<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>FoodServe: a self-service platform for restaurants</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Author(s)</strong></td>
<td>Chan, Chung Ngok (陳宗岳)</td>
</tr>
<tr>
<td><strong>Citation</strong></td>
<td>Chan, C. N. (2014). FoodServe: a self-service platform for restaurants (Outstanding Academic Papers by Students (OAPS)). Retrieved from City University of Hong Kong, CityU Institutional Repository.</td>
</tr>
<tr>
<td><strong>Issue Date</strong></td>
<td>2014</td>
</tr>
<tr>
<td><strong>URL</strong></td>
<td><a href="http://hdl.handle.net/2031/7465">http://hdl.handle.net/2031/7465</a></td>
</tr>
<tr>
<td><strong>Rights</strong></td>
<td>This work is protected by copyright. Reproduction or distribution of the work in any format is prohibited without written permission of the copyright owner. Access is unrestricted.</td>
</tr>
</tbody>
</table>
FoodServe: A Self-Service Platform for Restaurants

(13CS014)

Student Name : Chan Chung Ngok
Programme Code : BSCCS
Supervisor : Chow, Chi Yin Ted
1st Reader : Kwok, Lam For
2nd Reader : Yu, Yuen Tak
Abstract

Hong Kong, a gourmet paradise, has a wide variety of restaurants providing Eastern and Western dining services. Typically, restaurants provide dine-in, takeout and delivery services. Especially in crowd area, takeout and delivery ordering are popular in rush hours because restaurants might possibly full when people are coming late. Furthermore, takeout and delivery ordering are usually received by phone call. Have you ever been waiting for a long time in phone queue, and be denied finally? Or when you dine in restaurant, can you catch the attention of waiter for ordering during rush hours?

As a restaurant, you may want a software application to reduce cost, increase sales and improve quality of service. Software application helps a lot in reducing workload of staff, thus freeing staff for other important tasks. But after considering the development cost and required space, you may possibly say no.

As a customer, you may have difficulties in finding nearby restaurants that fits your interest. On the other hand, phone ordering is time consuming during holiday or rush hours because of the phone queue. It is difficult to place order by phone call since menu is not available while taking the phone call.
To solve these problems, FoodServe, a self-service platform for restaurants is developed. FoodServe would help restaurants process online ordering services in dine-in, takeout and delivery with customizable workflow. Customers can place order by smartphone while sitting on their chair comfortably without calling the waiter. Placing takeout and delivery order will not be nightmare to customers anymore. Meanwhile, restaurants benefit from more sources of customers. And of course, chance of making human mistake will be greatly reduced. Customers can enjoy the pace of self-service when using FoodServe and avoided the worst scenario that the waiter has taken a wrong order. Moreover, Customers can search nearby interested restaurants easily using their smartphone.

This project aims at providing a cross-platform system for restaurants with flexibility, usability and reliability. Our target platform will be on Android, iOS and web application. After using FoodServe, restaurants should be able to reduce the workload of staff, increase effectiveness on taking order and attract more customers. Meanwhile, customers can experience self-service dining provided by FoodServe with simple and clear interface on their smartphone.
Acknowledgement

I would like to express my deep gratitude to Dr. Chow Chi Yin, Ted. I appreciate his encouragement and kindly advice. Thanks for the arrangement of regular meeting that keep me going on the right track.

I would also like to thank Chris So, who gave me useful technical advice on web application development.

Finally, I have to thank my family for the understanding, encouragement and support throughout this year.
Table of Contents

1. Introduction ................................................................................................................ 1

1.1 Background........................................................................................................... 1

1.2 Motivations............................................................................................................ 2

1.3 Our Solution.......................................................................................................... 4

1.4 Project Objectives ................................................................................................ . 6

2. Literature review ......................................................................................................... 7

2.1 Cross-Platform Application.................................................................................... 7

2.1.1 jQuery Mobile .................................................................................................. 7

2.1.2 PhoneGap ....................................................................................................... 9

2.1.3 Conclusion .................................................................................................... 11

2.2 Online Payment .................................................................................................. 12

2.2.1 Braintree ....................................................................................................... 12

2.2.2 PayPal .......................................................................................................... 15

2.2.3 Conclusion .................................................................................................... 16
2.3 Map API .............................................................................................................. 17

2.3.1 Google Map .................................................................................................. 17

2.3.2 OpenStreetMap ............................................................................................ 19

2.3.3 Conclusion .................................................................................................... 21

2.4 Existing Applications ........................................................................................... 22

2.4.1 JUST EAT ..................................................................................................... 22

2.4.2 foodpanda ..................................................................................................... 25

2.4.3 OPEN DINING .............................................................................................. 27

2.4.4 Comparison .................................................................................................. 30

2.4.5 Conclusion .................................................................................................... 31

3. System Design ......................................................................................................... 32

3.1 Use case diagram ............................................................................................... 32

3.2 System Functions ............................................................................................... 35

3.3 System Overview ............................................................................................... 40

3.4 Entity Relationship Diagram ............................................................................. 41

3.5 Sequence Diagram ............................................................................................. 46
References

Appendices

Monthly Log:
List of Figures

Figure 1 - Screen capture of jQuery Mobile site, from http://jquerymobile.com .............. 8

Figure 2 - PhoneGap overview, from http://phonegap.com/about/artwork/ .................... 9

Figure 3 - Screen capture of supported feature by PhoneGap, from http://phonegap.com/about/feature/ ............................................................................. 10

Figure 4 - Screen capture of Braintree supported server side, from https://www.braintreepayments.com/developers .......................................................... 12

Figure 5 - Screen capture of Braintree supported client-side, from https://www.braintreepayments.com/developers .......................................................... 13

Figure 6 - Architecture of Braintree payment, from https://www.braintreepayments.com/docs/php/guide/getting_paid ....................................................... 14

Figure 7 - Screen Capture of PayPal Developer, from https://developer.paypal.com/ . 15

Figure 8 - Google Map, from https://www.google.com.hk/maps/ ................................. 17

Figure 9 - Marker on Google Map, from https://developers.google.com/maps/documentation/javascript/examples/marker-simple ............................................................................. 18

Figure 10 - OpenStreetMap, from http://www.openstreetmap.org/ ............................... 19

Figure 11 - Serving Tiles, from http://switch2osm.org/serving-tiles/ ......................... 20
FoodServe: A Self-Service Platform for Restaurants

Figure 12 - Screen capture of JUST EAT (UK), from http://www.just-eat.co.uk/ ............ 22

Figure 13 - Screen capture of JUST EAT (UK) search result, from http://www.just-eat.co.uk/area/sw11-clapham ........................................................................................................ 23

Figure 14 - Screen capture of JUST EAT (UK) menu, from http://www.just-eat.co.uk/restaurants-thamestandoori/menu .................................................................................................... 23

Figure 15 - Screen capture of foodpanda (Singapore), from http://www.foodpanda.sg/25

Figure 16 - Screen capture of foodpanda (Singapore) search result, from http://www.foodpanda.sg/restaurants/index/city/singapore ......................................................... 26

Figure 17 - Screen capture of OPEN DINING site, from https://www.opendining.net/ . 27

Figure 18 - Screen capture of OPEN DINING setup wizard, from https://www.opendining.net/admin/restaurants/5276b52431131cfd5200000a/setup ... 28

Figure 19 - Screen capture of OPEN DINING administrative back-end, from https://www.opendining.net/admin/home ........................................................................... 29

