COMP4801 Final Year Project
An online food catalogue based on open crowdsourcing
(Group 1)

Supervised by Dr. T. W. Chim

FYP17024 Final Report

By: Man Ting-Chun Matthew (3035178765)
Another group member: Yeung Ching Chun (3035177199)

Date of submission: 15/4/2018
Abstract

Currently, many smartphone applications provide crowdsourced restaurant reviews functionality. Users can know more about the restaurants based on the descriptions and reviews provided by other users before visiting one of the restaurants. However, a review process is needed for listing the restaurant on the ranking interface. Moreover, these applications rely on menu images on their searching lists, which causes ambiguity if people do not understand the language on menus. Low resolutions of photos and small font sizes bring further inconvenience when searching for specific kind of food. Apart from that, existing applications provide few accessibility features, which discourage users with disabilities. Another problem is that most of these applications provide similar functionalities, which are searching for restaurants by text and posting reviews of restaurants.

An online food catalogue based on open crowdsourcing is developed as the Final Year Project for solving the aforementioned issues. Instead of focusing on restaurants, this project mainly focuses on the food items. Food items can be added by typing in the information, or by uploading images of menus for Optical Character Recognition (OCR) analysis by using Google Vision Application Programming Interface (API), which can convert text in images into editable text. To facilitate searching, food items are stored as editable text format, and translation is applied to the inputted text, thus information of that specific food in different languages can be shown. An accessibility mode is also developed for users with visual or physical impairments, which supports scanning of food menus, variable font size, text-to-speech conversion, and speech-to-text conversion. New functions including searching for food by image, which is implemented in an Augmented Reality (AR) interface, and searching for a certain number of food items within a specific budget are also implemented in order to help users in a new way.

The deliverables for this project include an iPhone Operating System (iOS) application written in Swift, a web server developed with Node.js as well as a MySQL database, with the server and database deployed on a cloud computing platform, Amazon Web Services (AWS).
Acknowledgement

We would like to express our deepest appreciation for those individuals and organizations, who have kindly offer their help and support to this project.

We are highly indebted to our supervisor Dr. T. W. Chim for his opinions and guidance throughout the project. He has always spared time for our meetings and has given a lot of concrete feedbacks. We could not have achieved this much without his support.

We would also like to express our gratitude towards the Department of Computer Science, Faculty of Engineering, The University of Hong Kong for giving us the valuable opportunity for doing this project.
Table of Contents

Abstract 2
Acknowledgement 3
List of Figures 6
List of Tables 8
Abbreviations 8
1. Introduction 9
2. Objective 12
3. Vision and Scope 13
  3.1 Features 14
  3.2 Software Limitations 17
4. Methodology 18
  4.1 System Architecture Design 19
    4.1.1 Presentation Tier 20
    4.1.2 Logic Tier 20
    4.1.3 Data Tier 21
    4.1.4 Table design 22
    4.1.5 Cloud Computing 24
  4.2 Function Design 25
    4.2.1 Adding & Uploading Food Records 25
    4.2.2 Uploading Menus for OCR Analysis 27
    4.2.3 Searching for Food by Text 29
    4.2.4 Searching for Food by Image 30
    4.2.5 Searching for Specific Amount of Food Items Closest to a Fixed Budget 31
    4.2.6 Login/Register 33
  4.3 Task Schedule 35
5. User Interface 36
  5.1 Adding & Uploading Food Records 36
    5.1.1 Home View of locally stored food records 36
    5.1.2 Input View of food records 38
    5.1.3 Location Search View 39
  5.2 Function Design 41
    5.2.1 Adding & Uploading Food Records 41
    5.2.2 Uploading Menus for OCR Analysis 43
    5.2.3 Searching for Food by Text 45
    5.2.4 Searching for Food by Image 46
    5.2.5 Searching for Specific Amount of Food Items Closest to a Fixed Budget 47
    5.2.6 Login/Register 49
  5.3 Task Schedule 51
6. Conclusion 52
7. References 53
8. Appendix 55

List of Figures

Figure 1  The iterative cycle 18
Figure 2  Database design figure using Crow Foot Notation 22
Figure 3  System flow of Uploading Food Record(s) 25
Figure 4  System flow of Uploading Picture for OCR Analysis 27
Figure 5  System flow of Searching for Food by Different Languages 29
Figure 6  System flow of Searching for Food by Image 30
Figure 7  System flow of Searching for Specific Amount of Food Closest to a Fixed Budget 31
Figure 8  System flow of login/register 33
Figure 9  User interfaces of Home View 36
Figure 10  User interfaces of Input View 38
Figure 11  User interfaces of Location Search View 39
Figure 12  User interfaces of OCR Result View 40
Figure 13  User interfaces of Rank View 42
Figure 14  User interfaces of Food Detail View 43
Figure 15  User interface of Navigation View 44
Figure 16  User Interfaces of Filter View 45
Figure 17  User interfaces of Searching with Translation 46
<table>
<thead>
<tr>
<th>Figure 18</th>
<th>User interfaces of Searching by Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 19</td>
<td>User interfaces of Login/Register</td>
</tr>
<tr>
<td>Figure 20</td>
<td>User interfaces of User Profile</td>
</tr>
<tr>
<td>Figure 21</td>
<td>User Interface of Friendly Mode Rank View</td>
</tr>
<tr>
<td>Figure 22</td>
<td>User Interface of Friendly Mode Food Detail View</td>
</tr>
<tr>
<td>Figure 23</td>
<td>User Interface of Friendly Mode OCR Result View</td>
</tr>
</tbody>
</table>
List of Tables

Table 1  Features for this project  14-16
Table 2  Limitations for this project  17
Table 3  Schedule for this project  35

Abbreviations

AI  Artificial Intelligence
API  Application Programming Interface
AWS  Amazon Web Service
iOS  iPhone Operating System
JPEG  Joint Photographic Experts Group
OCR  Optical Character Recognition
S3  Simple Storage Service
SHA-512  Secure Hash Algorithm 512
SMS  Short Message Service
SNS  Simple Notification Service
URL  Uniform Resource Locator
1. Introduction

In the past, people frequently had difficulty in choosing restaurants because information about restaurants was hard to access. Due to advancement in mobile technology, information relating to restaurants is easily accessible by the public with the use of smartphone applications such as OpenRice [1] and Yelp [2]. These applications can be categorized as crowdsourcing applications. Crowdsourcing is defined as a kind of activity that an individual or an organization proposes a task, and it allows the crowd to participate voluntarily by providing knowledge, experience or other resources [3]. For applications like OpenRice and Yelp, reviewing restaurants is the proposed task for the public, and users (posters) are enabled to upload restaurants reviews, ratings or photos through the applications. Afterwards, other users (viewers) can read those contents to decide which restaurant to visit.

