Software Requirements Specification (SRS)

CMPT475 Spring 2022 – Group 10

Dezheng Zhong & Yuanying Mu & Wenqing Liu & Siwei Wang & Tongfang Zhang

1. Instruction

1.1 Purpose

The purpose of the software is to build an application in the mobile phone that all customers who use IOS or Android platform can get an accurate number about the calories in the food by taking the picture. This document provides a framework that all team members must follow. It will help the team complete the software's requirements. In order to make sure team members understand the requirements clearly, this document will list necessary and comprehensive demands for the project.

1.2 Scope

1.2.1 Name: Calorie Camera

1.2.2 Product Functionality:

The product will calculate the calories of the food user's intake in the picture. It will help monitor and analyze the user's diet and upload the data to the cloud database. The database will support almost hundreds of user's records around the world. Digital Health Inc can use these data to analyze eating habits for most people, understand the status of people's health and give a solution by making a healthy diet, creating a comprehensive intake system, completing a therapy to cure obesity, etc. For economic benefits, the results can be sold to clinics, dietary supplement companies or individuals who want to keep healthy. Most importantly, we hope this product can help users live healthy.

1.2.3 Product benefit, objectives and goals

Benefit:

First, it is convenient to use. As it is a mobile app, people can use it anytime if they bring their mobile phone. Second, for calculating the calories, people only need to use the camera of the phone and upload the pictures which is quite easy. Third, people can use it to make a healthy diet which benefits them. Last, the product uses Artificial Intelligence (AI), it can study by itself and improve the accuracy.

Objectives:

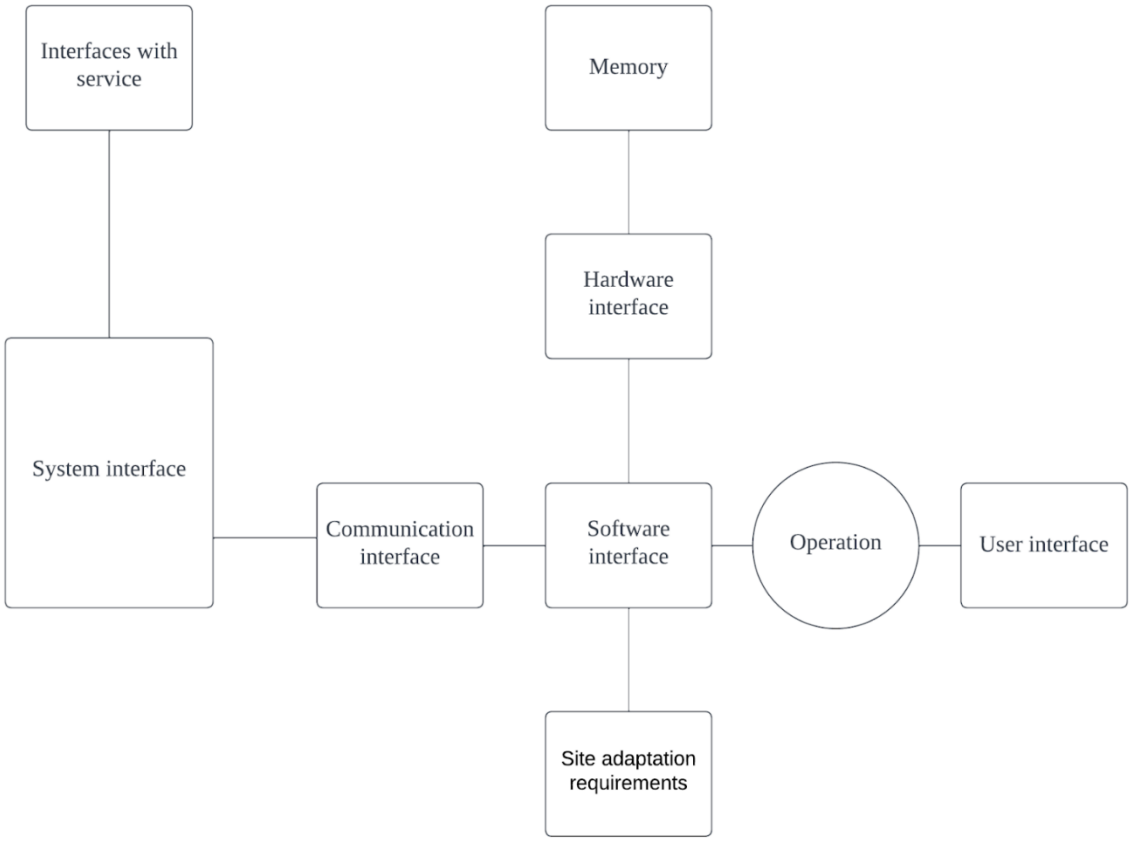
The objective of the software is to use big data to analyze people's health status.

Goals:

The first goal is to solve overweight problems by monitoring calories intake so that people can keep healthy. The second goal is that Digital Health Inc can get economic profit from the application and analyzing the data from the software.

1.3 Production overview

1.3.1 Product perspective



a) System interfaces

The system consists of the database and AI for analyzing the input image, and Amazon Cloud for calorie computation. The operation of the software relies on the huge data sets in the database and the analytical power of AI.

b) User interfaces

The GUI of the software application is easy-understanding and requires no prior knowledge of complicated interfaces. It contains icons at the bottom for choosing the account page, history page and more pages. Clicking the icon at the center of the page can turn to the page of taking pictures. Pictures shall be uploaded. Results of foods shall be displayed by charts.

c) Hardware interfaces

In hardware interfaces, plugs, sockets, cables, and electrical signals are passed through each other. The software only supports full-screen operation on mobile devices with a resolution of 1920x1080 or higher.

d) Software interfaces

The software application can run after version 12.0 of IOS on iPhone. iOS can be updated from Apple.

