|  |
| --- |
| **Senior Project Report**  The Development of a Complex Decision Making System “Edible Me”  Sam Yan  Student ID: 0930756  Adviser: Dr.Changjiang Zhang |



**TABLE OF CONTENTS**

[Preface 5](#_Toc24072)

[Acknowledge 6](#_Toc14204)

[Chapter 1 Proposal About Automatic Menu System: Edible Me 7](#_Toc29413)

[1.1 Keywords 7](#_Toc22353)

[1.2 Abstraction 7](#_Toc23620)

[1.3 Introduction 7](#_Toc5170)

[1.4 Problem Statement 8](#_Toc16326)

[1.5 Objective and outcomes 10](#_Toc3042)

[1.6 Methodology 11](#_Toc8247)

[1.7 Project Schedule 15](#_Toc2475)

[1.8 Resource Estimation 15](#_Toc26788)

[Chapter 2 Software Requirement Specification for Project: Edible Me 16](#_Toc3502)

[2.1 Introduction 16](#_Toc23782)

[2.2 Description 17](#_Toc26539)

[2.2.1 Perspective 17](#_Toc961)

[2.2.2 Functions 17](#_Toc18746)

[2.2.3 User classes and characteristics 17](#_Toc32464)

[2.2.4 Initial Platform 18](#_Toc4738)

[2.2.5 Constraints 18](#_Toc9921)

[2.2.6 Assumptions and Dependencies 18](#_Toc30193)

[2.3 Interface Requirements 18](#_Toc14022)

[2.3.1 User Interfaces 18](#_Toc12941)

[2.3.2 Software Interfaces 19](#_Toc26647)

[2.3.3 Hardware Interfaces 19](#_Toc25031)

[2.3.4 Communication Interfaces 19](#_Toc9980)

[2.4 Use Cases 19](#_Toc31999)

[2.4.1 Actor list 19](#_Toc5310)

[2.4.2 Use case list 20](#_Toc8221)

[2.4.3 Use case diagram 20](#_Toc31201)

[2.4.4 Use case details 21](#_Toc28201)

[2.5 Nonfunctional Requirements 22](#_Toc8164)

[Appendix A: Glossary 22](#_Toc20012)

[Chapter 3 Design Specification of “Edible Me” 24](#_Toc27480)

[3.1 Database Design 24](#_Toc1148)

[3.2 Package Design 31](#_Toc27910)

[3.3 Database-Application Interface Design 32](#_Toc17847)

[3.4 Class Diagram Design 35](#_Toc14887)

[3.5 Security Policies 36](#_Toc252)

[Chapter 4 “Edible Me” Testing Report 38](#_Toc8449)

[4.1 Project Structure 38](#_Toc145)

[4.2 Testing Strategies 42](#_Toc2932)

[4.3 Testing Results 45](#_Toc29892)

[4.4 Conclusion 48](#_Toc18133)

[Chapter 5 Future Work 49](#_Toc25295)

[5.1 Discussion of recent progresses 49](#_Toc25551)

[5.2 Suggestions for future work 49](#_Toc32699)

[Bibliography 50](#_Toc13857)

## Preface

This book is about my graduation project, developing a complex software information system. It has been actually 3 months since I started to prepare for my final graduation, during which I learned a lot. Those knowledge not only includes those software development phases from proposal to testing the project, but also the factors we have to consider, such as making full utilities of hardware and platforms that our software is running on. Even more, one course which impress me most was the education of code of conduct that we, together as a future computer scientists, shall follow. I feel extremely lucky and happy on going this process of approaching graduation for those moments are worth to be memorized.

My project is a complex decision-making system recommending diet for people to decide what they shall eat for every meal considering different factors. I name the application “Edible Me” for it is about food and drinks and this name is easy to remember.

In those development processes, I did not choose pretty “fashion” techniques such as Android programming or server-client programming, because there are several concerns. First of all, I value the “cores” (or in other words bones or structures) of a system more than its appearances, namely how it was developed. New technologies merge and disappear, only the ideas remains important. Secondly, my project shall be able to reflect what I learned in those 4 years, not what I am learning. It is true that I am learning server client programming, Android programming and QT programming this semester, but as for the four years passed I tend to focus more on the conceptual parts, namely data structures and algorithms, inter-disciplinary researches and the study of developing modes. Thus in my project, the technical issues may not be that fancy or fashionable, it is barely Java SE for desktop. However, I would like my project to reflect more on my ideas of solving real problems using algorithms, developing modes and the ideas of fulfilling inter-disciplinary requirements.

Also, this project is extendable and personally, I really want that someone can continue my job later even though I left the university after graduation for this could be a really good material for graduation for both computer science students who study the developing processes of information systems and biology students who study nutrition. This is the reason I try to specify my designs and specifications as clear as possible. **I am willing to share all of my works for others from any place of any ages to check, study, continue working and criticize. All the materials from this book and all the programming codes are available for checking and re-using from my personal side as long as properly accredited.**

Numbers of people have supported me of all kinds and due to my memory limitation, I may not list all of them in the acknowledging list and I am personally terribly sorry for this. However, I want to appreciate all of Wenzhou-Keaners for the pleasure experiences I had during the four years, especially those helped me with my graduation project, whose names I tried my best to put all into the acknowledging list. Due to the limitations of my knowledge, time and the page limitations of this book, there may be bias in this book and I am willing to listen to opinions and criticizes from all sides.

Sam Yan

Wenzhou-Kean University

# Acknowledge

My mum Mary Ruan and dad Ken Yan for supporting me for the 4 years’ study in computer science major, Wenzhou-Kean University

Professors from Wenzhou-Kean University, especially,

Dr. Changjiang Zhang

Dr. Pinata Winoto

Dr. Tiffany Tang

Dr. Toby Michelena

Dr. Yoyo Meng

All my fellow of computer science major of 2017 of Wenzhou-Kean University

All fellow students majoring computer science of 2018 and 2019

All my friends from other majors in Wenzhou-Kean University who constantly cares about my graduation project

All computer science major of 2016, especially,

Mr. Andrew Zhou

Mr. Freddy Wang

Mr. Hui You

Mr. John Zheng

Mr. Lincoln Lin

Mr. Nick Zhang

Mr. Relic Wang

Mr. Will Wang

Friends from other schools who had their opinions about my project, especially,

Maclaurin Mao, Medical University of Wenzhou

Alex Gong, financial-engineering graduation school of Rutgers University

# Chapter 1 Proposal About Automatic Menu System: Edible Me

### 1.1 Keywords

Decision making, Object-oriented design, Spiral Model, Expert System

### 1.2 Abstraction

This project is a decision-making application which automatically generate user-specific menu[[1]](#footnote-0) based on the theories of nutrition and the analysis of users’ health conditions, dietary habits and living habits. The name of the formal application is designed as “Edible Me”. This name is chosen because it sounds more direct of its meaning. In addition to providing general records about the calorie consumption or exercising records, this application tends to combine many aspects together according to a novel new algorithm to judge the health conditions of the users. Apart from this function, the application also combines factors such as the users’ current diseases conditions and current living activities to decide the recommended menu in a certain period. The important aspect of this application is that it provides different nutrition advice for various people. Because according to nutrition theories, different groups of people have a variety range of nutritional requirements. Thus, simply offering the exact same or even similar models to people with large differences of nutritional requirements might be biased. Apart from being general only about users’ calorie consumption by recording exercises or menus taken, this application provides specific guidance for different users based on their various requirements about nutrition, which is a main contribution compared to other applications of the similar theme in the market. Plus, by keeping records of users’ menu, the application provides a possibility to improve its recommendation algorithms and makes it possible for researchers to study the trend of people’s nutritional habits. As for the implementation of this application, the whole developing phases will be divided into 3 phases using the spiral model.

### 1.3 Introduction

The proper management of dietary menu every day is so important that it influences every aspect of how a person functions. In order to make a decision about what to take for meals, 8 aspects have to be taken into consideration.

Although there are different types of mobile or network applications about healthy life, controlling calories and providing exercising guidance, little have been found out that could give users specific menu guidance based on their health conditions and living habits. Namely, those applications tend to either record what the user ate or for how long did they exercise, but usually have little guidance about what kinds of food should the user takes later. As from another perspective, few of them really consider about food safety issues.

The information system developed in this semester is trying to make up those short edges of the applications in the market. It designs food plans for both customers and registered users. As for customers, the application simply provides general tips on healthy diet. As for registered users, it provides several healthy diet services based on their health conditions (Information gathered under the agreement of the user) listed in the function session of the designed information system. In additional to recording people’s calorie consumption and diets, it tries to “deal with” those recorded data to give different users different specific guidance about their future nutrition plans. In order to give qualitative idea about whether a user meets his or her health requirements, this application decides a novel new algorithm according to relative nutritional theories to measure the aspects that compose the concept “healthy” in a quantitative manner.

General applications in the market about the theme “health live” only have functions such as recording their menu or recording their exercises. Few of them are able to consider the factors of both sides of menu and exercise. Until this proposal has been written, no application is found out to be able to deal with those health data and provide specific guidance for users. Thus, the core contribution of this application is to stress the importance of dealing with gathered data. Besides only collecting data as other applications might also do, it generates guidance according to the nutritional health algorithm which considers nearly every important aspects of nutritional health.

The overall information system is developed following a spiral model using program language Java, Visual Paradigm and Netbeans 8.2, with the connection towards application database.

### 1.4 Problem Statement

The arrangement of menu is a process of decision making, which takes consideration of eight aspects:

1. Current personal health condition: Including but not limited to a person’s current BMI (Body Mass Index), gender, specific health problems that a person is suffering from (such as diabetes, allergic or heart problems), types of work (decides the person sits for more time or exercise for more time) and living habits.
2. Adequacy: This means that a health diet should take into consideration of all the six types of nutrients: carbohydrates, fats, proteins, vitamins, minerals and water.
3. Balance: This means that types of nutrients should be arranged in a proper manner, in fair amounts to each other.
4. Energy: This means the food that a person takes should provide the person with enough energy. Usually, carbohydrates such as starch-based food and food contains fats and proteins takes a major rule in providing energy supply for a person.
5. Nutrient Density: This measures the amount of nutrient that the food can provide per unit of energy (usually measured in calorie in food theories), the higher the better.
6. Moderation: This means the total amount of food taken by a person should be managed to a proper manner, not too much and not too little.
7. Variety: This means the menu should include different types of food.
8. Food safety: Food safety is an important and broad issue, especially in place such as China. This usually includes but not limited to:
   * The original materials of food:

There are types of food that are toxic if being taken over-dose and there are cases about those information frequently. Those food includes but not limited to: Nan Tianzhu (a special type of plant who can be used to make medicine for only a little portion), He Tun (a type of fish if the wrong organism being eaten will cause great toxic), industrial alcohol and industrial salts.

* + How the food is stored or processed:

If not processed or stored properly, non-toxic materials of food might become toxic. A famous food processing accident in China was the Sanlu Milk Powder event, during which the milk powder was contaminated by a special chemical. Self-made food, such as pickles, soused fish and crabs, soused meat, might also be toxic if not dealt properly.

* + Possible harmful reactions from the menu:

There are possibilities that different kinds of food, when combined together, will generate toxic through chemical reactions. Such typical combinations might include but not limited to crabs and persimmons, asprin and wine.

However, general applications do not consider all the eight aspects. Until the time this proposal has been written, no application is actually been found that considers all the 8 aspects of users’ diets. Usually, the applications about “healthy live” can be categorized into 2 aspects:

1. Applications that only focus on calorie consumption: Those applications are able to provide functions to keep records of different kinds of user exercises, such as running, swimming and cycling, to record their calorie consumption based on their exercises. The problem is that, if the users does not control calorie in-taking, the efforts on calorie consumption will then be in vain.
2. Applications that focus on both diet and consumption: One step forward than applications only considers calorie in-taking, those applications are able to consider more. They are able to record the in-taking according to the menu that users input. However, the problem is that simply recording does not help also because not all the users have knowledge about nutrition and healthy live and tips and guidance for different kind of users are different. Until this proposal has been written, no application has been found out to give users specific ideas about their calorie in-taking, not even to mention about considering all 8 aspects of healthy diets.