Figure 20 - Use case diagram of FoodServe ................................................................ 32

Figure 21 - System overview of FoodServe ................................................................ 40

Figure 22 - ERD of FoodServe, overview ................................................................... 41

Figure 23 - ERD of FoodServe, menu item .................................................................. 42

Figure 24 - ERD of FoodServe, order ......................................................................... 43
Figure 25 - ERD of FoodServe, workflow ................................................................. 44

Figure 26 - Hierarchy of restaurant menu ............................................................... 45

Figure 27 - Sequence Diagram of FoodServe, overview ........................................ 46

Figure 28 - Sequence Diagram of FoodServe, condition_dine_in_order ............... 47

Figure 29 - Sequence Diagram of FoodServe, browse_menu ................................ 48

Figure 30 - Sequence Diagram of FoodServe, add_item_to_cart .......................... 49

Figure 31 - Sequence Diagram of FoodServe, confirm_order ............................... 49

Figure 32 - Screen capture from FoodServe, web application login ...................... 50

Figure 33 - Screen capture from FoodServe, mobile application login ................. 51

Figure 34 - Approach of adding an item ............................................................... 52

Figure 35 - Screen capture from FoodServe, create an item: choose a category .... 53

Figure 36 - Screen capture from FoodServe, create an item: add a new category ... 53

Figure 37 - Screen capture from FoodServe, create an item: basic information ...... 54

Figure 38 - Screen capture from FoodServe, add an item: advance setup ............ 55

Figure 39 - Screen capture from FoodServe, add an item: choose existing size .... 56

Figure 40 - Screen capture from FoodServe, add an item: create a new size ....... 56
Figure 41 - Screen capture from FoodServe, add an item: set price of size .............. 57

Figure 42 - Screen capture from FoodServe, add an item: choose existing options .... 58

Figure 43 - Screen capture from FoodServe, add an item: create a new option ........ 59

Figure 44 - Screen capture from FoodServe, add an item: item's options ............... 60

Figure 45 - Screen capture from FoodServe, add an item: choose existing schedule . 61

Figure 46 - Screen capture from FoodServe, add an item: create a new schedule ..... 62

Figure 47 - Screen capture from FoodServe, add an item: summary ...................... 63

Figure 48 - Screen capture from FoodServe, add an item: successful ..................... 63

Figure 49 - Screen capture from FoodServe, browse item menu .......................... 64

Figure 50 - Screen capture from FoodServe, browse size .................................... 65

Figure 51 - Screen capture from FoodServe, browse options .............................. 65

Figure 52 - Screen capture from FoodServe, browse schedule ............................ 65

Figure 53 - Screen capture from FoodServe, add a combo: choose existing category 66

Figure 54 - Screen capture from FoodServe, add a combo: add a new category ...... 66

Figure 55 - Screen capture from FoodServe, add a combo: basic information ........... 67

Figure 56 - Screen capture from FoodServe, add a combo: choose component ........ 68
Figure 57 - Screen capture from FoodServe, add a combo: create new component ... 69
Figure 58 - Screen capture from FoodServe, add a combo: select item for component69
Figure 59 - Screen capture from FoodServe, add a combo: combo's component....... 70
Figure 60 - Screen capture from FoodServe, add a combo: schedule ....................... 71
Figure 61 - Screen capture from FoodServe, add a combo: summary.......................... 72
Figure 62 - Screen capture from FoodServe, add a combo: successful....................... 72
Figure 63 - Screen capture from FoodServe, select order type................................. 73
Figure 64 - Screen capture from FoodServe, list of restaurants............................... 74
Figure 65 - Screen capture from FoodServe, list of category................................. 75
Figure 66 - Screen capture from FoodServe, list of items................................. 76
Figure 67 - Screen capture from FoodServe, item information............................. 77
Figure 68 - Screen capture from FoodServe, finish add item................................. 78
Figure 69 - Screen capture from FoodServe, order cart................................. 79
Figure 70 - Screen capture from FoodServe, order cart part 2............................. 80
Figure 71 - Screen capture from FoodServe, receive order................................. 81
Figure 72 - Screen capture from FoodServe, kitchen screen............................. 82
Figure 73 - Screen capture from FoodServe, order updated ........................................ 83
Figure 74 - Screen capture of FoodServe, user registration ...................................... 84
Figure 75 - Screen capture from FoodServe, SMS confirmation code ....................... 85
Figure 76 - Screen capture of FoodServe, dine-in ordering ....................................... 88
Figure 77 - Screen capture from FoodServe, generate QR code ................................ 90
Figure 78 - Screen capture of FoodServe, setup delivery area ................................... 93
Figure 79 - Screen capture of FoodServe, setup delivery area in square shape ........... 94
Figure 80 - Screen capture of FoodServe, setup delivery area in triangle shape .......... 94
Figure 81 - Screen capture from FoodServe, Order Payment .................................... 101
Figure 82 - Screen capture from FoodServe, workflow customization ...................... 104
Figure 83 - Screen capture from FoodServe, modified workflow sequence .............. 105
Figure 84 - Screen capture from FoodServe, list of e-Coupon .................................. 106
Figure 85 - Screen capture from FoodServe, create e-Coupon ................................. 107
Figure 86 - Screen capture from FoodServe, push notification ............................... 108
Figure 87 - Screen capture from FoodServe, restaurant list with recommendation .... 109
1. Introduction

In this section, we would introduce the background of this project. Then identify existing problem and propose a solution.

1.1 Background

Hong Kong, a gourmet paradise, has a wide variety of restaurant providing Eastern and Western dining services. Everyone in Hong Kong dine every day, whenever they are busy or relaxed. Dining is important in Hong Kong culture because of the influence from China and colonial history [1].

Typically, restaurants provide dine-in, takeout and delivery services. Especially in crowd area, takeout and delivery ordering is popular in rush hour because restaurants might possibly full when people are coming late.

On the other hand, smartphone applications are popular nowadays. People are heavily relied on their smartphone for many services such as booking for holiday packages, paying bills, checking-in for flight. However, it is rare that restaurants in Hong Kong provide online ordering while it is common in other countries. Online ordering is often provided by some large scale restaurants as they can afford the high cost of developing and maintaining mobile application. Therefore, online ordering is demanded in Hong Kong since it provides business opportunity.
1.2 Motivations

In Hong Kong, restaurants are facing different problems. First of all, during rush hours such as lunch time or holiday, restaurant staff would be very busy in serving customers, delivering dish, handling customer payment and counting available table. Along with all these busy tasks, restaurant providing takeout or delivery services would suffer much heavier pressure. As a result, restaurants may lose order because of busy phone line and quality of service would be greatly reduced.

Secondly, waiting the restaurant may discourage time-pressured customers who are in their lunch break. They aim for quick dining instead of waiting. Customers would like to choose fast food or takeout in this scenario. The reason of long waiting time may due to most staff are spending time on handling dine-in customers. Response time of staff is affected by the number of customer. When they are busy, delay of staff response will reduce table turn, thus increasing the waiting time.

Thirdly, Restaurants may consider digital ordering to improve table turn and service quality. However, the cost of developing and maintaining a system is high. If mobile application is not adapted, restaurants may need extra space and resource for placing device on table. As a result, restaurants are greatly discouraged in using software application.