The software application can run after version 10.0 of Android OS. Android OS can be updated from Google.

The software shall have connection with the DBMS MySQL supporting 92 standards.

e) Communication interfaces
The software shall use TCP/IP protocol for secure communication with server

f) Memory
For the iPhone and iPod touch, it requires iOS 14.0 or higher versions or devices with Android 10 or above to run the application, and it is recommended to leave more than 10G of storage space for the secondary storage.

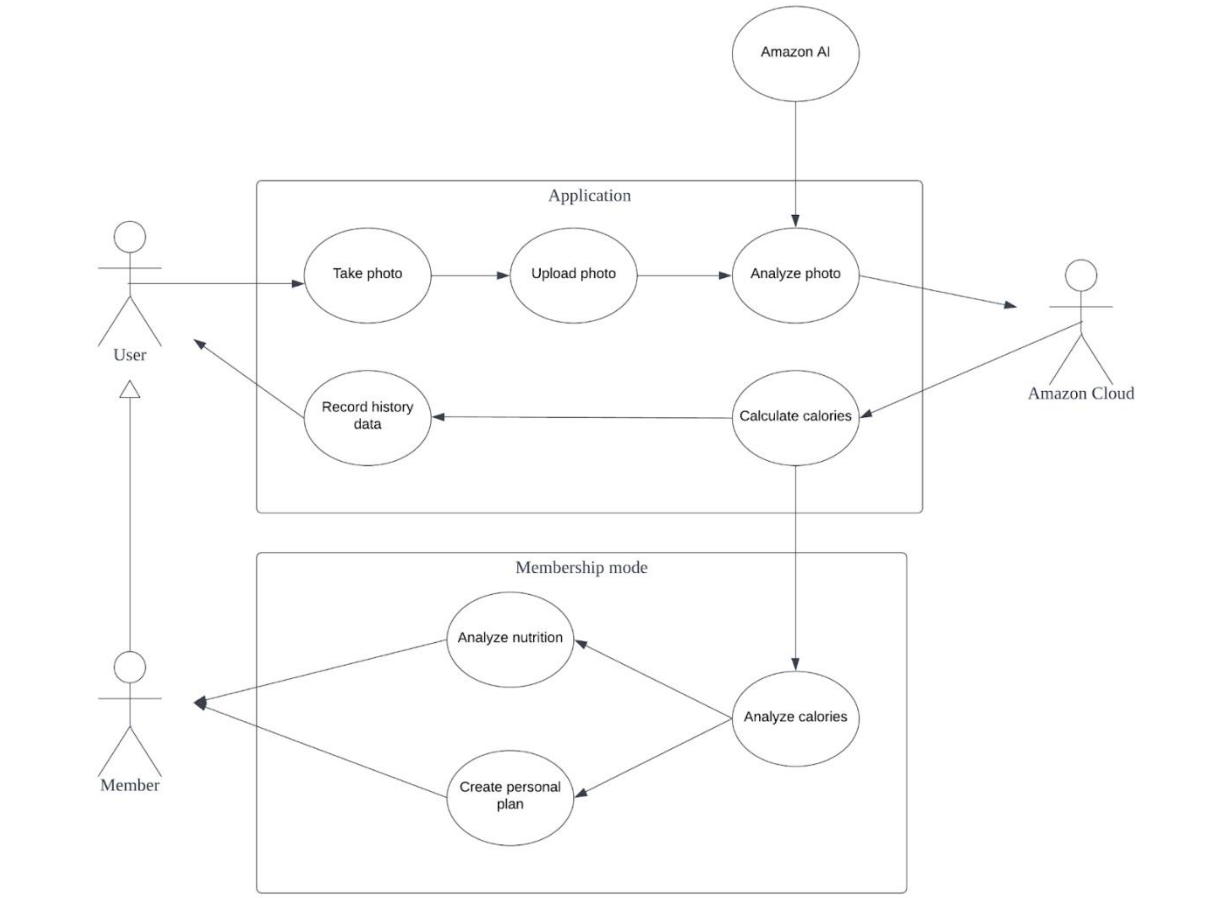
g) Operations
Initial mode: new users are automatically created based on the devices running this application.
User mode: users click the icon at the center of the main page to take pictures for calorie computation. This is the main function of the software.
Calculation data are returned by the Amazon Cloud web service and shown to the users after selecting pictures for computation.
The software also makes logs of all events and backs up data regularly to ensure fast recovery in case of system failure. Logs can also be used to monitor possible bad data after receiving results.

