Test Plan

Methodology

Testing Overview

There are three stages of testing, including development testing, release testing and user testing. Development testing, where the system is tested when developers are writing the code. The aim of development testing is to find bugs and defects in the development process. It is also called as Unit and Component Testing. Release testing requires a separate test team to do the testing before it is released to users. User testing is tested by user and customers. It is also called as alpha and beta Testing. Our testing plan is mainly for the development testing. The release testing and the user testing will be done after the project has been finished.

Testing goals

Validation Goals

Validation test is used to simulate user's normal input and system will get expected result. In this test, we will input normally, then system is expected to perform correctly. According to requirement document, we will test all functions in it to meet sponsor's request. Moreover, we will focus on both functional requirements like how system react to particular inputs and non-functional requirements such as reliability, response time and memory use.

Defect Searching Goals

Validation test is used to simulate user's abnormal input and find how system will react this situation. In reality, users may not input as we expect, thus we need to find systems bugs and avoid system behaving incorrectly under such circumstance. After finding defects of systems, we need to add more constraints to enforce and guide users input as we expected like limiting password must contain both upper and lower character and or improve server's reliability to avoid hostile attack.

Unit Testing

What it does

Unit testing is to test each part/unit of the software. It aims to validate individual components and make sure it runs as designed. A unit should be a basic part of an software. No matter how many inputs are, it usually has only one output. Therefore, a unit could be treat as a method, abstract class or child class.

Why it helps

1) Unit testing makes maintenance easier. After unit testing, we can assure each part can work independently. In other words, changing code will not cost a lot. Therefore, we must make our code modular for this purpose. And this could also make our codes more reusable.

3) Make the development much faster. We can test the whole software separately which means the codes should be in order and hierarchically. Hence, the time we spend in detecting the bugs and fixing them would be significantly reduced compared to poorly written codes. And to reach this performance, it needs us to think the structure clearly and design the architecture in a better way. If we have a good start, we may also have a better final work. And in a long term, we may take less effort in the further testing like system testing or acceptance testing cause the structure of the whole software is clear enough for us to detect and fix problems in a short time.

4) Easy for engineer to debug. In unit testing, only the latest changes impact the test result. However, a higher level testing would take several days even months to debug cause the bug may results from many aspects.

5) Make the documentation more clear. The developers can just read the document to know how to run each part and what these units actually do. Thus, if the developer want to make some changes based on some specific functions, it would be much easier to make change and the cost would not be expensive.

How to perform

We need to choose two kinds of test case that reflects that the component can work normally and finds out the defects in the product respectively.

We will choose appropriate inputs to make it happen. And we also need to include the corner case input to test the product to make sure it will not crash under any circumstances.

We still have to design inputs to force the system to generate all kinds of error messages, e.g. input buffer overflow, repeat input numerous times, too large and too small computation result.

Component Testing

What is Component Testing & Interface Testing?

Components are composite of several interacting objects, featuring specific functionalities. A page, screen, sub-systems out of a whole system are legal components. Component testing are performed to validate whether a system follows its specification/requirement and produces expected output, and to verify if error handling is properly executed when encountering system misbehavior or defects, such as illegal characters in inputs, out-of-range inputs, error messages.

Interface Testing

An interface is a communication between two components. Consider an example of data transfer between frontend and backend component aka. client app and database. Another example is how a payment page is integrated with a payment platform from other systems. Interface testing is applied to prove correctness of these communications.

Therefore, these two testing are often implemented simultaneously, due to the affinity that one tests individual component and the other tests connection between components.

