**3**				**X**		**X**						**X**	
UC11	**5**								**X**					
UC12	**4**								**X**					

6.2 Mathematical Model

First, our original data is in form of pure text of tweets. We need to formulate the input of k-NN algorithm in standards. To achieve this, we do Natural Language analysis on each tweet to extract vectors in two different properties. We consider that a tweet can be identify on the words which are most informative. We implement this in Term Frequency & Inverse Document Frequency (TFIDF). Here are some mathematical definition of TFIDF.

Term Frequency:

In the case of the term frequency tf(t,d), the simplest choice is to use the *raw frequency* of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of $t$by $f_{t,d}$, then the simple tf scheme is tf(t,d) = $f_{t,d}$. Other possibilities include

- Boolean "frequencies": tf(t,d) = 1 if t occurs in d and 0 otherwise;
- Logarithmically scaled frequency: tf(t,d) = 1 + log $f_{t,d}$, or zero if $f_{t,d}$ is zero;

- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times f_{t,d}}{\max\{f_{t,d} : t \in d\}}$$

Inverse Document Frequency:

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

With

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: Number of documents where the term t appears (i.e. $\mathrm{tf}(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$

In order to adapt this method with the clustering, we apply TFIDF along with cosine-similarity calculation for evaluating the extent of a tweet dependency to a group of labeled tweets.

Cosine Similarity:

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

Given two vectors of attributes, *A* and *B*, the cosine similarity, *cos(ϑ)*, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

After applying methods listed below, we are able to formulate one property of tweets.

Second, we will move on to the sentimental analysis of feelings of twitterers when posting tweet. Since our categories are based on extent of positive, motive and negative, the sentiment of certain training set like positive smoking and quit smoking will definitely show differences. We implement the sentimental analysis by using Natural Language Toolkit based on Naïve Bayes Classifier. We need to extract features from the original text rather than cleaned text.

Naïve Bayes Classifier:

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some *n* features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, \ldots, x_n)$$

For each of *K* possible outcomes or *classes*.

The problem with the above formulation is that if the number of features *n* is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' Theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

After analyzing sentiments with above methods, we retrieve another property of tweet.

Finally we apply the properties which have already been abstracted on numbers to k-Nearest Neighbor (kNN) Algorithm for classification of new tweet input.

K - Nearest Neighbor:

This is a supervised training algorithm. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance).

As we have been able to build up mathematical model, we use Euclidean distance for kNN.

7. Interaction Diagram

1. UC-1 REGISTER

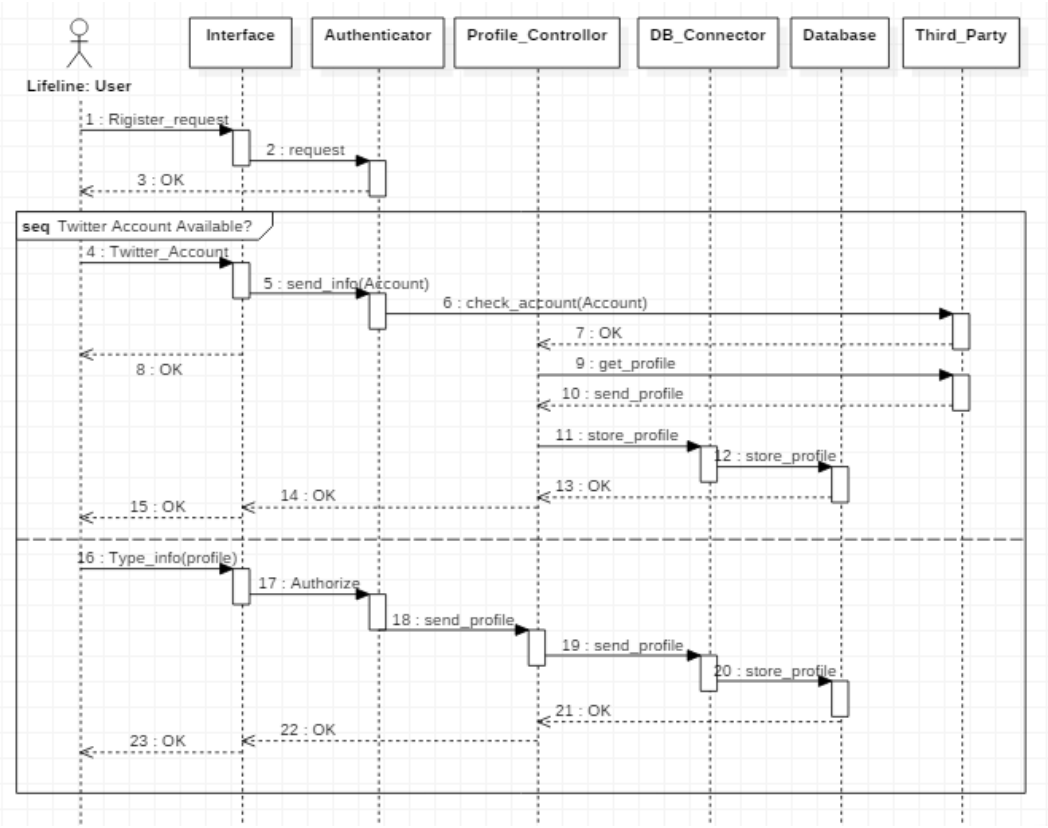


Figure 18: Register

The above interaction diagram corresponds to the UC1 Register. When the user clicks the “Sign Up” button on the log in page, the system displays the profile page in an editable condition. Then the user enters all the mandatory fields and then clicks “Save”. Then a request is sent from the system to the database to verify if this ID is already existing earlier and then send a query to verify the Twitter ID if it is existing. After the system gets both signal true from Database and Twitter, it then checks whether the two passwords are the same or not. If they are same, the profile is stored to the database, and it prompts a message to signal user the success.

2. UC-2 VIEW/EDIT PROFILE

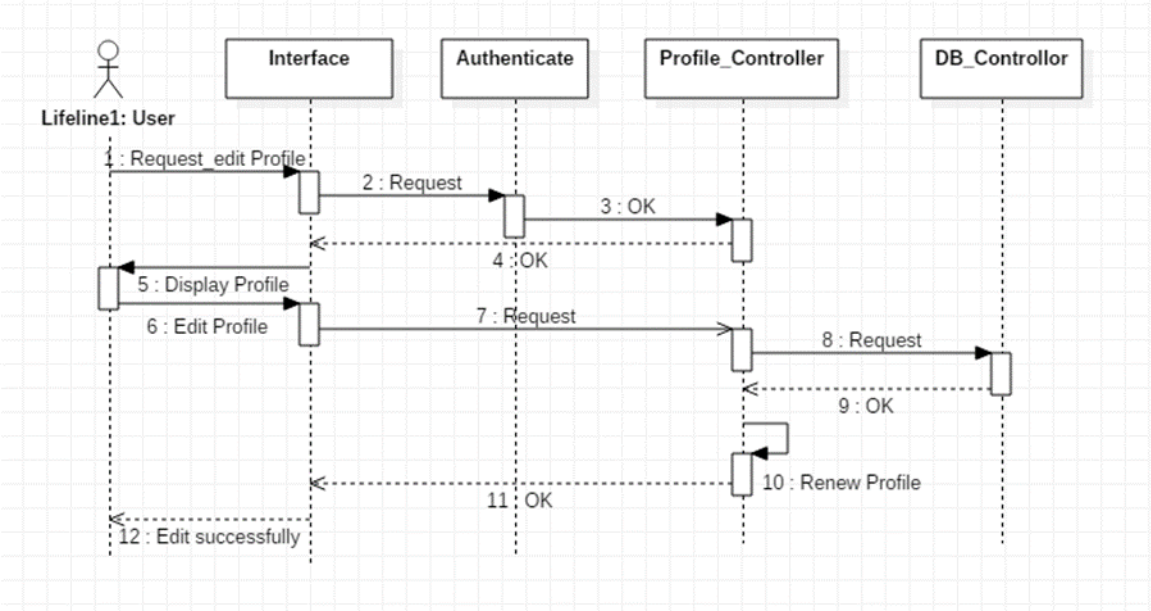


Figure 19: View/Edit Profile

The above interaction diagram corresponds to the UC2 View/Edit Profile. When the user clicks the "Profile" button on the homepage, then the system sends a query to the database for the profile. Database sends the profile to the system and then the profile is displayed.
When the user clicks the "Edit" button on the profile page, the system changes the profile page in editable option. User can now enter the new information and clicks "Save". System then stores the updated user information in the database. System then prompts a message to signal that the user has edited successful.

3. UC-9 COLLECT TWITTER INFO

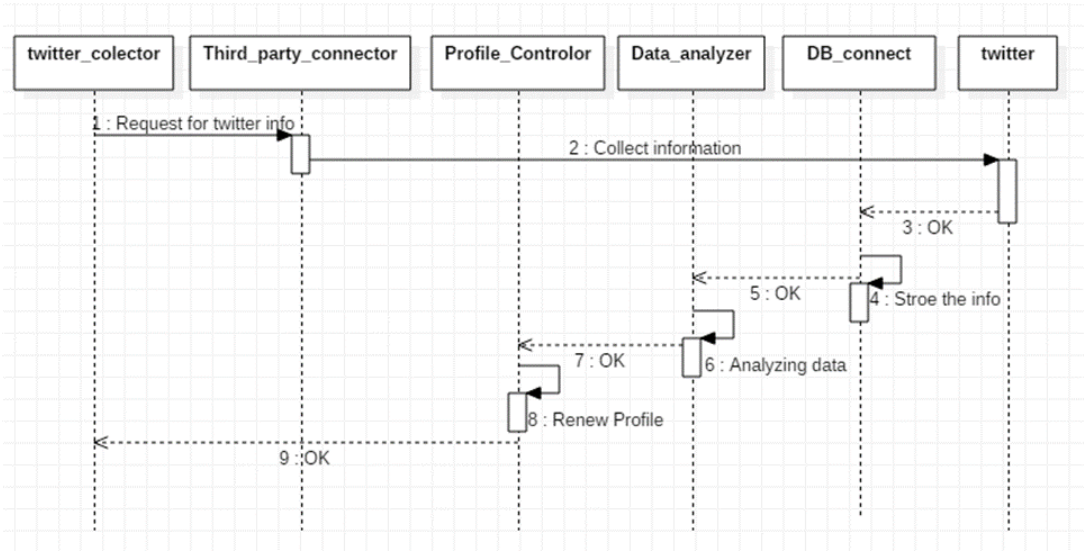


Figure 20: Collect Twitter Info

The above interaction diagram corresponds to the UC9 Collect Twitter Info.
Still, there exists default process: If we can't collect data from the twitter, and The Third_Party_Connector will directly show a wrong signal to Twitter Collector.

4. UC-10 COLLECT USER INFO

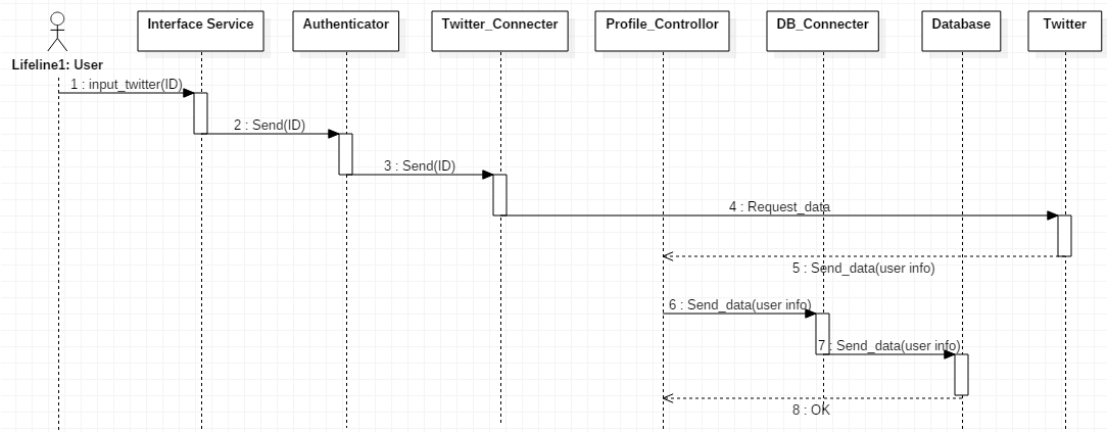


Figure 21: Collect User Info

The above interaction diagram corresponds to the UC10 Collect User Info. It is collecting information of users with a twitter account. The user types in his account ID and Authenticator will send the ID to Twitter_Connector to retrieve profile from Twitter to Profile_Controllor and store.

5. UC-3 SHOW GRAPHS

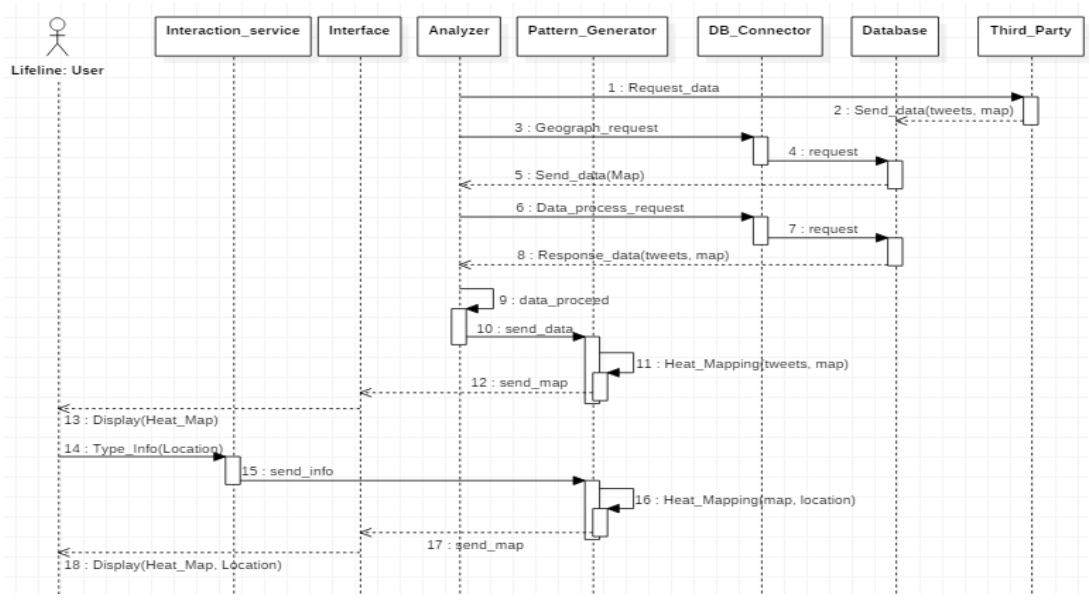


Figure 22: Show Graphs

The above interaction diagram corresponds to the UC3 Show Graphs. It shows in details of how to generate a heat map. First the Analyzer request enough data in tweets as well as request basic map information. These information are first stored in Database, and the Analyzer will then retrieve them from DB and proceed tweets information based on geographic location. After analyzing, the data will be send to Pattern_Generator to generate heat map. Heat map will be displayed through Web_Interface. When user try to interact with the heat map, user could click on locations they are concerning about and the location will be proceeded by Pattern_Generator to show specific heat map.