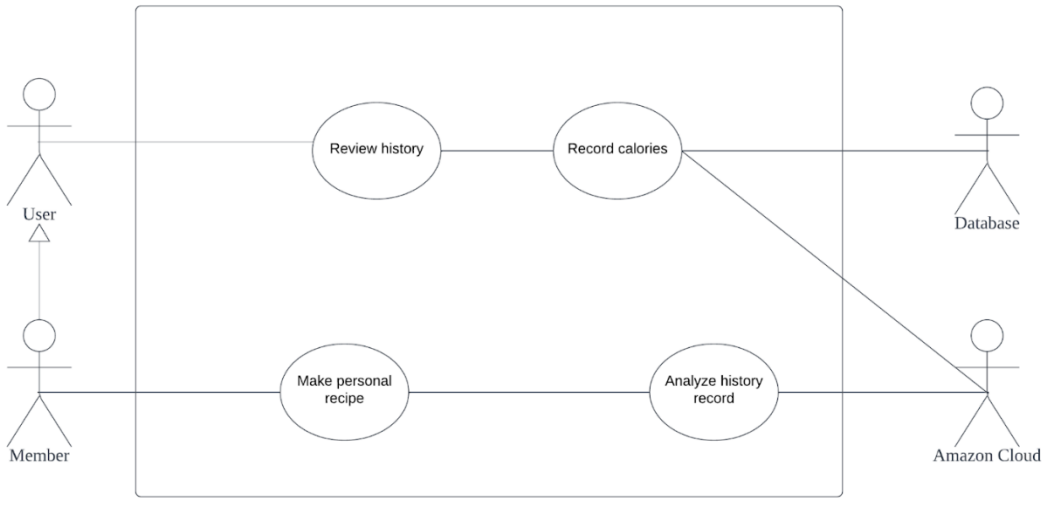
h) Site adaptation requirements

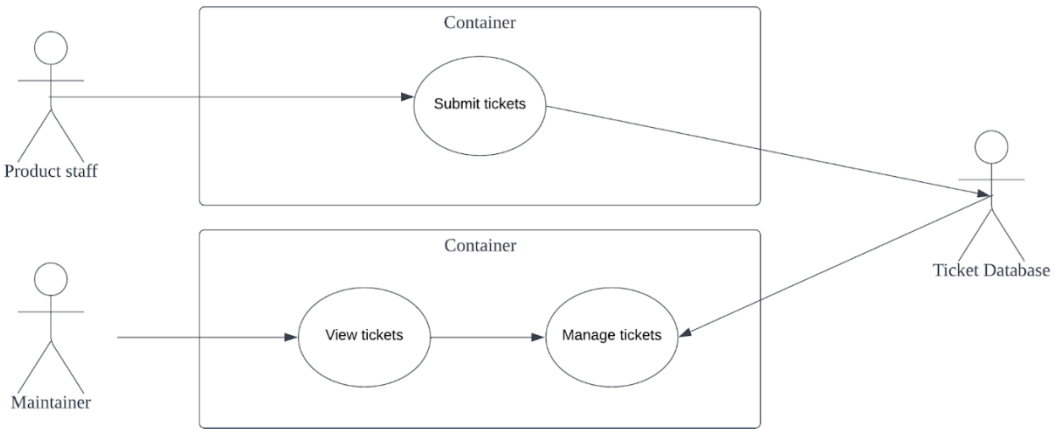
The default setting should be made based on standard configurations.

i) This product will use cloud computing power and big data provided by Amazon Cloud.

1.3.2 Product functions







a) Editing personal profile function
Users can edit their profiles, changing their personal information.

b) Transition function
Users can switch the screens between different sections of the software.

c) Capturing function
Users can capture new food images via the camera of the phone.

d) Uploading function
Users can upload captured images from the local memory of the phone.

e) Submitting function
Users can submit the created meal to the software server to get and calculate the result.

f) Deleting function
Users can delete the history calories intakes from the intaking logs.

g) Subscribing functions
Users can subscribe to the membership to get further services, such as better meal analysis, health plan, etc.

1.3.3 User characteristics

The characteristic of the software is that at noon and evening time, business volume grows significantly. It is because people always take meals during this time and they may prefer to use the software to calculate the calories before the meal.

1.3.4 Limitation

a) Regulatory Requirements and Policies
Under Canada's Personal Information Protection and Electronic Documents Act (PIPEDA), the organization may collect, disclose and use a certain amount of

information for such purposes as a reasonable person deems appropriate in the circumstances. There are some privacy restrictions when collecting user data and conducting analysis.

b) Hardware limitations

Due to server load, users are not allowed to continuously upload data and perform continuous analysis of food images.

c) Interfaces with other applications

The account of the software is bound to the mobile phone, so Apple's ID will be used to log in and record data.

d) Parallel operation

This account only allows login on one device. If the user is logged in on another device, the logged in device will be forced to log out.

e) Signal handshake protocols

The software will use XON-XOFF (software flow control). XON-XOFF MUST NOT appear in the data being transmitted without being mistaken for a flow control command. Therefore, any data that contains XOFF/XON codes must be encoded in some way for proper transmission, with corresponding overhead.

2. References

[2] Burak, A. (n.d.). Your 2022 Guide to Writing a Software Requirements Specification (SRS) Document. Relevant Software. Retrieved November 30, 2021, from https://relevant.software/blog/software-requirements-specification-srs-document

[3] Libor Bus, E. (2021, September 3). ReqView Software Requirements Specification. ReqView. Retrieved November 30, 2021, from https://www.reqview.com/doc/iso-iec-ieee-29148-srs-example

3. Requirements

3.1 Functions

Fundamental actions that have to take place in the software in accepting and processing the inputs and in processing and generating the outputs.

3.1.1 Validity checks on the inputs

The input images will be validated before performing any calculations to reduce unnecessary workload and space consumption. The system should reject invalid images which do not contain any visually detectable ingredients, such as a cat photo. If uploaded images pass the validity checks, the detection and calculation will be performed as normal.