### 1.5 Objective and outcomes

This project is tending to give users their own specific menu according to different characteristics of various users. The most essential problem of computer science, according to Zhong Qu in his book Introduction of computer science, is to study what kind of processes can be managed automatically and can be developed into information systems. Obviously, such a process of making decisions according to quantitative conditions (or qualitative rules that are measured using quantitative methods) can be set to rules of computer routines which forms together and formulate the information system as an integral. Thus, the major objective of this application is to automatically provide users specific guidance to make them be able to have an idea about healthy diet.

The importance of this application, namely, the output, may contain but not limited to following aspects:

1. It might save people’s life or mediate the possible harm towards users. Obviously by telling application user about the toxic food materials and their fatal amounts, the application will provide user an obvious warning that they are endangered, providing possibilities to safe user at the spot. Secondly, by telling users about possible harmful combinations of food, this application has a possibility of reducing the harm of mixing food that are not supposed to be mixed together in the same menu.
2. It provides specific guidance for specific requirements of different users. Traditional health-care applications tends to only record the user’s finished diets and only tell whether the calorie is overdue or not. Another typical function of traditional applications maybe only measuring the user’s exercise conditions and health conditions such as how long the distances the user has walked or what is the blood pressure of the users. Until this proposal is written, it has not been found out yet that any applications that can combine the current health condition and living habits of a user to his or her recommended diets. However, obviously, the diets for people suffering from heart diseases might have differences to people suffering diabetes. And their diets may be different from ordinary people who might have no essential health problems at this moment. This requires that the health-care applications have to be specific about a specific person.
3. It starts to gather data from its day one of running and will become more and more intelligent by applying some “learning” processes. At the beginning, the application just recommend users about their diets through theories and diets. However, there might be differences between what the users are recommended to take for their diets and the actual diets they are taking. This application also gathers what actually a user is taking every meal, analyze the differences and generate a more proper recommendation later if it found out the user’s diet is better for his case. Also, by opening interfaces for nutrition specialists, the recommendation plans will get better and better under the support of users classified as nutrition experts.
4. With more and more data gathered during the running duration of this application, the application will be able to seize a general picture about the dietary and living habits of different groups of people, providing them with rich researching gaps.

### 1.6 Methodology

**1.6.1 Related Work**

There are various kinds of applications that are related to the food and health care. Such as Live Strong, Huawei Health and Qing+. Those applications either just record the calorie consumption each day based on users’ activities (such as Huawei Health). Or they simply record what users’ take each day and corresponding calorie in-taking, such as Live Strong and Qing+. Especially for Qing+, they are able to provide menus for different groups of people with various needs.

However, until the time this proposal has been written, none of the application is able to provide specific diet recommendation for individual person according to the principles of all the requirements when arranging a menu, not even mentioning the track of the person’s habit and daily diets. Thus, those applications are still not specific enough. They tend to be not specific enough to help users to decide their diets based on their own specific conditions.

**1.6.2 Measurement Methodologies**

During the process of developing such an application, both quantitative and qualitative methods will be used. For example, nutritional balance is obviously a qualitative description. However, by measuring amounts and types of nutrition a user takes each day, the application tends to give this qualitative idea a quantitative description. By nutritional balance, the application will calculate the nutritional balance according to the proportion required for each type of nutrition materials, and gives out a score out of 50 in evaluating the general dietary health condition of the users. Another 50 scores consists of health profile information (20), BMI(10), activity frequencies and amounts (10) and calorie intake score (namely, whether or not a user’s calories meets his or her actual needs, 10). The general picture of the algorithm of measuring whether a person is dietary health is defined novel new in this project as following:

* Total Score: 100
  + Nutritional Balance, totally 50:
    - Water: 22
    - Energy source: 15
* Carbohydrates: 5
* Fat: 5
* Protein: 5
  + - Minerals and Vitamins: 13
  + Body Conditions, totally 50:
    - Health profile: 20
    - Calorie intake: 10
    - Activity frequencies: 10
    - BMI: 10

In each model, the score is calculated as:

Item score = FallMark \* (1 - | UserVal - Standard\_Val | / StandardVal) and the result is kept up to 2 digits. For example, suppose the standard value for BMI is 21.2, if the user’s BMI is 23.2, then the user’s item score is calculated as:

10 \* (1 - |23.2 - 21.2| / 21.2), and kept in 2 digits, which results in 9.06 scores.

The sum up of the scores indicates in 100 scores tends to provide a scientific measurement of the qualitative description of how health a person’s diet is.

**1.6.3 Object Identification**

The goal of this automatic health-care information system is to perform decision making for users requiring for specific goals according to certain theories of nutrition management and user specifications. Thus, the core design of this application is to provide a proper developing model, a proper object-oriented programming model and an enhanced database design which provides probabilities for adjusting. Below are several designs of this application:

* Roles of users:
* Customers: In this project, customers refer to those who just want to have a look at this application and got their food advice generally.
* Registered users: Those customers refer to the person who can fill in their personal health profiles and agree to sign into the application.
* Expert users: Those customers are groups of certified experts who has their experiences in the study of health, nutrition and food. They can also share the services provided to registered users as well.
* Application entities (objects):
* Food : An entity about all kinds of food, information includes Food-ID, food name, calorie, carbohydrates, fats, proteins, vitamins, minerals and water per 100 gram.
* Flavor: An entity that are related to the users’ flavor. This entity contains variables such as Flavor\_id and flavor\_name.
* Disease: An entity about different kind of normal diseases. The attributes include Disease - ID, Disease-Name and description.
* User: The attributes of registered users include userID, username, password, email, registered date, height and weight.
* Activity: Includes activity ID, sport items (such as run, sit, and sleep) and calories per unit.
* Health\_Score: A daily calculated score whose algorithm is following the one discussed in previous section. This entity includes UserID and score (of both individual items and total score).
* Processes: Contains information about usual food-processing mechanisms (e.g.boil, braise, fried, remaining\_raw and sauced), including process\_ID, process\_Name and process\_Description.
* Application relations (Relationship):
* Conflicts: This is a relationship in between different types of food that will generate toxic medical reactions when mixed together. The relation contains Food\_ID1 and Food\_ID2.
* Tastes: This is a relationship between food and flavor, it records the flavors of the food, including attributes Food-ID and Flavor\_id.
* Stimulates: This is a relationship between food and diseases, it indicates that what kind of food may stimulate the progress of what kind of diseases, so patients of such a disease should not be recommend to take such kind of food. This table includes Food-ID and Disease-ID.
* Flavor\_Simulates: This is a relationship between flavor and diseases, it indicates what kind of flavor may stimulate the progress of what kind of diseases, so patients of such a disease should not be recommend to take such kind of food that tastes such a flavor. The relation contains Disease-ID and Flavor-ID.
* Flavor\_Profile: This is a relationship recording the flavor of each user. The attributes include UserID and Flavor ID.
* Suffers: This is a relationship UserID-DiseaseID, indicating which user has which health problem.
* Performs: This is a profile between users and activities, which includes variables User\_ID, Activity\_ID, Activity\_Length and description.
* Recommended Diet: This is a relation which records the generated food for each meal for user, which includes variables Food ID, UserID and Time.
* Actual Diet: This is a relationship which records the actual menu that the user is using, including variables Food ID, UserID and Time.
* Howto: A relationship introduce how food will be processed, including food-id and process-id.
* Provides: Tells which process leads to which flavor, including process-id and flavor-id.
* Leads\_to: Tells which processes tend to lead to which types of diseases, including process-id and disease-id.

**1.6.4 Functions**

This application might be able to provide users with following functions that other similar kind of applications may not be able to provide (but may not only limited to those functions):

1. Keep recording about the users’ diets and exercising habits.
2. Evaluate users’ current health conditions based on their dietary habits and exercising habits according to the algorithm of this application.
3. Based on those evaluations and users’ current health conditions, recommend what to eat for them, considering menu needs of specific individuals.
4. Tells user about how their food will be processed and what are possible risks in those processes.
5. Automatically generate menu for specific users for about 2 weeks.

**1.6.5 Implementation**

This information system for automatic menu analysis and recommendation requires technologies of database (using MySQL), GUI (Graphic-User Interface) programming, Object-Oriented programming, searching and classifying algorithms and information analysis algorithms. Later on, it might also require machine learning, client-server communication, numeric analysis and simulation. However, those technologies are all available at the time when this proposal is being written.

Though using technologies that are available, this application does have several unique points from its original ideas to its implementation. The application will be implemented mainly using the Java programming language and Netbeans framework designing system.

**1.6.6 Testing and documentation**

As the application is implemented phase-by-phase using the spiral model, it will be tested model by model during the process of coding. Also, after the finish of each phase it will be tested to see whether the application meets the goal of each phase. And after phase 3, the application can be open for real-user testing. Testing results of phase 1 and phase 2 will be recorded by application developer and will serve as a reference for application instructors to check the ongoing of the progress.

The documentation of this application will consist of 2 main parts:

1. The final report which consists all the aspects of this application, including the proposal, requirement specifications, design specifications, testing document and the final report.
2. All the important sections of the project code will be documented using the conventional single line and multiple-line commends.

### 1.7 Project Schedule

This is a project which can be evolving endless because types of food and the complex relationships can be keep adding to the database. In this case, it is not proper to follow the general water-fall model from requirement specification down straight forward to the last stage of application running and maintaining. Instead, this project decides to follow a spiral model which evolves together with the development of this application.

In the first stage, the proposal and the design will be finished very quickly, with the finishing of building database and basic application classes in a quick manner. After the finish of stage 1, the users should be able to achieve information needed from terminal towards the database.

In phase 2, the database will be enriched and the graphic user interface will be built using Java with Netbeans. After the finishing of this phase, the application should be already able to be used for users of typical characteristics.

In phase 3, the whole application will be thoroughly tested and enriched before really going running, the potential bugs will be fixed and improved.

After 3 phases of develop, the application will be released to public for real testing, which enters the phase 4 of this application. And the goal of this project is to end up here. But the application can still be expanding based on users’ needs from more groups later on.

Phase 1 of the application will be finished before early March, around about 5th March.

Phase 2 of the application will be finished before early April, around about 2nd April.

Phase 3 will be available around the middle of April, nearly 20th April.

### 1.8 Resource Estimation

The developing processes requires use of software and hardware which follows following criteria.

6.1 Software:

Operating System: Windows 7

Development IDE: Visual Paradigm for UML, Java 8 and Netbeans 8.2

Database: MySQL5.7

6.2 Hardware:

Core i7-4700 HQ CPU, 2.40 GHz

RAM: 8.00 G

# Chapter 2 Software Requirement Specification for Project: Edible Me

## 2.1 Introduction

This project is a decision-making system which helps users to build up their diets based on their health profiles. Using the spiral model, 3 versions of applications will be developed and the final outcome of this project will be the version 3.0 of “Edible Me”. This document refers to the standard IEEE Software requirements specification template and the convention of assignment-submission works of Wenzhou-Kean University. The font is *Times New Roman*, of size 10.5.

This document of Software Specification is written for any potential users of this software, namely, nutritionists, athletes, physicians and any other application users who are interested in the topic food and healthy living. Particularly, this application will follow the user requirements specified by professors from biology department of Wenzhou-Kean University, who will act as the user group during the designing and developing phase of this application.

Apart from general users who are interested in self-specified decision making of dietary planning, this application also considers the user requirements for nutrition experts and cooperation vendors of food and drinks. It is a platform for cooperation vendors to sell their food and experts to give their opinions to general users and earn benefits by offering those advice to general users.

By finishing the requirement specification, this project is supposed to finish following tasks:

1. Gather and document users’ requirements, especially the explicit and implicit requirements that they are not able to be satisfied with from general applications of similar themes.
2. Finish the design work of phase 1 of the whole develop procedures and hopefully 20% of the work of phase 1 can be finished.
3. Half of the design work should be finished, the whole document of designing specification shall be submitted for approving one week after the official approving of this document.

The purpose of this application is to support users to generate their own plans of diets using the novel new algorithm specified in the approved proposal, with its core contribution of providing user-specific plans. Users shall be able to improve living qualities and improving health conditions from this application by following the plans made according to their health profiles that are made specifically for them.

Using this application, vendors from specific food or drink industries could have chances to advertise their products, serving as a potential business model. The payment amount of advertising products shall consider the original value of the product and the amount of users paying attention to the advertisement.

The whole document and the later documents and procedures shall follow the plans and schedules specified previously in the approved proposal.