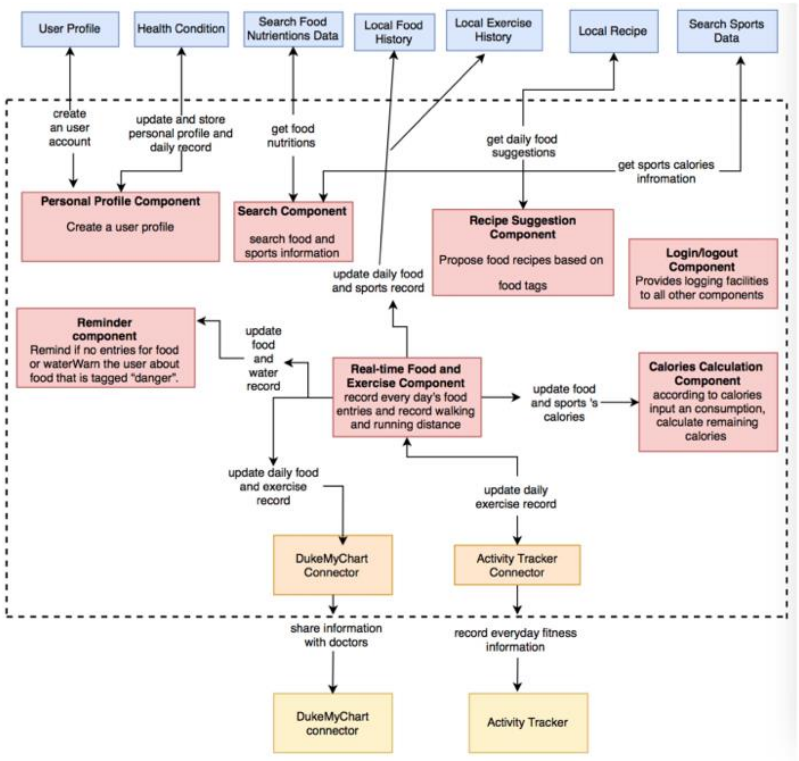
Why Component Testing & Interface Testing are important?

A component or a feature, is indeed a higher level of generalization from a cluster of function units. Even though testing on each function unit has validated and verified its completeness, integration of all components may generate extra problems with accessibility and usability which may force system errors, and crashes. Thus, a compatibility testing is required in-between units testing and release/system testing which may consider concurrency, stability, security, etc.

Also, interface failures possibly come from interface misuse, interface misunderstanding, and timing errors.

Implementation

The implementation is bound to our components diagram in C4 architecture design shown below:



Guidelines from the lecture would be useful, which lists some testing key points for component and interface testing, respectively. Component testing is supposed to cover all error messages, input buffer overflow, numerous repeating with same inputs, invalid outputs, two extreme ends of outputs. Interface testing should hunt down testing cases like extreme ends of range in passing parameters, pointer parameters with null pointers, re-ordering of component activation. Some examples may be shown here, but more details should be found in our plan in the final section.

1. Testing the UI part for usability and accessibility
2. Testing the Page loading to ensure performance
3. Trying SQL injection through the UI components to ensure security
4. Testing the login functionality with valid and invalid user credentials

System Testing

What it does

System testing is a level of software testing where a complete and integrated software is tested. It is the process of testing an integrated system that is intended for use. The purpose of it is to convince the supplier that the system is good to use and meet the requirements.

System testing is black box type of testing where the output performance is evaluated by the requirement documents. It is testing from users' view and do not required the knowledge about the system internal design.

Why it helps

First, it tests at the level of whole system. Testing verifies that the system meets the different requirements including, functional, performance, reliability, security, usability and so on. Second, the system is tested in an environment that is similar to the users' environment and hence the team can get a good idea of the user's reaction. Third, system testing can ensure the quality of the system. It finds the bugs and defects then helps to minimize the post release costs of support and service.

How to perform

The first step is to make a system test plan. In this step, we need to consider some points and strategies to ensure the quality of system test. Some common points include the goal of system test, critical areas to focus, test deliverable, testing schedule, entry and exit criteria, test environment, roles and responsibilities.

The second step is to create test cases. In this step, what should be considered is the different type of testing like Functional testing, Usability testing, Compatibility testing, Performance testing, UI software testing, Security testing, Reliability testing, Error handling testing, Installation testing, Accessibility testing and so on. When writing test cases, you need to check the test cases are covering all functional, non-functional requirements.

The third step is to execute the test cases. In this step, we can execute normal test cases and update test case if using any test management tool. After testing, the testing process should have bug reporting, bug verification.

Finally, we should repeat testing life cycle if required.

Summary

Software Testing is evaluation of the software against requirements gathered from users and system specifications. It can provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. The techniques we used include the process of executing a program with the intent of finding software bugs, and verifying that our software product is fit for use.