3.1.2 Exact sequence of operations

The sequence on the user side: take photo/upload from the album -> confirm image -> get result -> to navigate

The sequence on the client side: Writing logs -> uploading to the server -> display result in client

3.1.3 Responses to abnormal situations

a) Overflow

Once the meal image reaches the maximum size, then the client will pop up an error message to ask users to either edit the image or change a new one.

b) Communication facilities

The application should provide a report page for users to submit any suggestions or feedback they may have. The server will accept reports and send them to the development team. The process will require HTTP requests and responses to facilitate communication.

c) Hardware faults and failures

The application should commit changes within a 30 seconds interval. If the hardware faults and failures happen before committing, it should not cause unexpected data loss. The application should restore any new data into a log file in case of unexpected hardware issues during usage. Once the issue is resolved, the application should do the data recovery automatically by scanning the log to look for any uncommitted records.

d) Error handling and recovery

The client backend should check the log page to respond to different categories of logs within 3 seconds. For example, if the log is marked as ERROR, the application should pop up an error window for users and push it to the highest priority. If the log is marked as WARNING, then the application should pop up a warning window for users but not push it to the highest priority. Log files should be used in the process of data recovery in case of unexpected crashes.

3.1.4 Effect of parameters

a) Hardware

Different CPU and GPU can affect the speed of calories calculation. Also, the camera can make the resolution of images different which affects the AI detection. For example, Apple A15 with 6-core CPU and Snapdragon 888/888+ gives the most satisfactory performance.

b) Network

The speed of the network and stability can affect the speed of calories calculation. Bad network conditions can increase the upload time and delay the time to present the result. If the speed of the network is over 120 Mbps, the upload process will take no more than 7 seconds for any image with a size less than 50MB.

3.2 Performance requirements

3.2.1 The application should enter the welcome page within 3 seconds after users launch it from their mobile devices.

3.2.2 The application should automatically enter the main page after playing the welcome animation within 3 seconds.

3.2.2 The application should not display scrolling jerks longer than 150ms while users scroll up and down on the ingredients table.

3.2.3 The application only allows users to upload one image at a time and the size of the image cannot exceed 50MB.

3.2.4 The calories calculation time should be based on the size of the image and the complexity of the meal. The time can be calculated as follow:

$T < S * C$

T is the total amount of calculating time

S is the size of the uploaded image

C is a constant value, which is taken as 0.3s/MB

3.2.5 The payment stage should get responses from the server within 5 seconds after users confirm the payment. If the response is “success”, then the application should direct users to the updated account page within 2 seconds. If the response is “failed”, the application should pop up with an error message within 2 seconds.

3.3 Usability requirements

3.3.1 The whole application interface should appear easy and clear to users and for any operations users want to perform, it should not take users more than 7 minutes without giving any prior training.

3.3.2 The system should allow users to navigate to different pages and go back to the previous page without losing any new progress within 5 seconds.

3.3.3 All buttons and icons with the same purpose should appear exactly the same throughout the whole stage to avoid causing any potential ambiguities.

3.3.4 The system should generate and display error messages within 2 seconds of users performing some operations incorrectly. These messages should contain instructional keywords to help users correct wrong behavior within 1 minute.

3.3.5 The core taking-photo feature should be intuitive and placed in a noticeable place on the page so that users can start the feature within 10 seconds after launching the application.

3.3.6 Users should be guided to the history page after a meal calories calculation.

3.3.7 The subscription service should be navigated from anywhere in the application within 3 minutes and making payment procedures must not take the user more than 5 minutes.

3.3.8 Users should be able to confirm, edit, or cancel the order prior to completing the checkout stage.

3.4 Interface requirements

3.4.1 System interfaces

The application should be compatible with the two most common mobile platforms: Android and iOS. The front-end needs to ensure the application is responsive, meaning that it will work properly on all sizes of devices. One approach to implement the front-end is to use Swift for iOS and using Kotlin for Android. Another way is using Flutter or Xamarin to do a cross-platform development, which supports both iOS and Android, providing a faster and cheaper choice. Back-end implementation requires a tool that is compatible with both operating systems.

3.4.2 User interfaces

The user interfaces should be easy and pleasant to operate, having a good appearance that matches its functionality and topic. Here are the logical characteristics for each interface between the product and users:

a) Welcome page

The start page, showing the username. Customers can enter the application by clicking the “GO” icon.

b) Main page

Offers the “Calorie Camera” function. Customers can click the camera icon to enter the upload food page. There is also an entry to the account page.

c) Upload Food page

Customers could take photos or select from local albums to upload images of food. If no food is scanned in the photo, a pop-up window will inform the users to choose another picture. After the images are verified successfully by the embedded scanner, the application guides users to the result page.

d) Result page

Display the weight and calories of the food and each food ingredient. Customers can click each ingredient to check the introduction and nutrition facts in more detail. If customers save the result, the application will automatically move to the history page.

e) History page

Show all the user calculation history by date and time. Customers could change the name of the food or move it to another date. The “View More” button allows users to see detailed information. For easy viewing, the daily records can be collapsed or expanded by clicking the arrow icon. Returning from this page will guide users to the account page.

f) Account page

Display the username and user status. Customers can enter their height, weight, and age to calculate their daily calorie intake. After the calculation, the recommended daily calorie intake will show on this page. There are also entries to the history page and subscribe page.

g) Subscribe page

List all the subscription plans available to customers, showing the time duration and price of each package. After the users select the plan and click the arrow icon, it jumps to the payment page.

h) Payment page

List all the possible payment methods. After the users select the payment method and subscribe successfully, there will be a crown symbol attached to the user avatar, a subscription expiration time under the username on the account page to indicate the updated user status.

3.4.3 Hardware interfaces

a) Android or iOS mobile phone.

b) An internet connection to allow the data to be saved and transmitted.

3.4.4 Software interfaces

a) Data management system

Amazon Aurora is a MySQL and PostgreSQL compatible relational database built for the cloud. It delivers up to five times faster than a standard MySQL database and up to three times faster than a standard PostgreSQL database. It provides the security, availability, and reliability of a commercial database at 1/10th of the cost.