Some of these applications categorize photos for the restaurants into either menu photos, or general photos, such as food and environment, for users to check the menus more efficiently [4]-[5]. Being able to read the menus is crucial to users as they can get a preview of the food in advance and thus choose a suitable restaurants. However, providing mere pictures of the menus may be insufficient since the quality of images is not unified and images with low quality greatly reduce the readability of users. Indeed, existing applications provide little support for users with different background (e.g. users with different mother tongues) or with different needs (e.g. users with visual impairments). Those applications simply categorize photos with menus and assume that users find the menus useful – but those applications can go further.

Only providing pictures of food catalogues may be undesirable for certain users, in terms of language and the ability to search for food items, or for users with disabilities. First, the language of menus in the pictures may not be understood by all users. If menus in different languages are provided separately but users only upload the Chinese version, the content cannot be understood by some foreign users. Second, the ability to search for food items is limited because the menus are in picture format, instead of editable text format that can be searched by typing in characters and words. If only a few menu photos for a restaurant are uploaded, the restaurant cannot be
found by searching for food name because only the pictures are added to the restaurant. As for users with impaired visions, it can be hard for them to see the food catalogues easily, whether it is in physical format or digital image format. This may be due to the small font size, or no special design to cater for users with impaired visions.

The aforementioned problems can be solved by Optical Character Recognition (OCR) technology. With OCR, printed or written text in images can be converted into editable text [6], allowing the text to be further processed, by translating the scanned text into multiple languages or searching for the scanned text. Furthermore, the digital text can have variable font size based on user’s preference, and it can also be spoken out using speech-to-text function, which helps users with impaired vision to explore the food menus.

Apart from the problem of accessing digitized food records, users face another problem when they are inside the restaurant, which is choosing the suitable food items. Besides going through the food menu, users may often spot an attractive dish ordered by the next table, and then they want to order the same one. Since some food items may have no photos in the physical food menu, the users do not know the name of the dish, and they cannot order it. Even with all the food images in existing applications, it can be hard for users to go through all the images belonging to the same restaurant one by one until they find the one they want.

Image search can be used to solve the above problem. If food images are associated with food information, image search can be applied to a food image to find a best-matched food image that is in the database. Users can then find the food detail by taking a picture of the food they want to order.

Another problem that people often face is about creating a plan of food to order within a budget, which can be applied to one meal or multiple meals. For a single meal, the user may want to order a few food items in the restaurant within a budget. As for multiple meals, the user may want to spend a total budget for an amount of meals, and each meal consists of one food item. To
find an amount of food items within a certain budget, it becomes a form of the Knapsack problem, which is an NP-complete problem [7], meaning that there is no fast solution to such problem.

By using dynamic programming [8], the Knapsack problem can be solved effectively, allowing food records that are within a budget to be found easily. Therefore it is possible to help users to plan their meals by utilizing such algorithm.

This final report introduces a new smartphone application that integrates OCR to provide online food catalogues based on open crowdsourcing. The focus of the application is on the food items, unlike existing applications that focus on restaurants. Users can upload photos of food menus, then the images are converted into text for generating online food catalogues for restaurants. Searching for food information using any language is also possible, and all records with matching information can be found. Apart from searching by text, the application includes image comparison technology, which enables users to search for food information by uploading an image of the food item. Users can also input a certain amount of meals and a total budget, and then the application can select that number of food items that is closest to the budget. For users with visual or physical impairments, a friendly mode that incorporates accessibility features is developed to cater for their needs.

The remainder of this report proceeds as follows. First, the objective and the scope of the project are given. Next, the design strategies and methodology on how to implement this application are introduced. Afterwards, the user interface of the application, as well as the challenges encountered are described. Then, the user acceptance test results of the final product are provided. Lastly, this report closes with a conclusion and references.
2. Objective

The goal of this project is to develop a new smartphone application that benefits both restaurants and users. Comparing with existing applications that focus on crowdsourcing restaurants review, this application focuses on crowdsourcing food items, which provides features that benefit both restaurants and users.

For restaurants, online food catalogues can be generated easily using this application, by simply taking pictures of their existing food menus. Once the picture is scanned by OCR and the food records are uploaded, all food items shown on the menus are available for searching. In the past, only food items with reviews from customers were available for searching. This application helps restaurants to promote all food items, instead of limited to the most renowned ones that other users reviewed.

For users, other than searching for ranking of food and restaurants, searching for food or restaurant in any languages is also possible in this application. Translation is applied to the searching keyword that the user input, hence identical food name or restaurant name in different languages can be searched simultaneously. Instead of waiting for reviews on the specific food items, users can search for any food item available on menus once they are uploaded. Apart from searching for food by typing, searching for food by images is also possible. A meal planning function is available for users to input a certain amount of food and a total budget, and then that amount of food records can be found within the user’s budget. To cater for users with disabilities, a friendly mode is also included, which supports menu scan, variable font size, speech-to-text conversion and text-to-speech conversion.

For us, the developers, this project allows us to explore opportunities to collaborate with existing companies offering crowdsourcing applications. We can use this project to demonstrate how the new functions may benefit users, and then we can collaborate with other companies in order to incorporate more functions into our application or their application.
3. Vision and Scope

This project aims to develop a new application which allows users to upload food catalogue of restaurants and search for food more efficiently. Users upload food records instead of restaurant records. For simplicity, only English and Chinese are supported for OCR Analysis at the current stage.

Unlike existing applications that focus on restaurant reviews, this application focuses on food items. Instead of using photos to show food menus, this application uses editable text to store the food records. As a result, it is more efficient for users to look for menus in multiple languages or searching. Consequently, this application facilitates users in searching for not only restaurants, but also food. Since food images are associated with food records, searching for food by image is also possible. The application also incorporates a meal planning function which aims to help users in a new way. An accessibility mode is also developed to cater for the needs of users with visual or physical impairments. Below are the features and limitations of the application.
### 3.1 Features

Below is the table showing the features available to the users (see Table 1). The features are divided into four types: Uploading, Searching, Friendly Mode, and Login/Register. Each type consists of one or more features. The detailed implementations are illustrated in the later section 4.2 Function Design.