First, we specify the testing methodology, divide the testing process into different stages. Then we provide our testing goals, validation goals, unit testing, component testing and system testing. Finally, we give the detailed test plan both in sprint 1 and sprint 2.

Plan

Sprint 1

Unit Test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	POST account	Steven Jacobson	3/27	3/28	1. Check to make sure results from each API call are correct. 2. Interact with the database, possibly using external software (Postman) to make sure attributes of the objects can be created, uploaded, modified, deleted, and returned.	1. All API calls work as expected (as described in README.md). 2. Errors are handled gracefully. 3. Attributes of objects are created, uploaded, modified, deleted, and returned successively.
1	DELETE account		3/27	3/28		
2	GET health info		3/28	3/29		
3	POST health info		3/28	3/29		
4	GET food	Nan Ni	3/27	3/28		
5	POST food		3/27	3/28		
6	DELETE food		3/28	3/29		
7	GET exercise	Peiqi Xu	3/27	3/28		
8	POST exercise		3/27	3/28		
9	DELETE exercise		3/28	3/29		
10	Log in interface display	Qing Lu	3/27	3/28	1. I check situation when password does not match with username 2. click login button 3. click sign uo button	1. User can only input wrong password in 5 times 2. after clicking login button, it will enter main interface 3. after clicking sign up button, it will enter sign up interface
11	Sign up interface display	Qing Lu	3/28	3/29	1. I input wrong format email 2. I input wrong format password	1. warning will show" invalid email" 2. warning will show "Password must contain minimum 8 characters at least 1 Alphabet, 1 Number and 1 Special Character"

12	Main page profile display	Yuqi Ma	3/28	3/29	1. Click “Add Food” button 2. Click “Add Exercise” button	1. The Main page interface will jump to Add Food interface 2. The Main page interface will jump to Add Exercise interface
13	Add exercise interface display	Yuqi Ma	3/28	3/29	1. Click” Choose Enercise”, check if all kinds of exercises are listed in the pull-down menu 2. Choose “Walking” in the pull-down menu 3. Choose “Ball Games” in the pull-down menu 4. Choose “Swimming” in the pull-down menu	1. “Running”, “Walking”, “Swimming”, “Dancing”, “Yoga”, “Ball Games” are all listed in the pull-down menu 2. “Walking” is showed in the search bar 3. “Ball Games” is showed in the search bar 4. “Swimming” is showed in the search bar
14	Add food interface display	Yuqi Ma	3/28	3/29	1. Click search button to enter the food 2. Enter “apple” and click “DONE” button	1. Iinput keyboard will showed to type the food name 2. Food can be stored
15	okhttp3 package usage	ZIHAO WU	3/27	3/28	1. Send a POST request to sign up 2. Send a POST request to login 3. Send a POST request to edit personal profile 4. Since all information will be validated before being sent to server, so no defect testing could be crafted 5. Crafted some GET request to health information, food, exercises,	1. At the database side, to validate the request sent, I print out the message, and all testing work as expected 2. At the app side, the response received from database are all correct as requested, in JSON format
16	Transform JSON data into an object which has attributes match key-value pairs,		3/27	3/28	1. Parse to an object with attributes 2.Parse to an object with nested objects	1. all parsing worked as expected 2. actually, no worry about the Gson package, since the use

	then store it in local storage				3. Parse to an object with lists of objects 4. Parse to an object with Null values	is straightforward and simple
17	Activity redirection	XV RONG	3/27	3/28	1. Try all OnClick function for all buttons	1. all buttons function well
18	Username, Email, Password format validator		3/27	3/28	1. Try passwords which follow the prompt "Password must contain minimum 8 characters at least 1 Alphabet, 1 Number and 1 Special Character" 2. Try passwords which doesn't follow the instructions 3. Try password in length of 100 4. Try password in SQL injection Such as "xxxxxxx or 1=1" which attempts to get all passwords, or "DIO Robert'); DROP TABLE account; --" which intends to make destruction on database 5. Try email without "@" or "." 6. Try username over 20 characters	1. validation and verification are done and output as expected; 2. all misuse will be prompted by a message; all inputs to database will be sanitized by Django's middleware, and therefore SQL injection is not eliminated
19	Profile Edit validator		3/27	3/28	1. Try correct data type on corresponding blanks 2. Leave blanks 3. Try wrong data type	1. all wrong data type info will be prompted by a message 2. data length is not checked here, since the validation is done on database with digit limit set on model fields.