## 2.2 Description

### 2.2.1 Perspective

This application tends to be an improving version of current applications under the theme “health living”. As stated in the project proposal, current applications of this theme tends to record users’ diets or exercising behaviors only. Based on traditional usages of those applications, this application not only records data, but analyze data in order to fulfill the requirements of individual users according to nutrition theories and certain novel new algorithms specified by this project in the proposal part.

### 2.2.2 Functions

The functions are specified in section 4.4 of proposal already, apart from those, the application needs to add a function which is being able to let food vendors advertise their food, considering the requirements of the business model. The overall functions might be slightly modified in the final, but the modifications shall not affect the overall functionality of the original designed functions provided by this application.

### 2.2.3 User classes and characteristics

Users of this application are general users who have potential interests about healthy diet. However, considering the scope and the specialty of this applications, users are grouped into three roles as specified in the previous proposal, namely, customers, registered users and expert users. A special class till this moment this application has to add is the cooperation users and this is one of the most important roles in this application.

The characters of customers (visitors) are those who just want to have a general look about the application and who are not interested in specific dietary plans, thus those users are not that important compared to those who are able to register and are interested in user-specific dietary plans, expert users and cooperation users.

The registered users are those who are interested in specific healthy diet planning and are one of the most important roles of this application. The vast number of users serve as the potential buyers from cooperation vendors and customers of expert users.

The experts are users who have fully accesses towards the information about food and planning algorithms of users. They might benefit from providing more specific plans for individual users and are awarded if their plans are better than the automatic generated plans by the algorithm from this project.

The cooperation vendors should be able to advertise their products and shall be charged if their visitors reach a certain amount. They should also be satisfied because the vendors are the key of the business model of this application.

### 2.2.4 Initial Platform

The application is designed to be a desktop one on Windows system at its first phase and it shall be able to run on Windows 7 or later versions. To run this application, the internet connection is required.

### 2.2.5 Constraints

The development of this application has following constrains:

1. Time limitation: The application shall be reached to its first phase near around 5th March, its 2nd phase around 2nd April, so time limitation might be a constraint for this application.
2. In order to satisfy a real server-client model, a real server with abilities of holding huge amount of data is required, but it becomes a constraint since no server is available on this campus.
3. Due to private protection policies, the access towards users’ health profile and user registration information should be managed.
4. This application does not consider parallel access towards database, thus two users are modifying the same table at the same time, a problem might occur. The whole application is based on single thread at first two developing stages.

### 2.2.6 Assumptions and Dependencies

This application shall include some paying systems, thus may using API systems such as Ali API to get the information of whether users paid or not.

## 2.3 Interface Requirements

### 2.3.1 User Interfaces

This application mainly communicates with users through graphic user interfaces (known as “GUIs”). In order to satisfy the proposal and user requirements, following GUIs (but not limited to) shall be provided:

1. A GUI for user access control (log in panel). Based on user identities, this panel shall be able to guide users to their corresponding interface groups.
2. A “help” GUI (also introductory GUI). It provides introduction about this application and at the meanwhile serves as the interface for customer users.
3. A GUI which reflects registered users’ personal information.
4. A GUI which gathers and reflects registered users’ health information.
5. A GUI which gathers and reflects registered users’ diet information.
6. A GUI which provides registered users with their health evaluation and future suggestions (suggested food by certain experts and cooperation).
7. A GUI which provides connections between users and experts.
8. A GUI for expert users to check users that are connecting him or her.
9. A GUI for food vendors to post their food and pay their advertising fees.

### 2.3.2 Software Interfaces

In order for this application to run, following software interfaces shall be required:

1. MySQL 5.7
2. Windows 7 or above versions

### 2.3.3 Hardware Interfaces

This application does not have specific constrains about hardware interfaces. Any computing devices that runs operating system Windows 7 or above shall be fine.

### 2.3.4 Communication Interfaces

This applications require communication between the application and the database system, thus the connection between the application and the database is needed. During the implementation and testing phase, the local server (IP: 127.0.0.1 is used).

If time allowed, the application shall be made into real server-client mode, thus network sockets shall serve as communication interface in this case.

## 2.4 Use Cases

### 2.4.1 Actor list

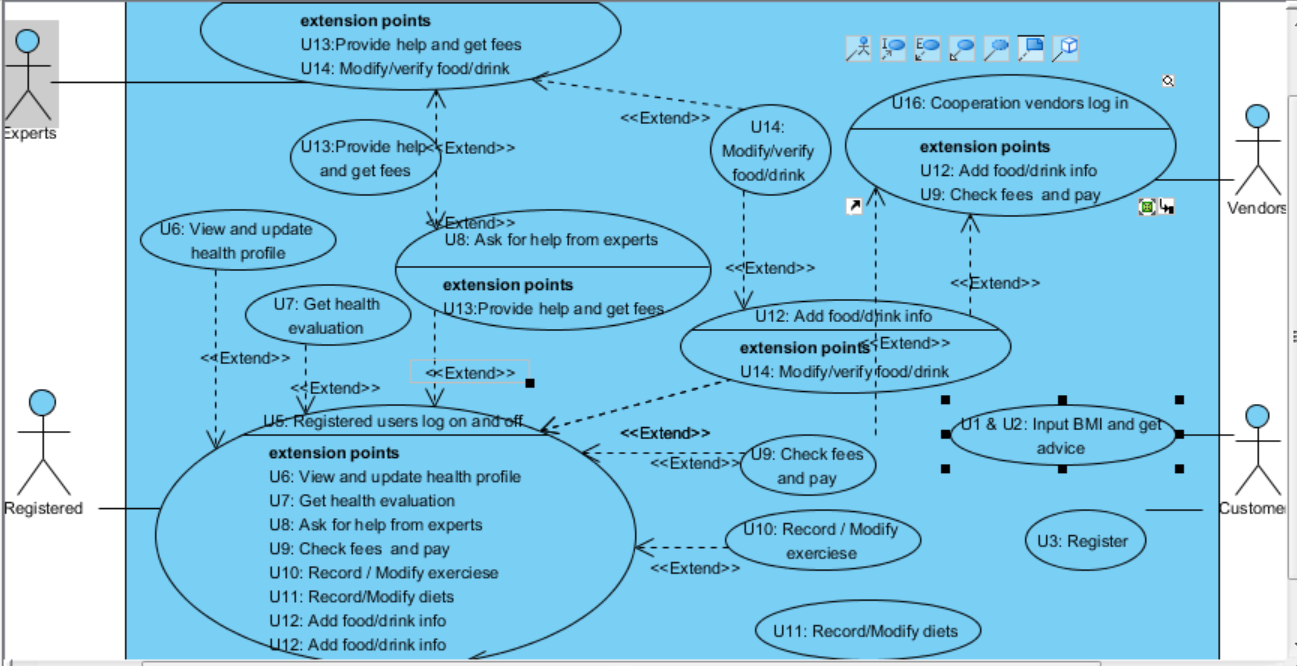
* + Customers: Any person who is interested in getting general ideas about healthy diet.
  + Registered users: Any person who is interested in getting specific plans about healthy diet based on the person’s individual conditions.
  + Experts: Qualified nutritionists who are able to give registered users specific dietary advice based on their health conditions.
  + Vendors: Those vendors of food and drink who would like to advertise their products.

## 2.4.2 Use case list

1. Customers can input BMI.
2. Customers can get advice according to his or her BMI.
3. Customers can register if they are interested in this application.
4. Registered users should be able to input and modify his or her health conditions and those information will be stored in “health profile”.
5. Registered users should be able to log in and log off.
6. Registered users can watch his or her health profile.
7. Registered users can get results about his or her health evaluation based on information he or she input to the system for free.
8. Registered users can ask help from expert users by paying consulting fees.
9. Registered users can check up the remaining of his consulting fees and add more if he or she wants to.
10. Registered users can record his or her exercises being performed.
11. Registered users can record or modify his or her diets.
12. Registered users can add information about food / drink, and if the information got qualified, he / she will get credits.
13. Expert users can modify the automatic generated diet plans for users and get payments.
14. Expert users can modify or verify the information about food / drink.
15. Expert users can log in and log off.
16. Cooperation vendors can log in and log off.
17. Cooperation vendors can add food/drink information they would like to sell.
18. Cooperation vendors can choose whether or not to advertise their food / drink.
19. Cooperation vendors can check his or her remaining fees.

### 2.4.3 Use case diagram

Below shows the use case diagram of this project.



### 2.4.4 Use case details

* U1 and U2: Customers can input BMI and can get advice according to his or her BMI:

Customer users can input his weight and height, the system calculates the user’s BMI and return the general information about nutrition in-taking.

* U3: Customers can register if they are interested in this application.

Customers can register if they are interested in this project and thus becomes registered user.

* U4: Registered users should be able to input and modify his or her health conditions and those information will be stored in “health profile”:

Registered users should be able to access to a frame called health profile and there they are able to modify their information about their health conditions.

* + U7: Registered users can get results about his or her health evaluation based on information he or she input to the system for free:

The system will automatically generate menu based on the input of the users for free. However, if the users are not satisfied with the results, they might contact experts to get better advice, in this case, the users have to pay for the experts.

* U10: Registered users can record his or her exercises being performed:

There should be a frame for users to input his or her exercise items and duration, the system should be able to calculate the calorie consumption and return the value to the users.

* + U11: Registered users can record or modify his or her diets:

There should be a page for user to record and modify his menu item by item, including: item name and item amount, the system should calculate the calorie and check the food safety issues of input item (whether they are “conflict” as specified in proposal).

* + U12: Registered users can add information about food / drink, and if the information got qualified, he / she will get credits:

If there exist food or drink that the system is not included, the users are allowed to input and will get credits (can be used to exchange money for getting help from experts). If there information about such food is verified by experts, they can get more credits.

* + U13: Expert users can modify the automatic generated diet plans for users and get payments:

If users ask for help, the experts can check whether the diet generated by system is suitable for him or her. If the result is suitable, the system does rollback the money paid by the users. However, if experts recommend a better diet, the system gives 90% of the money to experts and remains 10% for itself.

* + U18: Cooperation vendors can choose whether or not to advertise their food / drink:

If cooperation vendors choose to advertise their food/drink, the brand of the food / drink will be attached with the vendors so that their food/drink will be advertised and users are able to click on those to quickly check information. Otherwise, the system does not advertise their food/drink and their food/drink will simply be stored in the database. The system will charge the vendors based on the amount of users clicking the link of their advertisements.

### 2.5 Nonfunctional Requirements

Apart from functional requirements that are specified in this document and the previous proposal, there are some non-functional requirements of this application, including but not limited to:

1. This application shall be able to provide access management for different users, for users’ health profile information shall be privacy-related information and should be protected. Even for experts, they shall not allowed to see the real name of users, but fake name shall be available in this case.
2. The system shall be able to modify its models when an expert gives suggestions.
3. Registered users shall be able to input their diets in various ways, from “quantative” to “qualitative” for many of the users might not able to provide exact accurate amount of the diets they are taking.
4. Other Requirements

In the final phase, the multiple-language support version for this application shall be provided. During the implementation and testing phase 1 - 2 however, the application is made for English/Chinese users only.