6. UC-4 DISPLAY CALENDAR

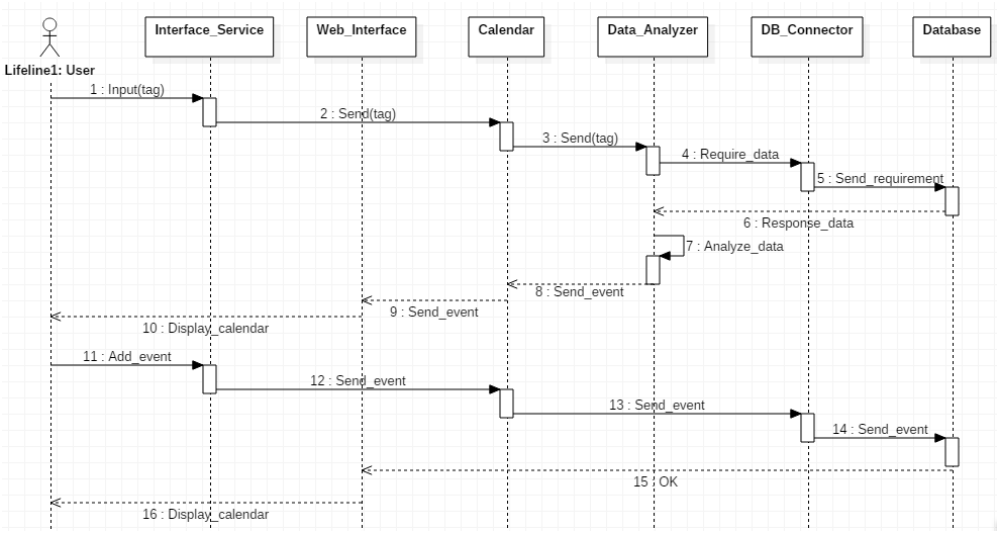


Figure 23: Display Calendar

The above interaction diagram corresponds to the UC4 Display Calendar. Basically, user chooses his favorite theme and input a tag, the tag will be send in sequence to Analyzer, then Analyzer will request data in tweets from Database and cluster the relative tweets in the way user wants. The relevant tweets will be analyzed by dates & events and show on Calendar. If user wishes to add his schedule, it would be easily a click and edit of his events and the Calendar will show changes in it. These events will be stored in Database as well.

7. UC-7 SHOW REAL-TIME TWEETS

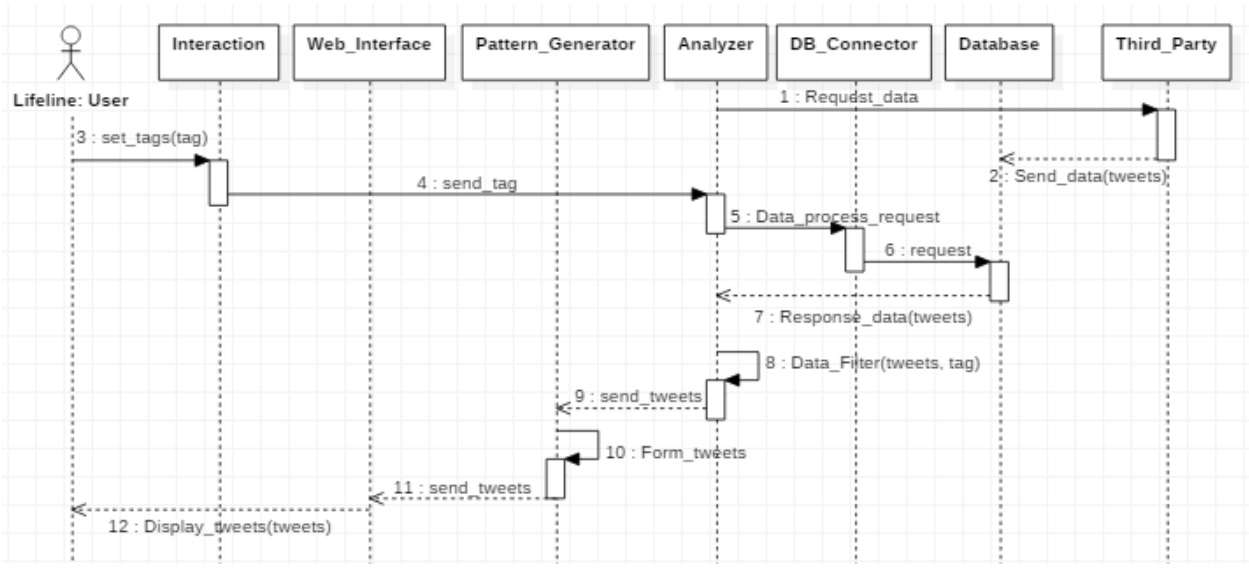


Figure 24: Show Real-Time Tweets

The above interaction diagram corresponds to the UC7 Show Real-Time Tweets. Analyzer is constantly requesting tweets from twitter API. When I user want to sort the tweets and show what he is interested in, he can barely type his tag and Analyzer will filter tweets from Database in his tag. Finally, the filtered tweets will also be constantly sent to Pattern_Generator to form the format of tweets and then display through Web_Interface.

8. UC-8 BBS SECTION

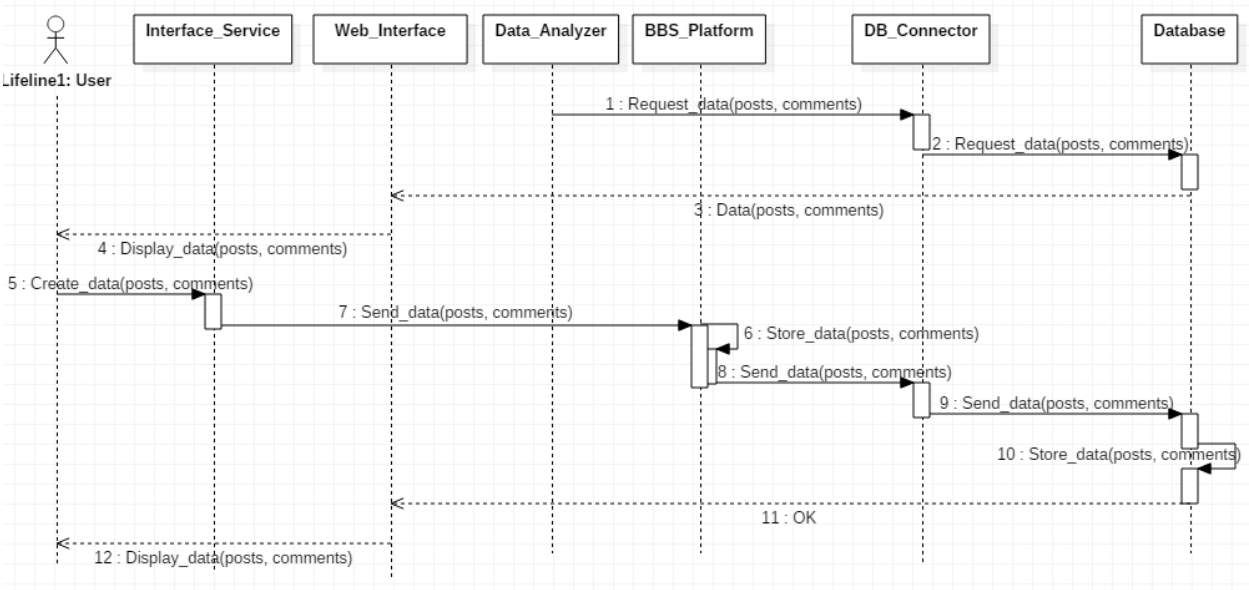


Figure 25: BBS Section

The above interaction diagram corresponds to the UC8 BBS Section. First system shall request data from database to retrieve former posts & comments on BBS and use Web_Interface to show user the BBS Platform. Default Operations are:

- Users without login can go through the BBS.
- Authenticated user can post a new discussion or comment on other one's post. The new posts & comments will also be stored through platform and finally stored in Database.

8. Class Diagram and Interface Specification

8.1 Class Diagram

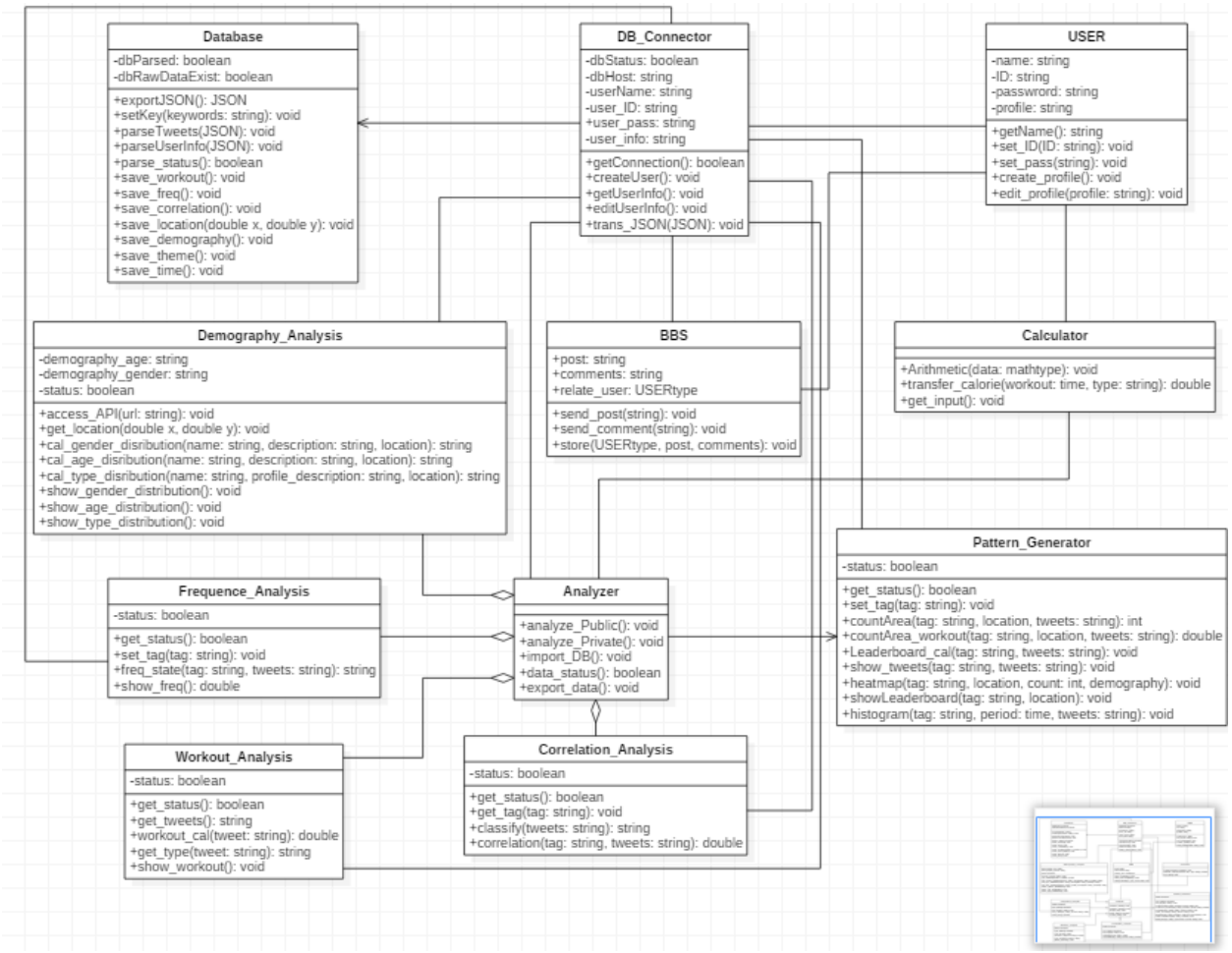


Figure 26: Class Diagram

8.2 Data Types and Operation Signature

1. Authorizer:

OAuthExample.java, OAuthTokenSecret.java, OAuthUtiles.java. These three classes are used to pass the authentication.

2. DataCollecter:

In our data collector section, we mainly use two APIs. The first one is Streaming API, the inputs are keywords concerning smoking, diet, drinking. Then the program connects to Twitter, and extracts tweets containing one of these keywords. It is a streaming API, which means that the tweets extracted are just posted by Twitter users. In our program, when you run "StreamingApiExample.java", the extracted data will be directly stored into our local database (Mongo dB) as JSON files. Then we are transmitting the tweets in MongoDB to MySQL. So far we have collected around 50000 tweets.

3. DataBase (Mongo dB)

MongodbSaver is a class that has functions to save or load data to/from mongo dB. These functions can be easily called in other programs. "getUserLocation" is actually a very flexible function. You can simply change the argument in command "loctmp = obj.getString ("*coordinates*");" to get all the other user information. MongodbViewer is a small class to run the functions in MongodbSaver. Tweets are collected from Streaming API and placed into MongoDB as DB objects.

4. DataBase (MySQL)

Class SqlSaver is the bridge between Eclipse and database SQL. "createUser()" is used to create a new table for twitter user information in a database in SQL and declare the variables in this table. "insertUser()" is used to add a new twitter user with attributes "user id, followers_count, statuses_count, name, screen_name, profile_image_url_https, location, coordinates" into this new table. getUserInfo() is for extracting user profile. addUserInfo is for storing new user profile. Tweets relevant to our hashtags in consideration are stored in MongoDB initially. These tweets are later extracted into MySQL since it's easy to data analysis if the data is stored in a relational database. Also, It's easy to extract information unto User Interface if we have our data stored in MySQL. The UserID, TweetID, TweetText and CreationTime are stored in these tables.

5. User

Class profile is for user to view and edit their information. Class register is for user to register as they can access their homepage.

6. Histogram

This class can extract users' history tweets from Twitter and analyze these tweets to get the number of exercising tweets in each month.

7. Calculator

This class allow user to type weight and height to get advice.

8.3 Traceability Matrix

	Class									
Domain Concepts	Database	DB_connector	User	BBS	Demograph_Analyzer	Frequence_Analyzer	Workout	Correlation_Analyzer	Calculator	Pattern_Generator
LB Organizer	X	X	X	X						
Calendar	X	X	X							
Analyzer	X	X			X	X	X	X		
Twitter Collector	X	X		X	X	X	X	X		
Authenticator	X	X								
Pattern Generator	X	X								
Profile Creator	X	X	X							X
Calculator			X						X	

8.4 Design Pattern

The **Design Pattern** we are trying to follow is Implementation Strategy Pattern. The classes are all related to exactly different members' work. As a result, the division of responsibilities becomes clear which allows us to do parallel program and makes the development more efficient. For example, we split up the Data_Analyzer and Pattern_Generator classes in case the analysis section can be completely done with sub-team with skills of data mining and the graphs are assigned as responsibility to sub-team who is familiar with UI and Database. In reality, we proceed the concept of parallel programming which proved to be a good choice.

9. System Architecture and System Design

9.1 Architecture Style: 3-Tier

Our Personal Health Companion system uses the three-tier architecture. The three tier architecture has three layers namely Presentation Tier, Business Tier and the Data access tier describes as below

1. Presentation Tier: It occupies the top level and displays the information related to the services available on a website. It deals with the user interface and also communicates with other tiers by sending results to the browser and other tiers in the network. It helps the user to interact the system in a smooth manner giving the ability to understand the results.
2. Application Tier: It is also called the middle tier or logic tier or business logic or logic tier. This tier is pulled from the presentation tier and it's a processor of our system. . It helps in processing various computations and algorithms which give user an understandable result of the raw data retrieved by the database.
3. Data Tier: The data access tier stores all the tweets data and also the logic tier retrieves data from this tier for processing. Data in this tier is kept independent of application servers.

In our system the presentation tier is handled by web which will translate tasks and result to user can understand. The logical tier is handled by Java and database is MySQL.