Component Test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	Login Connection	Qing Lu	3/31	4/2	1. I enter the login interface. I input the username and password to check if I can enter the profile interface 2. I enter the login interface. I click signup button to check if I can enter the signup interface to register	I can enter the personal profile or sign up interface though the login button or signup button.
1	Sign up and information register interface interaction	Qing Lu	4/1	4/3	1. I enter the signup interface, after registering username, password and email, I can enter the information register interface to input more personal information	The connection between sign up and information interface works well.
2	Information and personal profile interface route	Xi Li	4/1	4/3	1. I check if I input information, then click the update button, I can enter the personal profile interface. 2. I check the content of personal profile interface is the same as the information registered	The jump between two interfaces display well. And the information is the same as the information registered.
3	Edit personal profile route	Xi Li	4/2	4/4	1. I check when I click edit button at the personal profile interface, I can enter the edit profile interface. 2. I can modify information and update information. 3. After I click confirm button, I can return the personal profile	The interaction between edit personal profile and personal profile works well.

					interface and the information in that interface is updated. 4. After I click cancel button, I return to the personal profile and the information is not changed.	

System Test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	Log in and Sign up	Qing Lu	4/3	4/4	1. sign up with email, set username and password. 2. Log in with account	Registration is successful. Login with registered account is successful Proper error reminder like password is not correct.
1	Information and personal profile Test	Xi Li	4/3	4/4	1. enter user information 2. edit prosomal profile	User can input their information and change their information.
2	Edit profile Test	Xi Li	4/3	4/4	1. input information in editText view 2. return to personal profile to check information 3. If check cancel button, the information will not change	Confirm the information changed, the modified information will update in the personal profile. Cancel the change, personal profile will not change, too.
3	System Security	ZIHAO WU	3/27	3/28	1. May run DDoS attack test using an open source program called GoldenEye e.g. set up 200 users to visit the database, each with 100 concurrent sockets. The server may be not accessible any more i.e. 504 gateway time-out response	1. Attacks successful on Django 2. Strategies may be required to lock and ban suspicious users in the future

4	System Security	ZIHAO WU	3/27	3/28	1. Use Wireshark to grab packets on the database ip	1. The attack succeeds to steal username and password, since the database running is still a development server which does not support HTTPS protocol, and thus, packets are sent in plaintext

Sprint 2

Unit Test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	AddUSDAFood(food)	Steven Jacobson	4/9	4/10	1. Call this function when wanting to access and add food from USDA database. 2. Insert item into database.	If food exists in USDA database, it should be added to the user's history.
1	TagFood(food)	Steven Jacobson	4/9	4/10	1. Tag a food as danger food. 2. Return all danger foods for one user.	Any food should be able to be tagged as "danger" by the use and displayed by the user.
2	ViewRecipe(keyword)	Steven Jacobson	4/10	4/11	1. Access online database API to view recipes given a keyword. 2. Access recipe url.	1. Given any keyword, be able to view all potential recipes. 2. Be able to access the webpage from recipe url.
3	FindNumCalories()	Nan Ni	4/9	4/10	1. Ask for the number of calories remaining in the day. 2. Change health target, repeat step 1. 3. Change amount of exercise / food	Be able to accurately find number of calories remaining based on any health target, and any type of food or exercise completed that day.