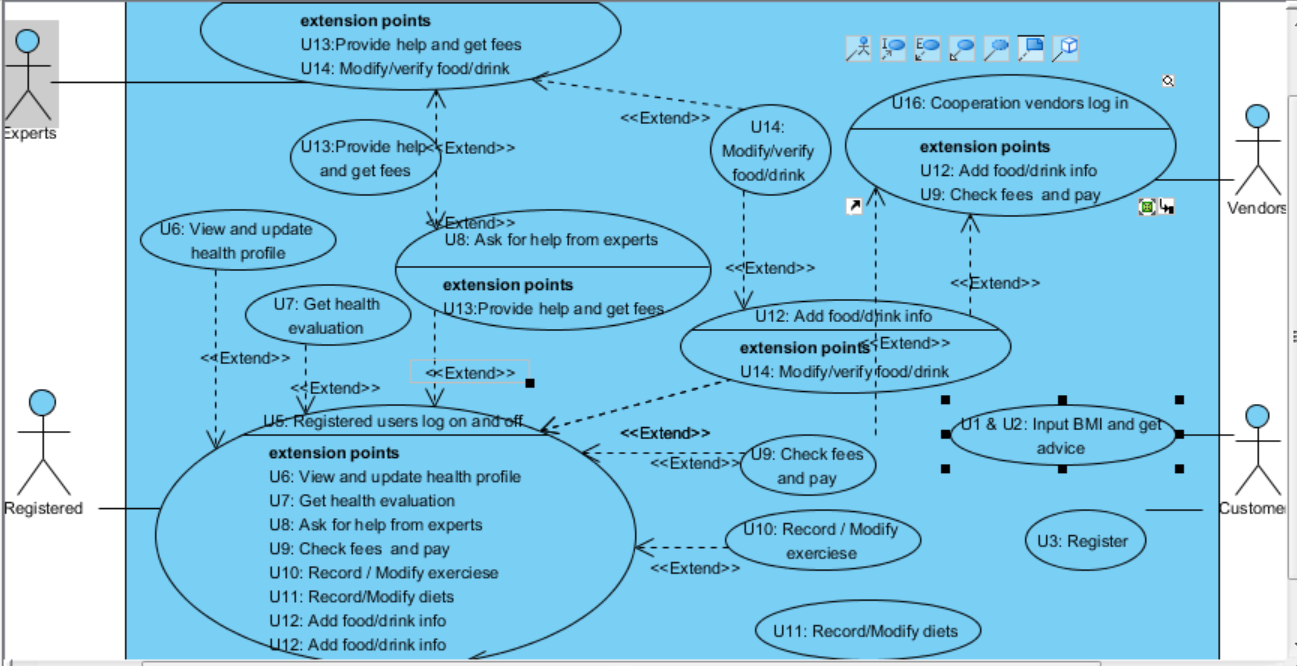
## Appendix A: Glossary

* Menu: If not specified, menu refers to the diets that users take for meals.
* Flavor: How the dishes taste, for example, the flavor of vegetables shall be salty, the flavor of yogurt shall be sour.
* Activity: The exercises that users of the application are performing.
* Process: How the food is made. Examples are fried, sauced and so on.

# Chapter 3 Design Specification of “Edible Me”

This design specification document is used for the further implementation of previous approved project “Edible Me” as specified by the project proposal and software requirement specification. All the design of this project is based on the proposal and the software requirement specification approved before. The process of design considers the detailed database design, package design, database interface design and security policies till this moment.

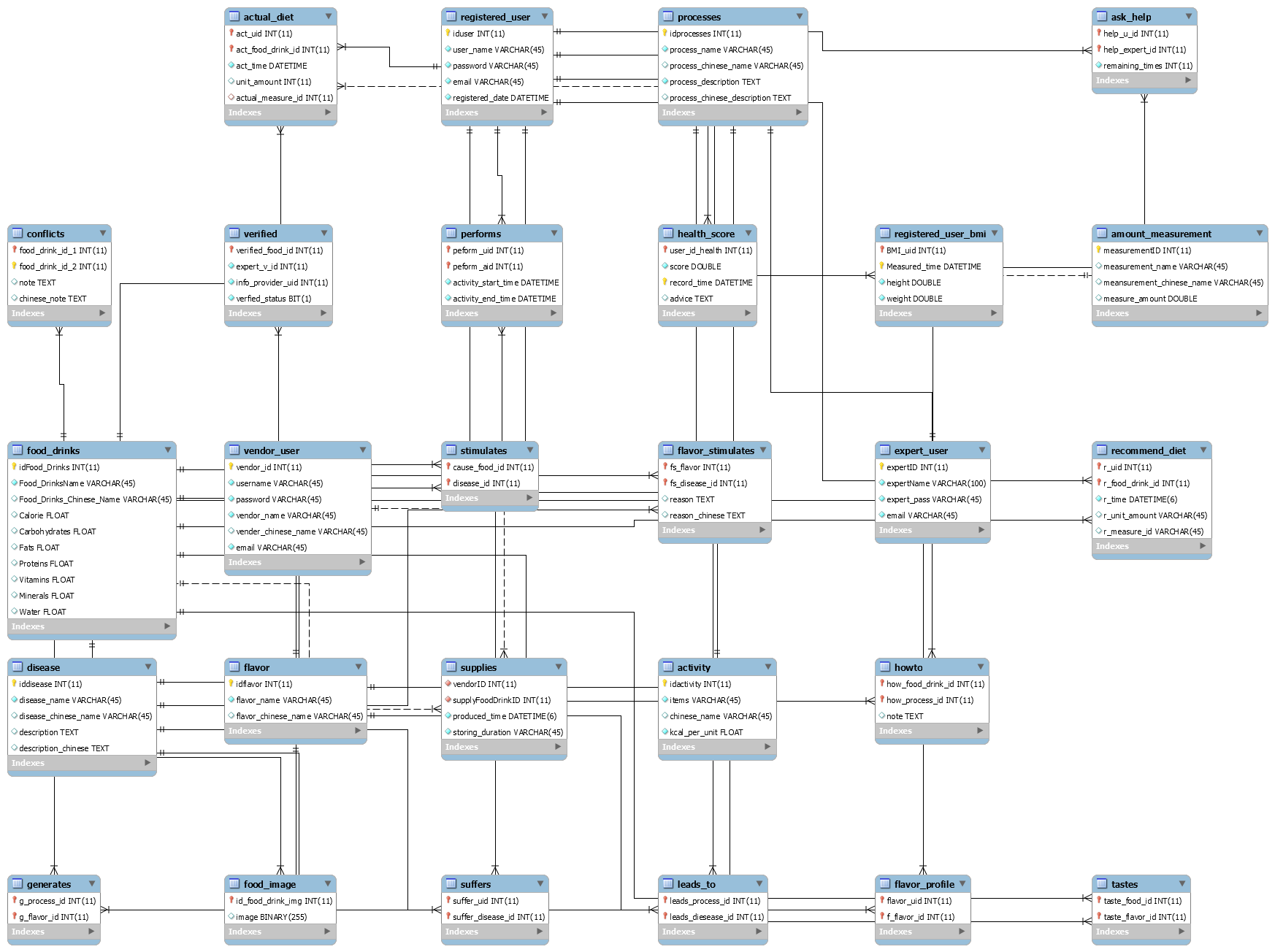
In general, the whole design processes follows the principle of “bottom up”. It starts from database design and ends up to class design. As user interface design works are done previously in the software specification requirement, this stage does not consider about user interface for a second time. This design takes into consideration of grouping different classes for different usages into different packages. It separates the linkage between database API which links to the database and project objects (classes), considering the possibility that objects might be used for purposes other than linking to database. Furthermore, it uses SHA1 as encryption policy for users’ one-way encryption data such as users’ password. The whole design refers to the previous specified user cases as shown below:



Use case diagram of project “Edible Me”

## 3.1 Database Design

The database design is the basic core of this project. The goal of the implemented database is that it should support full data of the daily running of this application. During the design of this database, E-R model is used and the convention of 4NF is followed. In order to be more specific and more easier to check, maintain and extend potential transaction requirements at later stages, the entities and relations are further categorized into three categories, namely, entities without constraints, entities with constraints (functional entities) and relationships.



E-R Diagram of database design of this project

Tables that are entities without foreign keys:

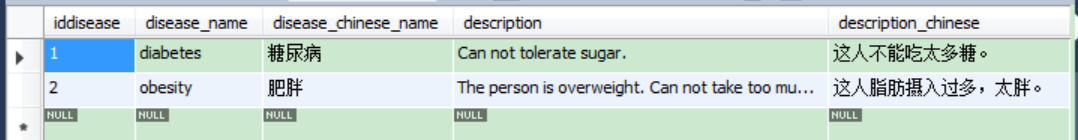
* Activity: Entity which records the user’s exercise information, satisfying user case 10. During the implementation of phase 1, actual data is already put into the activity table. This table allow **users to add information only**, but experts can either **add, modify or delete information.** Data organization example:



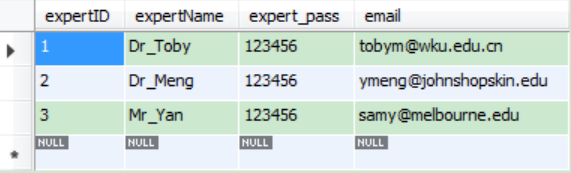
* Amount measurement: Entity which contains information about the usual measurement of amounts. During the implementation of phase 1, tested data will be used. Data organization example:



* Disease: Entity which records different types of diseases. This entity is related to possible further usage of many other user cases. During the implementation of phase 1, fake data was entered into this table. This table allow **users to add information only**, but experts can either **add, modify or delete information.** Data organization example:



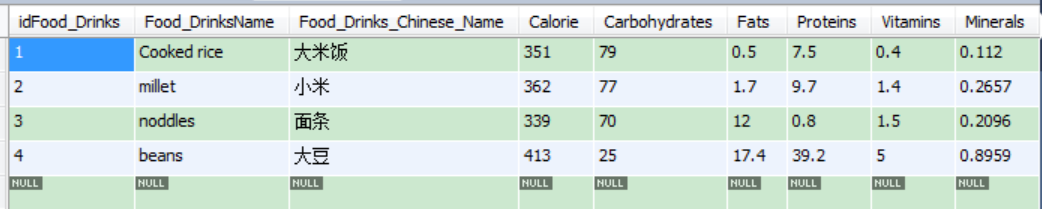
* Expert users: Entity which records basic information about experts. During the implementation of phase 1, test data was used in this table. Data organization example:



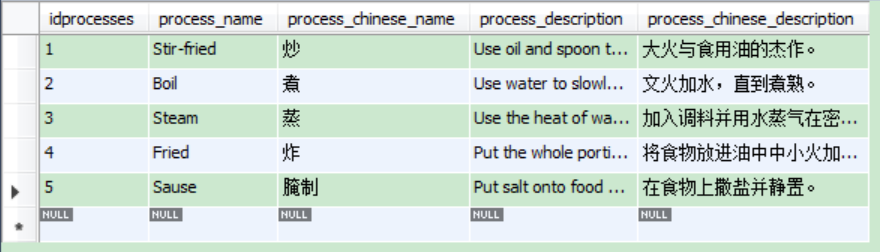
* Flavor: Entity which contains flavors of different dishes, such as bitter, sour or sweet. During the implementation of phase 1, real data will be used. **This table allows experts to access to add, modify or delete only.** Data organization example:



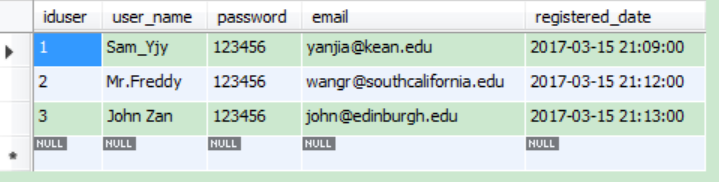
* Food & Drinks: Entity which contains information about food and drink this application refers to. In the implementation phase 1, this table contains partially finished actual data. This table allows users to add data and experts to access to add, modify and delete data. Data organization example:



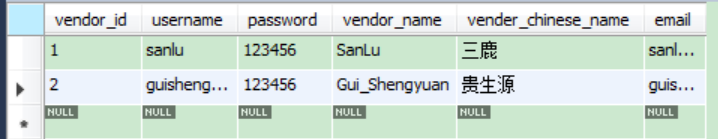
* Processes: Entity which records processes of how food and drinks got processed. This table usually does not have to be modified and only expert users are allowed to reach the table for adding, modifying or deleting data. During the implementation of phase 1, fake data will be used. Data organization example:



* Registered users: Entity which contains information about registered users. This table allows registered users to change their username, password and email address. During the implementation of phase 1, test data will be used. Data organization example:

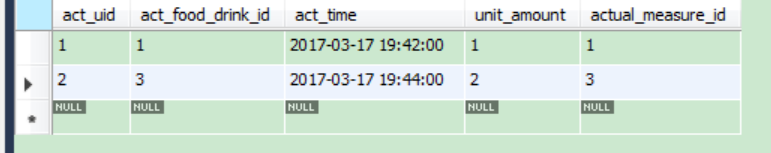


* Vendor users: Entity which contains information about vendors. This table allows cooperation vendors to change their username, password, name of cooperation and email. During the implementation of phase 1, test data will be used. Data organization example:

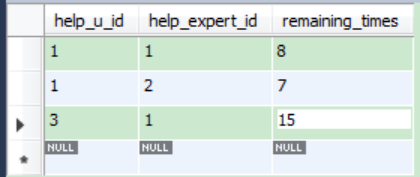


Tables that are entities with foreign keys: (Functional tables)