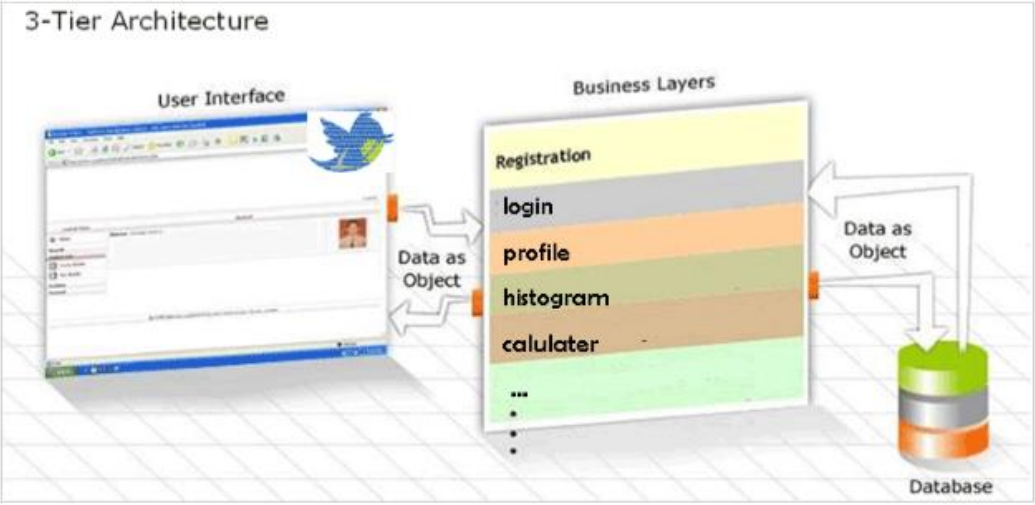


Figure 27: 3-Tier Architecture figure of HMA

9.2 Identifying Subsystems

A subsystem is a special kind of package that only has interfaces as public elements. The interfaces provide a layer of encapsulation, allowing the internal design of the subsystem to remain hidden from other model elements. The concept subsystem is used to distinguish it from the "ordinary" packages, which are semantic-free containers of model elements; the subsystem represents a particular usage of packages with class-like (behavioral) properties.

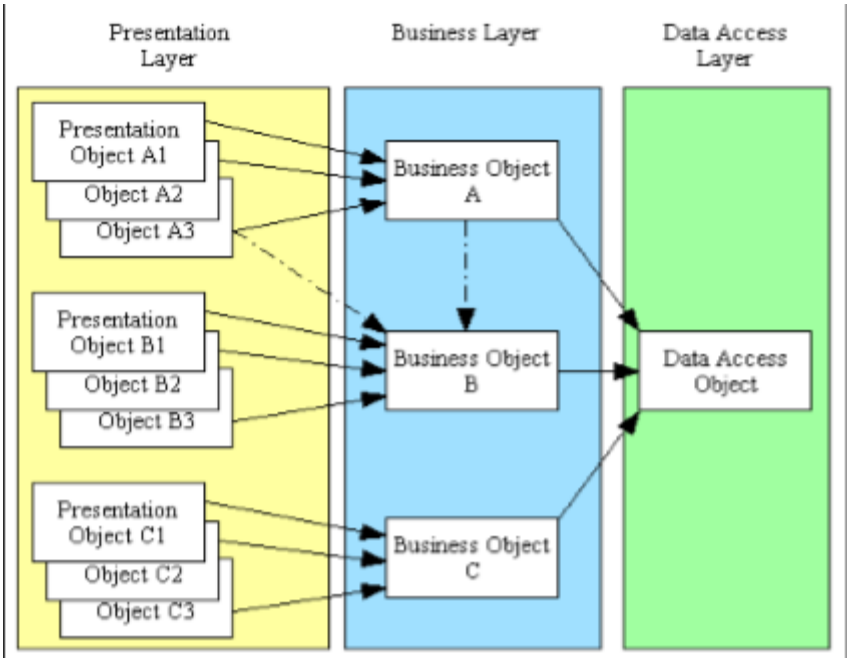


Figure 28: HMA Subsystem[6]

9.3 Mapping Subsystems to Hardware

Following the user/ server architecture, the whole system is mapped into two sets namely the web browser for the client side and web server for the server side. The clients use the web browser to get access to the web.
The server collects the information through an API from the twitter and stores the information in the Mongo DB and later sends them to MySQL. The server also solves the request from the web browser.

9.4 Persistent Data Storage

Basically, our system uses Mongo Db to store the data coming from Twitter API and this data comes in JSON document format. Then after a set of operations, the data will be stored in rational database MySQL. With the database, the application will maintain the data for the next running. The data stored in the database consist of these categories: the user history tweets information, current tweets information. The user history tweets information is obtained by server and stored in local database for last several days. And the current tweets information is grabbed by server and stored in MySQL all the time.
The user history tweets information is stored into the local database. The current tweets information is kept in the MySQL. User history tweets information is used to show some statistic data and histogram data. Current tweets information is used to demonstrate latest exercise-related tweets data. UserID, TweetID, TweetText and CreateTime are the columns which will be used.

9.5 Network Protocol

Our software is a client-Server application. Therefore, the software parts use the internet communication network protocols HTTP between webserver services (Database) and the client (Browser).
PHP functions as a grab-data-only protocol in the system-server computing model. It is interpreted by a web server with a PHP processor module, which generates the resulting web page: PHP commands can be embedded directly into an HTML source document rather than calling an external file to process data. Our web server will work with a web server database and keep obtaining exercise-related tweets.
HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and an application running on a computer hosting a web site may be the server. The client submits an HTTP request message to the server. The server, which provides resources such as HTML files and other content, or performs other functions on behalf of the client, returns a response message to the client. The response contains completion status information about the request and may also contain requested content in its message body.

9.6 Global Control Flow

Our system is event-driven. When a user visits our website, the website would wait for the click operation to display corresponding interface. Our system uses a timer to reload the data from feature tables to make our system real-time.

9.7 Hardware Requirement

Now, we almost collect 50+ thousand tweets, and we will collect more in the next stage of our project. So, to ensure the achievement of all features of our website, the server computer should have enough disk storage for tweets (Based on the number of tweets at the end).
Also, a server is needed to host the website, as well as to allow us to store tweets. MongoDB, Python, and Java IDE must be supported by the server.

Additionally, the computer should also be equipped with 4GB RAM and high performance CPU for calculate large amount of data. The client computer is just the normal PC used in daily life which could open browser and link to Internet.

Some of the data collection snapshots are displayed here: (according to the hashtags we shortlisted)

	TweetID	TweetText	UserID	CreateTime
Edit Copy Delete	527566517672890368	Pilateando en Olimpic !! Ya no puede entrar nadie ...	2358627272	2015-10-30 12:43:24
Edit Copy Delete	527566523301642240	Pilateando en Olimpic !! Ya no puede entrar nadie ...	2358627272	2015-10-30 12:43:24
Edit Copy Delete	527566533946793984	Buenas noches familia! #SoyDeAgua #shark #surf #wa...	2358627272	2015-10-30 12:43:24
Edit Copy Delete	527566538640203776	If you want to lose weight contact http://t.co/oRh...	2365562754	2015-10-30 12:43:24
Edit Copy Delete	527566540531838977	Because what's a day with out my weight loss coffe...	88336290	2015-10-30 12:43:24
Edit Copy Delete	527566554267807744	Entrenando con demasiado calor #GYM espalda y tric...	2756704688	2015-10-30 12:43:24
Edit Copy Delete	527566558722146304	#WeightLoss #Program Health Tip: Practice a Well-B...	534928178	2015-10-30 12:43:24
Edit Copy Delete	527566559783313409	#Healthy #Nutrition How to Lose Weight With Thermo...	598370918	2015-10-30 12:43:24
Edit Copy Delete	527566559921729536	#EmmaWatson #Fitness #Diet ###Fitness Fitness on t...	559369958	2015-10-30 12:43:24
Edit Copy Delete	527566561129672704	#FatLoss #Healthy 5 Secrets To Weight Loss While G...	601794252	2015-10-30 12:43:24
Edit Copy Delete	527566562564132864	#Workout #GetFit Tips to Lose Fat http://t.co/erCn...	769756746	2015-10-30 12:43:24
Edit Copy Delete	527566563617284096	Después de una sesión de gym toca una duchita fres...	573865263	2015-10-30 12:43:24
Edit Copy Delete	527566565911179264	#JohnMayer #Fitness Paleo Diet Success Stories htt...	784182600	2015-10-30 12:43:24
Edit Copy Delete	527566584429428736	8 Min Abs The Classic - Level 1 #workout #abs #six...	189855189	2015-10-30 12:43:24
Edit Copy Delete	527566588455559168	Bem melhor assim... #Run #Mar #Floripa #VemVerão #...	98040902	2015-10-30 12:43:24
Edit Copy Delete	527566595267510274	Hoy recorrimos 10 kilómetros de #running con temas...	384937373	2015-10-30 12:43:24
Edit Copy Delete	527566634643632129	Before And After Weight Loss Pictures Women | http...	826488764	2015-10-30 12:43:24
Edit Copy Delete	527566635734159360	RT @MisTillas: Nike LunarGlide 5 de ricardo_spv #M...	83247301	2015-10-30 12:43:24
Edit Copy Delete	527566644407988224	Los Alamitos Creek Trail - equestrian route. #autu...	31481099	2015-10-30 12:43:24
Edit Copy Delete	527566658282729472	#toronto #biking : HOW-TO: 8 tips for riding home ...	384179652	2015-10-30 12:43:24

Console

Figure 29: Extracted Tweets

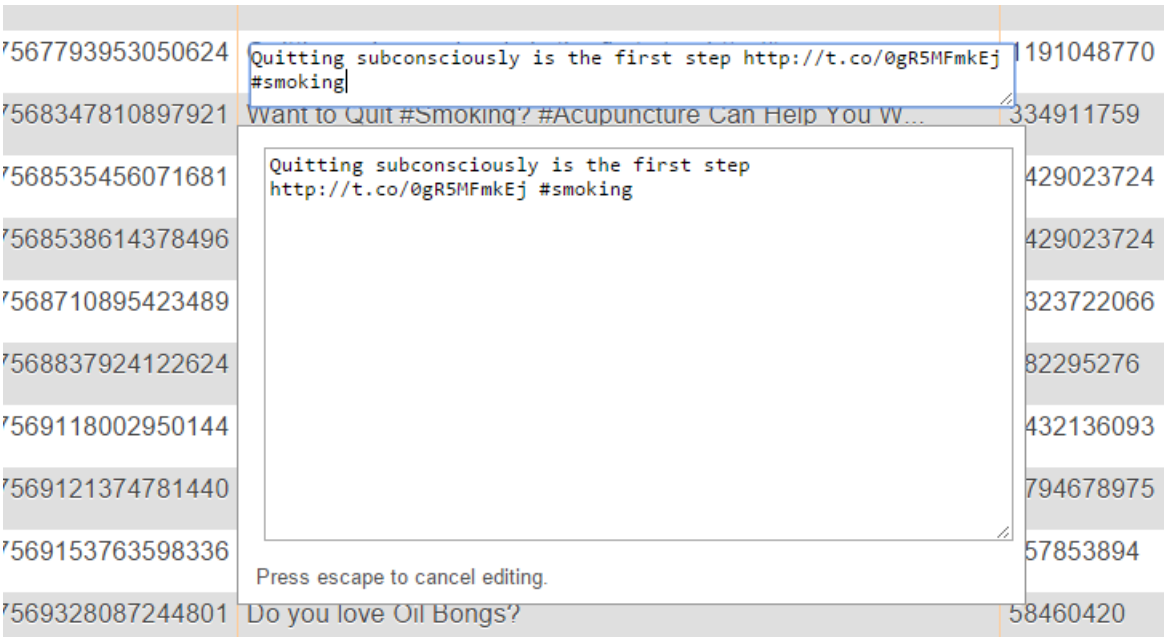


Figure 30: Sample tweet depicting how people tweet about being motived to quit smoking.

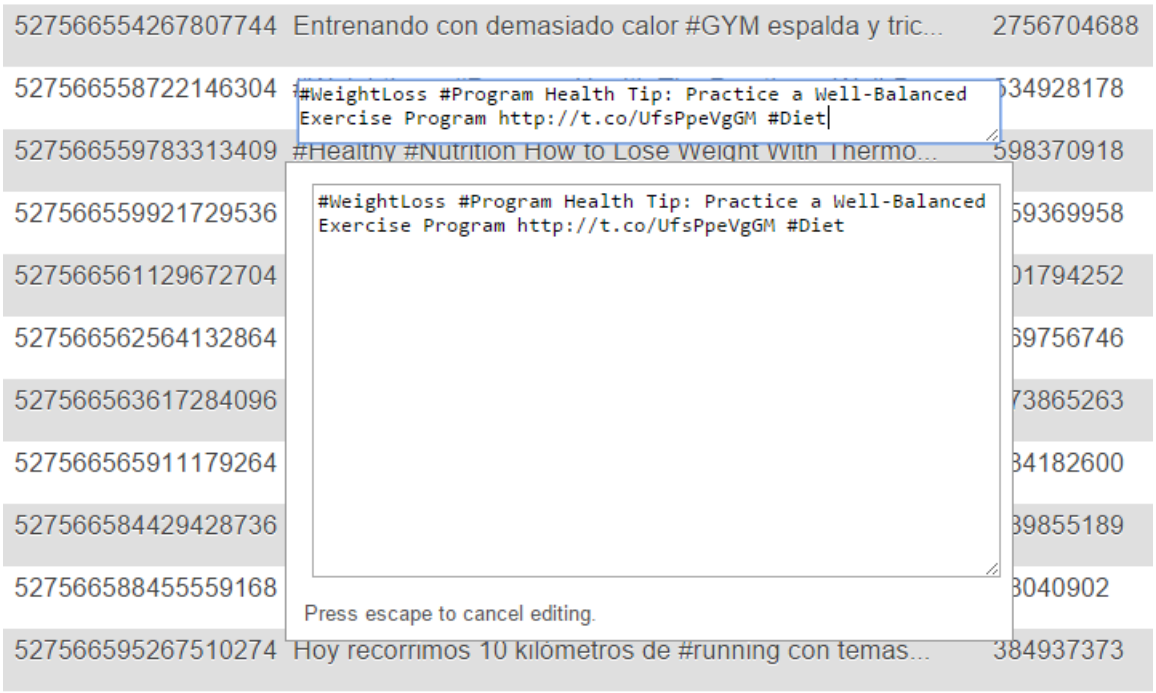


Figure 31: Sample tweet depicting how people tweet about being motivated to lose weight.