<table>
<thead>
<tr>
<th>Type</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uploading</strong></td>
<td>Upload food menu for OCR Analysis</td>
<td>Food menu photo is uploaded and stored in the server. The photo is then passed to the Google Vision API for OCR Analysis. Scanned food records consist of food names and their corresponding prices are returned to the user’s application afterwards. Users can view their uploaded menu photos in the user profile view.</td>
</tr>
<tr>
<td></td>
<td>Edit information of food items</td>
<td>After uploading menu for OCR Analysis, food records are returned to the user’s application. The information of each of the records can be edited by the user. Food name, food price, restaurant name and restaurant location are the compulsory fields of a food record.</td>
</tr>
<tr>
<td></td>
<td>Give tags to food items</td>
<td>Tags can be given to enrich the searchability of the food.</td>
</tr>
<tr>
<td></td>
<td>Give ratings to food items</td>
<td>Rating from 1 to 5 can be given for each food.</td>
</tr>
<tr>
<td></td>
<td>Upload food pictures</td>
<td>Food picture can be attached for each of the food record. Users can view their uploaded photos in the user profile view.</td>
</tr>
<tr>
<td><strong>Searching</strong></td>
<td>Search for food by name/tag</td>
<td>Keyword can be entered to search for the corresponding food.</td>
</tr>
<tr>
<td>Search for food by restaurant name/location</td>
<td>Keyword can be entered to search for the corresponding food records of the restaurant. Apple Map showing the estimated route, walking distance and time is provided.</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Search for food by price range</td>
<td>Range of price can be adjusted to search for the corresponding food.</td>
<td></td>
</tr>
<tr>
<td>Search for specific amount of meals closest to a fixed budget (Knapsack Problem)</td>
<td>Given a specific amount of meals and a specific budget, the application find the specific amount of meals so that the total price is closest to the specific price.</td>
<td></td>
</tr>
<tr>
<td>Search for food by image (AR)</td>
<td>Users can take a photo and find out what the food is in a specific restaurant. The food name and price are be displayed in the form of AR Text.</td>
<td></td>
</tr>
</tbody>
</table>

**Friendly Mode**

<p>| Search for food by name/tag               | Keyword can be entered to search for the corresponding food. |
| Search for food by restaurant name        | Keyword can be entered to search for the corresponding food of the restaurant. Apple Map showing the estimated time and route is provided. |
| Adjust Word Size                          | Users can adjust the word size of words in the view. |
| Speech-to-Text                            | Searching keyword can be inputted by speech recognition, the keyword can then be used to search for food by name/tag/restaurant name |</p>
<table>
<thead>
<tr>
<th>Text-to-Speech</th>
<th>Synthesized audio can be outputted for each food record and for each OCR record, generating audio for the food information.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Login/Register</td>
<td>Login/Register &amp; Verify for an account</td>
</tr>
</tbody>
</table>

*Table 1. Features for this project.*
### 3.2 Software Limitations

Below is the table showing the limitations of the APIs, the web services used and the front-end application (see Table 2). These limitations define the bottleneck of the functionalities.

<table>
<thead>
<tr>
<th>Limitation Aspect</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>API</strong></td>
<td></td>
</tr>
<tr>
<td>Picture Dimension for OCR Analysis</td>
<td>Picture size should be minimum of 640 x 480 pixels, and recommended to be of 1024 x 768 pixels [9]</td>
</tr>
<tr>
<td>Picture Size for OCR Analysis</td>
<td>Maximum of 4 MB is supported by Google Vision API [9]</td>
</tr>
<tr>
<td>File Format for OCR Analysis</td>
<td>Only JPEG, PNG8, PNG24, GIF, Animated GIF (first frame), BMP, WEBP, RAW, ICO are supported [9]</td>
</tr>
<tr>
<td>Text Languages for OCR Analysis</td>
<td>56 Languages are supported [10]</td>
</tr>
<tr>
<td>Text Languages for Google Translate API</td>
<td>104 Languages are supported [11]</td>
</tr>
<tr>
<td><strong>Web Services</strong></td>
<td></td>
</tr>
<tr>
<td>Picture(s) Storage for each user</td>
<td>Maximum storage size of 5GB is supported by AWS Simple Storage Service (S3) [12]</td>
</tr>
<tr>
<td><strong>App</strong></td>
<td></td>
</tr>
<tr>
<td>Language for Food Records</td>
<td>Chinese and English [13]</td>
</tr>
<tr>
<td>Language for Text to Speech</td>
<td>Chinese and English [13]</td>
</tr>
<tr>
<td>Language for Speech Recognition</td>
<td>English</td>
</tr>
<tr>
<td>Maximum Budget entered in the search filter</td>
<td>$100000</td>
</tr>
</tbody>
</table>

*Table 2. Limitations for this project.*
4. Methodology

For this project, an iterative and incremental approach has been used, and repeated iterative cycles have been carried out (see Figure 1). In each iteration, part of the system is developed, and the system is a functional product on its own to prove the iteration is completed. Apart from providing a working deliverable, each iteration builds on top of the previous iteration such that the system grows incrementally.

The iterative and incremental approach has been used instead of some traditional approaches such as the Waterfall model. The major benefit of the iterative and incremental approach is that early feedback can be obtained from stakeholders, such as the supervisor or potential users. Stakeholder’s requirements can be better understood with those feedback, allowing modification to be made to the system in the next iteration to fulfil those requirements. The traditional Waterfall model contains only one iteration of the iterative and incremental approach [15], and the product cannot be seen by the stakeholders before the implementation phase, therefore it is more likely that the their expectations cannot be met by the system.

Figure 1. The iterative cycle [14].
4.1 System Architecture Design

For this project, a three-tier architecture has been used. Multi-tier architecture (also known as N-tier architecture) means that the system is divided into multiple parts physically and logically, and each part is responsible for a specific type of tasks [16]. The traditional basic client-server architecture is regarded as a two-tier architecture, which consists a client tier for user interface and data processing, and a data tier for data storage and retrieval [17]. In a typical three-tier architecture, the tiers are the presentation tier, the logic tier and the data tier [18]. The presentation tier consists of the user interface and some operations to present meaningful result to users, the logic tier is for processing of data or calculation, and the data tier is for data storage and retrieval.

A three-tier architecture has been used in this system because it has better scalability and security than a two-tier architecture [19]. Scalability is defined as the ability to support increasing amount of work [20]. In a three-tier architecture system, more resources can be easily allocated to the logic tier, allowing the overall system to have better performance. Security is stronger because data access can only be performed through the server but not directly by the users.