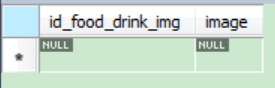
* **Actual user diet:** This entity contains information about who actually ate what at when of what amount, which serves as a very important table managing the information of user health. During the implementation of phase 1, test data will be used for this table. Data organization example:



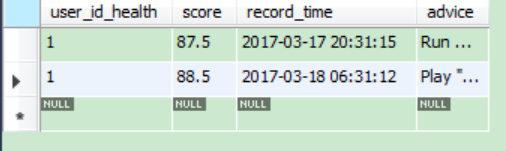
* Ask help: This entity allows registered users to maintain a relationship with certain expert(s) by telling how many times remained for a certain user to ask for help from certain experts. During the implementation of phase 1, test data will be used for this table. Data organization example:



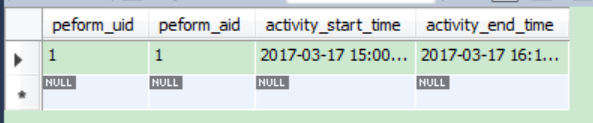
* Food image: This entity contains image of each kind of food. If image(BLOB data structure) are not able to be stored in database, they will be stored as I/O files instead. This table is temporarily not considered during the implementation of phase 1. Data organization example:



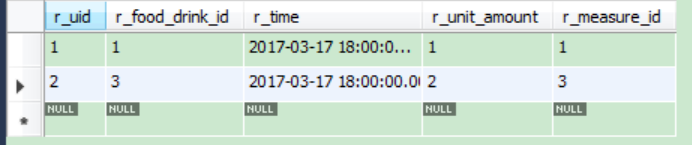
* **Health score:** Users can ask the system to automatically evaluate their health based on all the aspects of data they input to the system. The system can record their health conditions and provide advice for them. This table uses tested data during the implementation of phase 1. During the implementation of phase 2 (and later), real scores shall be evaluated based on the algorithms specified. Data organization example:



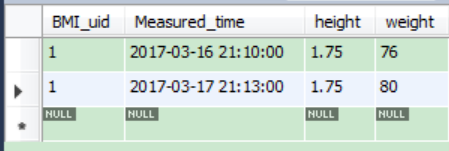
* **Performs:** This table records information about the exercising information of each user. Testing data will be used during the implementation of phase 1. Data organization example:



* Recommend diet: This table records information about recommended diet each meal (denote breakfast as 07:00, lunch as 11:30 and dinner as 18:00) for users. Fake data will be used during the implementation of phase 1. Data organization example:



* Registered user BMI: This table records information about users’ BMI. Testing data will be used during the implementation of phase 1. Data organization example:

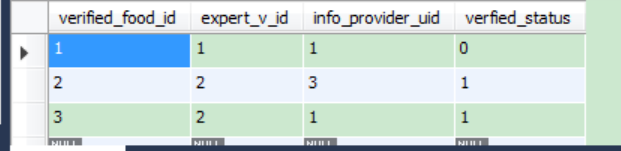


* Verifies: This table records information between food and experts, which means food information is verified by (or waiting to be verified) by experts and the quality of food is ensured. Specifically, this table uses bit 0 to indicate the food is not verified, using 1 to suggest the food is qualified. Thus, the accessing of data may look like following:

INSERT INTO what\_to\_eat.verified

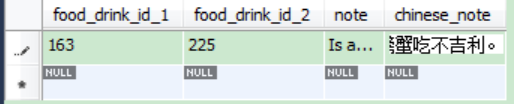
VALUES ('3', '2', '1', b'1');

During the implementation of phase 1, testing data will be used. Data organization example:

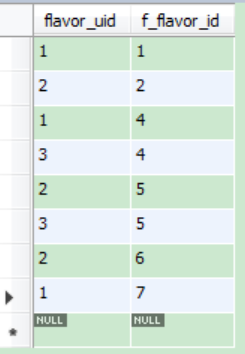


Tables that are relations:

* Conflicts: This is a one-to-one relationship between different types of food, showing the combination of which two food may cause potential problems. The note columns indicates how the conflict may cause problems. Fake data are used during the implementation of phase 1. Only experts are allowed to achieve this table. Data organization example:



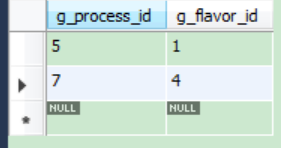
* Flavor\_profile: Record which user prefers which kind(s) of flavors. This table is available for each users to add / modify their own information about their flavor preferences and for experts to view when users are needing help from experts. Data organization example:



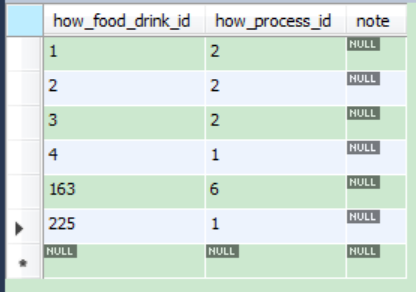
* Flavor\_Simulates: This is one-to-many relationship indicating the possible incurs of certain diseases due to taking too much specific kind of food / drink of similar flavors. For example, eating too much rice may exacerbates the situation of diabetes. Testing data are used during the implementation of phase 1. This table is designed only for experts to add / modify or delete information. Data organization example:



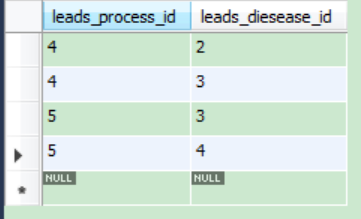
* Generates: This table contains the relationship between processes and flavors that this certain process tends to lead to. For example, the process honeydew tends to lead to sweet flavors while the process sauce tends to lead to salty flavors. Testing data are used during the implementation of phase 1. Data organization example:



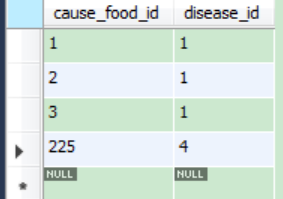
* How to: This is a one-to-one relationship showing how food is usually processed. During the implementation of phase 1, tested data will be used. This table allows all kinds of users to access / modify and delete. Data organization example:



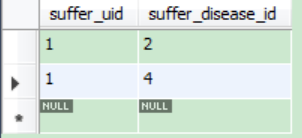
* Leads to: This table specifies which processes may lead to which kind of diseases, can be add or modified by expert users. Fake data will be used during the implementation of phase 1. Data organization example:



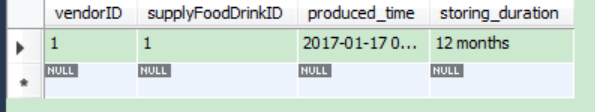
* Simulates: This table builds a many-to-many relationship between food and cause of diseases, which means which food might cause which diseases. Fake data will be used during the implementation of phase 1. This table allows expert users and registered users to add and modify. Data organization example:



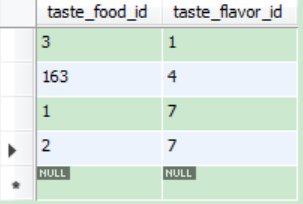
* Suffers: This table forms a many-to-many relationship between users and diseases. Testing data will be used during the implementation of phase 1. Data organization example:



* Supplies: This table forms a one-to-many relationship between vendors and food / drink. It specifies which food / drink is sold by which vendors. Data organization example:



* Tastes: This table provides information about the tastes (flavor) of different food/drinks. This table allows both experts and registered users to add / modify information. Testing data will be used during the implementation of phase 1. Data organization example:

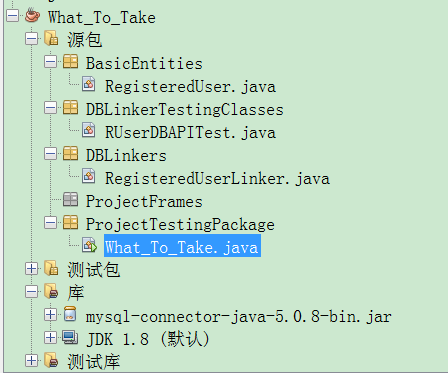


Note: Table names that are in bold are considered to be the essential supports of this application and thus should be implemented first and as soon as possible.

Tables that are entities without foreign keys are most important tables to consider. Tables that are relations are second important to consider. Tables that are entities with foreign keys are mainly work as auxiliaries to this application.

## 3.2 Package Design

This application consists of various components, thus the management and design of packages[[2]](#footnote-1) for different usages becomes particularly important. The whole project package structure is shown as below:



Pic: Package management structure of this application

The whole package structure is divided into two parts: application packages and testing packages. The packages for application packages are:

* Basic Entities: This package used to store user roles and application entities, such as the RegisteredUser class, the FoodDrink class and the Activity class.
* DBLinkers: This package is used to provide database linking application interface (mainly for entities). The usage of package will be further demonstrated in section III of this document.
* ProjectFrames: Which provides user interfaces for this application to interact with project users of all kinds of roles.

The packages for testing are:

* + DBLinkerTestingClasses: Classes in this package are used to test the linking between database and basic entities.
  + ProjectTestingClasses: Classes which are written to test certain objects or frameworks.

## 3.3 Database-Application Interface Design

For each entity and relation in this project, there should be a database application interface designed. The interface subroutine provides only basic add, modify and update functions for this certain entity or relation. For example, the database application interface of entity registered user is shown as below (Due to the scale of the document, the body part of constructors or methods are omitted):

package DBLinkers;

//import statements, omitted.

/\*\*

\* @author Sam Yan

\*/

public class RegisteredUserLinker {

//Variables for linking database:

private Connection connect;

private PreparedStatement preparedStatement;

private ResultSet resultSet;

//Variables for registered user information.

private RegisteredUser ruser;

//Initialize linking environment

public RegisteredUserLinker() {}//end constructor

/\*

\* Set and Get: Accessors for entity/relationship able to acess this interface.

\*/

public void setRUser(RegisteredUser ruser){}//end method

public RegisteredUser getRUser() {}//end method

//Read information according to information already known (such as primary key):

private void readFromDB() {}//end method

/\*Method of registering user to database. Including reading his/her id

and write his/her information into database.

In this case, it is a special method being written in the interface\*/

public void registerUser() {}//end method

//Typical writing to database function

private void addToDB() {}//end method

//Typical updating function.

public void updateDB(boolean isPasswordChanged) { }//end method

}//end class

Code Example of showing a typical database application interface for this project

There are 28 entities and relationships in total, which means that 27 such database linking interfaces shall be built. They are packed in the project package DBLinkers.

Each application interface shall then be tested by subroutines in the test package, one testing example of the database application interface is designed as following:

package TestingClasses;

import BasicEntities.RegisteredUser;

import DBLinkers.RegisteredUserLinker;

/\*\*

\*

\* @author Sam

\*/

public class RUserDBAPITest {

private RegisteredUserLinker userLinker;

public RUserDBAPITest() {

userLinker = new RegisteredUserLinker();

testRead();

testWrite();

testUpdate();

}//end cons

public void testRead() {

// Testing reading from registered user db link:

RegisteredUser oldUser = new RegisteredUser();

oldUser.setRe\_userid(1);

userLinker.setRUser(oldUser);

oldUser = userLinker.getRUser();

System.out.println("Old user: " + oldUser.getRe\_username());

System.out.println("Password: " + oldUser.getRe\_password());

}//end method test read

public void testWrite() {

//Testing register user to db link:

RegisteredUser newUser = new RegisteredUser();

newUser.setRe\_username("Sam J");

newUser.setRe\_password("123456");

newUser.setRe\_email("samy@163.com");

newUser.setRe\_registeredTime("2017-03-18 00:00:00");

userLinker = new RegisteredUserLinker();

userLinker.setRUser(newUser);

userLinker.registerUser();

}//end method test write

public void testUpdate() {

// Testing registered user to db link:

RegisteredUser updatingUser = new RegisteredUser();

updatingUser.setRe\_userid(1);

//The second bool is for whether user is registering.

userLinker = new RegisteredUserLinker();

//Set the user object first in order to get information later:

userLinker.setRUser(updatingUser);

//Read the user according to id:

updatingUser = userLinker.getRUser();

System.out.println("Username: " + updatingUser.getRe\_username());

//Change the user's information

updatingUser.setRe\_password("123457");

updatingUser.setRe\_email("223560119@qq.com");

//Set user

userLinker.setRUser(updatingUser);

System.out.println(updatingUser.getRe\_username());

//password changed is true.

userLinker.updateDB(true);

}//end method test update

}//end class

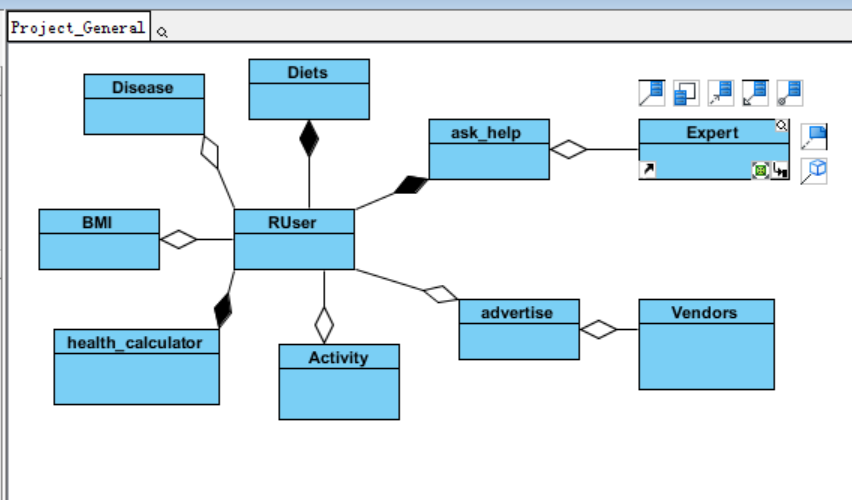
The testing result above indicates that all inserting, updating and reading operations for

registered users’ table works pretty well. Thus, this database application interface is considered as designed and implemented properly.