It is ineffective for restaurants to handle takeout and delivery order by phone call. Staff is affected due to the busy and increase waiting time. And there is a chance of misunderstanding because of bad communication [2]. Also, queuing experience will bring negative influence on customer satisfaction [3]. Therefore, customers are not willing to dine in restaurant.
Also, lack of advertisement is a common problem for small scale restaurants. It is expensive to advertise on magazine or newspaper. So, if there are foods that cannot be store over-night, it would be a waste and bring negative effect on cost control. Under this situation, it is difficult for restaurants to sell those foods even with lower price because restaurants do not have a proper way to post an advertisement in a short time.

Lastly, customers may found that they would like to try for a restaurant that they have not visit before. However, customers may not know the quality of restaurants and is difficult to find restaurants that close to their interest.

Hence, we would like to solve the problems of restaurants and customers by using latest technology and implement an application. As a result, restaurants and customers can enjoy the benefits of technology nowadays.
1.3 Our Solution

Smartphone application is popular nowadays. [4] reported that 11,498,184 people are subscribed to 2.5G, 3G/4G mobile. They are heavily relied on their smartphone for many services such as booking for holiday packages, paying bills, checking-in for flight. These are examples of self-service mobile applications. Development on self-service application is continually growing [5]. Self-service helps the service provider to reduce cost, increase sales and enhance customer service [6].

To improve restaurants effectiveness and service quality, a self-service platform for restaurants are proposed. Self-service can bring benefits to both restaurants and customers. To restaurants, they can reduce cost, increase sales, and provide better service. To customers, they can save time on catching staff attention, tracking the order status, and experience self-service in restaurant.

In this project, we aim to develop a cross-platform application with high flexibility and user friendly interface. Restaurants can upload their menu through our system, and customize the behavior of mobile application that suits their needs. Customer can place dine-in, takeout or delivery order using smartphone easily.

Restaurants can customize the business workflow, namely dine-in, takeout and delivery. The mobile application will behave in different ways according to business workflow. For example, restaurants can choose the moment of customer payment, after placing an order, or before they leave the restaurant. This provides flexibility to restaurants since the nature may differ.

Restaurants can create an e-Coupon and issue them to customers in a very short time with user friendly interfaces. e-Coupon will find nearby customers in order to improve the chance of the coupon being used.
Customers can find nearby restaurant, or recommend by FoodServe according to their rating history. When customers check in to restaurant, they can use their smartphone to scan QR code on table to start ordering. They can control the pace of ordering while they do not need staff service. Then, order will be sent to kitchen through the Internet. Customers can pay for their order by credit card via our system.

After adapting our system, restaurants can take more orders and improve table turnover rate during rush hours. Human mistake made by staff will not happen again because digital order is reliable. Staff can save effort on serving customers while they can spend time on providing better service. Customers can search for interested restaurants and enjoy the benefits of using mobile device to place order.
1.4 Project Objectives

The objectives of project should be accomplished as following:

- **Usability**
  
  o It is important for smartphone application to be usable. As the screen size of smartphone is usually small, developing user friendly interface would be essential for customer experience.

- **Flexibility**
  
  o Since Hong Kong has lots of restaurant, each of them may have different culture, business workflow and requirement. Flexibility of customizing the ordering application is necessary.

- **Reliability**
  
  o During rush hours, our system should behave well and provide stable service to customer. Furthermore, it should handle order properly for restaurant because kitchen would rely on digital order. There should not be mistakes on order and payment as they are important for restaurant.
2. Literature review

In this section, we would discuss different technologies that may help to develop this project and evaluate existing application.

2.1 Cross-Platform Application

Mobile applications are common in Hong Kong, but there are different mobile platform such as android, iOS, BlackBerry, etc. Each platform has their app store for distributing application, thus different versions of application for every platform is required if it needs to be popular. However, there are problems when developing on multi-platform. First, different knowledge of each platform is required for development which cost huge time to learn. Also, maintaining different version of source code for each platform is exhaustive. Therefore, frameworks for cross-platform application development are necessary for this project in order to save effort.

2.1.1 jQuery Mobile

jQuery Mobile is a framework for development on smartphone and tablet by HTML5. jQuery Mobile supports lots of mobile platform including android, iOS, Blackberry, Windows Phone, Palm, Symbian, etc. Furthermore, it is compatible with web browser such as Chrome, Firefox and Internet Explorer.
The benefit of using jQuery Mobile is easy to learn. Since jQuery is popular and widely used in web development, learning jQuery Mobile would be quick without learning completely new knowledge. Furthermore, as mention before, it is cross-platform. It is a major advantage of using it can save time and effort on development.

However, there are limitations of jQuery Mobile. First of all, the performance is slower than native. Even with high quality smartphone hardware, the performance of mobile web application is noticeably slow. Also, the capability of jQuery Mobile is limited. As it is running on a web browser, it does not gain access to many feature of mobile device such as camera, sensor and GPS.
2.1.2 PhoneGap

PhoneGap is a free and open source framework that allows developer to create cross-platform mobile application with HTML5, JavaScript and CSS. It supports popular mobile platform like android, iOS, Blackberry, Windows Phone, etc. Developer can warp their web code by PhoneGap and then deploy to different mobile platform easily.
Other than creating mobile application by web technologies, PhoneGap also provides API for smartphone native function such as accelerometer, camera, compass, geolocation, etc. So, without learning different programming language of each mobile platform, developer can simply enable mobile features by using JavaScript with the help of PhoneGap.

Furthermore, there are lots of plugin provided by PhoneGap. These plugin boost the availability of native feature with simple usage. For example, it allows application to enable the Bluetooth, speech recognizer, NFC, etc.
2.1.3 Conclusion

As mentioned before, cross-platform is important for mobile application because of the variety of platform. By using jQuery mobile, developer can create mobile web application easily with less coding effort. And with PhoneGap, developer can deploy mobile application developed along with jQuery and enable smartphone feature if necessary. Combining jQuery Mobile and PhoneGap would help to develop a cross-platform mobile web application with native feature.
2.2 Online Payment

Payment is a must for every customer after serving by restaurant. For software application, there are different implementations for online payment method. In this section, different online payment method will be discussed.

2.2.1 Braintree

Braintree is a payment platform that allows developer to enable payment acceptance to their application. Braintree is international which accept credit card payments in 130+ currencies. This suit the nature of Hong Kong as an international city since it is not limited to Hong Kong credit card only.

![Figure 4 - Screen capture of Braintree supported server side, from https://www.braintreepayments.com/developers](https://example.com)
Furthermore, Braintree provides cross-platform support of payment, on both server side and client side. It supports server side such as php, .NET, Java, Perl, Python, etc. It also supports client side including android, iOS, Windows Phone and web browser. So, Braintree has good flexibility on development of web/mobile application.
The architecture of Braintree is simple. The payment form in our client side will encrypt sensitive credit card information by Braintree library, and then transmit to our server. After that, our server communicates with Braintree server by server-to-server call to process the credit card payment. Therefore, the complicated credit card payment process is done by Braintree server while our server and client are responsible for sending, receiving and handling data only.
2.2.2 PayPal

PayPal is a way for online payment services. It allow customer to pay through their PayPal account, bank account or credit card. It supports 26 currencies around the world, including Hong Kong.