The three-tier architecture is like a bank queueing system. The presentation tier consists of all the customers with different requests, the logic tier is all the counters that serves the customers, and the data tier is the bank’s centralized database with all customer’s data. During peak hours, the queueing system may have many customers, just like many users may be using our application in certain hours. The queueing system can add more counters during peak hours, just like more resources can be added to the logic tier any time to handle more user requests. The three-tier system is secure because users cannot directly manipulate the data, which is similar to bank system that customers can only manage their account through the counters or standardized ATM machines.

The details regarding technology used in different parts of the project is discussed below.
4.1.1 Presentation Tier

For the presentation tier, an iOS application written in Swift has been developed for this system. iOS and Android are two popular smartphone operating system on the market. iOS is selected because iOS is less fragmented when comparing to Android [21]-[22], meaning that more iOS users have updated to the latest operating system. Since certain functionality may only exist in more recent operating system, iOS is easier to manage. Apart from fragmentation of operating system, less device models exist in iOS platform, which makes it easier to cater for all screen size or ratios of the models. Regarding programming language choices, Swift and Objective-C are the two major programming languages for iOS, and Swift has been used because it has better performance and it is easier to maintain than Objective-C [23].

4.1.2 Logic Tier

For the logic tier, Node.js has been used for server development. Some other popular scripting languages for servers are PHP, Django and Rails. However, Node.js has better performance and database integration than PHP, Django, Rails servers [24]. Thus, to achieve better performance, Node.js has been used as the programming language on the server side.
4.1.3 Data Tier

For the data tier, a MySQL relational database has been used. Databases can be categorized into either relational database or non-relational database (also known as NoSQL database). Relational database has been used because it has better data integrity when comparing to non-relational database due to usage of foreign key. Apart from data integrity, using relational database is also better because it creates less problems than non-relational database [25].

A MySQL relational database has been used because it is an open-source software, meaning that it can be used or modified freely [26]. Some other popular relational databases are Oracle RDBMS or Microsoft SQL Server [27]. However, license fees are required for these databases. Due to budget limitation for the project, a MySQL relational database has been used.
4.1.4 Table design

![Database design figure using Crow Foot Notation](image)

**Figure 2.** Database design figure using Crow Foot Notation [28].

Multiple tables and relationships exist in a relational database.

First, the restaurant name and the location of the restaurant are stored in the “Restaurants” table (see Figure 2). The field restaurant_id is auto-generated which is unique for each restaurant branch. Both restaurant name and location are stored such that multiple branches of the same restaurant can be identified.
Second, the user_id, phone_number and bucket_name of users are stored in the “Users” table as shown in Figure 2. The field user_id is an auto-generated number that is unique for each user, and it is used to uniquely identify a particular user. In addition to the user_id, the phone_number is the phone number used by the user for login, and the bucket_name is the image storage directory path for the user to store the photos uploaded by the user.

Third, the restaurant_id, tags, user_id, name, price, ratings and photo_filename given to the food are stored in the “Food” table in Figure 2. The field “tags” is similar to the hashtag function in Instagram [29], and it has been included for users to search for food items easily by using the “tags”. The field “photo_filename” is the image storing address for the food photo.

Fourth, the food_validation_id, food_id, isValid, and user_id are stored in the “Food_Validation” table as shown in Figure 2. The food_validation_id is an auto-generated unique number for each validation record. The food_id and user_id are foreign keys from the “Food” and “User” table for associating each validation record to a food record, and the isValid field is for checking whether the record is validate or not. This table is added to allow users to give a validation score to the food images, and the average score is presented to other user when they look at the food images(see Section 5.3.2 for more details).

Fifth, the food_label_id, food_id, description, score and user_id are stored in the “Food Label” table as appeared in Figure 2. The food_label_id is a unique number for each record. The food_id and user_id are foreign keys from the “Food” and “User” table to associate each label record to a food record. The description is a phrase that describe the Food image, while the score is a percentage about how likely the phrase is an accurate description for the image. The description phrases are stored to facilitate searching for food information by images(see Section 4.2.4 for more details).
4.1.5 Cloud Computing

Both the server and database have been deployed on the Amazon Web Services (AWS), which is a cloud computing platform [30]. A cloud computing platform is used because it can be easily managed by multiple administrators, instead of using a desktop computer as a server, which can only be operated by one member. There are many cloud computing platform existing in the market, and the most popular ones are AWS, Microsoft Azure and Google Cloud Platform [31]. AWS has been used because it provides the best documentations and most functions among the three platforms [32] - [33].
4.2 Function Design

This application consists of six major functions: 1. Adding & Uploading Food Records, 2. Uploading Menus for OCR Analysis, 3. Searching for Food By Text, 4. Searching for Food By Image, 5. Searching for Specific Amount of Food Closest to a Fixed Budget, and 6. Login/Register. In the remaining sections of this part, users’ step-by-step workflow and the corresponding back-end working components are shown. All of the six major functions are fully implemented.

4.2.1 Adding & Uploading Food Records

![Figure 3. System flow of Uploading Food Record(s).](image)

User can upload the locally-stored food record(s) from the smartphone application to the server (see Figure 3, step 1). Next, the server adds those record(s) to the database (step 2). If any food image is attached to a food record, the image is redirected and stored in the uploader’s S3 bucket (step 3). After that, the server passes the Uniform Resource Locator (URL) pointing to the photo in the S3 Storage to the Google Vision API for image analysis (Step 4) for finding food description phrases. Finally, the corresponding URL of the image, together with the description phrases, are stored in the database for later retrieval.
The image is stored after adding the related food record. It is because the image file name is generated using the food ID. This naming facilitates the debugging process since related images can be easily found by their file names. Cross-checking among database records and S3 bucket photos has become easier.

Google Vision API has been used for image recognition to find food description phrases. The food descriptions are stored in the database in order to facilitate searching for food by image (see Section 4.2.4 for more details). Another popular approach to perform image recognition is by training a deep learning model [34]. Although deep learning models should be more accurate, it requires a lot of time to train the model. Since our application is based on crowdsourcing, it is difficult to generate a model that can cater for all kinds of food. Therefore Google Vision API has been used to search for food description.
4.2.2 Uploading Menus for OCR Analysis

![Figure 4. System flow of Uploading Picture for OCR Analysis.](image)

User can upload the photo of a food menu from the application to the server (see Figure 4, step 1) by taking a picture with the camera instantly or by selecting a photo from the photo library. After the image is received by the server, it finds the corresponding AWS S3 user bucket in the Database (Step 2), and stores the photo in the corresponding bucket for the user in the Amazon S3 Storage (Step 3). After that, the server passes the URL pointing to the photo in the S3 Storage to the Google Vision API for OCR analysis (Step 4). Lastly, the Google Vision API returns the analysed data and the server passes it back to the application.