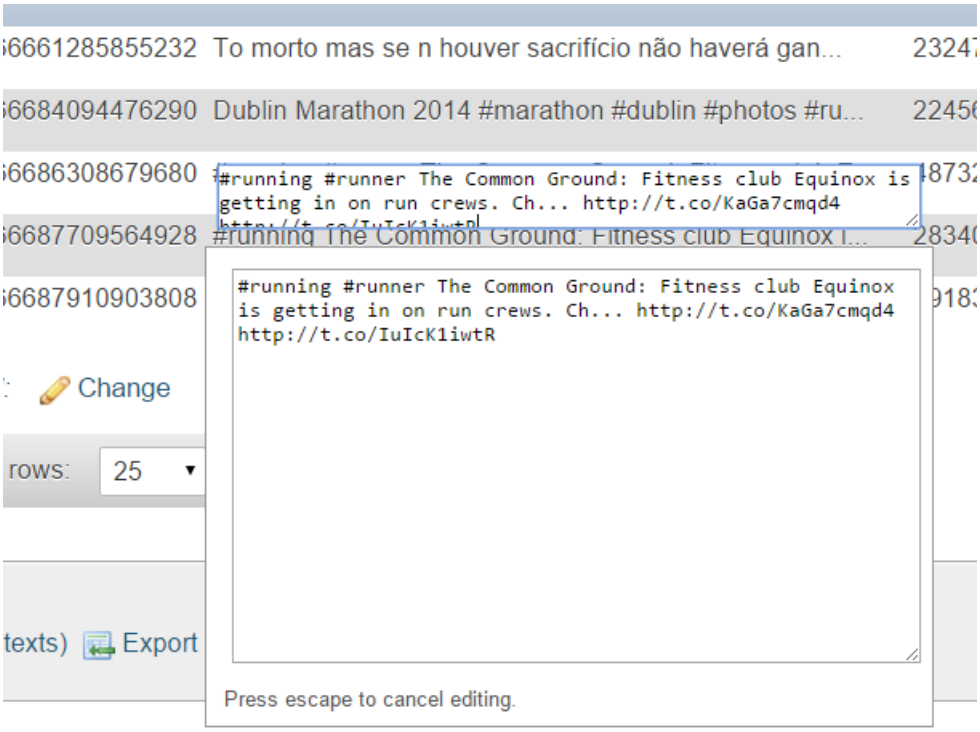


Figure 32: Tweet regarding fitness

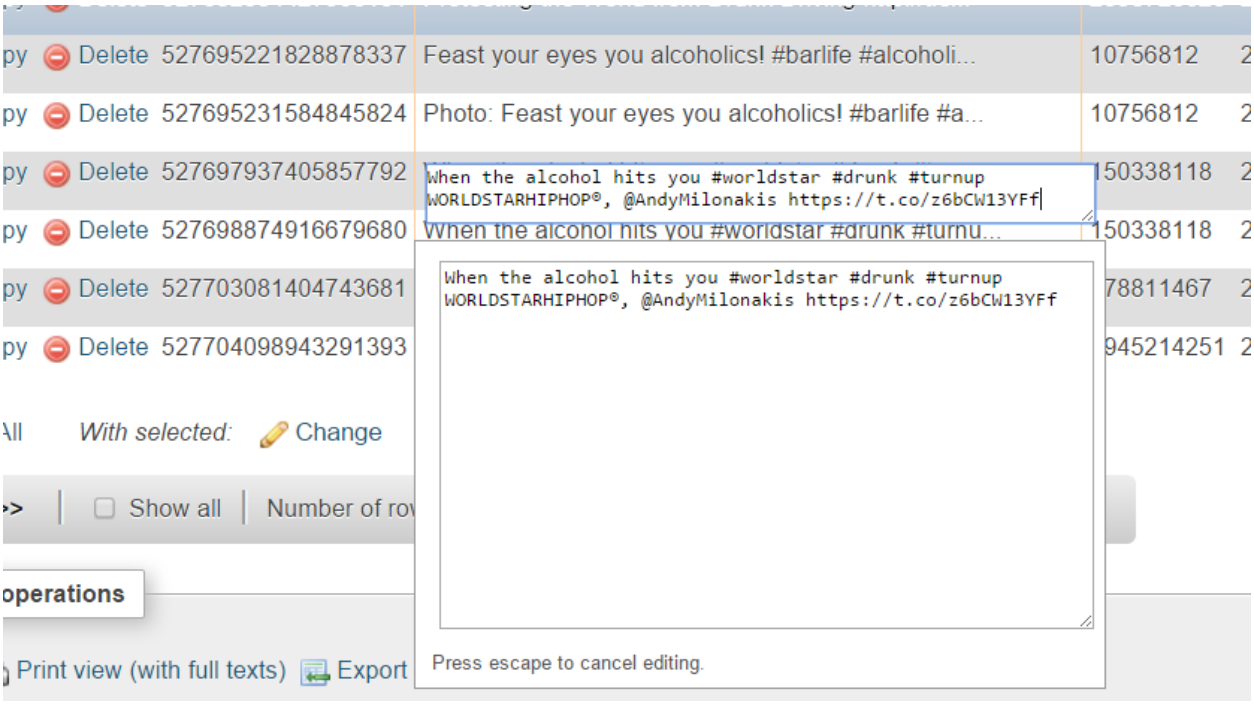


Figure 33: Tweet by a person who is into drinking.

10. Algorithms and Data Structures

10.1. Algorithms

For the data we are collecting from twitter, we face a problem to optimize the text categorization. As the tweets even are been collected by defined hashtags may have more than one label, we want to categorize them, and get the most fit set. So we chose to use the kNN method (multi-label K-nearest neighbor method).

10.1.1. Introduction

Multi-label K-nearest neighbor method works like this: we have an existing set of example data, our training set. We have labels for all of this data—we know what class each piece of the data should fall into. When we're given a new piece of data without a label, we compare that new piece of data to the existing data, every piece of existing data. We then take the most similar pieces of data (the nearest neighbors) and look at their labels. We look at the top k most similar pieces of data from our known dataset; this is where the *k* comes from. (*k* is an odd integer and it's usually less than 20.) Lastly, we take a majority vote from the k most similar pieces of data, and the majority is the new class we assign to the data we were asked to classify.

10.1.2 Mathematical Description

1. Data Pre-process

We convert all data to lower case, remove punctuation, numbers, URLs, and do stemming first. After than we can make sure all data in Tweets are unique meaning words. Then use these cleaned tweets, we can get a term-document matrix which row label is the words (terms), column label is the tweet numbers (documents), and the content is the term frequency. Once we get the matrix, we can do the clustering. The sample term-document matrix (1980-2000 terms, 500-550 tweets) is as following:

	Docs																								
Terms	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549
smokefetishist	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokefree	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokefreegov	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokefreeteen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokeless	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokers	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
smokey	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokinfitgirls	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smoking	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
smokingampriding	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingban	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingcessation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingdp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingfetish	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokinggirl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingicknow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smokingthe	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
smoko	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snoopdogg	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
snuf	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

When we get the matrix, it is easy to get the frequency of terms, just the sum of the row numbers.

Then after removing sparse (we set sparsity as 0.95), we can cluster terms.

Hierarchical Clustering

For the hierarchical clustering, we use Wald's method using a set of dissimilarities for the n objects being clustered. Initially, each object is assigned to its own cluster and then proceeds iteratively, at each stage joining the two most similar clusters, continuing until there is just a single cluster. At each stage distances between clusters are recomputed by the Lance–Williams dissimilarity update formula, which is as following:

Lance and Williams (1967) established a succinct form for the update of dissimilarities following an agglomeration. The parameters used in the update formula are dependent on the cluster criterion value. Consider clusters (including possibly singletons) i and j being agglomerated to form cluster $i \cup j$, and then consider redefining the dissimilarity relative to an external cluster (including again possibly a singleton), k. We have:

$$d(i \cup j, k) = a(i) \cdot d(i,k) + a(j) \cdot d(j,k) + b \cdot d(i,j) + c \cdot |d(i,k) - d(j,k)|$$

where d is the dissimilarity used – which does not have to be a Euclidean distance to start with, insofar as the Lance and Williams formula can be used as a repeatedly executed recurrence, without reference to any other or separate criterion; coefficients $a(i), a(j), b, c$ are defined with reference to the clustering criterion used (see tables of these coefficients in Murtagh, 1985, p. 68; Jambu, 1989, p. 366); and $|.|$ denotes absolute value.

The Lance-Williams recurrence formula considers dissimilarities and not dissimilarities squared.

A number of different clustering methods are provided. *Ward's* minimum variance method aims at finding compact, spherical clusters. Two different algorithms are found in the literature for Ward clustering. We implement the criterion Murtagh (1985) and Legendre (2012).

We start with (let us term it) the Ward1 algorithm as described in Murtagh (1985).

It was initially Wishart (1969) who wrote the Ward algorithm in terms of the Lance-Williams update formula. In Wishart (1969) the Lance-Williams formula is written in terms of squared dissimilarities, in a way that is formally identical to the following.

Cluster update formula:

$$\delta(i \cup i', i'') =$$

$$\frac{w_i + w_i''}{w_i + w_{i'} + w_{i''}}\delta(i, i'') + \frac{w_i' + w_i''}{w_i + w_{i'} + w_{i''}}\delta(i', i'') - \frac{w_i''}{w_i + w_{i'} + w_{i''}}\delta(i, i')$$

$$\text{and } w_{i \cup i'} = w_i + w_{i'} \tag{3}$$

We now look at the Ward2 algorithm described in Kaufman and Rousseeuw (1990) and Legendre and Legendre (2012).

At each agglomerative step, the extra sum of squares caused by agglomerating clusters is minimized, exactly as we have seen for the Ward1 algorithm above. We have the following.

Cluster update formula:

$$\delta(i \cup i', i'') =$$

$$\left(\frac{w_i + w_i''}{w_i + w_{i'} + w_{i''}}\delta^2(i, i'') + \frac{w_i' + w_i''}{w_i + w_{i'} + w_{i''}}\delta^2(i', i'') - \frac{w_i''}{w_i + w_{i'} + w_{i''}}\delta^2(i, i')\right)^{1/2}$$

$$\text{and } w_{i \cup i'} = w_i + w_{i'} \tag{4}$$

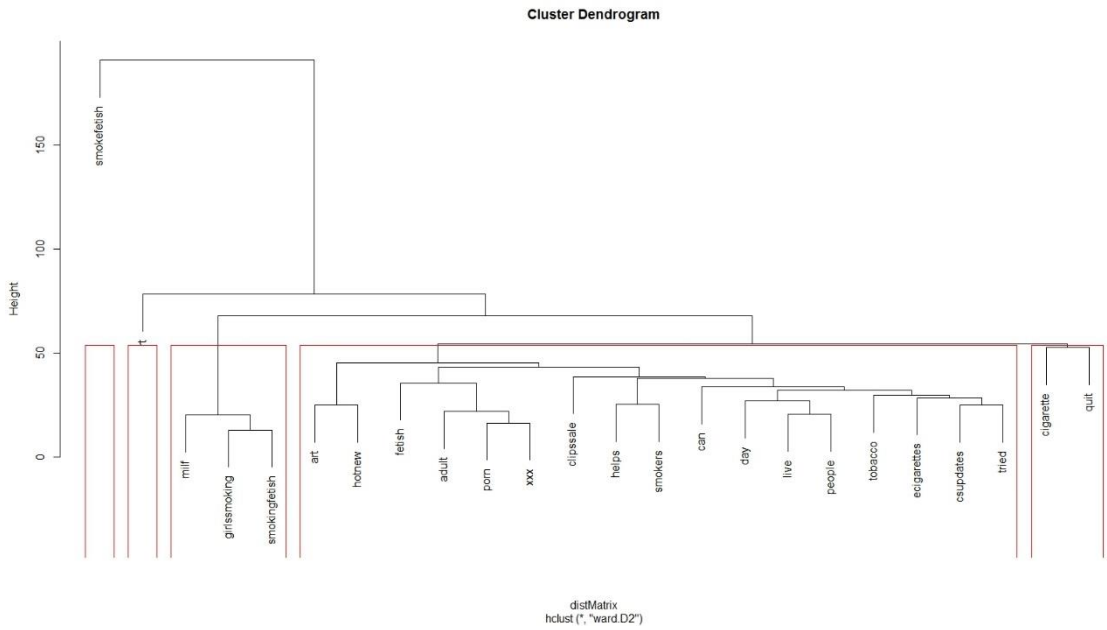
Contrary to Ward1, the input dissmilarities are Eucludean distances (not squared). They are squared within equation (4): $\delta^2(i, i') = \sum_j (x_{ij} - x_{i'j})^2$. It is such squared Euclidean distances that interest us, since our motivation arises from the error sum of squares criterion.

If members != NULL, then d is taken to be a dissimilarity matrix between clusters instead of dissimilarities between singletons and members gives the number of observations per cluster. This way the hierarchical cluster algorithm can be 'started in the middle of the dendrogram', e.g., in order to reconstruct the part of the tree above a cut. Dissimilarities between clusters can be efficiently computed only for a limited number of distance/linkage combinations, the simplest one being *squared* Euclidean distance and centroid linkage. In this case the

dissimilarities between the clusters are the squared Euclidean distances between cluster means.

In hierarchical cluster displays, a decision is needed at each merge to specify which subtree should go on the left and which on the right. Since, for n observations there are n-1 merges, there are 2^{(n-1)} possible orderings for the leaves in a cluster tree, or dendrogram. After ordering the subtree so that the tighter cluster is on the left (the last, i.e., most recent, merge of the left subtree is at a lower value than the last merge of the right subtree). Single observations are the tightest clusters possible, and merges involving two observations place them in order by their observation sequence number.

We use R to implement these methods, two of results are as following, the first using Murtagh (1985) and the second using Legendre (2012):



K-means clustering

K-means clustering is the most popular partitioning method. It requires the analyst to specify the number of clusters to extract. A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters. Conceptually, the K-means algorithm:

Selects K centroids (K rows chosen at random)

Assigns each data point to its closest centroid

Recalculates the centroids as the average of all data points in a cluster (i.e., the centroids are p-length mean vectors, where p is the number of variables)

1. Assigns data points to their closest centroids
2. Continues steps 3 and 4 until the observations are not reassigned or the maximum number of iterations (R uses 10 as a default) is reached.
3. Implementation details for this approach can vary.
4. R uses an efficient algorithm by Hartigan and Wong (1979) that partitions the observations into k groups such that the sum of squares of the observations to their

assigned cluster centers is a minimum. This means that in steps 2 and 4, each observation is assigned to the cluster with the smallest value of:

$$SS(k) = \sum_{i=1}^{n} \sum_{j=0}^{p} (x_{ij} - \bar{x}_{kj})^2$$

5. Where k is the cluster,xij is the value of the jth variable for the ith observation, and xkj-bar is the mean of the jth variable for the kth cluster.

6. K-means clustering can handle larger datasets than hierarchical cluster approaches. Additionally, observations are not permanently committed to a cluster. They are moved when doing so improves the overall solution. However, the use of means implies that all variables must be continuous and the approach can be severely affected by outliers. They also perform poorly in the presence of non-convex (e.g., U-shaped) clusters.

7. The format of the K-means function in R is kmeans(x, centers) where x is a numeric dataset (matrix or data frame) and centers is the number of clusters to extract. The function returns the cluster memberships, centroids, sums of squares (within, between, total), and cluster sizes.

Since K-means cluster analysis starts with k randomly chosen centroids, a different solution can be obtained each time the function is invoked. Use the set.seed() function to guarantee that the results are reproducible. Additionally, this clustering approach can be sensitive to the initial selection of centroids. The kmeans() function has an nstart option that attempts multiple initial configurations and reports on the best one. For example, adding nstart=25 will generate 25 initial configurations. This approach is often recommended.

2. Data Analysis

First, our original data is in form of pure text of tweets. We need to formulate the input of k-NN algorithm in standards. To achieve this, we do Natural Language analysis on each tweet to extract vectors in two different properties. We consider that a tweet can be identify on the words which are most informative. We implement this in Term Frequency & Inverse Document Frequency (TFIDF). Here are some mathematical definition of TFIDF.

Term Frequency:

In the case of the term frequency tf(t,d), the simplest choice is to use the *raw frequency* of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of $t$by $f_{t,d}$, then the simple tf scheme is tf(t,d) = $f_{t,d}$. Other possibilities include

- Boolean "frequencies": tf(t,d) = 1 if t occurs in d and 0 otherwise;

- Logarithmically scaled frequency: tf(t,d) = 1 + log $f_{t,d}$, or zero if $f_{t,d}$ is zero;
- augmented frequency, to prevent a bias towards longer documents, e.g. raw frequency divided by the maximum raw frequency of any term in the document:

$$\mathrm{tf}(t,d) = 0.5 + \frac{0.5 \times f_{t,d}}{\max\{f_{t,d} : t \in d\}}$$

Inverse Document Frequency:

The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

$$\mathrm{idf}(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

With

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: Number of documents where the term t appears (i.e. $\mathrm{tf}(t,d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$

In order to adapt this method with the clustering, we apply TFIDF along with cosine-similarity calculation for evaluating the extent of a tweet dependency to a group of labeled tweets.

Cosine Similarity:

The cosine of two vectors can be derived by using the Euclidean dot product formula:

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

Given two vectors of attributes, *A* and *B*, the cosine similarity, *cos(ϑ)*, is represented using a dot product and magnitude as

$$\mathrm{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

After applying methods listed below, we are able to formulate one property of tweets.

Second, we will move on to the sentimental analysis of feelings of twitterers when posting tweet. Since our categories are based on extent of positive, motive and negative, the sentiment of certain training set like positive smoking and quit smoking will definitely show differences. We implement the sentimental analysis by using Natural Language Toolkit based on Naïve Bayes Classifier. We need to extract features from the original text rather than cleaned text.