![Figure 7 - Screen Capture of PayPal Developer, from https://developer.paypal.com/](https://developer.paypal.com/webapps/developer/docs/classic/)

For software developer, PayPal provided Classic APIs and Representational State Transfer API (REST API) for developer. For Classic API, Developer needs to implement the payment gateway in server side using server-to-server call to PayPal service endpoint and then process the payment. For REST API, it provides simple operation using PayPal or credit card, with limited cities. PayPal also provide SDK for different mobile platform.
2.2.3 Conclusion

Comparing Braintree and PayPal, Braintree provides simpler usage on both server side and client side, with well-organized documentation and code sample. It is much easier to learn Braintree rather than complicated PayPal API.

Furthermore, PhoneGap PayPal plugin only support REST API. The plugin allow the application to process payment by PayPal account. But for credit card payment, the supported cities do not include Hong Kong. On the other hand, since Braintree client side provides JavaScript, PhoneGap can works well with Braintree without side effect.

To conclude, if PhoneGap is adapted for cross-platform development, PayPal is not preferred as it does not support Hong Kong, while Braintree can work along with PhoneGap and PHP.
2.3 Map API

2.3.1 Google Map

Google Map is a web map provided by Google. Google provides API for developer to integrate Google Map into their application.

![Google Map](https://www.google.com.hk/maps/)

Figure 8 - Google Map, from https://www.google.com.hk/maps/

Basically, Google Map API support different platform including web application, Android application and iOS application. For web application, Google Map API provides JavaScript library to developer. The library allows developer to display a map in their application as a basic function. Developer can customize the map using the API. For instance, developer can style the map, add custom marker on the map, draw shape on the map, etc.
Other than map customization, Google Map API provides Geocoding service too. Geocoding service can convert an address into geographic coordinates in form of longitude and latitude. The service is useful when placing marker with reference to an address.

However, there are limitations of Google Map API. First of all, there are usage limits on Google Map. If the site or application generates 25000 map loads or more each day, for more than 90 consecutive days, the developer will need to pay for the extra usage [7].
2.3.2 OpenStreetMap

OpenStreetMap is a free and editable web map supported by OpenStreetMap Foundation. It is open source and participated with over one million contributors [8].

![OpenStreetMap](http://www.openstreetmap.org/)

**Figure 10 - OpenStreetMap, from http://www.openstreetmap.org/**

To enable OpenStreetMap in web application, OpenLayer is required. OpenLayer is an open-source JavaScript library for deploying map services. OpenLayer allows basic map displaying. For advanced feature, developer can add marker, draw shape or style the map according to their needs. On the other hand, OpenStreetMap provides Geocoding service for converting address to geographical coordinate.
Since OpenStreetMap is open-source, it is possible to build a private server to store the map data. Unlike the official tile server, private server will not have the limitation of usage policy. Another benefit is developer can modify the map in their server without changing the OpenStreetMap official site, therefore it is highly flexible.

Developer can gather raw data from OpenStreetMap server into the private tile server. Then the front-end application can use the data from private tile server instead of OpenStreetMap server.
2.3.3 Conclusion

The advantage of using Google Map is the API provided is well-documented with rich example. It is easy for developer to implement the map function by referring to the example. Moreover, Google Map provides better support on drawing on map meaning the performance of drawing is better. OpenStreetMap is similar to Google map but it is open-source.

However, there is a usage limitation of Google Map and OpenStreetMap. Although OpenStreetMap server has usage limitation, OpenStreetMap allows developer to build their own tile server which does not have limitation.

To conclude, both maps API provides similar functionality. It is better to use Google Map if the usage of function is low. But if the usage of map is very high such as mobile application, OpenStreetMap with private tile server will be a better choice.
2.4 Existing Applications

Currently, there are lots of restaurant related applications around the world. In this section, we would evaluate the strength and weakness of some application, thus improving our system.

2.4.1 JUST EAT

JUST EAT is an online take-away ordering service which support in 13 countries including Belgium, Brazil, Canada, Denmark, France, India, Ireland, the Netherlands, Norway, Spain, Sweden, Switzerland and the UK. JUST EAT allows restaurant to provide online ordering services for take-away. Meanwhile, customer can easily search and order their food through JUST EAT.

Figure 12 - Screen capture of JUST EAT (UK), from http://www.just-eat.co.uk/
Figure 13 - Screen capture of JUST EAT (UK) search result, from http://www.just-eat.co.uk/area/sw11-clapham

Figure 14 - Screen capture of JUST EAT (UK) menu, from http://www.just-eat.co.uk/restaurants-thamestandoori/menu
The user interface for customer ordering is simple. Customer can search takeaway by post code and their favorite type of food. Then a list of restaurant is shown with details including delivery distance, cost, user rating. These help the customer to make their decision when choosing restaurant. After choosing a restaurant, customer can review the menu. Items are categorized by a category clearly. Each item has their user rating. Customer can place order easily with sufficient information of restaurant and meal item.
2.4.2 foodpanda

foodpanda is an online delivery ordering service that support 30 cities. It provides a platform for restaurant to enable online delivery ordering services. Customer can search restaurant by their address. Address is especially important for delivery because it can be used for estimating delivery time and area. The representation of menu is similar to JUST EAT, meaning the item is well categorized, and have user rating.

Figure 15 - Screen capture of foodpanda (Singapore), from http://www.foodpanda.sg/
Figure 16 - Screen capture of foodpanda (Singapore) search result, from http://www.foodpanda.sg/restaurants/index/city/singapore
2.4.3 OPEN DINING

Figure 17 - Screen capture of OPEN DINING site, from https://www.opendining.net/

OPEN DINING provides online ordering service for restaurant. OPEN DINING has lots of feature, such as customizable ordering application, customer engagement, site integration and administrative back-end.

OPEN DINING has a guiding tour for first time user when setting up their new restaurant. The setup wizard lead user to create an order service step by step, from basic to complicated. This presentation skill is useful when introducing complicated process to first time user.
Welcome to Open Dining!

Thank you for signing up. We'll have you processing online orders in no time! Please follow the steps below to complete your basic setup:

- **Restaurant Information**
  - Provide basic information about your restaurant.

- **Complete Menu**
  - Setup your menu and options.

- **Receiving Orders**
  - Tell us how to send you orders.

- **Accept Credit Cards**
  - Process credit cards through our payment partner, Mercury Payments.

- **Web Setup**
  - Add ordering to your website or send instructions to your web developers.

- **Start Taking Live Orders**
  - Start your live trial and take orders today!

Questions?
We’re here if you need help:
- **U.S.A.**
  - 1-800-922-6026
  - (Mon-Fri 9:00 A.M. to 5:00 P.M. Mountain Time)

Figure 18 - Screen capture of OPEN DINING setup wizard, from https://www.opendining.net/admin/restaurants/5276b52431131cfd520000a/setup
Restaurant can manage their information and menu through administrative back-end. From the back-end, they can also setup the order payment method, order notification, manage coupons, layout of mobile application, etc. These features help restaurant to customize the ordering system to suit their needs thus provide great flexibility and rich feature. After setting up the restaurant ordering information, restaurant can publish the application to Facebook, mobile web application or embed to their web site.
### 2.4.4 Comparison

<table>
<thead>
<tr>
<th></th>
<th>Just-eat</th>
<th>FoodPanda</th>
<th>Open Dining</th>
<th>FoodServe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delivery ordering</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Take-out ordering</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Dine-in ordering</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Search restaurant by location</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Customization</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Cross-platform application</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Recommendation</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>e-Coupon</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

FoodServe supports more types of ordering like take-out and dine-in ordering. Also, since search restaurant by location is common in existing application, FoodServe would apply similar technique to the application. Furthermore, customization is a nice and flexible feature for restaurant such that FoodServe would adapt that too. At last, cross-platform application is necessary as it is common in existing applications.
2.4.5 Conclusion

From the existing application, it is obvious that there are needs of online ordering. Moreover, these ordering applications provide simple and user-friendly interface because ordering process should not be complicated. Keep the interface clean and clear helps customer to locate their target restaurant and meal. Also, for takeaway and delivery, address is important for ordering and should be noted in our proposed solution. It helps to determine the delivery area, cost and time, for both restaurant and customer. Lastly, customizable is a nice feature for ordering application because different restaurants have different culture and operation.
3. System Design

In this section, we would introduce the design of proposed solution, including use case, system function, system overview and database design.