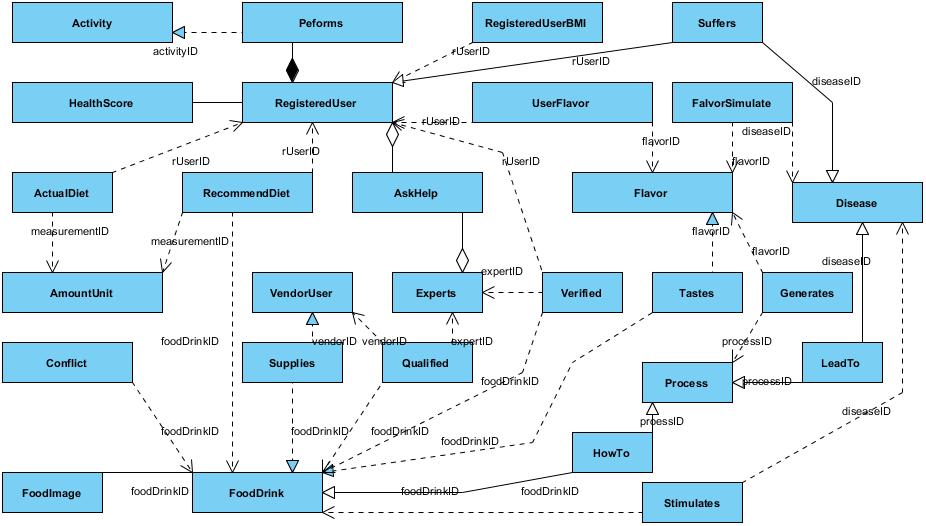
There shall be 28 such testing examples, one entity / relation each. They are all grouped together in the project’s package DBLinkerTestingClass.

## 3.4 Class Diagram Design

Except from database application interfaces and entity classes which works together to support the basis of this project, some interactive classes shall be considered in order to satisfy the use cases and functions specified in previous documents. Those classes are depicted as a further project general class diagram as shown below.



Pic: Very Brief Interactive classes Diagram of the project



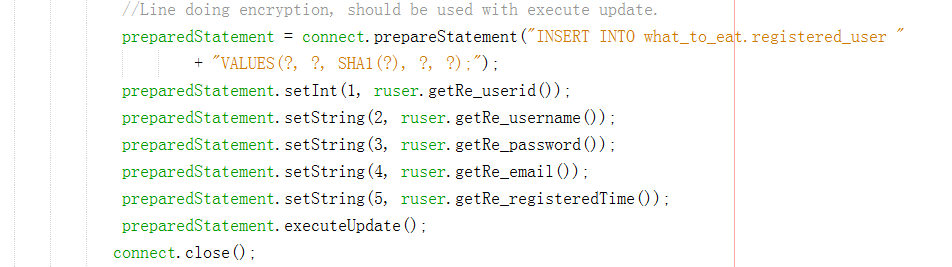
Pic: Detailed class diagram of this project

This diagram is a role-based class diagram considering the interactive functions between different roles of the application user. The class AskHelp forms a many-to-many relationship between registered user and expert users. Once a user is asking help from experts about his or her nutritional plans, the expert than can access users’ different data as shown above.

Also, through Advertise class, which forms a one-to-many relationship between user and vendors (a user can view many advertisements), the cooperation vendors are able to advertise their products.

## 3.5 Security Policies

According to official police report cases, users tend to use their password for critical services (such as bank services) for general services as well. Thus, this project provides one-way security guard using SHA1 algorithm for user passwords. In later stages, user-sensitive data such as users’ health profile shall also satisfy the security policies. However, due to the health profile is a bidirectional information, thus AES algorithm shall be used to encrypt and decrypt such information. For example, while registering a new user, this application uses following subroutine to write encrypted password into database:



Also, similar cases applies when users try to update their password.

# Chapter 4 “Edible Me” Testing Report

To test this implemented project, the structure of the program is studied according to the design specification. The program structure includes three layers, the first is the database design which was implemented using MySQL 5.7. The second layer is the database linker which is used to link between the database and the subroutines. This layer is used to process the the interactions between programs and user data, including reaching data from database and storing data back into database. Furthermore, in order to consider the possible extension of the user requirements in the future, the data linking layer could serve as server side programs if later on the Server-Client model needs to be implemented. The third layer is the project framework which directly interacts with application users, dealing with their input and outputs and provides certain functions to satisfy software specifications.

The testing strategies are formed according to those 3 levels of the whole project. The database design and implementation is tested according to the rules of 4NF database design requirements, the program linker phase is each tested by using sample testing classes while the project framework is tested by running this application in the virtual machine environment to estimate whether the instructions are clear for users and whether the graphical user interfaces follow the rules of user friendly and hinting properly.

## 4.1 Project Structure

The database is designed and implemented according to the requirements of 4NF. The 4NF database relationship specifies that there should be no multiple value dependencies between non-trivial and non-functional relationships. It is based on the prerequisites of BCNF, 3NF, 2NF and 1NF. Among those normal forms, 1NF specifies the requirement that their should be primary key(s) in the database which is designed. 2NF specifies that each column of the database should form one-to-one relationship to its corresponding primary key(s). 3NF specifies that their should not exists inductive relationships between columns and BCNF is the improving normal form of 3NF which specifies that each of the primary key(s) should decide(s) certain one-to-one factors. The database design following the rule of 4NF and the implementation using MySQL 5.7 as specified in the software requirement together works as the first layer of the application.

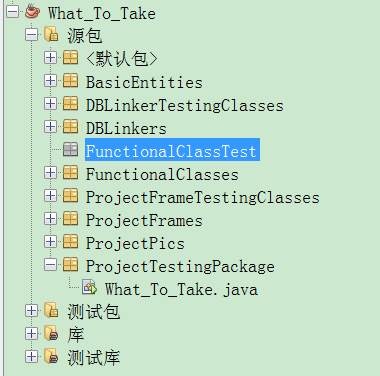


Fig: Package management of “Edible Me”

In the implementation of this application, the package management strategy is used to divide classes or contents of different subroutines into different layers or function specifications. For example, package BasicEntities is implemented according to the project class diagram design specified in design specification. Those entities are implemented using Visual Paradigm for UML version 10.0, by directly generating and importing the corresponding project codes into the project path after the finishing of project designing phase.

DBLinkers forms “links” from program subroutines to databases, thus, the separation from programs to data is satisfied because user programs are not allowed to manipulate data in the database directly in this case, instead, those subroutines are supposed to first achieve data in the database and then processes data based on their interactions with users.

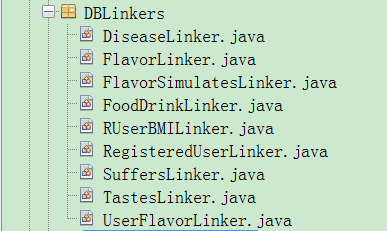


Fig: Subroutines linking to database

For example, DiseaseLinkers in the package DBLinkers works as the link between the information of various kinds of diseases in database and it provides subroutine functions as shown below:

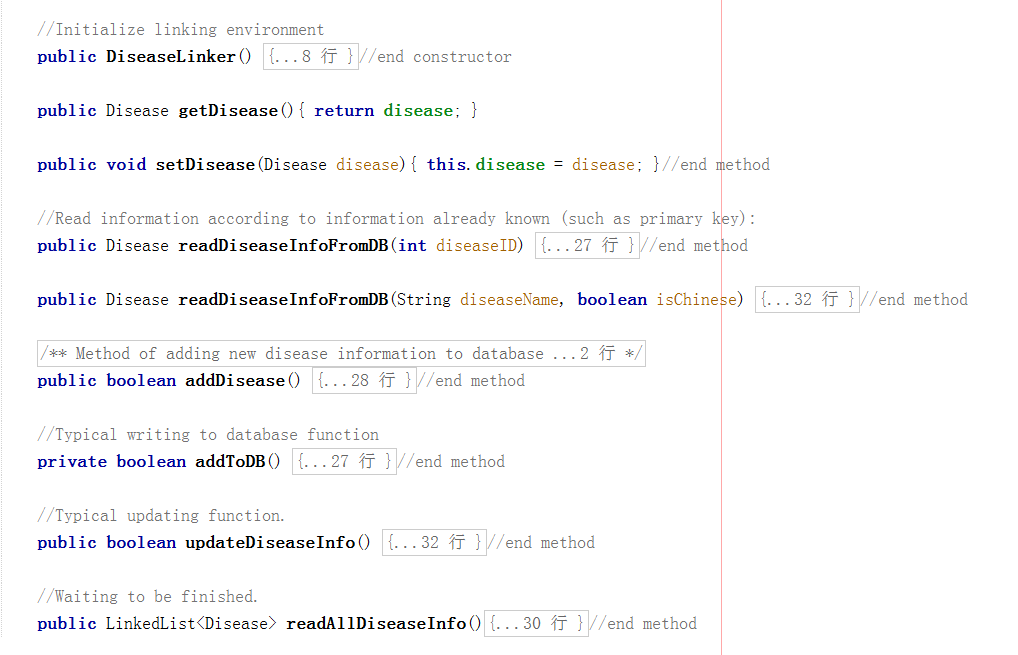


Fig: Subroutines of DiseaseLinker class

Thus, for other programs or routines that needs to reach information about different diseases users might suffers from, they could possibly call the function of this program. For example, to inquiry about the full information about certain type of disease, either the name of the disease (can be both in English or Chinese, taking into the consideration of possibly extending a Chinese version of this application later on) or the ID of the disease (possibly achieved from other applications) shall serve as a possible input for this linking class. The output in the case is the full information of this type of disease to the caller program. Also, caller programs might possible require to record information about new diseases into database, thus they could use this link class by first set the full information of certain disease that they could provide, and then after setting the information about the disease by calling either addToDB() or updateDiseaseInfo() function to achieve their goals.

Other subroutines of this package works similarly as DiseaseLinker class. For example, FlavorLinker reads and writes information about different flavors, FlavorSimulatesLinker keeps information about which types of flavors tend to simulate what kind of diseases and FoodDrinkLinker keeps information about various food and drinks.

The third level of this project is the user interface level, which also includes some functional classes which combines different functions in order to satisfy different purposes specified in software specification. This level of subroutines are included in both ProjectFrames and FunctionalClasses.

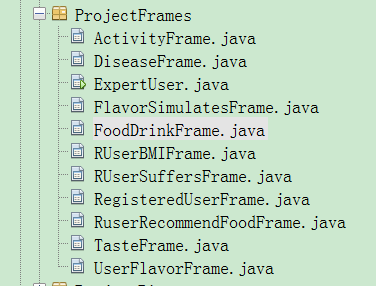


Fig: Structure of ProjectFrames

ProjectFrames are Graphical user interfaces which directly interacts with application users. They are developed using GUI Designing system of Netbeans 8.2.