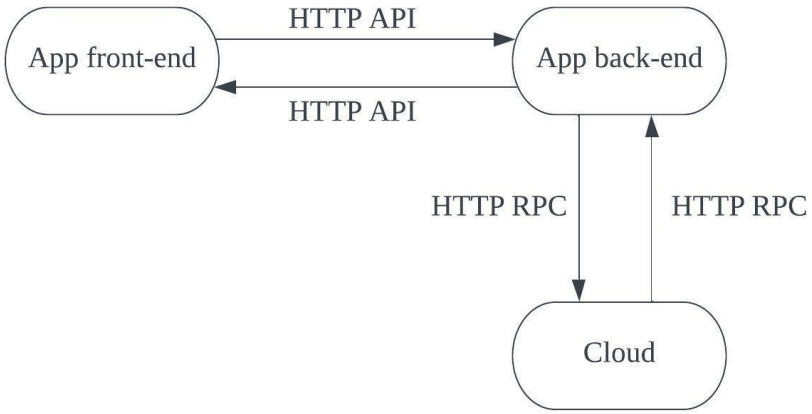
Mnemonic: Amazon Aurora
Version number: Aurora MySQL version 2.07.
Source:aws.amazon.com

b) Operating system
For mobile devices: iOS (version 14 or above), Android (version 10 or above)

c) API management
APIs can access data and functionality from your back-end services. Amazon API Gateway is a fully managed service that helps easily create, publish, maintain, monitor, and secure APIs of any size. It has no minimum fees or start-up costs. You only pay for the API calls you receive and the amount of data you pass out.
Mnemonic: API Gateway
Version number: Amazon API Gateway Version 1
Source:aws.amazon.com

3.4.5 Communications interfaces

Frontend and backend communicate with each other via HTTP requests. Backend and cloud communicate through RPC service.



3.5 Logical database requirements

3.5.1 Types of information used by various functions

a) Login: [username] and return a handle for current use.
b) Upload food: [user handle, food image, food name]
c) Result: [user handle, food id, food result, list of ingredients, list of ingredient result]
d) History: [user handle, date, time, lists of results]
e) Subscribe: [user handle, subscription plan, payment]

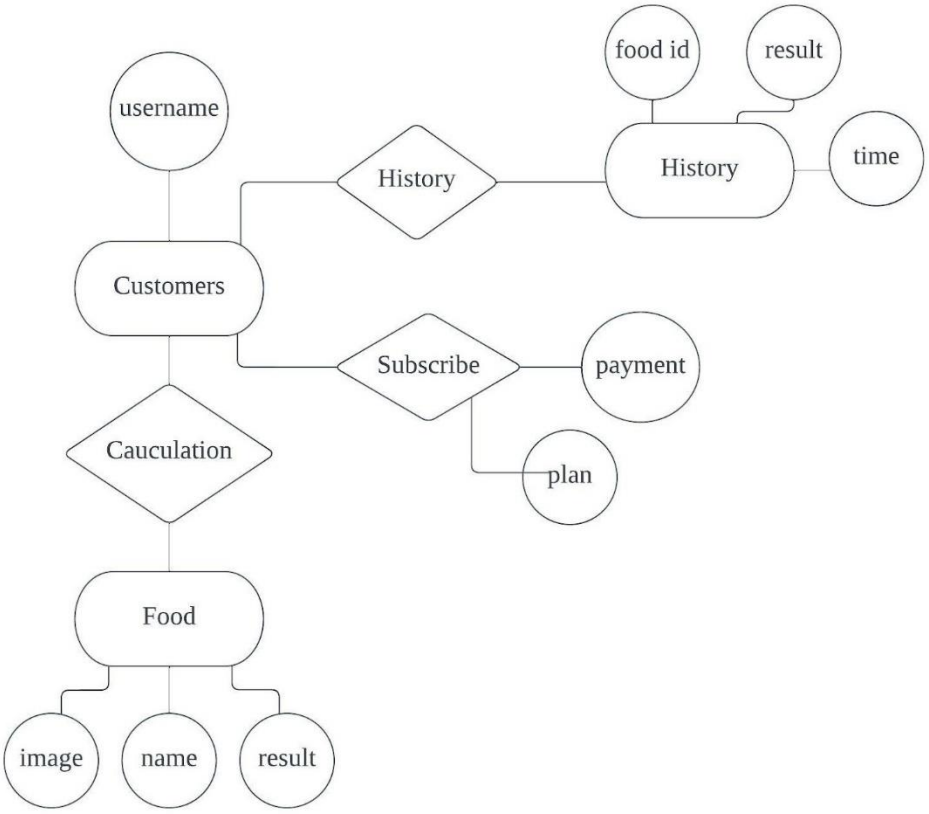
3.5.2 Frequency of use

The database and cloud services can concurrently process millions of levels of input data elements per day.

3.5.3 Accessing capabilities

Depositing data in a public access data repository. Build a ladder-shaped authority management system. To access the higher priority level of the database, requires the permission of database maintainers.

3.5.4 Data entities and their relationships



3.5.5 Integrity constraints

a) All columns have a not null constraint that forces a field to always have a value.

b) Each food in the Food table has a unique key name.

c) History has a foreign key food_id that refers to Calculation to ensure consistency.

3.5.6 Security

Use the ladder-shaped authority management system for the security classification of the data. All sensitive data are always sensitive.

3.5.7 Data retention requirements

The data retention period needs to be at least three years. Save energy by finding a good home on tape for infrequently accessed historical data. Use automated systems to securely delete data that no longer falls within the required data retention period.