3.1 Use case diagram

![Use case diagram of FoodServe](image.png)

Figure 20 - Use case diagram of FoodServe
## Actor description:

<table>
<thead>
<tr>
<th>Actor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Restaurant</td>
<td>Restaurant is a service provider who serves meal to customer.</td>
</tr>
<tr>
<td>Customer</td>
<td>Customer is someone who will place order to restaurant.</td>
</tr>
</tbody>
</table>
## User case description:

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Description</th>
<th>Actor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Place order</td>
<td>User can place order after choosing restaurant and browse menu. An order will be sent to restaurant after place order.</td>
<td>Customer</td>
</tr>
<tr>
<td>Register account</td>
<td>User needs to register an account before placing an order. An account can record user's information such as e-mail, name, address and phone.</td>
<td>Customer</td>
</tr>
<tr>
<td>Create menu</td>
<td>User can create menu by setting up each item or combo of their restaurant. The created menu will be browse by user when placing order.</td>
<td>Restaurant</td>
</tr>
<tr>
<td>Customize workflow</td>
<td>User can customize detail of workflow, such as the sequence of ordering process, delivery area and manage QR code.</td>
<td>Restaurant</td>
</tr>
<tr>
<td>Manage order</td>
<td>User can receive order and change order status for processing.</td>
<td>Restaurant</td>
</tr>
</tbody>
</table>
3.2 System Functions

**Web application**

Create an item

Restaurants can create an item for the menu.

Create a combo

Restaurants can create a combo for the menu. Combo is a set of grouped items.

Browse menu

Restaurants can browse the item menu and combo menu they have created. Restaurants can enable or disable an item according to their needs.
Workflow

Setup workflow sequence

Restaurants can customize and rearrange the workflow sequence of an order type.

Setup delivery area

Restaurants can define the delivery area on a map in a shape of polygon.

Display kitchen screen

Restaurants can see the dish to be prepared in the kitchen screen.

Generate QR code

Restaurants can generate QR code according to their number of tables. QR code can be used for dine-in ordering while customers can place order by scanning the QR code.
Order

Receive order

Restaurants can receive orders in real-time.

Update order status

Restaurants can update the status of order after receiving an order.

View report

Restaurants can view the report of their restaurant. Such as number of order, top sales item and top sales combo.

e-Coupon

Restaurants can create an e-Coupon for advertisement. e-Coupon will be sent to nearby customers in real-time through push notification.
**Mobile application**

**Register user account**

Customers can register a user account by entering email, password and phone number. An SMS confirmation code will be sent for registration confirmation.

**Search restaurants by location**

Customers can search restaurants by selecting locations.

**Recommendation**

Customers can find interested restaurants by recommendation provided.

**Place order**

Customers can place a dine-in order, take-out order or delivery order to a restaurant by adding items or combos into order cart.
Payment by credit card

Customers can pay their order by online payment using credit card information.

Browse order history

Customers can browse orders they have been placed before. Status of order will be updated in real-time.

Update user profile

Customers can update their personal information in user profile.
3.3 System Overview

For client side, PhoneGap is adapted for cross-platform development. Android and iOS would be the target platform of our system. It is responsible for interface presentation. For server side, PHP and MySQL are used. It is responsible for processing data received from client side.

Figure 21 - System overview of FoodServe
3.4 Entity Relationship Diagram

Figure 22 presented the database design for menu item. To begin with, category has multiple items, and each item has different size, schedule and option. Item may belong to a component. Combo category has multiple combos, and each combo have components, schedule and option. Item size, option and schedule are shared by different item.
Figure 23 - ERD of FoodServe, menu item

Figure 23 presented the database design for menu item. To begin with, a restaurant may have one or more category. And a category may have multiple items. For each item, it may have multiple sizes, options. Also, an item may have a schedule which defined by restaurant.
Figure 24 shows the ERD for modeling order. First of all, an order is belongs to a restaurant and a user, while the user is referring to the customer. Each order may have combo or item. For each item, there may be more than one choice when placing order. The field payment reference in order is used for saving the credit card payment reference number. Note that the information of item, combo and choice are not refer to their original table, because the information of these table may change over time. To maintain the information of order in a correct state, the data of item, combo and choice are saved in order related table instead of referring to original table.
Figure 25 presented the modeling of workflow. For each restaurant, they may have a customizable workflow. There will be different type of workflow, such as workflow for dine-in, workflow for take-out or workflow for delivery. A workflow will have a set of component in an arranged sequence. A component will have a name, actor, description and type.
Figure 26 - Hierarchy of restaurant menu

Figure 26 shows the hierarchy of restaurant menu in a simplified way. The menu is divided into two sets, which are item and combo. Also, the size, option and schedule of item may share with other items such that it is easier to manage by restaurants and able to reuse, thus saving time when creating menu.
3.5 Sequence Diagram

Figure 27 - Sequence Diagram of FoodServe, overview

Figure 27 shows the sequence diagram of the ordering function. Entry with prefix UI_ is the interface that customer interact with their mobile phone. Order is an entity class that store order information and control the flow of ordering. In this scenario, customer will start placing order beginning with selecting the order type, and then a list of restaurant will show up. Customers then select a restaurant on the list. After selecting the
restaurant, customer can browse the menu and add item into cart until they are satisfied. The last process is to confirm the order.

Figure 28 - Sequence Diagram of FoodServe, condition_dine_in_order

Figure 28 is the sequence diagram of handling dine-in condition. When the order is dine-in, table number will be required. Customer can scan QR code on table to capture the table number. After setting the table number, a list of category will be shown. If the order type is not dine-in, category list will be shown immediately for menu browsing.
Figure 29 shows the flow of processes in browsing menu. First of all, the customer will select a category in the category list. Secondly, a list of item will be shown according to the selected category. Then customer can select an item from the list and item information will be displayed. Finally, the customer can add the item to the order and the order cart will be shown.
Figure 30 - Sequence Diagram of FoodServe, add_item_to_cart

Figure 30 shows the function flow of adding an item into cart. First of all, the customer can select the size, quantity and choice. Then add the item to order. Data will be stored in order entity, and the screen of order cart will show up.

Figure 31 - Sequence Diagram of FoodServe, confirm_order

Figure 31 shows the function flow of confirm order. A function call will be made to order entity when confirm order. The order entity will send the order to backend for further processing. Backend server will insert a new record in database. Lastly, a success will be returned from backend server, and then show the order information in another interface.
4. System Implementation

In this section, we will introduce the implementation detail of system functions.

4.1 Login

There are two login interfaces in FoodServe. First interface is for restaurants which is a web application.