Take the example of how RegisteredUserFrame works. This frame is used for registered users to record their food and drink taken during the day, to record their flavor preferences, body conditions and health conditions. Also, through this frame, users can get their recommended dishes. The frame implements most of the requirements specified in the software requirement specification for registered users. As for expert users and vendor users, their frame structure is similar registered users’ frame.

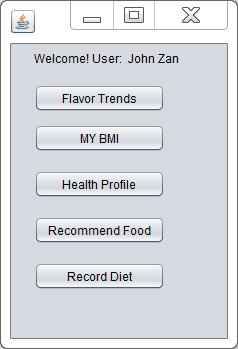


Fig: Registered user interface

Based on users’ needs, the user can then click on different buttons designed for different functional requirements. By clicking flavor trends, for example, the user can update information about his or her preferred flavors.

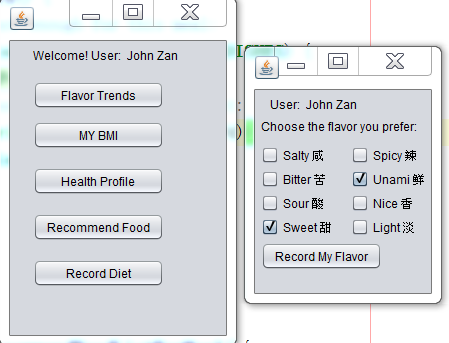


Fig: Menu of Users’ flavors

Similarly, other functional requirements shall be satisfied by clicking other different buttons on user menu.

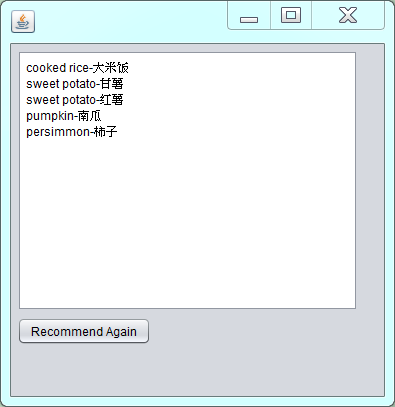
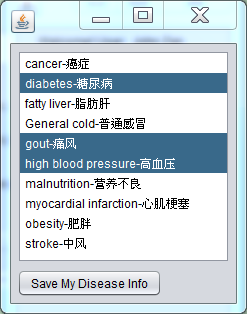
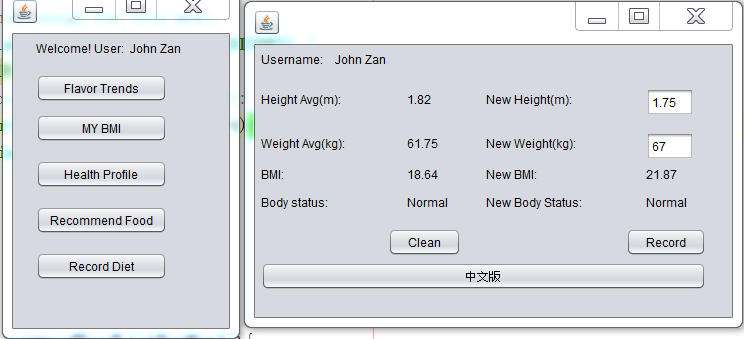




Fig: Corresponding results by clicking on different user frames for registered users

## 4.2 Testing Strategies

* Database Linking layer test

For each of the database linking subroutine, a program which simulates using this linking subroutine while running the program is written. For example, for the data linking class DiseaseLinker, which fetches disease information or write reported new diseases should be tested its functions of reading a certain disease information from database, writing a certain disease into database and update information about a certain disease in the database. This is the “white-box” testing of a DBLinker class because it “goes into” the subroutine to test every section of its implementation details. If after executing all the procedures the result returns as expected, the “black-box” testing in this case in then no need because in this case the inputs and corresponding expected outputs are guaranteed.



Fig: General view of testing codes for DiseaseLinker

Then, to test reading disease information from database, some simulated cases of requiring getting a type of disease is written, as shown below:

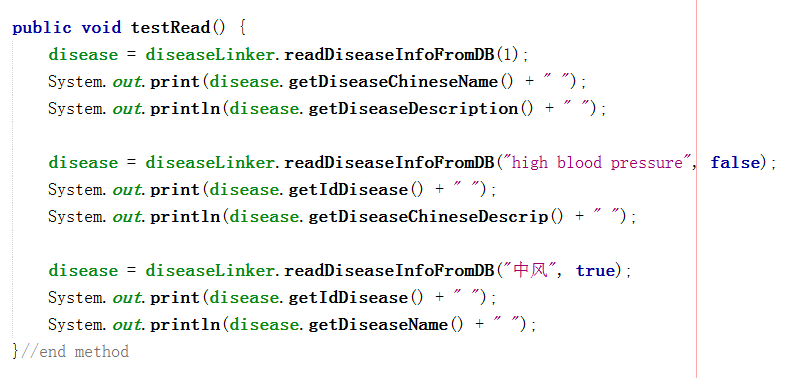


Fig: Black testing of reading certain diseases from database

Similarly, to test writing disease and updating disease information from database, simulated cases are designed and tested, the corresponding subroutines are as shown below:

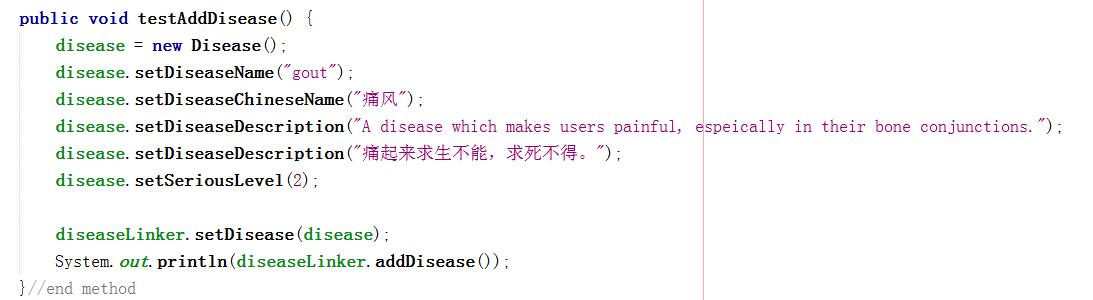


Fig: Black testing of adding a new disease information from database

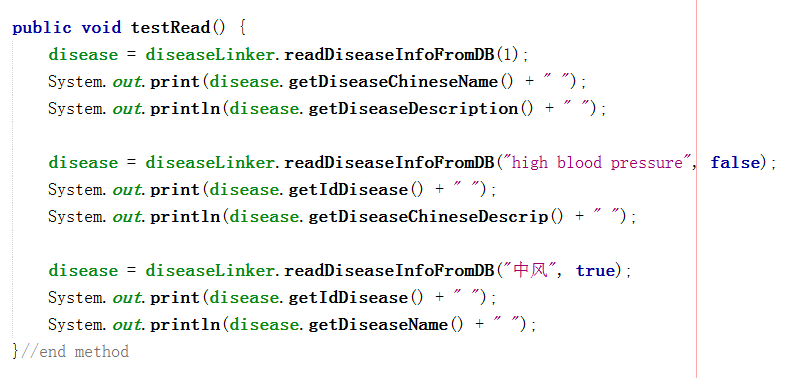


Fig: Black testing of updating the information of a certain disease

As for the testing cases of other database linking subroutines, the strategies are similar to this subroutine.

* Project Framework test

The project frameworks are used to link to database linking subroutines to read and write data from the user side, while they directly interact with program users. Thus, it is important for those interfaces to be able to interact well with databases and application users. During the phase of testing user frameworks, both the project and the database application are under observation. However, due to the finishing of linker program tests, the linking programs are treated as “black boxes” which are able to get correct input and give out expected outputs. Thus if problems occur, it is more likely that the framework sides are biased.

Another issue needs to be considered is exceptional inputs from users. Take the example for reading the calorie number of certain kinds of food. The expected value is actually of the format floating points, but in real usages it might be possible for users to input non-value texts, such as “h” for users may accidentally press wrong buttons. The user frameworks shall be robust enough to deal with unexpected user inputs.

Thirdly, the user-friendly cases of the interfaces shall be taken into consideration. Issues such as whether hints and instructions for users are clear or not, whether the frame achieve all the functional goals shall all be taken into consideration.

* Security policy test

This project uses SHA-1 to encrypt users’ passwords using API (Application programmable interface) commons-codec-1.10.jar. Thus, the correctness of this API also needs to be tested, making sure that it is able to produce expected results. Because this application already distinguishes different users by their different username and password, it is considered that their information is confidential if their password is not being exploited for their information for other aspects are decided one-to-one to this user only, plus without their agreement, those information is not even achievable by expert users who will be strictly being checked during their registration. However, attackers are always not guaranteed to be inaccessible, all the security policies for this application can only lower down the possibilities that users are attacked.

## 4.3 Testing Results

* Database design test

During the design and implementation on platform MySQL5.7, all the requirements of 5 different normalization forms are checked one by one. The checking results suggests did not find out any pitfalls which fails to fulfill any level of normalization specified above.

* Demonstration of database linking tests

By applying the strategies stated in database linker test, all the linker classes are tested. Below shows the testing of RUserLinker class. This class links between registered user and the login frame for users. The testing case simulates user trying to log in (when username and password needs to be compared), user registering and user modifying their information, testing case subroutine is shown below:

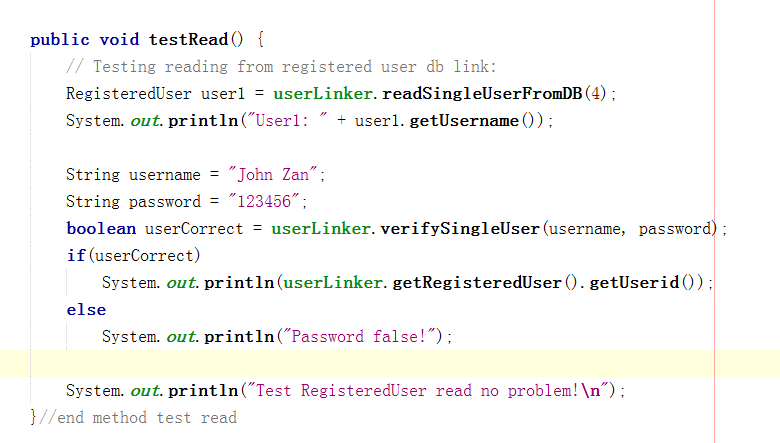


Fig: Subroutine of simulating user log in

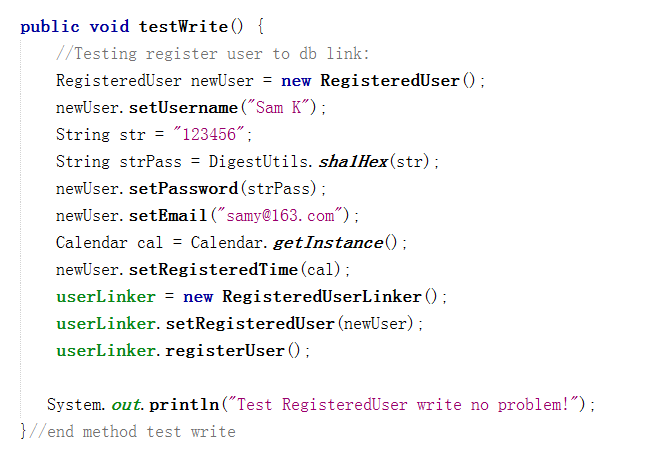


Fig: Subroutine of simulating user registering

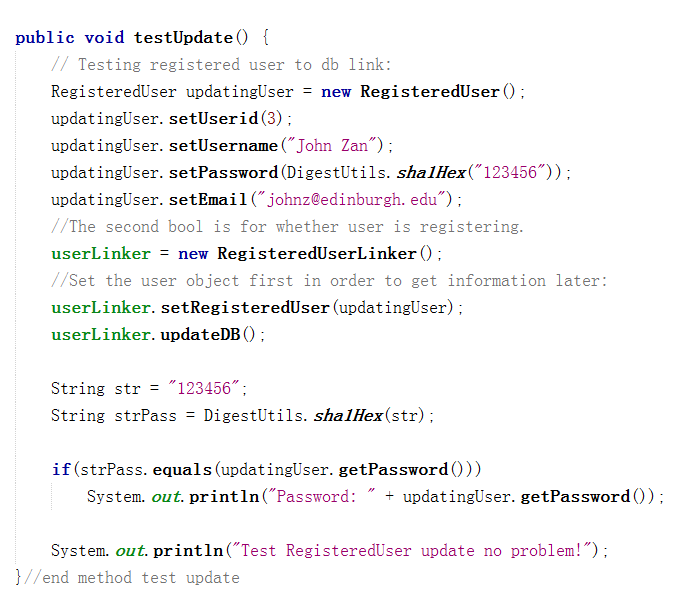
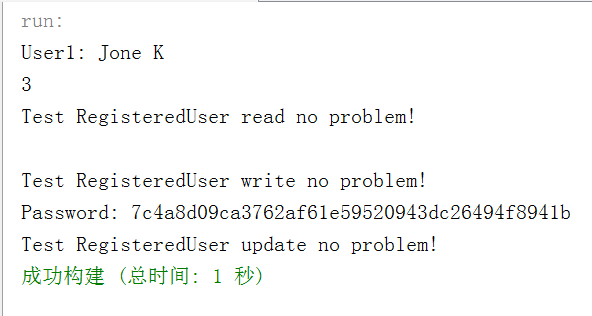