Google Vision API has been used to perform OCR analysis. There are some other OCR technologies such as Tesseract OCR [35]. However, Tesseract OCR is an Artificial Intelligence(AI) program, hence the program needs to be trained to get accurate results. Due to time limitation and lacking of experience in AI program training, Google Vision API, which does not require any AI programs training, has been used in this project.

The client application is not allowed to directly connect to the Google Vision API. The major reason is that a secret passcode is required when accessing Google Vision API. If the passcode is
included in the client application, hackers may reverse engineer the client application to see the programming codes [36], which includes the passcode. Instead of placing the passcode in the client application, it is stored in the server such that other users cannot use it for their own applications. Since the passcode is stored in the server, the server is used to connect to the Google Vision API.
4.2.3 Searching for Food by Text

**Figure 5. System flow of searching for food by different languages.**

User can search for specific food or restaurant in the application. The keyword is sent from the client application to the server (see Figure 5, Step 1). Then, the keyword is translated into different languages using Google Translate API (Step 2), and the translated keywords in multiple languages are passed to the Database for searching (Step 3). Matched records in multiple languages are therefore shown on the application. Translation allows the Database to show more results with the same meaning and minimize the language barrier.

The client application is not allowed to connect to the Google Translate API directly due to similar security reason as mentioned in 4.2.2. Apart from security, since the translated text is less useful for the user but only for the system to search for food in multiple languages, therefore the translated text is not sent back to the user, and thus reduces network traffic.
4.2.4 Searching for Food by Image

![System flow of searching for food by image.](image)

Figure 6. System flow of searching for food by image.

User can search for food information by taking a picture of the food, and an optional restaurant name can be included to obtain more accurate results (see Figure 6, Step 1). The image is redirected to the Google Vision API for detecting objects in the photo in order to generate a set of object description phrases (Step 2). After retrieving the object descriptions, the descriptions are used to search for records that matched the description and the restaurant name (Step 3). Finally, all the images of records with matching descriptions and restaurant name are compared with the user uploaded image, and the one with the highest similarity can be found (Step 4). The food name and price of that record is then returned back to the user application (Step 5), and these information are presented to the user as AR text (Step 6).

To find the food image with the highest similarity, hamming distance has been used to compare the images, since hamming distance is fast and accurate on image retrieval [37]. As for the representation of the food information, the information is presented in AR text format to provide a more immersive experience for the user because it allows the text to be shown on top of the physical food item.
4.2.5 Searching for Specific Amount of Food Items Closest to a Fixed Budget

User can search for specific amount of food items closest to a fixed budget by inputting a budget, the specific amount of meals, and other filtering parameters, which includes food name, restaurant name, restaurant location and maximum price for each food item (see Figure 7, Step 1). Afterwards, the server receives the data and translates the food name and restaurant information into multiple languages (Step 2). Then the server uses the text in different languages to fetch for food records from the database that satisfy the criteria specified (Step 3). Finally, the food records are used as the input for the algorithm that can find the specific amount of food which has the closest total cost to the specified budget (Step 4).

The problem of finding the specific amount of food with total cost closest to a certain budget is identified as a form of Knapsack problem. Knapsack problem is that given n items, each with a weight and a value, and a knapsack of capacity W, choose a subset of the items such that total weight smaller than W, and get the maximum total value in the knapsack [7]. In the case of this application, the total budget is treated as the knapsack, and the food records in our database are the items to choose to put in the knapsack. This problem differs from the standard Knapsack problem in the value of items. Since a suitable number of food records is required, the values of

Figure 7. System flow of Searching for Specific Amount of Food Closest to a Fixed Budget.
all the records are all set to 1. And instead of the maximizing the total value, an exact number of records is needed in the knapsack.

Since Knapsack problem is regarded as an NP-complete problem [7], there is no polynomial-time solution. In this application, dynamic programming, which is one of the most popular approach to solve the Knapsack problem, has been used. The dynamic programming solution has a time complexity of \(O(nW)\) [8], where \(n\) is number of items and \(W\) is the capacity of the knapsack. A 2-dimensional array with size \(nW\) is created to store the calculated results, and the results are calculated row by row. For any entry \((i,j)\) in the array, it checks whether the \(i\)-th item can be added to the knapsack, or shall be added to a knapsack with capacity \(j\) if it maximizes the value, and it checks the value of \((i-1,j)\) and \((i, j - i\text{-th.price})\) results for comparisons.

Setting the values to 1 caused a problem, because there is no difference between the first-item value to the other-item value, dynamic programming always takes the first item as result. Therefore, in this application, the algorithm has been modified. First, the food items are sorted according to their price in descending order before running the dynamic programming algorithm, which can ensure that food items with highest price are always considered first. Second, after running the dynamic programming algorithm, the data at the last-row-last-column of the 2D array represents the maximum number of food records to fit in the total budget. Therefore, an extra linear upward scan is required to search the last-column starting from the last row, in order to find a number that matches the number of food records specified by the user. Sorting is done in the database, and it should be in \(O(n^2)\) for the worst case algorithms such as bubble sort, while the linear scan is done in the server with \(O(n)\) time. Hence the total time complexity should be \(O(n^2+nW)\) (or \(O(nW)\) if \(W > n\)) for the worst case.
4.2.6 Login/Register

Figure 8. System flow of login/register.

User needs to register before using other functions of the application. User must first enter and submit his or her own phone number in the application (see Figure 8, step 1). Next, the phone number is passed to the server. After that, a confirmation message containing a one-time passcode is sent to the user through Short Message Service (SMS) by the AWS Simple Notification Service (SNS) (Step 2). If the client enters the passcode correctly (Step 3), a new user record is inserted into the server database (Step 4). The one-time passcode is used to ensure that the phone number entered by the user is valid. Hence the database could use the phone number for user identification.

Once verified, the application keeps the user logged in. The user can auto-login with the verified phone number any time.

To improve security, hashing is applied to the verification passcode before it is stored. Hashing means converting a string into a value by some algorithms, and it is impossible to regenerate the value back to the original string [38]. By using hashing, even if the server is accessed by malicious users, they cannot regenerate the user’s verification code and login to those accounts. There are different popular hashing algorithms available on the market, including the Secure
Hash Algorithm 512 (SHA-512) and MD5. In this project, SHA-512 has been used because it is more secure than MD5 [39].
### 4.3 Task Schedule

The development of this project is completed according to the task schedule as shown below (Table 3).