3.6 Design constraints

3.6.1 Project limitations

a) Budget

The employer sets a certain budget. It may be necessary to use some inventory to reduce costs. We need to explore ways to ensure that delivering a finished product that meets the goals while keeping the budget in mind.

b) Style guide

Digital Health Inc. has provided some similar services before. They may have their own brand style. Our design needs to match the identity of the brand.

3.6.2 Device specification

For our mobile application, we need to think about how to make the product or feature as simple and user-friendly as possible, so that users can use it anytime, anywhere. There is a need to understand why users choose a particular device and tailor the experience to their needs in those situations.

3.6.3 Compliance

a) The data naming rules should follow hump nomenclature.

b) The coding should be clear, readable, efficient, and have an appropriate number of comments.

3.7 Software system properties

3.7.1 Reliability

a) Size and complexity of the code

The project source code, including the front-end, back-end, and algorithmic models, should be of significant complexity and size to allow consideration of each situation and aspect of the edge case in the logical design, thereby enhancing reliability and robustness.

b) Error avoidance and detection

Every possibility of error should be avoided as much as possible during the software development process. A typical one is input validation, such as entering user information, importing images, etc. Some errors may still exist in the software rather than using the best methods to avoid them. Therefore, in order to obtain highly reliable software, every error should be detected. The error detection process is done in the form of reliable tests. The whole project should include unit tests for each functional module and integration tests for the whole process.

c) Fault tolerance

Fault tolerance means giving the required correct results in spite of system failures. For our large projects, it is almost impossible to be error free. Therefore, we need to set up fault-tolerant methods to accept a certain level of errors while the process still guarantees correct output from the client. Typical methods are retry mechanisms, recovery blocks, and rollback recovery.

d) Accuracy Improvement

The core part of our project is the trained AI model. Developers should keep in mind to improve accuracy, but avoid problems such as overfitting that may lead to low actual prediction accuracy.

3.7.2 Usability

a) Checkpoints

After the current commit passes the main tests, the development process should have several checkpoints as stage markers in some of the main development branches, also in the data management system. In case of failure, the last checkpoint will be used as a recovery point.

b) Recovery and restart

For some unpredictable situations or errors that occur, we use recovery logic models as part of the software design, such as hot rollback, hot import, hot restart, so that the main service on the client side is to fix or update the system without stopping.

3.7.3 Security

a) Use of certain encryption techniques

We will use certain encryption techniques to secure the transmission of messages in the network to avoid intermediate attacks. We start by using https as the network protocol. For plain text messages like JSON files, we will use SHA-256 for encoding. For the image part, we will use MD5 encoding as the image identifier and transmit it gently.

b) Retaining specific log or historical data sets

We will set up a system logging module for the distribution system in the cloud service and for our clients. Log output files should be categorized and placed in a database for future error tracking.

c) Assigning certain functions to different modules

We import sub-modules into our project design. For the current project, we can only use the current imported submodules of the specified version (tag/commit). The own development process is only visible in the repository of the submodule project. This creates an effective barrier between different modules or projects. For DHL, we will use the AI model project containing the functionality as a submodule of the main DHL project with front-end and back-end.

d) Ensuring data privacy

A stepped permission management system that assumes a trade-off of access capabilities and provides different access needs for different people

3.7.4 Portability

a) Front-end

There are two approaches here: the first approach is non-portable, for IOS you implement the front-end using Objective C or Swift, for Android you implement the front-end using Kotlin; the second approach is portable, for both systems you implement the front-end using Flutter, which may reduce development costs and time.

b) Backend

The back-end programming language is definitely portable. Considering speed and memory safety, candidate portable languages are C++, Rust.

3.8 Support Information

3.8.1 Sample input/output formats

Here we consider only the main input/output formats of the terminal.

a) Food detection

For example

Input

{user_handler:100, fine_name:today_lunch, image_path: [/user/local/album/foo1.jpg, /user/local/album/food2.jpg]}

Output

{user_handler:100, metal_id:0x213abd, component_view:tomato}

b) History.

For example

Input

{user_handler:100, time_period:d (days)}

Output

{user_handler:100, time_period:d(days), monthly list.

[{meal_number:0x12345, data:2022-03-12, food_name:wrap}, {meal_number:0x12345, data:2022-03-12, food_name:Cheeseburger}

{Meal number: 0x12346, data: 2022-03-12, food name: salad}]

}

3.8.2 Problem description

From the user's point of view, the DHL software will address the problem of quantifying the user's food intake and provide some statistics about the user's food intake history with suggestions or feedback.

From the software developer's point of view, the software needs to address communication between tools in different programming languages, between the client and the server, and between the server and the AI model part. Usually, we use the FFI package for language binding; in addition, the software needs to address the accuracy of the AI model with appropriate training data sets and proper parameter tuning to achieve high accuracy.

4. Verification

4.1 Functions verification

4.1.1 Validity checks on the inputs

The AI algorithm should scan the uploaded image and verify whether or not it contains any visually detectable ingredients. If AI marks the image as invalid, the system should notify users of error messages.

4.1.2 Exact sequence of operations

The backend should record requests from the user side and save them as the sequence of logs to send to the server. After the requests are responded to, the backend should return the result back to the user side.

4.1.3 Response to abdominal situations

Once an abdominal situation is detected, the system should then generate a warning message and send it to users. Moreover, if users attempt to bypass rules to do something bad maliciously, users accounts will be blocked, and will not allow using any service.

4.1.4 Effect of parameters

When users launch the application for the first time, the system will detect phone models and run the internet speed test automatically in the back-end. The system will notify users with an estimated time for using features. If there are some abnormal conditions, the system will also remind users with a pop-up warning window. The estimated upload time and calculation time will be displayed on the screen as well.