Naïve Bayes Classifier:

Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\mathbf{x} = (x_1, \ldots, x_n)$ representing some *n* features (independent variables), it assigns to this instance probabilities

$$p(C_k|x_1, \ldots, x_n)$$

For each of *K* possible outcomes or *classes*.

The problem with the above formulation is that if the number of features *n* is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' Theorem, the conditional probability can be decomposed as

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}.$$

After analyzing sentiments with above methods, we retrieve another property of tweet.

Finally we apply the properties which have already been abstracted on numbers to k-Nearest Neighbor (kNN) Algorithm for classification of new tweet input.

K - Nearest Neighbor:

This is a supervised training algorithm. The training examples are vectors in a multidimensional feature space, each with a class label. The training phase of the algorithm consists only of storing the feature vectors and class labels of the training samples.

In the classification phase, *k* is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the label which is most frequent among the *k* training samples nearest to that query point.

A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance).

As we have been able to build up mathematical model, we use Euclidean distance for kNN.

10.1.3 Pseudo Code (kNN Method)

For every point in our dataset:
calculate the distance between inX and the current point
sort the distances in increasing order
take k items with lowest distances to inX
find the majority class among these items
return the majority class as our prediction for the class of inX

More symbolized pseudo-code:

Input: D, the set of k training objects, and test object $z = (\mathbf{x}', y')$

Process:

Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every object, $(\mathbf{x}, y) \in D$.

Select $D_z \subseteq D$, the set of k closest training objects to z.

Output: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

10.2 Text Classification – Language Analysis

There are two parts in this tweet analysis – K-NN algorithm being the first level of removal of garbage tweets which is implemented in R. The algorithm is discussed in the previous section. The second step is the tweet analysis which is performed by the language analysis. The tweets are classfied into different categories. In our project, our main motive is to categorize the tweets into the following categories:

1. A separate category for garbage(garbage)
2. People who follow a positive a diet (pos.diet)
3. People who follow a negative diet (neg.diet)
4. People who need motivation for following a positive diet (mot.diet)
5. People who drink liquor (pos.alcohol)
6. People who want to quit drinking alcohol (quit.alcohol)
7. People who smoke (pos.smoking)
8. People who want to quit smoking (quit.smoking)
9. People who workout (pos.exercise)
10. People who require motivation for workout (mot.exercise)

Tweet classification involves assigning a document to category by human means. The analysis tool provides a classification facility that takes training tweets for each of the above mentioned categories which is generated by us. The analysis tool learns how to classify the tweets that is in the SQL database based on what it learnt with the training tweets given by us.
For each category, around 50 training tweets are manually given. The training tweets are taken from Twitter using input from a website called Hashtagify which actually mentions the most related hashtags.
Using the training tweet dataset we train a set of character based language models for each category. This training process processes the data in 12 character sequences as specified by the NGRAM_SIZE. We initially chose an NGRAM_SIZE of 6 , but it was insufficient to get satisfactory results.

For example,

Tweet 1- I do not like running
From this example, one can see that with an NGRAM size of 6 , we get two sequence of characters that are,
1. I do not
2. like r
3.unning

Tweet 2- I like running.
1. I like
2. runnin
3. g

With this size, Tweets 1 and 2 will have almost 50% probability of being placed in both categories. But, we know that Tweet 1 goes into motivational exercise and Tweet 2 goes into positive exercise. So we need a larger NGRAM_SIZE. We chose an NGRAM_SIZE of 12. Though we could use a larger size, 12 proved to give a better classification and performance level in terms of the time it takes to classify a million tweets. The training is done on this data set provided by us for each category and then classification happens based on the training received.

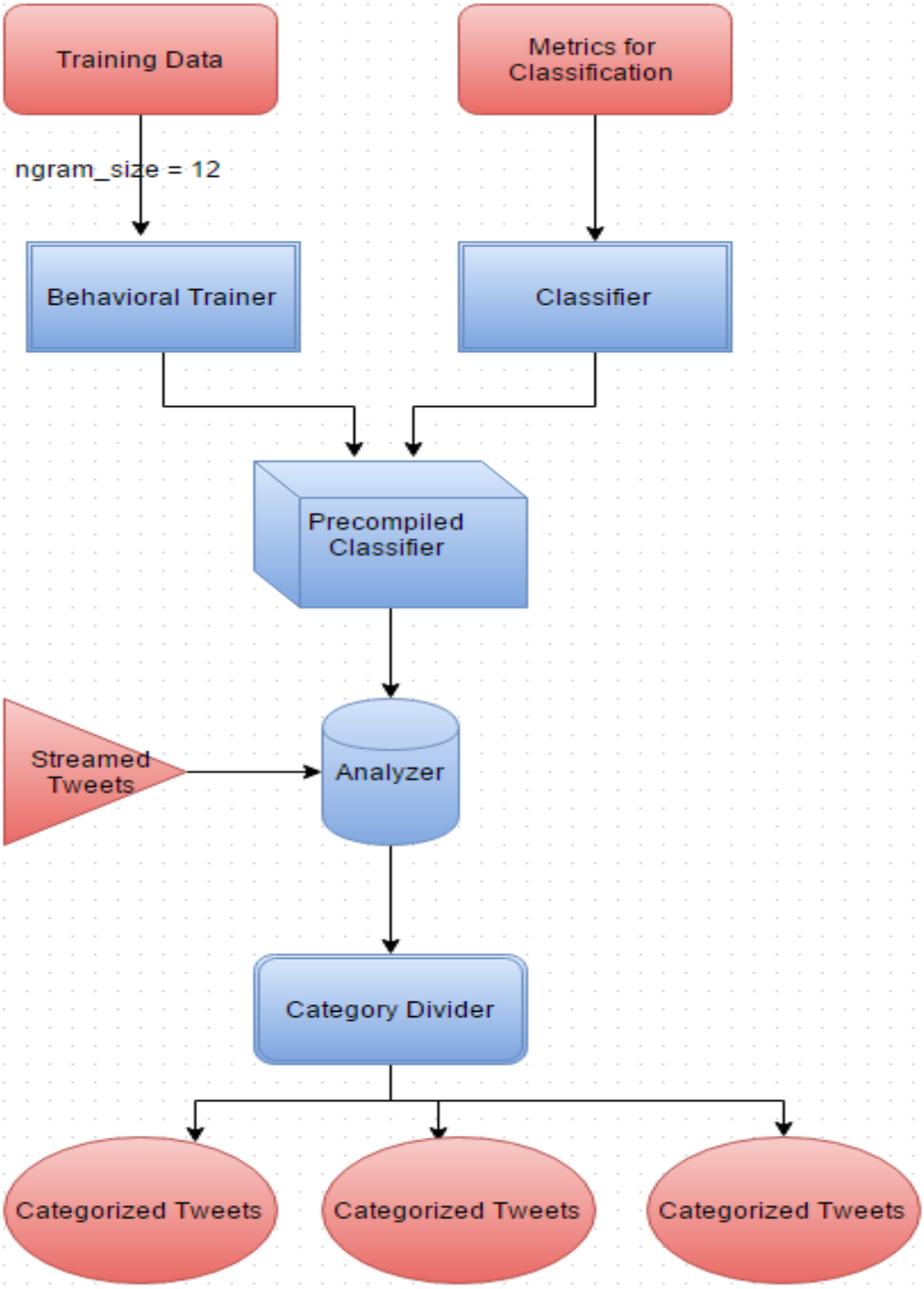


Figure 34: Text Classification of Tweets

Dynamic Classification:

Before using the trained classification, we pre-compile the classifier to produce the more efficient compile version which is pretty fast in the classification process. The code takes more time to compile so already compiled binaries prove much efficient.
Classification is actually classifying the tweet based on our categories, it takes the training tweets as a comparison matrix and runs the comparison on the input tweets. The following output shows the categorization of a tweet with respect to all the categories.

```
---------------
Testing on Tweet : Seriously need to start the #gym
Got best category of: mot.exercise
Rank  Category  Score  P(Category|Input)   log2 P(Category,Input)
0=mot.exercise -1.496112393251283 0.9999991238499288 -50.86782137054362
1=mot.diet -2.100313123366865 6.546258515056686E-7 -71.41064619447341
2=garbage.all -2.1462898242995343 2.2152395933099748E-7 -72.97385402618417
3=pos.exercise -2.7256835237995154 2.602044311942475E-13 -92.67323980918353
4=neg.diet -3.2069670654907974 3.0858021994177213E-18 -109.0368802266871
5=quit.smoking -3.330557216229261 1.6765651632222177E-19 -113.23894535179487
6=pos.alcohol -3.53548912975657 1.3394861716678278E-21 -120.20663041172338
7=quit.alcohol -3.565592969801908 6.589080811676464E-22 -121.23016097326487
8=pos.diet -3.7619455100165937 6.443957786874213E-24 -127.90614734056419
9=pos.smoking -3.7664197399929567 5.799076717001385E-24 -128.05827115976052

---------------

---------------
Testing on Tweet : I ran 3.17 mi with @MapMyRide.  #run #running http://www.mapmyride.com/workout/811185395
Got best category of: pos.exercise
Rank  Category  Score  P(Category|Input)   log2 P(Category,Input)
0=pos.exercise -2.8527878863654347 1.0 -262.45648572962
1=garbage.all -4.134556830482898 3.174806669032881E-36 -380.3792284044266
2=mot.exercise -5.472824622120887 2.746075747642947E-73 -503.4998652351216
3=pos.alcohol -5.8499575541069015 9.865216567972878-84 -538.196094977835
4=pos.diet -6.204123501972317 1.5330836026502418E-93 -570.7793621814532
5=quit.smoking -7.22456193441882 8.409472685313616E-122 -664.6596979665314
6=quit.alcohol -7.360285168829972 1.4653670268712897E-125 -677.1462355323574
7=neg.diet -7.787121250333351 2.212196743493594E-137 -716.4151550306683
8=mot.diet -7.810135402614176 5.098616344398212 4E-138 -718.5324570405041
9=pos.smoking -8.31873640164336 4.186681591308139E-152 -765.323748951189

---------------
Testing on Tweet : I spent 35 minutes on an elliptical machine. 545 calories burned. #LoseIt
Got best category of: pos.exercise
Rank  Category  Score  P(Category|Input)   log2 P(Category,Input)
0=pos.exercise -1.2436480492192274 1.0 -93.27360369144206
1=garbage.all -4.196181202014098 2.18736790881841 88E-67 -314.71359015105736
2=pos.diet -4.747524905133622 7.8000501374302 55E-80 -356.0643678850216
3=pos.alcohol -4.923838715990108 8.154859170745637E-84 -369.2879036992581
4=quit.smoking -5.01115252656083 8.71185752193385E-86 -375.83643949206225
5=neg.diet -5.0143028881384994 7.39578477466825E-86 -376.0727166103875
6=quit.alcohol -5.379770830959665 4.146897214521268E-94 -403.4828123219749
7=pos.smoking -5.444742539989052 1.415274802152266E-95 -408.35569049917893
8=mot.exercise -5.517904760108809 3.155288423528776E-97 -413.8428570081607
9=mot.diet -5.556683390958515 4.2026597756439786E-98 -416.7512543218886

---------------
```

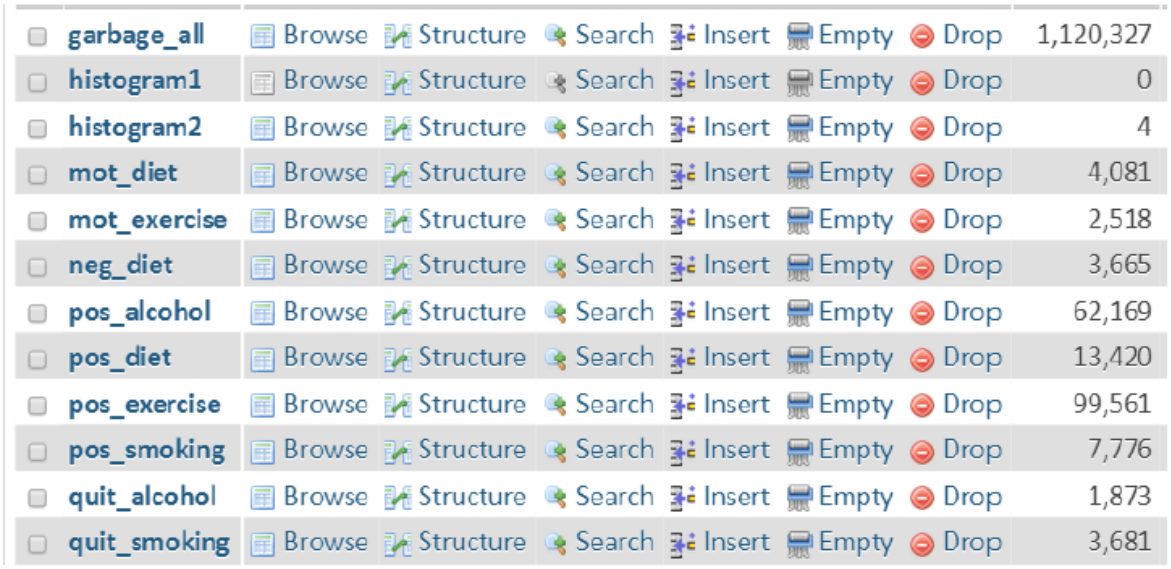
The above figure shows a sample tweet and the output of classifier. The classifier actually tests that tweet for all the categories and gives the expected probability of each category. All the categories are given ranks, and the tweets are finally categorized belonging to the classification which has lowest rank i.e.; Rank 0. For the sake of garbage collection and to maintain correctness

and accuracy of the data analysis we have a category ***"Garbage"***, if the classifier has no matching category it puts that tweet into the garbage category.

The introduction of garbage category proves to be beneficial not only to improve correctness of analysis but also provides us with a metrics to calculate the fractional useful tweets. We can use total no. of tweets and garbage tweets to find out what fraction of the total tweets are not useful.

Once the classification is done we move on to tagging tweets to their categories. This tag is used later on for the purpose of analysis. As soon as the Rank is determined, the tweet is tagged and is moved to the corresponding category table in SQL Database.

For the sake of convenience and dependable tagging process, we have introduced tables in SQL Database. Each category has its own table. The below figure clearly depicts our DB arrangement for tagging process.



garbage_all	Browse	Structure	Search	Insert	Empty	Drop	1,120,327
histogram1	Browse	Structure	Search	Insert	Empty	Drop	0
histogram2	Browse	Structure	Search	Insert	Empty	Drop	4
mot_diet	Browse	Structure	Search	Insert	Empty	Drop	4,081
mot_exercise	Browse	Structure	Search	Insert	Empty	Drop	2,518
neg_diet	Browse	Structure	Search	Insert	Empty	Drop	3,665
pos_alcohol	Browse	Structure	Search	Insert	Empty	Drop	62,169
pos_diet	Browse	Structure	Search	Insert	Empty	Drop	13,420
pos_exercise	Browse	Structure	Search	Insert	Empty	Drop	99,561
pos_smoking	Browse	Structure	Search	Insert	Empty	Drop	7,776
quit_alcohol	Browse	Structure	Search	Insert	Empty	Drop	1,873
quit_smoking	Browse	Structure	Search	Insert	Empty	Drop	3,681

As soon as the categorization id finished, copying of tweets into corresponding table takes place at the same time, while the tweet is getting copied to SQL DB the java analyzer takes the next tweet in queue and starts its analysis.

Is classification Legitimate?

For the process of classification, asking a question about its legitimate output is crucial for data analysis. Our Analysis process gets into jeopardy once the classifier starts segregation of tweets in unexpected manner. To make sure that the classification is done properly we use probabilistic determination of tweets provided categories. Each tweet is given the probability for each category. The probabilistic determination is described below:

A JointClassification is a conditional classification derived from a joint probability assignment to each category and the object being classified. The conditional probabilities are computed from the joint probabilities, but an additional score may be provided for ordering. These scores must be ordered in the same way as the joint probabilities. For example, the language model classifiers implement the score as an entropy rate to allow between-document comparisons.