<table>
<thead>
<tr>
<th>Task</th>
<th>Completion Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>First iteration</td>
<td>30 September 2017</td>
</tr>
<tr>
<td>● Interface to list all stored food records</td>
<td></td>
</tr>
<tr>
<td>● Interface to add or edit food records</td>
<td></td>
</tr>
<tr>
<td>● Set up server and database</td>
<td></td>
</tr>
<tr>
<td>● Connection between server and database</td>
<td></td>
</tr>
<tr>
<td>● Connection between the client application and the server</td>
<td></td>
</tr>
<tr>
<td>Second iteration</td>
<td>15 November 2017</td>
</tr>
<tr>
<td>● Interface for location search</td>
<td></td>
</tr>
<tr>
<td>● Interface for food ranking</td>
<td></td>
</tr>
<tr>
<td>● Upload picture for OCR analysis</td>
<td></td>
</tr>
<tr>
<td>Third iteration</td>
<td>31 December 2017</td>
</tr>
<tr>
<td>● Search for food by text with translation</td>
<td></td>
</tr>
<tr>
<td>● Login function with hashing</td>
<td></td>
</tr>
<tr>
<td>Fourth iteration</td>
<td>31 January 2018</td>
</tr>
<tr>
<td>● Navigation to restaurants</td>
<td></td>
</tr>
<tr>
<td>● Interface for User Profile</td>
<td></td>
</tr>
<tr>
<td>Fifth iteration</td>
<td>20 February 2018</td>
</tr>
<tr>
<td>● Search for food by images with Augmented Reality</td>
<td></td>
</tr>
<tr>
<td>Sixth iteration</td>
<td>15 March 2018</td>
</tr>
<tr>
<td>● Search filter with Knapsack Problem</td>
<td></td>
</tr>
<tr>
<td>Seventh iteration</td>
<td>31 March 2018</td>
</tr>
<tr>
<td>● Friendly Mode</td>
<td></td>
</tr>
<tr>
<td>Overall Testing and Debugging, User Acceptance Test</td>
<td>15 April 2018</td>
</tr>
<tr>
<td>Deliverable of Final Report</td>
<td></td>
</tr>
<tr>
<td>Final Presentation</td>
<td>18 April 2018</td>
</tr>
<tr>
<td>Exhibition</td>
<td>2 May 2018</td>
</tr>
</tbody>
</table>

*Table 3. Schedule for this project.*
5. User Interface

All the user interfaces are designed and implemented. In the remaining subsections of this part, the detailed user interfaces of each function are introduced.

5.1 Adding & Uploading Food Records

5.1.1 Home View of locally stored food records

User’s record table is shown in Figure 9, and it is also the home view of the application, which can be accessed by clicking on the house icon(🏠) on the left of the bottom tab bar. The essential information for each food record are the food name, restaurant name with its location, and the price of the food. Records with missing information are considered as incomplete, and they are highlighted in yellow and the corresponding missing field is presented in red (see the first record in Figure 9a). When the user clicks on the “+” button on the bottom-right of the screen, which indicates adding a new record, 3 options appears. The user can choose between...
input by typing (“Type”) or input by OCR (“Scan”) (Figure 9b). Upon clicking the “Help” button, the user is redirected to our project website, which contains the documentation for the application.

All records shown in the view in Figure 9 is stored locally and persistently, meaning that the users can retrieve the records with no internet connection required, and the data persist after they exit and then reopen the application. And when the user click on the top-right “cloud” button ( ), all completed food records with the essential information is uploaded to the server.
5.1.2 Input View of food records

This view is shown whenever the user clicks on a record (Figure 10a, shown when click on second row in Figure 9) or the choose “Type” option(Figure 9b). Users can add or edit the food name, price, restaurant name with its location, tags, ratings from 1 to 5, and the food photo in this view. The rating is selected by clicking on one of the five stars, and the application automatically fills the stars on the left, which indicates the rating. When the user clicks on the picture, another image can be selected from the album or taken from camera. If no picture is selected, a temporary image is displayed to allow image adding by users (Figure 10b).
5.1.3 Location Search View

This view is shown when the user clicks on the location row within the detailed view (see Figure 11a, shown when click on location row in Figure 10). The table automatically updates with latest search result upon typing (Figure 11b). The arrow button( ▲ ) on the top-left of the interface is for clearing the text field and performing search based on user’s current location. After selecting one of the locations, the location name and location address are presented in the location row, as shown in Figure 10.
5.2 Uploading Menus for OCR Analysis

Figure 12. User interfaces of OCR Result View.

When user clicks on the “Scan” button in the view of Figure 12a, user can choose between taking a new picture with the camera or selecting an image from gallery. The image captured or selected is then uploaded to the server for OCR analysis. After the analysis, the result is shown in the format as shown in Figure 12b. The first row of the view is for selecting a location for the restaurant, which works identically to the location search function as mentioned in Section 5.1.3. The table below the first row is a matched-record table, which contains food name and price that are matched by the OCR analysis. The table located at the bottom-right of the screen is the price table, which contains scanned price values from the OCR analysis. Having two tables allows the user to “drag” the price from the price table and then “drop” it on the corresponding record on the matched records table to update the price of a row(Figure 12c). And on the left of the price table there are two text fields for users to manually input records that are not recognized by the OCR.
Since the OCR analysis may be inaccurate sometimes, the drag-and-drop function between the two tables can facilitate users to match the food name and price efficiently without the need to manually input price data that are scanned but are incorrectly matched.
5.3 Searching for Food by Text

5.3.1 Rank View

![Figure 13. User interfaces of Rank View.](image)

The view in Figure 13 is shown when the user clicks on the middle icon of the tab bar at the bottom of the screen. It is the Rank View, which shows the ranking of all food, sorted according to their average ratings in descending order. If the image of the food record is loading, the view shows a temporary image as shown in the second and third row in Figure 13a. If there is no photo uploaded for this food record, the image with a red line striking through a picture is shown, as in the first row in the figure. When the loading of image is completed, the image appears for each food record (see Figure 13b).
5.3.2 Food Detail View

![Food Detail View Screenshots](image)

(a) ![Food Detail View Screenshots](image)

(b) ![Food Detail View Screenshots](image)

**Figure 14. User interfaces of Food Detail View.**

When the user clicks on one of the food item in Figure 13, a food detail view is shown (Figure 14). The food detail view contains all information of the food record given by users. Since food records with the equal food name, restaurant name and restaurant location are treated as the same food item offered by the restaurant as mentioned in the Section 4.1.4, it is possible that the same food record can be uploaded by multiple users. Hence the differences between those records can be the tags, rating and attached image. For tags, all tags given to the food record from different uploads are displayed. For the rating, the average rating given to the food record is shown. For image, if there is no photo uploaded for this food record, the photo with a red line striking through a picture is shown (same as the one in Section 5.3.1); else if there is/are photo(s) uploaded, user can view and swipe on the image to see other images for the food record. Full screen viewer with zooming is also supported by tapping on the image.
A percentage is put below the image, and it represents the validation score that the image receives. The validation score is given by users, such that the image with the highest score is shown first, and as the user navigate to other photos by swiping, images with lower score are shown, and the rightmost image is the one with the lowest score. 100% is the maximum score, meaning that all users who viewed the image thought it represented the food name correctly (see Figure 14a). If an image is newly uploaded and no one has rated it, it shows “Not verified” (see Figure 14b).