![Screen capture from FoodServe, web application login](image)

Figure 32 - Screen capture from FoodServe, web application login
Second interface is for customers which is a mobile application.

![Figure 33 - Screen capture from FoodServe, mobile application login](image)

After logging in, backend system will validate the email and password accordingly. First checking would be the email is exists in the system or not. If the email is valid, then check if the password is match or not. The last checking is the user type, access will be denied if the user type does not have the permission. If the user login success, the information of user will be store in session storage to indicate the login is success.
4.2 Menu Creation

Restaurants can create menu by adding items and combos for their menu. However, adding an item or combo will require a large set of information such as category, size, option and schedule. It is not user friendly to ask for all information in a single page because some information is optional result as excessive input confusing restaurants.

Therefore, a setup wizard like process flow is adapted for adding an item / combo. That means the process of adding an item would be in divided into different parts and process in linear approach. Instead of single page input, linear approach can minimize the input required in each page, thus providing clear and simple interface for restaurants. Following is the approach of adding an item.

![Figure 34 - Approach of adding an Item](image-url)
Following is the steps to add an item to the menu.

Step 1: Choose or create a category for the item

Figure 35 - Screen capture from FoodServe, create an item: choose a category

Figure 36 - Screen capture from FoodServe, create an item: add a new category
Step 2: Enter basic information of the item

Figure 37 - Screen capture from FoodServe, create an item: basic information
Step 3: Choose suitable advance setup

Figure 38 - Screen capture from FoodServe, add an item: advance setup

If no advance setup is needed, skip to the summary.
Step 4: Choose or create sizes for the item

Figure 39 - Screen capture from FoodServe, add an item: choose existing size

Figure 40 - Screen capture from FoodServe, add an item: create a new size
Press "Add to item" to assign the size to the item. User can set different for different size of the item.

Figure 41 - Screen capture from FoodServe, add an item: set price of size
Step 5: Choose or create options for the item

Figure 42 - Screen capture from FoodServe, add an item: choose existing options
User can set additional price for the choice. The "Required" means customer must choose one of the choices when ordering this item.

Figure 43 - Screen capture from FoodServe, add an item: create a new option
Press "Add to item" to assign the option to the item.

Figure 44 - Screen capture from FoodServe, add an item: item's options
Step 6: Choose or create schedule for the item

Figure 45 - Screen capture from FoodServe, add an item: choose existing schedule
Or create a new schedule instead of choosing existing one.

Figure 46 - Screen capture from FoodServe, add an item: create a new schedule
Summary

Figure 47 - Screen capture from FoodServe, add an item: summary

Press confirm to complete the process.

Figure 48 - Screen capture from FoodServe, add an item: successful
After adding the item, restaurants can browse the menu. They can enable or disable items according to their needs.

![Figure 49 - Screen capture from FoodServe, browse item menu](image)

FoodServe: A Self-Service Platform for Restaurants
Restaurants can view the created size, option and schedule.

**Figure 50 - Screen capture from FoodServe, browse size**

**Figure 51 - Screen capture from FoodServe, browse options**

**Figure 52 - Screen capture from FoodServe, browse schedule**
Following is the steps to add a combo to the menu.

Step 1: Choose or create a category for the combo

Figure 53 - Screen capture from FoodServe, add a combo: choose existing category

Figure 54 - Screen capture from FoodServe, add a combo: add a new category
Step 2: Enter basic information of the combo

Restaurants needs to provide the name, description and price of the combo. Image is optional.

![Figure 55 - Screen capture from FoodServe, add a combo: basic information](image_url)
Step 3: Choose or create component for the combo

If restaurant had created components before, they can choose from existing component instead of creating a new one.

Figure 56 - Screen capture from FoodServe, add a combo: choose component
When creating a component, press the "+" to add item for the component.

![Figure 57 - Screen capture from FoodServe, add a combo: create new component](image1)

Restaurants can select more than one item for the component to form a group.

![Figure 58 - Screen capture from FoodServe, add a combo: select item for component](image2)
Press "Add to Combo" to assign the component to the combo.

Figure 59 - Screen capture from FoodServe, add a combo: combo’s component

After setting the component, customer can choose one item from each component when ordering this combo.
Step 4: Choose or create schedule for the combo

It is same as item, restaurants can choose no schedule, choose existing schedule or create a new one.

Figure 60 - Screen capture from FoodServe, add a combo: schedule
Summary

![Screen capture from FoodServe, add a combo: summary](image1)

**Figure 61** - Screen capture from FoodServe, add a combo: summary

Complete

![Screen capture from FoodServe, add a combo: successful](image2)

**Figure 62** - Screen capture from FoodServe, add a combo: successful
4.3 Order Processing

Customers can place order through their smartphone using FoodServe mobile application. Following is the step to place an order.

![Figure 63 - Screen capture from FoodServe, select order type](image)

Customers can select seat-in, take-out or delivery order then proceed to next step by pressing next.
After selected order type, a list of restaurants will be shown. Customers can filter restaurants by location.
A list of item category will be displayed after selecting a restaurant. Customer can select either an item category or a combo category.
According to the category that the customer had chosen, a list of items or combos will be shown. Customers can select an item to view its information.
In this page, customers can view the information of item such as the name, description, size, options. Customer can add this item into order cart.
Figure 68 - Screen capture from FoodServe, finish add item

After adding an item into order cart, customer can either add more items or check out.
Figure 69 - Screen capture from FoodServe, order cart

In the order cart, a short summary of ordered item will be shown including the name of restaurant, type of order and order detail.
The customer can choose to pay by credit card or by cash. After submit order, the page will be redirect to payment page or order information page.

![Figure 70 - Screen capture from FoodServe, order cart part 2](image)

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seafood Pizza (Large), $130 x 1</td>
<td>$131</td>
</tr>
<tr>
<td>Extra Cheese + $1</td>
<td></td>
</tr>
</tbody>
</table>

Total Amount: $131
In the meantime, after submitted an order, the responsible restaurant can see the upcoming order through web application. The status of an order will be updated in real-time using AJAX call.
In the kitchen screen, the restaurant can see the dishes that need to be done. Unlike the order receiving page, the information of kitchen screen is reduced into minimum amount because kitchen does not required other information such as the price of item, customer name, address, etc. After completed a dish, kitchen can press DONE to notify and update related order.

Figure 72 - Screen capture from FoodServe, kitchen screen
Figure 73 - Screen capture from FoodServe, order updated

When all dish of an order is completed, the order status will be updated to “Dish Out”. The restaurant can press “End Order” to end the transaction.
4.4 User Registration

User register is an important step for customer when using the system. There are some concerns about user registration. First of all, the registration procedure should be simple and quick because customer would not like to spend lots of time on filling forms. Therefore, we tried to limit the required information as few as possible.

Figure 74 - Screen capture of FoodServe, user registration
Also, phone number is important for contacting and identifying a customer. It is necessary to ensure the phone number is valid and unique. So, our system has a mechanism to confirm the user registration. A Short Message Service (SMS) with randomly generated confirmation code will be sent to user’s phone. User has to enter the confirmation code to validate the phone number holder.