In addition to the score and conditional probability methods, this interface adds a method to retrieve joint log (base 2) probability by rank, jointLog2Probability(int).

The conditional probability estimate of the category given the input is derived from the joint probability of category and input:

P(category|input) = P(category,input) / P(input)

where the joint probability P(category,input) is determined by the joint probability estimate and the input probability P(input) is estimated by marginalization:

P(input) = Σcategory P(category,input)

In the study of probability, given at least two random variables X, Y, ..., that are defined on a probability space, the joint probability distribution for X, Y, ... is a probability distribution that gives the probability that each of X, Y, ... falls in any particular range or discrete set of values specified for that variable.

10.3. Data Structure

The most important data structure in our system is JSON format. JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is used primarily to transmit data between a server and a web application, as an alternative to XML.

Although originally derived from the JavaScript scripting language, JSON is a language independent data format, and code for parsing and generating JSON data is readily available in a large variety of programming languages.

The tweets information extracted from Twitter is stored in the JSON format. Each tweet contains 24 JSON objects being counted which you can find detail explanations at a website here: https://dev.twitter.com/docs/twitterids-json-and-snowflake. These are some example tweets information we extracted in JSON format.

Here is an example of tweet in JSON:

```
u'source': u'<a href="http://twitter.com/#!/download/ipad" rel="nofollow">Twitter for iPad</a>',
u'text': u'RT @edgarvnovobot: I want to ride my bicycle.  #bicycles #bikes #cycling http://t.co/HVr9OVfjHL',
u'timestamp_ms': u'1444427298663',
u'truncated': False,
u'user': {u'contributors_enabled': False,
          u'created_at': u'Sun Apr 27 13:58:14 +0000 2014',
          u'default_profile': True,
          u'default_profile_image': False,
          u'description': None,
          u'favourites_count': 33636,
          u'follow_request_sent': None,
          u'followers_count': 279,
          u'following': None,
          u'friends_count': 898,
          u'geo_enabled': False,
          u'id': 2512712213L,
          u'id_str': u'2512712213',
          u'is_translator': False,
          u'lang': u'en-gb',
          u'listed_count': 98,
          u'location': None,
          u'name': u'David Clark',
          u'notifications': None,
          u'profile_background_color': u'C0DEED',
          u'profile_background_image_url': u'http://abs.twimg.com/images/themes/theme1/bg.png',
          u'profile_background_image_url_https': u'https://abs.twimg.com/images/themes/theme1/bg.png',
          u'profile_background_tile': False,
          u'profile_banner_url': u'https://pbs.twimg.com/profile_banners/2512712213/1398843396',
          u'profile_image_url': u'http://pbs.twimg.com/profile_images/461585157938765824/QeoLwJdb_normal.jpeg',
          u'profile_image_url_https': u'https://pbs.twimg.com/profile_images/461585157938765824/QeoLwJdb_normal.jpeg',
          u'profile_link_color': u'0084B4',
          u'profile_sidebar_border_color': u'C0DEED',
          u'profile_sidebar_fill_color': u'DDEEF6',
          u'profile_text_color': u'333333',
          u'profile_use_background_image': True,
          u'protected': False,
          u'screen_name': u'davidpj70clark',
          u'statuses_count': 33085,
          u'time_zone': None,
          u'url': None,
          u'utc_offset': None,
          u'verified': False}}
```


Figure 35: Sample JSON

As we can see, tweets JSON data contains the useful information such as the user's profile, tweets context, location, etc. So we can extract them out and use them for analysis based on each user and community

11 User Interface Design and Implementation

11.1 Preliminary Design

The user interface as discussed before was developed keeping Requirements and Use Cases in mind. The sole purpose was to help user in getting the required information investing minimum effort. The interface mock ups provided previously were able to meet the requirement of embodying the Use cases completely. In order to provide ease of use for users and easy understanding of the UI just by a glance at it is crucial in developing the UI. We are using HTML to develop our web framework and extend its functionality by using JSP, graphs and statistical information are calculated and displayed on the web page using java.

11.2 Purposed Improvements

Here are parts of our websites showing some of our features mentioned above.

Figure 36: Homepage (1)

When users enter our websites, they well see a welcome interface.

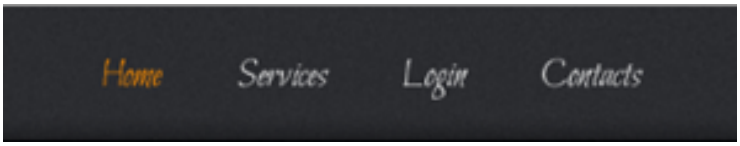


Figure 37: Navigation

Users can choose to login by click “Login” on the top right of the web to find more personalized detail. And they can find out main features of our project after clicking “Services”. And “Home” will allow users to return the home page.

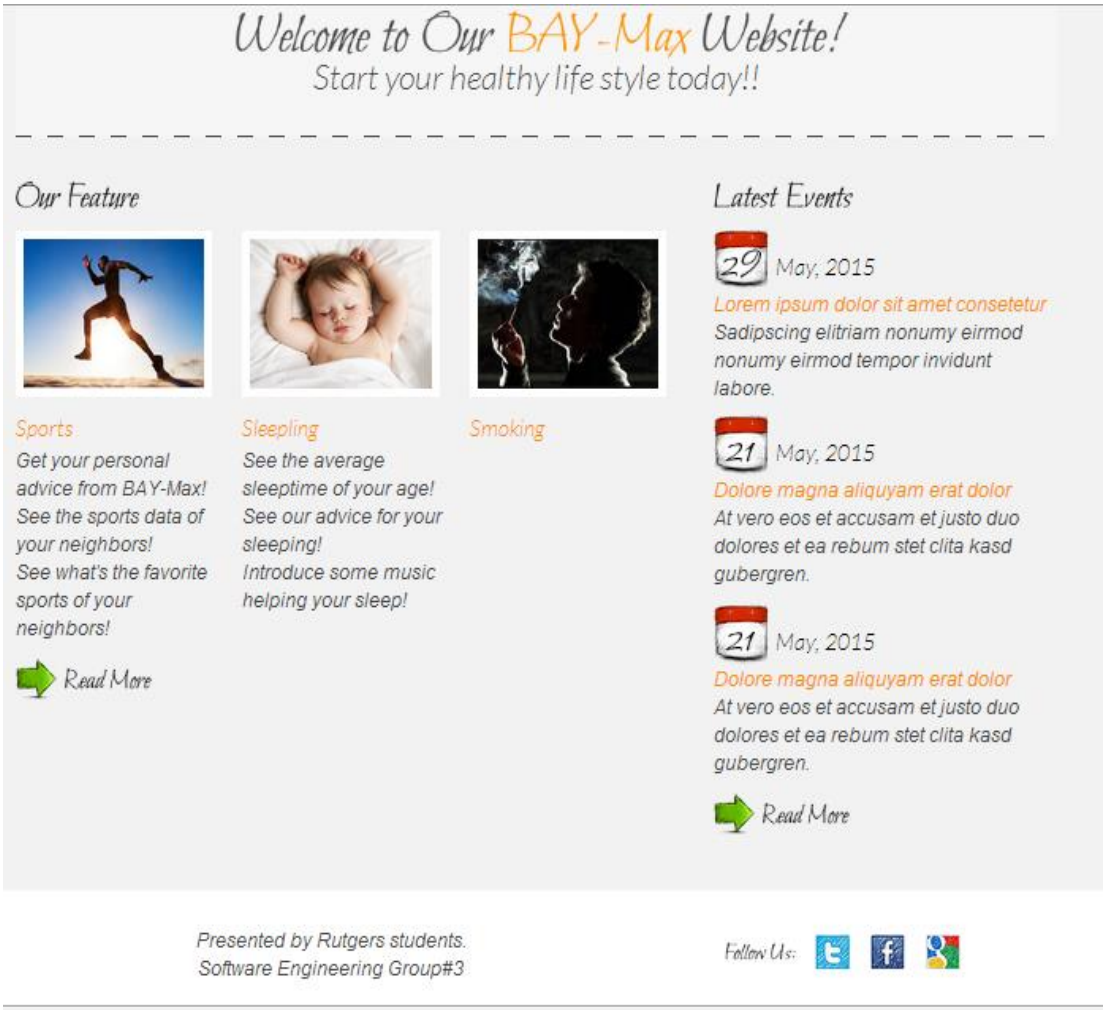


Figure 38: Homepage (2)

On the left side, users are also available to access those features from links below the welcome interface.

And on the right side, users will find out recent events happened nearby from the calendar displayed above, and figure out what they are interested in then add it to their own schedule.

Figure 39: Services page

Users can click “Services” and enter this page.

On this page, users are able to find a heat map showing the density of, for example, people who exercise or smoke of different regions.

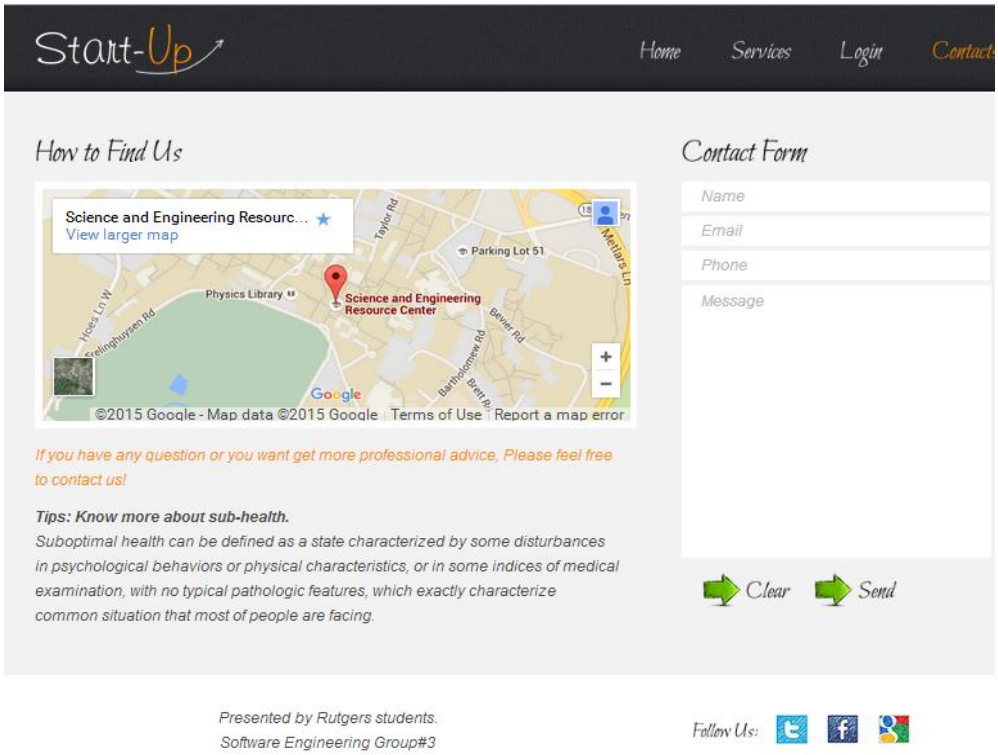


Figure 40: Contact us page

Users will enter this page after clicking “Contacts” and they can easily find the location of our team from the map.

And they can also write down advises about our project on the contact form showed at the right of this website.

11.2 Implemented Website

When you click into our website, you will see the flowing home page:

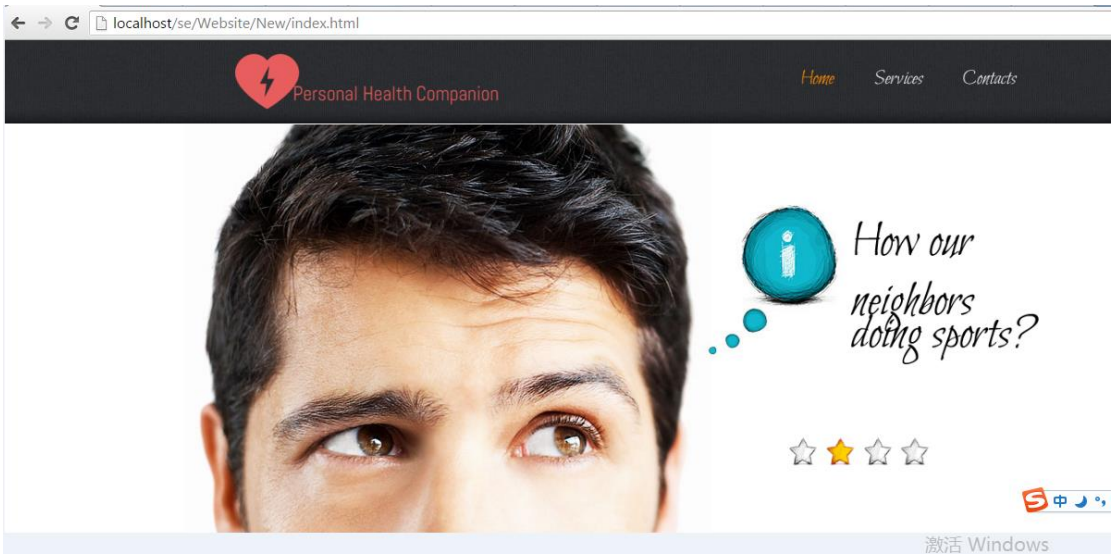


Figure 41: Home page

When user click into the service, which you can find in the navigation part:

Figure 42: Homepage navigation

User will see such a table for our features. And User can choose any of the features that they are interested in and click the button “Read More”, and it will jump into the corresponding html file.

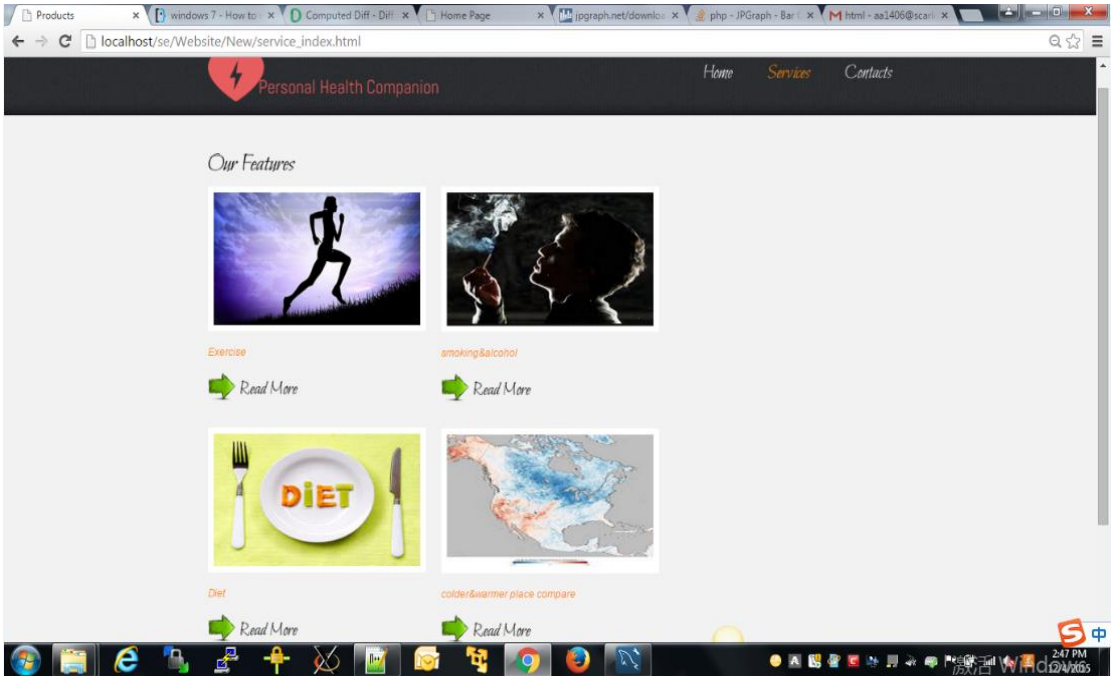


Figure 43. Features page

If I click "Exercise", the website will goes into the exercise.html, showing the weekly exercise trends.