![Image](226x242 to 385x525)

Figure 15. User interface of Navigation View.

If the address in Figure 14 is clicked, a navigation view is shown (Figure 15). The user’s current location is shown as the blue circle, while the restaurant’s destination is shown as a red pin. A route is generated based on the path from the user’s current location to the destination, and the distance and estimated time of travel by walking are shown at the bottom of the screen. Clicking the top-left cross icon (×) can close the navigation and go back to the Food Detail view.
5.3.3 Filter View

When the user clicks on the top-left icon (a) in the rank view (Figure 16b), a filter view is shown (see Figure 16a). In this view, there are two sections. The first section is “Meal Planning”, which allows the user to input a total budget and a number of food items wanted. This is the function as describe as in Section 4.2.5, which helps the user to plan how to spend the budget on the food items. And after the user typed in the information and click on the “Apply” button, the rank view is shown, with the records selected within the total budget (Figure 16b). The second section is “Filtering”, which allows the user to specify the food name, restaurant name, restaurant address, and the price range for each food item.

**Figure 16. User Interfaces of Filter View.**
5.3.4 Search with Translation

Translation is automatically applied to the search keyword inputted by the user. The server automatically detects the language of the keyword and translates it to English and Chinese as they are the most popular languages in Hong Kong. In spite of the overwhelming Cantonese speakers, English is perceived as the most common language used by the government, as well as the legal, professional and business sectors [13]. The server then queries for all the records that includes the original keyword or the translated keywords in the corresponding search fields. Finally only the matched records are presented in the table. As shown in Figure 17a, when the user inputs “oolong” in the food name search bar of the rank view, the record that includes the translated version of “oolong” (translates to “烏龍” in Chinese) is shown. In Figure 17b, the user inputs a Korean word “사과” in the food name search field, and the food record with food name “apple” is shown because the English version of “사과” is “apple”. As shown in Figure 17c, the Chinese words “蛋糕”, which is “cake”, is inputted by the user in the restaurant search field, and the two records with restaurant names that includes “Cake” are shown.

Figure 17. User interfaces of Searching with Translation.
5.3.5 Search by Image

![User interfaces of Searching by Image.](image)

When the user clicks on the top-right camera icon (.snapshot) in the rank view as shown in Figure 13, the augmented reality interface appears (see Figure 18a), which is for searching for food record by image. User can point the camera at the physical food, and tap on the screen to start searching for the food record. The user can also type in the yellow text field at the bottom of the screen to filter restaurant name for obtaining more accurate result. The 3D text “loading” appears while the server is searching for the food record. Once the recognition is completed, the food name and price are shown on the screen in 3D text object format. In Figure 18a, the scene is scanned and the text “apple $5” is shown on top of the physical apple after the search. Resizing of the 3D text object can be done by using the pinch gesture.

While waiting for the search results, a temporary interface appears, which asks the user to check whether the given image and food name matches (see Figure 18b). This is for validating the
image, and the validation score for the image can be found in the food detail interface, as shown in Figure 14. The user can exit this validation interface by clicking on the top-left cross, or by answering the question by clicking on one of the three options at the bottom of the screen. This interface will also be automatically closed once the search result is returned to the user.
5.4 Login/Register

When the user launches the application for the first time, a login interface is shown (see Figure 19a). After the user input his or her phone numbers into the application, the phone number is passed to the web server. Upon receiving the phone number, the web server delivers a one-time passcode to the user through SMS message for verification (see Figure 19b). To enhance security, the passcode is only stored in the server side for 15 minutes. If the user does not submit the passcode within 15 minutes, a new passcode is needed for the user to login to the application. This method limits the time for guessing the passcode, which leads to better security.

The user’s phone number is stored in the “Keychain” after the verification process. Keychain is a service API provided by Apple which encrypts and stores data in the file system [40]. It ensures that only authorized applications can gain access to the data stored by the keychain, meaning that only this application can obtain the user’s stored phone number. The user can thus safely
auto-login to the application afterwards even if the application is closed. Each time when user launches this application, the home view as shown in Section 5.1.1 is shown if the phone number is stored; and if not, the login View is shown (see Figure 19a). To logout, please see the details in the next section, Section 5.5.
5.5 User Profile

![User Profile](image)

**Figure 20. User interfaces of User Profile.**

The logout button is implemented to destroy all stored “Keychain” items for this application, and the user needs to verify again upon pressing the logout button. The “Logout” button is placed at the bottom of the user profile page, which can be accessed by clicking on the right icon on the bottom tab bar(see Figure 20a).

The user profile page also contains all the food or menu images uploaded by the user. The user can click on a menu photo, and then he or she can see the upload date and time of the image(see Figure 20b). As for the food photo, the user can see the uploaded information, which are the food name, restaurant name and restaurant address(see Figure 20c). Resizing of photo is also supported when the user is viewing a photo.
5.6 Friendly Mode

When the user clicks on the top-left “Friendly Mode” button in the user profile (as shown in Figure 20a), the interface of Friendly Mode Rank View appears (see Figure 21a). The Friendly Mode is a special part in this application which is specially designed for users with accessibility needs, especially for visually impaired users or physically impaired users. Adjustable font sizes, speech-to-text, and text-to-speech functions are three accessibility features that can help users with impairments [41], and the Friendly Mode has supported all these features in all the interfaces.

For the interface in Figure 21a, which is a modified version of the Rank View, there is no image on the screen, and word size can be adjusted by clicking on the “+” or “−” button at the bottom of
the screen. Figure 21b shows the same view with enlarged font size, which can help visually impaired users to see the text clearly. Apart from variable font size, users can click on any one of the speaker icons( ), and then the synthesized audio for the food information that appeared on the corresponding row is voiced out by the phone. This design helps visually impaired users to know the information of the food record without the need of actually reading the text. For users with physical disability and cannot type with the virtual keyboard, they can click on the microphone icon( ) for speech-to-text input. After the input, the text is shown in a popup window that shows the text(see Figure 21c, which the user spoke the word “Chicken”). The user can then choose to search for food or restaurant with the recognized words, or to discard the text with the “Done” button.