Fig: Subroutine of simulating user updating information



IMG_256

Fig: Result of first trial, new user is registered and reading / uploading not affected

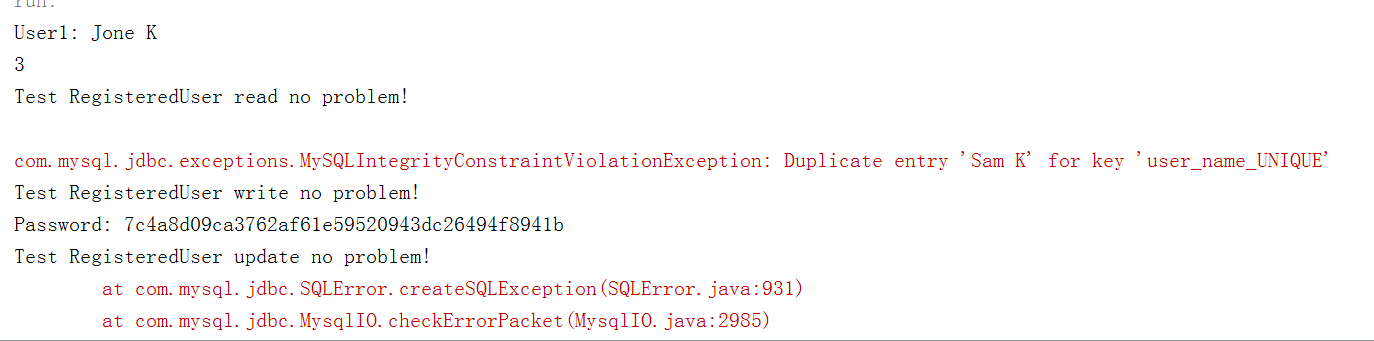


Fig: Result of the second trial, registering new user fails, while reading and updating user information not affected

Note that in the second trial, writing the new user “Sam K” into database fails, because users are supposed to be unique. This testing shows that the database system can respond correctly when meeting with the case registering two users, preventing duplicated registration and possible flooding through keeping registering zombie accounts.

Similarly, all the other database linking interfaces are tested using this approach. The results of this level of testing suggests that the linking programs all work as expected.

* Demonstration of project framework tests and evaluations

As specified before, for the testing of framework, each function of the frame is clicked to see the action of the programs.

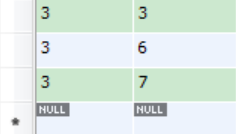
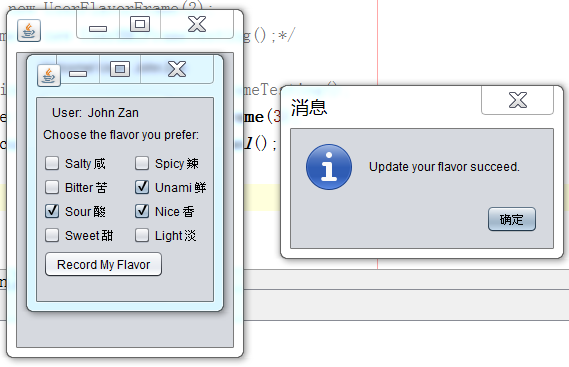


Fig: Testing results of user specifying their flavor tendencies

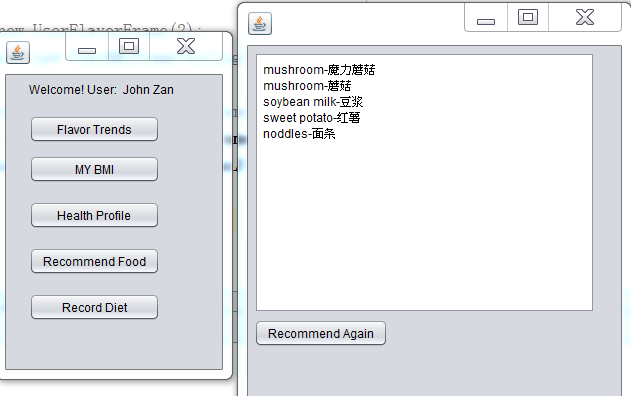


Fig: Testing results of user trying to get his/her recommended dishes

Similarly, other testings related to different registered users and expert users are tried. The results turns out to be that except getting recommended food, all other functions works as expected. However, the program might stuck sometimes when running button “Recommend Again”. Further checking suggests that the reason is due to a “dead loop” requiring to get 20 dishes as a cache, however, the actual size of all dishes is only 16. The problem shall be solved by entering more information about food and drink, for the designed sample of food / drink list is more than 20, thus automatically solving this “dead loop”.

Another issue is that those user interfaces are not friendly enough. The font is too small, together with the problem that the recommended diets are not vivid even though tool tips are designed for those buttons and columns. Plus, when users are trying to input exceptional data, the program only hints user writing to database fails, which may not be specific enough.

* Demonstration of security policy tests



Fig: Sample testing for encryption API

In order to prove that the encryption API generates expected encrypting message, example above is used, and the result is shown below:

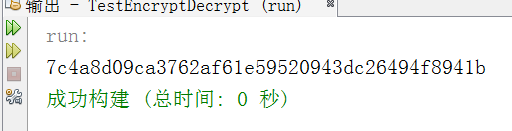


Fig: Result of encrypting message “123456” to encrypted message

This result is exactly the same by applying SQL language “INSERT INTO what\_to\_eat.registered\_user VALUES (XX, SHA1(‘123456’)”, proving the encryption API shall work as is expected.

IMG_256

Fig: Result of registering user using SHA1 and whose password is 123456

## 4.4 Conclusion

In the testing phase of this project, all the three levels of this application, namely database designing and implementation layer, database linker layer and project framework layer are all fully tested. The results suggests that the database layer and 90% of the database linker layer programs are working as expected. Though 80% of the project frameworks are working as expected and 83.3% user functions are supported, the pitfalls of less elegantly designed graphical user interfaces needs to be further improved.

# Chapter 5 Future Work

## 5.1 Discussion of recent progresses

Till the moment that this report is finished, this project already nearly reached upon its phase 2 works. Namely, the basic structure of the database and the application is built and the application is succeeded in achieving its goal as a decision making system which automatically helps users by recommending food and drinks based on their preferences and body conditions.

However, extending possibilities still remain for the whole system. First of all, due to time and condition limitations, this system is not able to be implemented as a server-client one. If it could be implemented as a server-client one, then it can have real values for many purposes such as data gathering, nutritional conditional researches among certain groups of people. Secondly, at this moment the project only considered functions for expert users and registered users, it only remains the interfaces for vendor users however remains not implemented, which means its commercial model is still not fully implemented. Last but not least, other than the requirements specified by this book, users in the future may come up with more needs of measuring their nutritional conditions. For example, if they require more accurate measurements of different kinds of vitamin inside the food, some parts of all the three layers of this application might need to be modified to satisfy this changing. In this particular example of measuring more items about nutrition, the suggestion is to take apart the fooddrink table as two tables, one is still its own, another is the vitamin table. The attributes (columns) of the new vitamin table contains foodDrinkID, vitaminA, vitamin B12, for example, and so on.

## 5.2 Suggestions for future work

Because the basic works of this project is done, meaning actually the requirements from most important user registered users and secondly important user the expert users are satisfied, this application can actually be put into real use. So the most emergent task to put it into real use is to develop the server-client model for this application. Other issues such as adding more features and extending more functional purposes can be implemented in a later approach.

A good thing about the three level designing of this application is that adding features in one layer does not interfere with functions of another layer. Take the example of measuring more accurate issues of vitamin specified by users. Adding a vitamin table only requires to change issues related to food and drink in the database design, it does not require the application to change other parts of the data at all, not even mentioning to change other parts of the application implementations.

# Bibliography

1. Abraham Silberschartz, Henry F. Korth and S.Sudarshan, Database Concepts sixth edition, in 2011, McGrawHill, New York
2. Anne Gardiner and Sue Wilson, The Inquisitive cook: Discover How a Pinch of Curiosity Can Improve Your Cooking, in 1998, Henry Holt and Company, New York
3. CCTV program “Laws on line”, report cases of collation, URL: <https://v.qq.com/x/cover/d2uhjthvr0w1301/y00233mfntv.html?ptag=tips.xw&pt_src=3&ADUIN=896298116&ADSESSION=1489818134&ADTAG=CLIENT.QQ.5515_.0&ADPUBNO=26657>
4. Calorie consumption of different activities: URL: <http://fitness.39.net/a/161013/4990792.html?open_source=weibo_search>

<http://wenku.baidu.com/link?url=dbabDB1SuPL1AP9NrfOmzAkXwgsXnBpF6kWG2sifd_ehq1W_xqxJn8lZcjJ4GICfTplzPiNoFklIAyk_TQNHLu5_vjNf0IRlUeah7M9IMhq>

1. Fan Weihong, Proper and improper combinations of food quick pedia, in July 1st, 2014 Zhejiang Science Publishing House
2. IEEE, template of software requirement specification, URL: <https://www.google.com.hk/?gws_rd=ssl#q=template+software+specifications+document,> used on 2nd Mar, 2017
3. Li Hui, Proper and improper combinations of food, in Jan 2009 Beijing Yanshan Publishing House
4. Liqi, Ou, Big Talk Software Testing, in 2014, Publishing House of Electronics Industry
5. Mike O’ Docherty, Object-Oriented Analysis & Design: Understanding System Development with UML 2.0, in 2005, John Wiley & Sons, Ltd
6. News reports about food toxic accidents or events, URL: http://news.fznews.com.cn/shehui/20170120/5881801e2cfa1.shtml, used on 15th Feb, 2017
7. News reports about food toxic accidents or events, URL: http://news.ifeng.com/a/20170125/50628244\_0.shtml?\_zbs\_sogou\_bd, used on 15th Feb, 2017
8. News reports about food toxic accidents or events, URL: http://zj.qq.com/a/20170203/014554.htm, used on 15th Feb, 2017
9. Wang S., Sa SX, Concepts of Database system, fifth edition, in February 2016, Higher Education Press
10. Wenzhou Lu, Qt 5 Develop and Examples 2nd Edition, in Publishing House of Electronics Industry, in Beijing, Dec. 2015
11. W. Stallings and L.Brown, Computer Security–Principles and Practice, 2nd Ed., in 2015 Prentice Hall
12. Toby Michelena, course slides of BIO3200 Food and Nutrition, in 2016 Winter session, Wenzhou-Kean University
13. Xie Fuen, Huang Hongzhong and Zeng Xun e.t., Similar Sources of Food and Medicine, in Aug, 2012, Guangzhou Science and Technology
14. Y. Daniel Liang, Introduction to Java Programming Comprehensive Version, ninth edition, in 2013, Pearson
15. Zhong Qu, Lin Liu, Anping Xiong etc., Introduction to Computer Science, 4th Edition, in Mar 2014, Tsinghua University Press

The cover and the table of content of the book is designed using @KingSoft’s automatic cover and table of content generation system.

1. If not specified, menu in this proposal refers to users’ dietary plan. [↑](#footnote-ref-0)
2. Here, package means a file structure that groups similar classes of objects, rather than packages on Internet protocols or other occasion. [↑](#footnote-ref-1)