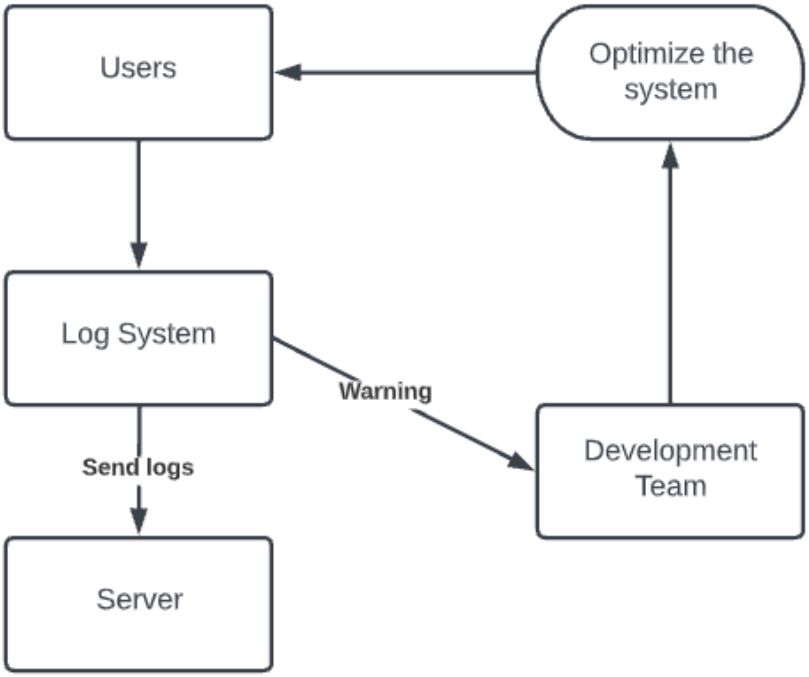
4.2 Performance verification

4.2.1 The system should have a backend log system to hold the time measurement for each operation. The log history should be sent to the backend server.

4.2.2 The performance testing should have acceptable performance criteria for each operation. If the operation takes a longer time than expected, the log system should report a warning to the development team.

4.2.3 The log system should generate a large volume of time data from internal testing, alpha, and beta testing. These records should be used to judge the performance of the system.

The overall performance verification procedures can be shown as follow:



4.3 Usability verification

4.3.1 Internal testing

The software engineers will test the usability in the development stage. Then, the QA group will validate the process to make sure they all match usability requirements. They all need to measure the time taken to send requests and get responses.

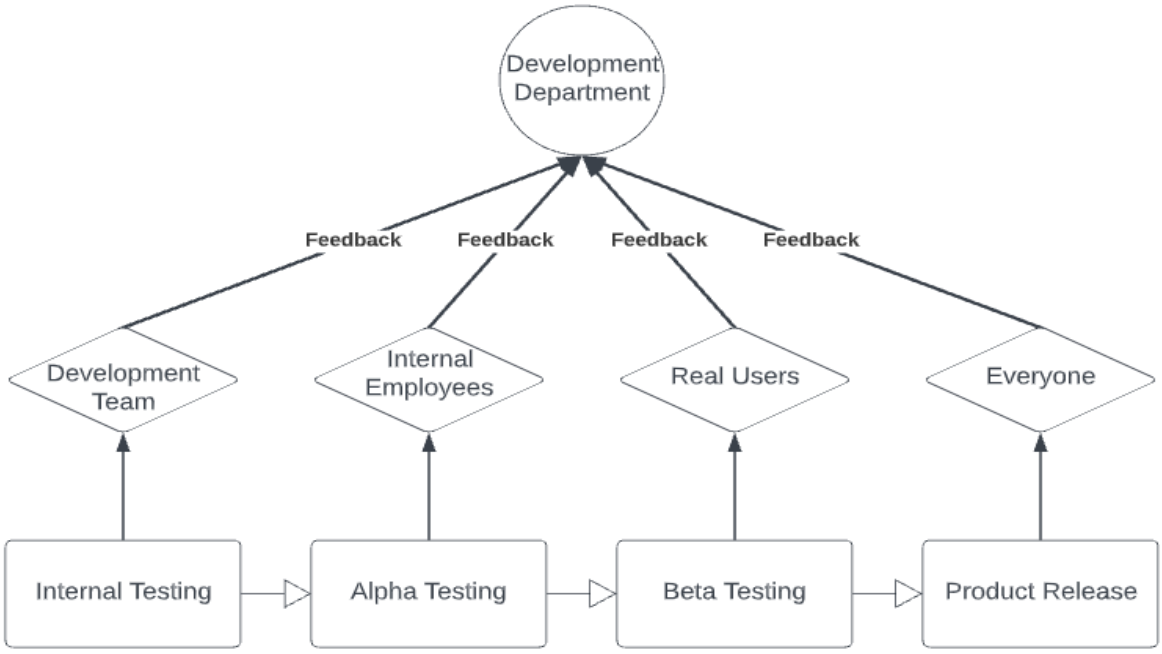
4.3.2 Alpha testing

Alpha testing is carried out by the testers who are internal employees of the organization to identify all possible issues and bugs before releasing the final product to the end-users. They can be given questionnaires or surveys to reflect users' experiences. Then, the development team can adjust the product according to feedback.

4.3.3 Beta testing

The organizations should invite “real users” of the software application to perform the test. Direct feedback from customers should be the guidance for improvement before shipping a product to the customers.