If the user wish to return to the normal mode, there is a “Normal Mode” button on the top-left of the view that allows the user to get back to the normal mode.
The user can click on any of the food record in the Friendly Mode Rank View (see Figure 21a,b), and then the Friendly Mode Food Detail View appears (see Figure 22). This interface contains all the information of the food record, including the food name, price, restaurant name and location, tags and rating. This view also contains the word size modifier that works the same way as the Friendly Mode Rank View, and it also contains the speaker icons that can speak out the particular information of the food record.
The user can click on the top-right camera button (📷) on the Friendly Mode Rank View (see Figure 21a,b), and the camera is opened. The user can take a picture of a food menu, then the photo is analyzed by the OCR, and the recognized food name and price are presented to the user (see Figure 23). This view also contains the word size modifier that works identically to the Friendly Mode Rank View, and the speaker icon that can speak out the food name and price for one row. It is designed to help users with visual disabilities to see the food information that are on the food menu but they cannot see them clearly with the physical menu. The user can quit this view by clicking on the “Cancel” button on the top-left of the view, and the application goes back to the Friendly Mode Rank View.
6. Encountered Challenges

In this section, the encountered challenges and the corresponding solutions used are described. These challenges did not affect much of our project as the problems are solved or alternatives have been found.

6.1 OCR Accuracy

*Problem:*
OCR is not with 100% accuracy. Typical frameworks like Google Vision API and Tesseract OCR give an accuracy rating on the scanned text. Factors affecting the accuracy include file size, image resolutions or text layouts [9]-[10]. Given that this application generates food catalogue by open crowdsourcing, it is hard to put restrictions on the images uploaded by users.

*Solution:*
After the image is scanned for OCR analysis, our server responds back the recognized text to the client application. Users can make final adjustment to the data before uploading them to the database. The “drag” and “drop” function is developed specifically to facilitate users to make adjustments to the mismatched OCR results (as describe in Section 5.2).

6.2 Map Usage

*Problem:*
Apple Map does not always return the most accurate current location, especially in Hong Kong because there are a lot of buildings, and multiple restaurants may be in the same building, but Apple Map can only tell user’s location by the GPS coordinates. When users try to upload the location of a specific restaurant, one may upload wrong location. This limitation might hinder the searching function if the wrong location is stored.
Solution:
A search field with auto-update functionality is provided (as shown in Section 5.1.3), allowing users to enter part of the place name or address and then choose the location from the list of auto-update suggestions for the corresponding restaurant by Apple Map. Hence, accurate restaurant locations can be chosen.

6.3 Response Rate on OCR & Searching

Problem:
Photo uploading and downloading hinders the response rate the most. On one hand, menu photos need to be uploaded for OCR analysis. On the other hand, food photos need to be downloaded each time when the user searches. Frequent transferring of photos increases the loading time, which worsens the user experience.

Solution:
JPEG format is used for uploading and storing the photos in server. Testings have been carried out and no negative impact was made for the OCR analysis. But the photo file size has been reduced from about 2MB to around 300KB, and the uploading time has been reduced from roughly 1 minute to approximately 10 seconds.

The food photos are downloaded asynchronously so that text are loaded before images while searching (see Section 5.3.1). The loading time is now be greatly reduced to 1 to 2 seconds.
7. User Acceptance Test

Several schoolmates and family members have been found for the user acceptance test. They are required to use the perspective of a normal person, a visually impaired person (by wearing translucent eye masks) and a physically impaired person (by putting weights on their arms to restrict their hand and arm movements) to test this app.

The results are positive and the results show that our application provides an accessible environment for the disabled. They also think that our application is useful as an online food catalogue. The detailed results are as follows:

7.1 Perspective of a normal person

The testers found the application really useful. First, most of the testers commented that the idea of food searching instead of restaurant searching is helpful, as most of the time they would like to search for a type of food, rather than finding a specified restaurant. Second, they gave high appreciations to the OCR analysis and the function of searching for food by image. They think these functions are useful and convenient, and they have never thought the technology can be used in this way. Third, the testers mentioned that the meal planning function can help them to plan their daily meals, which is a function that they have never seen before.

7.2 Perspective of a visually impaired person

The testers found the word size adjustment and the function of text-to-speech particularly useful. All the older testers, who are above 50 years old, commented that the word size adjustment is helpful because they often have trouble reading text clearly on their smartphones, but they can read the text on our application easily after enlarging the font size. As for the text-to-speech function, the testers think of it as the last resort and used it when they think that the enlarged words are still too small for reading. With this function, users do not need to read the words but just listen to the audio generated, which makes the application easier to use, especially for visually-impaired users.
7.3 Perspective of a physically impaired person

The testers found the function of speech to text useful. As their hand and arm movement are restricted, typing the keyword for searching is time-consuming. But with the function of speech to text, they could search for food records easily. They agree that this function is beneficial to a physically impaired person.
8. Conclusion

The idea of an online food catalogue based on open crowdsourcing is the key to coming up with the unique functions incorporated in this project. Instead of taking restaurant reviews as the main focus like all other existing applications, the food items on the menus are treated as the major focus of the project, hence allowing new functions to be implemented, such as searching for an amount of food items within a budget, or searching for food by images. It is impossible to search for an amount of food items within a budget if the food information are not stored independently, and it is hard to search for food by image if the images are not associated with food information.

The smartphone application have been fully implemented. It consists of six major functions, which are adding and uploading food records, uploading menus for OCR analysis, searching for food by text, searching for food by image, searching for specific amount of food items closest to a budget, and login and register. Furthermore, a friendly mode is also developed for users with visual or physical impairments. It is anticipated that this application will be highly useful for both restaurants and customers, as it turns finding restaurants into a two-way conversation. On one hand, customers can contribute to the application by simply taking a photo for OCR analysis and uploading food records. This encourages customers getting involved if they want to recommend some good restaurants to their friends, instead of being told by the application passively. On the other hand, restaurant owners could enjoy free labour (customers) to promote their restaurants. Both restaurants and customers can benefit from this application.

The initial objectives for this project have been achieved, but there are possibilities for future developments that can focus on some aspects that are not covered. First, an Android version of the application can be created. Second, machine learning models can be developed for the OCR or image comparison functions. Third, approximation algorithm can be used to solve the Knapsack problem, but the trade-off between speed and accuracy when comparing approximation algorithm to dynamic programming should be analysed.
9. References