Figure 44: Weekly Trends for exercise

Similarly we have diet and Smoking & Alcohol weekly trends as below:

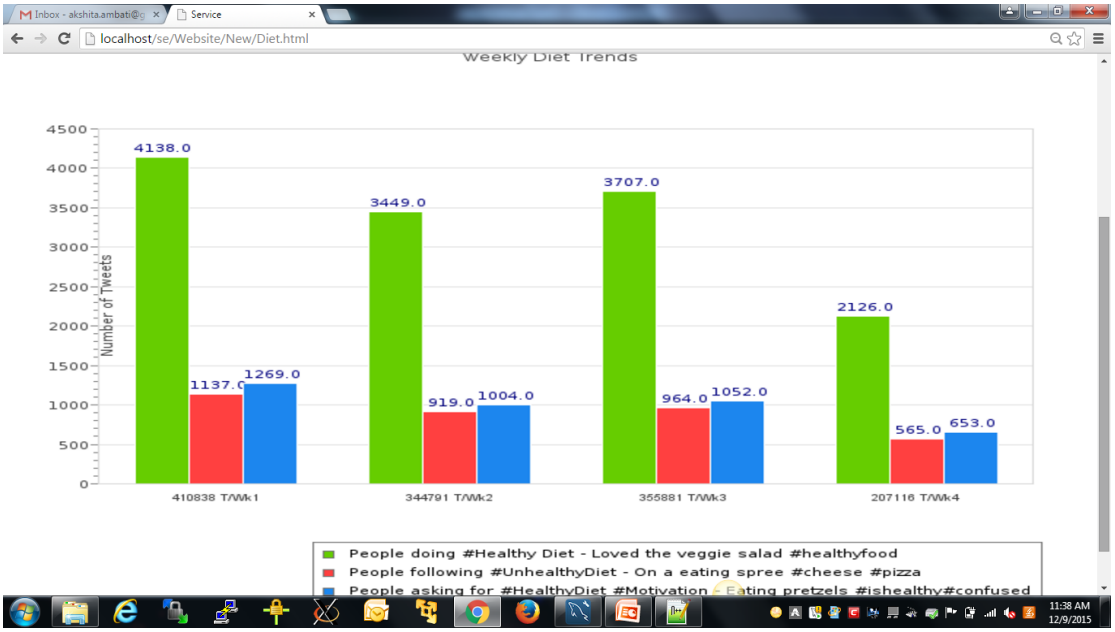


Figure 45: Weekly Trends for Diet

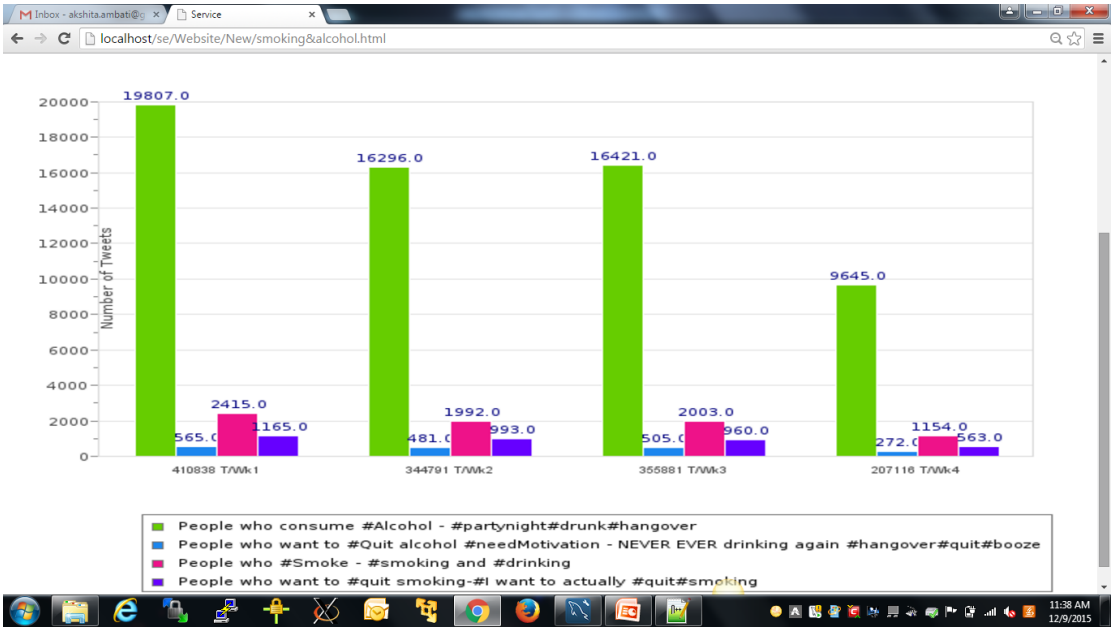


Figure 46: Weekly Trends for Smoking & Alcohol

As you can see, there is a navigation part too. And when you click into certain feature, it will show a drop-down list based on our different features.

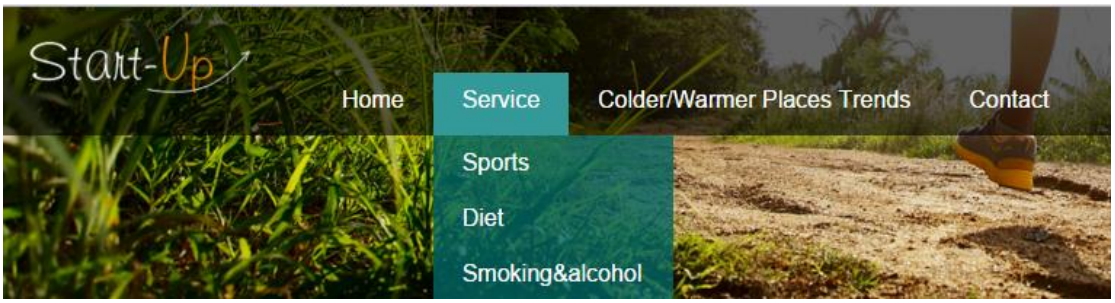


Figure 47: Navigation for features

When I want to know the difference trend between the warm and cold place, you can click any of the features as you like. For example, I will click " diet ".

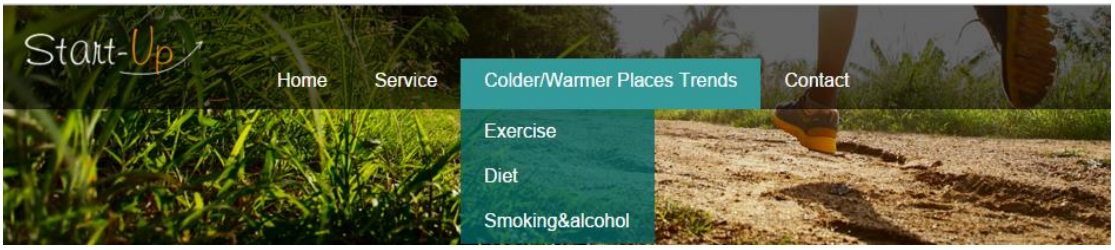


Figure 48. Navigation for features

The following three screenshots are the Exercise, Diet and Smoking & Alcohol trends for the Colder and Warmer places

Figure 49. Weekly Exercise Trends for Colder/Warmer places

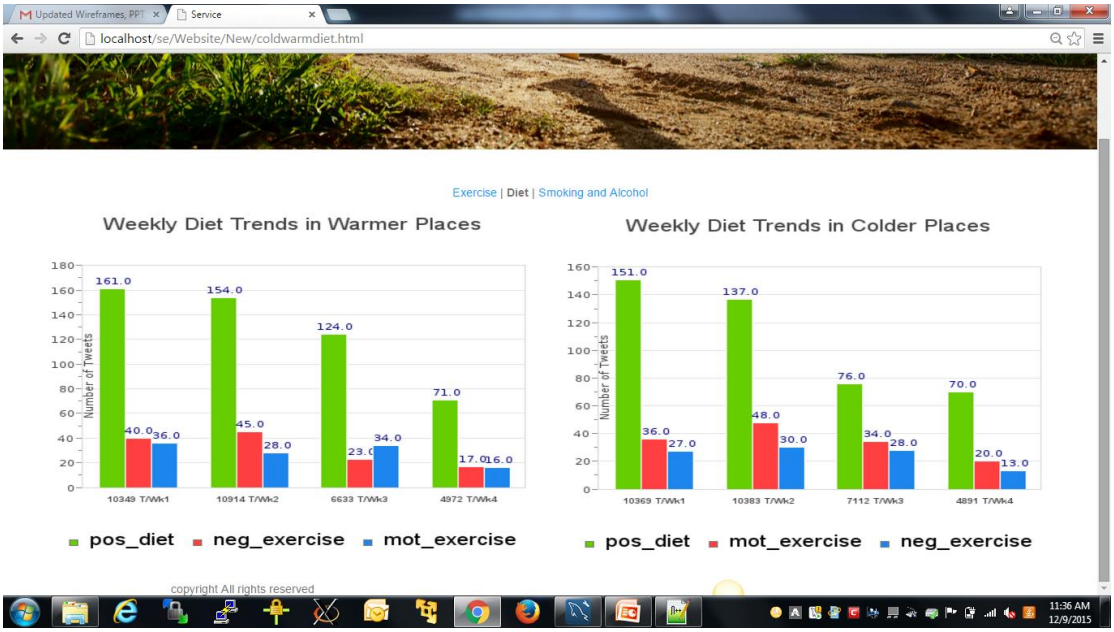


Figure 50: Weekly Diet Trends for Colder/Warmer places

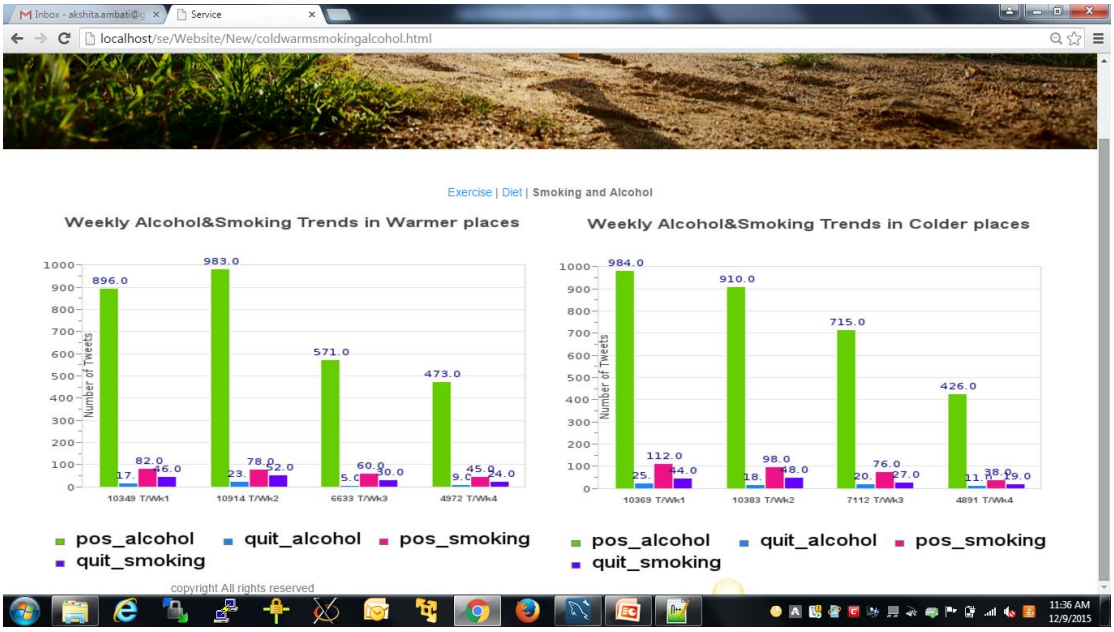


Figure 51: Weekly Smoking & Alcohol Trends for Colder/Warmer places

Users will enter this below page after clicking “Contacts” and they can easily find the location of our team from the map.

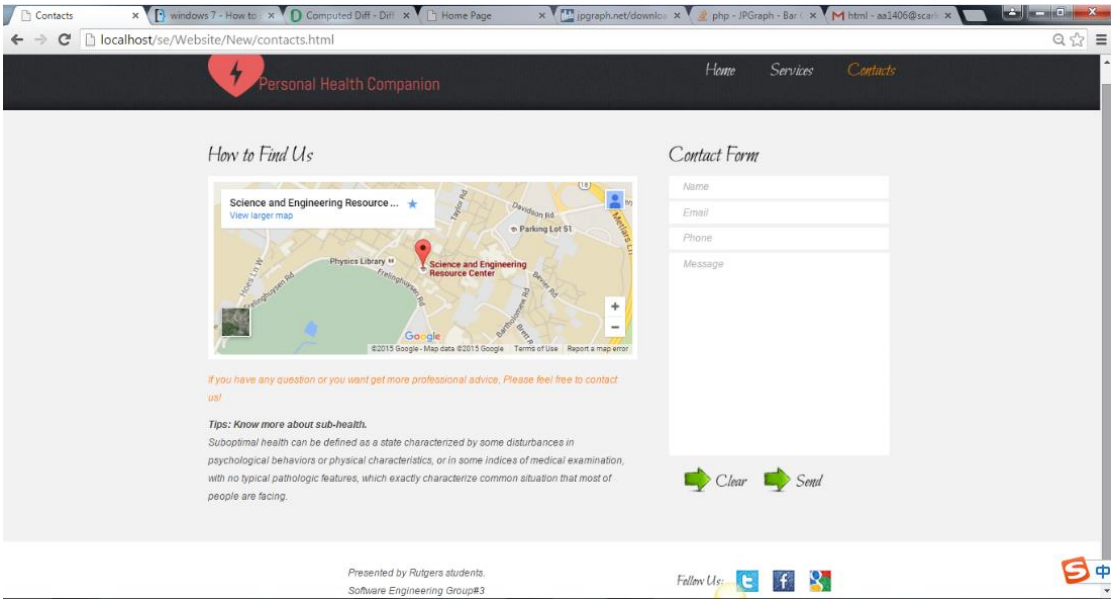


Figure 52: Contact page

And they can also write down advises about our project on the contact form showed at the right of this website.

12. Design of Tests

12.1. Test Cases

12.1.1. Function Test

The use cases are divided into independent parts, which mean that the use cases can be easily tested as a unit. So in this part, we decide to list and describe the test cases by the use case list.
Use Case Function list:
1. Leaderboard
2. Histogram
3. Calculator
4. Calendar
5. Twitter collection

1. Test ID: TC1_Leaderboard

Input Requirement	Expected Output	Pass/Fail	Comments

User enters the website and clicks on the leaderboard.	The database sends back the data of leaderboard. The website shows the leaderboard as a graph.	Pass if the website shows the graph of the current leaderboard. Fail if the website shows nothing or out of- date leaderboard.	The test is to make sure that the data collected from twitters can be updated on time and displayed as leaderboard on the website.

2. Test ID: TC2_WorkoutHistogram

Input Requirement	**Expected Output**	**Pass/Fail**	**Comments**
User enters the website and clicks on the button of different histogram (diet, exercise and smoke).	The database sends back the specific data of histogram. The website shows the histogram.	Pass if the website shows the graph of the current histogram. Fail if the website shows nothing or out-of-date histogram.	The test is to make sure that the website can receive the histogram choose and display the required histogram.

3. Test ID: TC3_Calculator

Input Requirement	**Expected Output**	**Pass/Fail**	**Comments**
User uploads the individual situation in the calculator blank.	The calculator calculates the individual situation by mathematics formula and shows the result.	Pass if the website shows calculator result correctly. Fail if the website does not receive the uploaded data or the website cannot show the result correctly.	The test is to make sure that the website can calculate the data that uploaded by the users.