The confirmation code is randomly generated using timestamp as the seed. To send SMS to user's mobile phone, we used Nexmo API to achieve this goal. Nexmo provides an API for developer to send SMS to any country.
Following is php source code of sendSMS function.

```php
// Send SMS to mobile phone
function sendSMS($from, $phoneNo, $message) {
    $url = 'https://rest.nexmo.com/sms/json?';

    $text = $message;

    $fields = array(
        'api_key' => 'my_api_key',
        'api_secret' => 'my_api_secret',
        'from' => "$from",
        'to' => "$phoneNo",
        'text' => urlencode($text),
    );
    $field_string = '';

    foreach ($fields as $key => $value) {
        $field_string .= $key . '=' . $value . '&';
    }
    $field_string = rtrim($field_string, '&');
    $url .= $field_string;
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

    $result = curl_exec($ch);
    curl_close($ch);
    return $result;
}
```
A request containing parameters including the key and secret of api, sender, receiver and message of the SMS, is sent to the URL through HTTP GET method. After submitting request to the Nexmo API, an HTTP response will be received. The HTTP response will contain a JSON object that stores the receipt and status. The HTTP response will look like this:

```
{
  "message-count":"1",
  "messages":[
    {
      "status":"0",
      "message-id":"00000123",
      "to":"44123456789",
      "remaining-balance":"1.10",
      "message-price":"0.05"
      "network":"23410"
    }
  ]
}
```

After sending the request to Nexmo API, a SMS should be sent to the receiver’s phone successfully.
4.5 QR Code

Typically, customer would enter the table number by themselves. However, customer may incorrectly enter the table number, and it is not user friendly for customer to type in textbox. Therefore, QR code scanning would be an alternative for getting the table number. QR code is durable, small and able to be recognized by mobile phone [9].

![Screen capture of FoodServe, dine-in ordering](image)

**Figure 76 - Screen capture of FoodServe, dine-in ordering**
With the help of PhoneGap plugin, we can simply use a function call in JavaScript to enable QR-code scanning:

```javascript
this.scanQR = function() {
    window.plugins.barcodeScanner.scan(function(result) {
        var QRjson = $.parseJSON(''+result.text);
        currentTableNumber = QRjson.table_number;
    }, function(error) {
        alert("Scanning failed: " + error);
    });
};
```

Since the function is written in JavaScript, we don’t need to write another set of code for different mobile platform. PhoneGap will detect the type of mobile device and use corresponding plugin automatically. Therefore, the advantage of using PhoneGap is it is easy to implement and able to cross-platform.
On the other hand, Restaurants can generate QR code by entering the number of table in their restaurant through the web application.

![Figure 77 - Screen capture from FoodServe, generate QR code](image)

Our system will generate QR code accordingly by using an open source library, “PHP QR Code”. Following is php code for QR code generation:

```php
// String to generate QR code
$qrString = '{"restaurant_id":'.$restaurant_id.'", "table_number":'.$tableNumber.'}';
$filePath = $dir.'/qr_table_'.$tableNumber.'.png';

// Call the api to convert the string into QR code image
QRcode::png($qrString, $filePath, QR_ECLEVEL_H, 4);
```
The library provides options for generating QR code. For example, we can configure the Error Correction (EC) level of QR code. There are 4 levels of EC, which are high, medium, low and quartile. The EC level will affect the performance of decoding the QR code. Therefore, when a QR code label is damaged and become incomplete, it is still possible to decode it if the EC level is high enough.

On the other hand, our system allows restaurants to download the QR code one by one, or download all QR code as a zip file. Following is the php code of compressing the files:

```php
$restaurant_id = $GET['restaurant_id'];
$dir = 'qrCode/zip/'.$restaurant_id;
if (!file_exists($dir)) {
    mkdir($dir, 0777, true);
}

$source_dir = "qrCode/$restaurant_id/";
$now = date("Y-m-d");
$zip_file = "qrCode/zip/$restaurant_id/QRCode_$now.zip";
$file_list = Util::listDirectory($source_dir);

if (file_exists($zip_file)) {
    unlink($zip_file);
}

$zip = new ZipArchive();
if ($zip->open($zip_file, ZIPARCHIVE::CREATE) === true) {
    foreach ($file_list as $file) {
        if ($file != $zip_file) {
            $zip->addFile($file, substr($file, strlen($source_dir)));
        }
    }
    $zip->close();
}
header("location: tableQR_ZipDownload.php?zip_file=".$zip_file);
This function will zip the images of QR code that belongs to the restaurant into a .zip file and name it with the creation date. And then redirect to the php for downloading the zip file. Following is the code for downloading the zip file:

```php
$zip_file = $_GET['zip_file'];
$file_name = basename($zip_file);
header("Content-Type: application/zip");
header("Content-Disposition: attachment; filename=" . $file_name);
header("Content-Length: " . filesize($zip_file));
header("Pragma: no-cache");
header("Expires: 0");
readfile($zip_file);
```

After changing the header to zip file, then read the target zip file, download will be trigger in browser automatically.
4.6 Delivery Ordering

For delivery ordering, the area of delivery is important for both restaurant and customer. Restaurant should be able to determine the delivery area. Meanwhile, customer should not be able to place an order that is out of delivery area.

To determine the delivery area, usually it would be by distance as radius. Yet, it is not accurate enough since the area would be result in circle. Hence, we decided to determine the delivery area of the restaurant by polygon instead. Polygon provides better flexibility than circle because it is possible to draw in any shape to cover the area they want or ignore the area they don’t want while the area of a circle is fixed according to the radius.

Figure 78 - Screen capture of FoodServe, setup delivery area
Restaurant can draw a polygon on Google map easily. And drag the pin point on the map to modify the shape. The advantages of using polygon as delivery area is restaurant can see the area immediately which helps them to define delivery area accurately.
Following are implementation details of Google API:

```javascript
function initialize() {
    var polyOptions = {
        strokeWeight: 0,
        fillColor: '#FF0000',
        fillOpacity: 0.35,
        editable: true
    };

    // Creates a drawing manager attached to the map that allows the user to draw
    // markers, lines, and shapes.
    drawingManager = new google.maps.drawing.DrawingManager({
        //drawingMode: google.maps.drawing.OverlayType.POLYGON,
        drawingControl: false,
        polygonOptions: polyOptions,
        map: map
    });

    google.maps.event.addListener(drawingManager, 'overlaycomplete', function(e) {
        // Switch back to non-drawing mode after drawing a shape.
        drawingManager.setDrawingMode(null);

        // Add an event listener that selects the newly-drawn shape when the user
        // mouses down on it.
        var newShape = e.overlay;
        newShape.type = e.type;
        deliveryArea = newShape;
    });
}
```

In the initialize function, we used the “DrawingManager” of Google Map for handling the drawing function. We defined the options for polygon, such as the color, opacity and most importantly, the polygon should be editable. After created the drawing manager, the next step is to handle the event when the drawing is completed. Therefore, we add an event listener to the overlaycomplete event. In the event listener, we switch the drawing mode to non-drawing mode first to stop the drawing action. Then, we get the overlay, which is the shape that drawn on the map, and store into a variable deliveryArea for reference.
After initialization, we have a drawing manager and the event listener for overlaycomplete. However, the drawing will not start until we set the drawing mode.

```javascript
function startDrawShape() {
    deliveryArea.setMap(null);
    drawingManager.setDrawingMode(google.maps.drawing.OverlayType.POLYGON);
}
```

Therefore, when the create delivery area is clicked, startDrawShape() is called. startDrawShape() will first clear the shape that currently on the map, then set the drawing mode in polygon.
For mobile application, it is important to limit the customer delivery order to a certain area. Address of customer would be required when placing delivery order, then, our server would check whether the address is valid or not. Valid address means it could be identified by Google Map, and is within the delivery area.

To check the address is valid or not, we used the Geocoder of Google Map API. Geocoder can convert an address into geographic coordinates like latitude and longitude, which can be used for determining the location. Following is JavaScript code for using the geocoder.