					consumed, repeat step 1.	
4	CalculateTargetCalories()	Nan Ni	4/9	4/10	1. Calculate the target number of calories for a user. 2. Repeat step 1 with different target.	Based on a listed health target (from the user), be able to calculate the suggested number of calories to consume for a day.
5	RemindFood()	Peiqi Xu	4/9	4/10	1. Set amount of time to remind user to eat in health profile. 2. Wait amount of time, and make sure user is reminded. 3. Repeat steps 1 and 2.	1. Be able to set different time lapses in between meals. 2. Consistently remind the user about time lapse of food, over many trials, without failure.
6	RemindWater()	Peiqi Xu	4/9	4/10	1. Set amount of time to remind user to drink in health profile. 2. Wait amount of time, and make sure user is reminded. 3. Repeat steps 1 and 2.	1. Be able to set different time lapses in between drinking water. 2. Consistently remind the user to drink water based on time lapse listed.
7	Log out interface	Qing Lu	4/9	4/10	1. Click logout button	1. Be able to exit application
8	Food recipe interface	Qing Lu	4/10	4/11	1. Click one picture	1. This interface show several food recipe with pictures 2. After clicking one picture, the interface show detailed information of recommended food recipe
9	Tag food interface	Xi Li	4/9	4/10	1. Click the food chocolate to tag 2. Input the tag “High Calories”	Chocolate will be stored with tag” High Calories” in local database
10	Reminder interface	Yuqi Ma	4/9	4/10	1. Set the remind time of	1. Remind the user “It’s time to drink

					"drink water" every 2 hours 2. Set the remind time of "Yoga" at 7:00 pm. 3. Add food "chocolate" in daily food	water" every 2 hours. 2. Remind the user "It's time to do Yoga!" at 7:00 pm. 3. Remind. User immediately "It's dangerous food!"
11	JOSN parser	ZIHAO WU	4/10	4/11	1. Request food nutrition from USDA's REST API 2. Request food nutrition from database 3. Request recipes from various resources, most with REST API	1. parse with no problem
12	Planner interface	ZIHAO WU	4/10	4/11	1. try some food/recipe with extreme food indexes 2. try to add food with danger tag 3. try food/recipe with nutrition in whole less than target	1. warning message is supposed to prompt to inform user of incorrect food/recipe combination 2. kick out of food with danger tag.

Component test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	Add food interface route	Yuqi Ma	4/9	4/11	1. I click add breakfast, add lunch, add dinner, add snack in the main interface 2. I can enter the add food interface 3. after I input enter food, I can return to main interface and I can see what I input in the main interface.	Click the buttons related to add food in the main interface, I can enter the add food interface, then can return to main interface.

1	Add exercise interface route	Yuqi Ma	4/10	4/11	1. I click add exercise button in the main interface. 2. I check whether I can enter the add exercise interface, then return back to main interface	The add exercise interface and main interface can connect correctly.
2	Food recipe test	Qing Lu	4/11	4/12	1. I enter the recipe interface, I can see the food recipes. 2. I check whether I can enter different recipe without constriction.	User can select the recipes according to their interest.
3	Log out Test	Qing Lu	4/11	4/12	1. I check when clicking log out button in the main interface, I can log out the application	User can exit the app
4	Tag food interface Test	Xi Li	4/10	4/11	1. I Click the food chocolate, I can tag the food as high calories. 2. after tagging, I can return back.	The tag food interface can connect correctly.
5	Reminder interface Test	Yuqi Ma	4/10	4/11	1. I check after the reminder is in the screen, I can click cancel to return back the current interface	The reminder can connect with any interface and It will not affect the current task after cancelling it.

System test

Test ID	Test Name	Owner	Start	Finish	Test Steps	Expected Result
0	Log out Test	Qing Lu	4/10	4/11	1. Click log out, I can exit the application 2. Enter the app again, I need to log in again	User can use logout securely.
1	Tag food interaction Test	Xi Li	4/10	4/11	1. I check when food is tagged, It can show its tag after that food display	User can tag food according to health condition and the tag can be displayed.
2	Reminder interaction Test	Yuqi Ma	4/10	4/11	1. I check whenever the reminder occurs, it will not	The reminder can remind user but not

					affect the current interface's work 2. cancel the reminder, I can continue the current task	disrupt the current work.
3	Load Test	ZIHAO WU	4/10	4/11	1. Stress test is done with numerous users sending requests concurrently 2. different combinations of requests in parallel to test different resources contention 3. send significant data in stress test, since performance depends on data size	The server could handle in a relative high performance, aka. low latency + high bandwidth.
4	Acceptability Test	XV RONG	4/10	4/11	Let a group of students to try the system, and collect feedback to analyze if the system is acceptable to most of students	It may depend on many factors like performance, custmonmizability, community, etc.