4. Test ID: TC4_Calender

Input Requirement	**Expected Output**	**Pass/Fail**	**Comments**
User enters the website.	The website displays the calendar correctly.	Pass if the website shows calendar correctly.	The test is to make sure that the website can calculate the data that uploaded by the users.

		Fail if the website cannot show the calendar correctly.	

5. Test ID: TC5_CollectTwitterInformation

Input Requirement	Expected Output	Pass/Fail	Comments
The data needed by the Health Monitor Analytics occurs on the twitter.	The data needed by the Health Monitor Analytics is collected successfully in JSON and stored completely in the Mongodb.	Pass if the Mongodb stores all the data that the system needed. Fail if data collects the wrong data or the system crashes.	The test is to make sure that the data can be collected correctly and successfully. The database can store all the data collected.

12.1.2. Data Relevant Test

As a Personal Health Companion System, we should keep our system data source more accurate to ensure our website to be credible. Since we collect the information from the twitter, the information noise is unavoidable. So we make the effort to filter the twitter data by some Algorithms (like KNN method).It is necessary to do the Data Relevant Test except the Function test (11.1.1).The test is started by the raw data collected by the twitter. We search the #word in Twitter. We will get the raw relevant data like the figure below. In this example, we search #run.

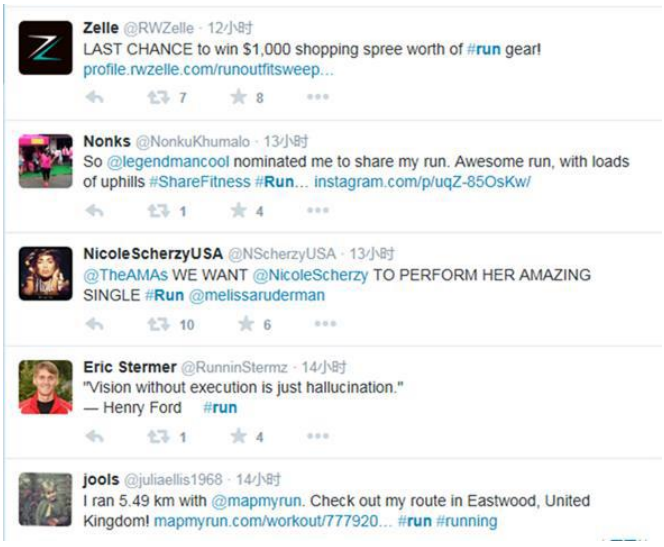


Figure 53: Tweet Example

The data in the figure will be collected completely in the database as the raw data. We use our Analytics algorithms to filter the data. The filtered data is stored as a list. As a test, it is impossible to analyze all the data. We will take a hundred filtered twitters from the list randomly. Then we judge the data relevance manually and count the number of the relevant data. At last, we can get the accuracy percentage of the analytics algorithms.
Certainly, we will make several parallel groups of the same analytics algorithm and achieve the mean accuracy percentage. If the accuracy percentage passes the threshold value, we will regard the analytics algorithm as a qualified algorithm. We will test all the filter algorithms in our system in this way.

12.1.3. Acceptance Test

ATC1.01
Enter the website URL and click on Sign Up button. The user should be able to see the registration page.
ATC2.02
After successful registration, the user should be able to login with registered user ID and password.
Acceptance Test Case 2
After successful login user should be able to click on profile tab and be able to edit their personal information
User should be able to save and edit the user profile any number of times after a successful login.
Acceptance Test Case 3
The system should be able to assign a weightage to each workout style in order to compute the amount of workout done by the user.
Acceptance Test Case 4
After successful login, the user will be able to view the leader board containing the colder and warmer places comparison on terms of dietary, smoking/alcohol and exercise patterns
Acceptance Test Case 5
When a registered user clicks on Exercise tab, the user should be able to view the Exercise data analysis performed on twitter data.
When a registered user clicks on Smoking & Alcohol tab, the user should be able to view the Smoking and Alcohol data analysis performed on twitter data.
When a registered user clicks on Diet tab, the user should be able to view the Dietary data analysis performed on twitter data.
Acceptance Test Case 6
Registered user should be able to view the data analysis performed by the system on the website, after a successful login.
Acceptance Test Case 7
A registered user belonging to a community should be able to invite other registered user to join that community.

12.2 Test Coverage

We use test coverage to get the degree to which the specification or code of a software program has been exercised by tests. As mentioned in 11.1.2, we divide the test by use case list. So it belongs to the State-based testing.
State-based testing defines a set of abstract states that a software unit can take and tests the unit's behavior by comparing its actual states to the expected states. State based testing is used where some aspects of the system can be described in what is called a 'finite state machine'. This simply means that the system can be in a number of different states, and the transitions from one state to another are determined by some rules. We use the State-based unit for test coverage because the website system is like the objected oriented system. The function is defined as the object. It works as an individual state independently. So it will be easy to test in this way.

12.3 Integration Testing Strategy

In this system, we decide to use **Vertical Integration Testing Strategies** to test our codes for different reasons.
The codes in our system have several characteristics as listed below:
1. The use case functions are divided independently.
2. Most of the use cases in our system give the feedback of the data as the function of the website.
3. The system is designed for the customers who want the data relevant to health.
4. The system is designed for daily uses.
The **Vertical Integration Testing Strategies** have several characteristics as listed below:
1. The vertical integration approaches to develop the user stories in parallel for testing the code.
2. Each story is developed in a feedback loop, where the developers use unit tests in the inner loop and the customer runs the acceptance test in the outer loop.
3. Each cycle starts with the customer/user writing the acceptance test that will test a particular user story. Based on the acceptance test, the developer writes the unit tests and develops only the code that is relevant, i.e., needed to pass the unit tests.
4. The unit tests are run on daily basis, soon after the code is written, and the code is committed to the code base only after it passes the unit tests. The acceptance test is run at the end of each cycle (order of weeks or months).
So the Vertical Integration Testing Strategies fit our aim to test the code in our system. We believe that Vertical Integration Testing Strategies can fit our code completely and properly.

13. Product Ownership

As, suggested by the professor we divided the group into 3 sub teams comprising of 2 members each.

Sub Team 1: Shikha Kakar and Harika Matta

Sub Team 2:Akshita Ambati and Yueyang Chen

Sub Team 3: Jianyu Zhang and Ruiqi Lin

1. Functionality:

Task/ Feature	Ownership
Dietary habits trends of the last 4 weeks	Sub-team 1
Exercising trends based of last 4 weeks	Sub-team 2
Smoking/Alcohol trends of last 4 weeks	Sub-team 3
Data analysis on how people after getting help on their lifestyle make amends and how progressively that is visible.	Sub-team 1,2,3
Analysis by comparing the Dietary habits trends of the last 4 weeks between colder and warmer places	Sub-Team 1
Analysis by comparing the exercise habits trends of the last 4 weeks between colder and warmer places	Sub-Team 2
Analysis by comparing the Alcohol/Smoking habits trends of the last 4 weeks between colder and warmer places	Sub-Team 3

2. Qualitative Property:
   All the sub-teams will work on tuning their respective sub-parts and once the whole project has been integrated, all the team members will work to improve the performance of the whole system.

14. History of Work, Current Status and Future Work

14.1 Project Coordination and Progress

In short, we have most analysis part finished and less pretty for user interface.

We have adjusted the directions and weight more into the data analysis after demo1. Considering most of our use cases are basically the requirements for the user interface but little on data analysis, we decided to stress more on analysis and push off the implementation of several use cases. The redirection makes great sense since the results we show now are quiet pretty. Before demo2, we enlarged our database along with more tweets collected, which made our analysis more representative.

For the analysis part, we made great progress after taking more efforts. We are able to finish the state-wise analysis as well as periodic trends. In addition, we did some cross comparison over different seasons and find some interesting result which will be helpful for user. For the seasonal analysis we wanted to take 1 year of data because of the constraints of the resources as the size of the archive was 33 GB .We evaluate the importance of Clustering Algorithms and the Natural Language Process because their accuracy ensure that our system is credible. How to improve the accuracy still remains a big problem, and our geological analysis are inadequate for now. The heat map and some real-time analysis like calendar have already been proceeded but not completely finished. All these are different aspects of Natural Language Processing and we can foresee in the future these consecutive use cases will be implemented.

For the user interface part, we simply create the homepage in a pretty style and implement basic functions. It is now becoming a public reference website. The login use case and things integrate to the login are weakened or abandoned for now. We only keep basic functions of showing all kinds of graphs for now and other functions are future works.

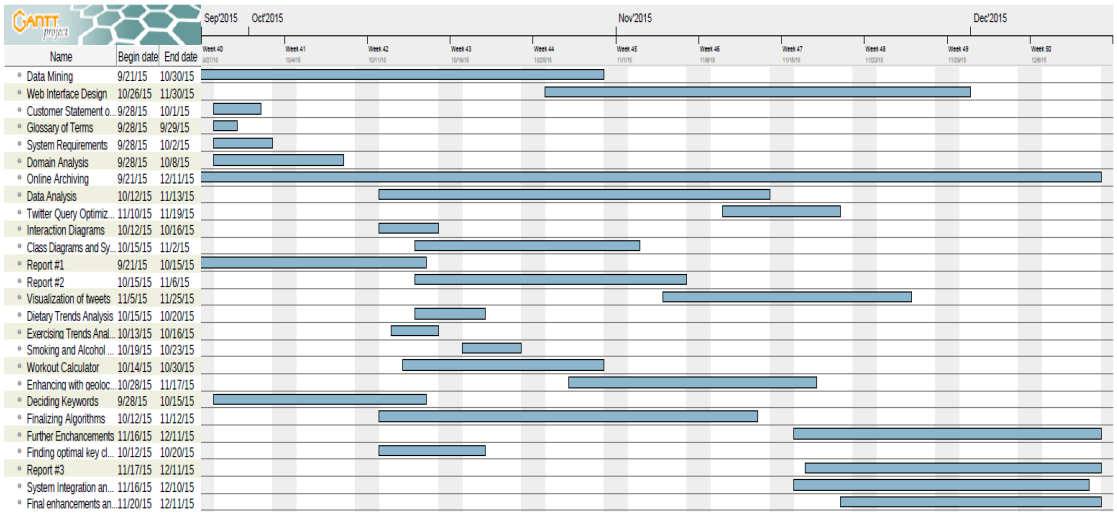
14.2 Former Plan of Work

Tasks

Name	Begin date	End date
Data Mining	9/21/15	10/30/15
Web Interface Design	10/26/15	11/30/15
Customer Statement of Requirements	9/28/15	10/1/15
Glossary of Terms	9/28/15	9/29/15
System Requirements	9/28/15	10/2/15
Domain Analysis	9/28/15	10/8/15
Online Archiving	9/21/15	12/11/15
Data Analysis	10/12/15	11/13/15
Twitter Query Optimization	11/10/15	11/19/15
Interaction Diagrams	10/12/15	10/16/15
Class Diagrams and System Architecture	10/15/15	11/2/15
Report #1	9/21/15	10/15/15
Report #2	10/15/15	11/6/15
Visualization of tweets	11/5/15	11/25/15
Dietary Trends Analysis	10/15/15	10/20/15
Exercising Trends Analysis	10/13/15	10/16/15
Smoking and Alcohol Consumption Analysis	10/19/15	10/23/15
Workout Calculator	10/14/15	10/30/15
Enhancing with geolocation	10/28/15	11/17/15
Deciding Keywords	9/28/15	10/15/15
Finalizing Algorithms	10/12/15	11/12/15
Further Enchancements	11/16/15	12/11/15
Finding optimal key cluster size	10/12/15	10/20/15
Report #3	11/17/15	12/11/15
System Integration and Testing	11/16/15	12/10/15
Final enhancements and testing	11/20/15	12/11/15

We have described the timeline for all our tasks here. Most of our work sticks on plan.

Gantt Chart

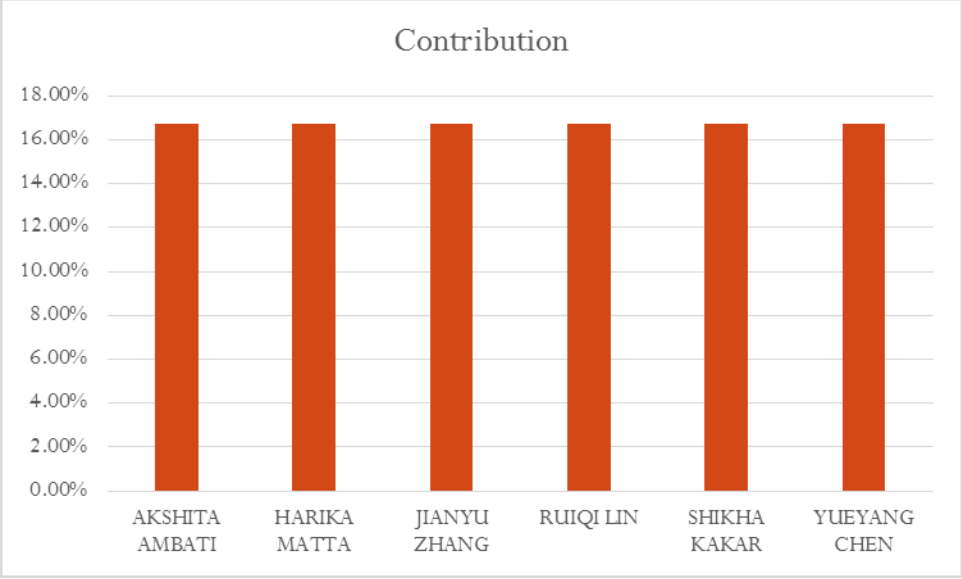


Responsibility Matrix:

Everyone contributes equally!

Responsibilities	Akshita	Harika	Jianyu	Ruiqi	Shikha	Yueyang
Project management(10)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Customer Statement of Requirement(9)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
System Requirements(6)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Functional Requirements(30)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
User Interface Specs(15)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Domain Analysis(25)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Interaction Diagrams(30)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Class Diagrams & Interface Specification(10)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
System Architecture & Design(15)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%

Algorithm & Data Structure(4)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
User Interface Design & Implementations(11)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Design of Test(12)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Product Ownership(18)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Plan of Work(5)	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Display of Histogram	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
History of Work, Current Status and Future Work	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Data Collection	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Database Management	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
R Studio	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
Language Analysis	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%
References	16.7%	16.7%	16.7%	16.7%	16.7%	16.7%



15. Reference

[1] “Software Engineering” A book by *Ivan Marsic*

[2] “Twitter Data Analytics” by *Shamanth Kumar, Fred Morstatter and Juan Liu*

[3] Http://www.who.int/en/ : “World Health Organization”

[4] Global Burden of Disease Study 2013 Collaborators*(Theo Vos; Ryan M Barber; Brad Bell; et al) [GBDS] (2013). “Global, regional, and national incidence, prevalence, and years lived with disability for 301 acute and chronic diseases and injuries in 188 countries, 1990–2013: A Systematic Analysis for the Global Burden of Disease Study 2013.” *The Lancet Articles, Volume 386, No.9995, p743-800, 22 August 2015.* DOI: 25 September 2015http://dx.doi.org/10.1016/S0140-6736(15)60692-4

[5] Li, Guolin; FuxiaXie; Siyu Yan; Xiaofei Hu; Bo Jin; Jun Wang; Jinfeng Wu; Dazhong Yin; QingjiXie (2013). "Subhealth: definition, criteria for diagnosis and potential prevalence in the central region of China". *BMC Public Health* **13** (1): 446.doi:10.1186/1471-2458-13-446. ISSN 1471-2458

[6] http://smartwebexperts.blogspot.com/2014/07/concepts-of-3-tier-n-tier-mvc.html

[7] *Java Look and Feel Design Guidelines* : http://java.sun.com/products/jlf/ed2/book/

[8] Software Engineering Project: Health Monitoring Analytics
http://www.ece.rutgers.edu/ marsic/books/SE/projects/HealthMonitor/ analytics.html