The overall usability verification procedures can be shown as follow:



4.4 Interface verification

4.4.1 System interfaces

Install and run the application on two operating systems: iOS and Android. To check whether the application is compatible with the two platforms.

4.4.2 User interfaces

a) Welcome page

Verify if the login of the user based on the phone is successful. In the software phase, the system should return the user handle.

b) Main page

Verify if the user could enter the upload food page and account page.

c) Upload Food page

Verify if the user could upload images from either album or taking photos. Verify if the application gives an error message when the user uploads a photo that is not food. After clicking the “Use Photo” button, check the waiting time for the results and check whether the results are valid.

d) Result page

Verify the consistency of the total calories of the food with the sum of the calories of each ingredient. Check whether there are ingredients or details that have been left out. After clicking the “SAVE” button, verify if the application guides to the history page.

e) History page

Verify if each result is under a date category. Check whether the user can edit the history successfully. Verify if the user could enter the account page.

f) Account page

Verify if this page displays the username and user status correctly. Check if the daily calorie calculator is available. After entering data and clicking the “Calculate” button, check if the daily calorie is displayed correctly. Verify if the user could move back to the history page and enter the subscribe page.

g) Subscribe page

Verify if the subscription plans are displayed with the time duration and price. After choosing the plan, check if the application leads to the payment page.

h) Payment page

Verify if the user could pay successfully. Check the updated user status after subscription. Check whether the prime functionality is activated.

4.4.3 Hardware interfaces

Verify the mobile phone and internet are available.

4.4.4 Software interfaces

a) Data management system

Run a small demo that has Amazon Aurora as its data management system. Test with databases of different scales to check its ability.

b) Operating system

The same approaches as 4.4.1

c) API management

Run a small demo that has Amazon API Gateway as its API management tool. Test to check its ability.

4.4.5 Communications interfaces

Verify the communication methods by large files that reach the set limit. Check the transmission time and transmission accuracy.

4.5 Logical database verification

4.5.1 Types of information used by various functions

Evaluate each type of information. Verify there are no missing types or unreasonable structures.

4.5.2 Frequency of use

Analyze system performance by increasing the amount of data in the database. Test to determine the behavior of a database when accessed by multiple users at the same time.

4.5.3 Accessing capabilities

Verify the level assignments for each database are reasonable. Verify that all levels of access are valid. For example, whether higher levels require approval from database maintainers before they can be accessed.

4.5.4 Data entities and their relationship

Verify there is no missing data element for each table. Verify the reasonability of each relationship.

4.5.5 Integrity constraints

Check the schema, tables and triggers of the database by conducting database testing. Verify the integrity and consistency of the data. Ensure that the data values and information received and stored in the database are valid. Re-evaluate the current constraints in the real situation. Update the integrity constraints if needed.

4.5.6 Security & Data retention

Perform security testing to identify system vulnerabilities, threats and risks and prevent malicious attacks by intruders.

4.5.7 Data retention requirements

Use stress testing to assess robustness and error handling under severe load conditions.

4.6 Design constrains verification

4.6.1 Project limitations

a) Budget

Plan carefully and check frequently to ensure stay within the budget.

b) Style guide

Verify the style of the application follows a style guide for the brand that constrains the colors and layout of designs.

4.6.2 Device specification

The same approaches as 4.4.1. Also, check the usability of the application on different devices.

4.6.3 Compliance

Do the code review periodically to check the code style.

4.7 Software System Attributes Verification

4.7.1 Reliability

a) The QA shall validate various valid inputs in all input boxes of the App to check if invalid inputs can be validated after the input is completed, some invalid inputs include:

i) Not the required input type
ii) Inputs that exceed the maximum length
iii) Inputs that may trigger information leakage in the backend

b) Refer to the verification method in 4.5.6 for fault tolerance

c) For AI models, our AI development team should periodically make predictions on AI models using test image sets collected from daily use and then try to improve the algorithm from the prediction results.

4.7.2 Usability

a) In the development environment, trigger pre-designed database crash events to verify rollback, recovery, backup, and restart functionality.

4.7.3 Security

a) Perform unit tests for each encryption technique including SHA-256, OTP, and MD5, noting edge cases. Pay attention to each encoding and decoding part.

b) Validate the distribution logging system by the following metrics.

i) Each sub-logging system should be consistently synchronized with other sub-logging systems without loss of information
ii) Each logging system should have levels to indicate various messages, such as errors, warnings, debugging, messages, traces.

c) Assign certain functions to different modules. Verify it through regular code review meetings.

4.7.4 Portability

If we do a non-portable approach, we can refer to 4.4.1. If we do a portable approach, we should validate it against the approach in 4.4.1 and the performance compared to the non-portable approach.

4.8 Supporting information verification

4.8.1 Example input/output formats

This should be done on the backend, validated in the final application, and all valid inputs and outputs follow the correct output format in this predefined example specification.

4.8.2 FFI

After selecting the programming language set, we will verify that the FFI package works by creating a demo. This demo should consider common FFI issues.

a) Variable type mapping, especially null types.
b) Memory allocation and release.
c) FFI communication interface design.

4.8.3 Special Packaging Instructions

We will validate the organization's packaging instructions with a packaging demo project and a checkpoint project to see the results. All processes of packaging, including pipeline inspection, packaging and labeling, shall be notified at the final release.

5. Appendix

AI - Artificial intelligence is the ability of a computer or a robot controlled by a computer to do tasks that are usually done by humans

AWS - Amazon Web Services, provides servers, storage, networking, remote computing, email, mobile development, and security.

iOS - iPhone Operating System, is a Unix-derived operating system powering all of Apple's mobile devices.

Inc. – incorporated, a company's business structure is a legal corporation.