```javascript
var geocoder;
var geocoder_result;
var geocoder_status;

function codeAddress(address, callback) {
  geocoder_initiated = false;
  geocoder.geocode({'address': address}, function(results, status) {
    geocoder_status = status;
    if (status == google.maps.GeocoderStatus.OK) {
      geocoder_result = results[0].geometry;
      callback();
    } else {
      showToast("Invalid Address");
    }
  });
}

function initialize() {
  geocoder = new google.maps.Geocoder();
}
```

The function codeAddress accept 2 parameter, address and callback. In this function, it will convert the address into geometry as a result. If the status of geocode is not OK, meaning the address cannot be recognized by the Google Map API, thus invalid. If the geocoding is success, we will call the callback function for further processing if necessary.
To check if the address is inside the delivery area polygon, ray-casting algorithm is adapted. Ray-casting algorithm can be used for determining that a point is within a polygon or not, which suits our situation.

Following is PHP code of ray-casting algorithm.

```php
function rayCrossesSegment($targetLatLng, $a, $b) {
    $px = $targetLatLng->lng();
    $py = $targetLatLng->lat();
    $ax = $a->lng();
    $ay = $a->lat();
    $bx = $b->lng();
    $by = $b->lat();
    if ($ay > $by) {
        $ax = $b->lng();
        $ay = $b->lat();
        $bx = $a->lng();
        $by = $a->lat();
    }
    // alter longitude to cater for 180 degree crossings
    if ($px < 0) $px += 360;
    if ($ax < 0) $ax += 360;
    if ($bx < 0) $bx += 360;
    if ($py == $ay || $py == $by)
        $py += 0.000000001;
    if (($py > $by || $py < $ay) || ($px > max($ax, $bx)))
        return false;
    if ($px < min($ax, $bx))
        return true;
    $red = ($ax != $bx) ? (($by - $ay) / ($bx - $ax)) : INF;
    $blue = ($ax != $px) ? (($py - $ay) / ($px - $ax)) : INF;
    return ($blue >= $red);
}
```
The function rayCrossesSegment is used by point in polygon function. This function will return true if the ray starting from P intersect with segment AB.

The variable px and py represent the starting point of the ray. ax and ay represent the end-point of the segment with the smallest y coordinate. bx and by represent the end-point of the segment with the largest y coordinate. Therefore, A must be below B. Then, alter the longitude when necessary. if (($py > $by || $py < $ay) || ($px > max($ax, $bx))), that means it is out of bound, thus it is not intersect the segment AB. So if the py is valid, if ($px < min($ax, $bx)), then it is intersect the segment AB. Red represent a line draw from ax to bx while blue represent a line draw from ax to px. When blue is greater than red, that means P is intersect with the segment AB.

```php
function isInPolygon($path, $lat, $lng) {
    $targetLatLng = new LatLng($lat, $lng);

    $crossings = 0;

    foreach ($path as $key => $val) {
        $a = $val;
        $j = $key + 1;
        if ($j >= count($path)) {
            $j = 0;
        }
        $b = $path[$j];
        if (rayCrossesSegment($targetLatLng, $a, $b)) {
            $crossings++;
        }
    }
    // odd number of crossings?
    return ($crossings % 2 == 1);
}
```
The function isInPolygon will check the number of crossing between the point and the path. If the crossing is odd number, it implies that the point is in polygon. The explanation of ray-casting is as follow.

Assume there are point P, and a polygon. If we move P towards the polygon, when it crosses a border, the crossing would be inside-outside or outside-inside. If the last crossing is outside, that means P is starting inside a polygon. So if the crossing is odd number, which means it could give:

- 1: inside-outside
- 3: inside-outside, outside-inside, inside-outside
- 5: inside-outside, outside-inside, inside-outside, outside-inside, inside-outside
- ...

Therefore, when the number of crossing if odd number, P must be in a polygon.
4.7 Payment Gateway

Credit card payment is convenient for customers because it avoids the trouble when exchanging cash. To adapt the Braintree payment gateway, we need to implement on both client side and server side. For the client side, which is the mobile application, we used JavaScript provided by Braintree to encrypt the data that send to server side.
Javascript code as following:

```javascript
var braintree = Braintree.create("my_private_key");
braintree.onSubmitEncryptForm('paymentForm', ajax_submit);

var ajax_submit = function(e) {
    form = $('form');
    e.preventDefault();
    $('#submit').attr('disabled', 'disabled');
    $.post(
        serverPath + "process_payment.php",
        {
            "order_id": currentOrder.order_id,
            "amount": currentOrder.getTotalAmount(),
            "number": $('txt_card_number').val(),
            "cvv": $('txt_card_cvv').val(),
            "month": $('txt_card_month').val(),
            "year": $('txt_card_year').val(),
        },
        function(data) {
            // console.debug(data);
            if (data.success) {
                showToast(data.msg);
                $('#submit').attr('disabled', false);
                currentOrder.status = parseInt(data.orderStatus);
                currentShowOrderId = -1;
                $.mobile.changePage("showOrderDetail.html");
            } else {
                showToast(data.msg);
                $('#submit').attr('disabled', false);
            }
        },
        'json'
    );
    return false;
};
```
For the server side, the php code of payment processing is as follow:

```php
Braintree_Configuration::environment('sandbox');
Braintree_Configuration::merchantId('my merchant id');
Braintree_Configuration::publicKey('my public key');
Braintree_Configuration::privateKey('my private key');

$order_id = $_POST['order_id'];
$amount = $_POST['amount'];

$result = Braintree_Transaction::sale(array(  
    "amount" => $amount,
    "creditCard" => array(  
        "number" => $_POST["number"],
        "cvv" => $_POST["cvv"],
        "expirationMonth" => $_POST["month"],
        "expirationYear" => $_POST["year"]
    )
));
$data['success'] = false;
if ($result->success) {
    // update database
} else if ($result->transaction) {
    $data['msg'] = "Error processing transaction";
} else {
    $data['msg'] = "Validation errors";
}
echo json_encode($data);
```

To begin with, we set up the configuration of Braintree library with the id, public key and private key. Then, we can start the transaction by calling Braintree_Transaction::sale(), passing the information of credit card and amount. The decryption of credit card information is done by the Braintree library automatically. The result will indicate the transaction is successful or not.
4.8 Workflow Customization

Workflow is a sequence of ordering process. However, each restaurant may have different practice of order handling. FoodServe provides a function that allows restaurants to modify the sequence of ordering process, result as different behavior of ordering. Firstly, FoodServe support three type of ordering, namely dine-in, take-out and delivery. Each type of ordering will have different set of ordering process.

![Figure 82 - Screen capture from FoodServe, workflow customization](image)

For example, the ordering process of dine-in would be the customer browse the menu and place an order, and then the customer will pay the order. After that, kitchen will be notified for meal preparation. When the meal is prepared, the dish will be out for the customer. And the last process is customer finished their meal and the ordering process would be completed. However, this process may suitable for fast food restaurant but not Cha Chaan Teng.
So, in order to provide flexibility for other type of restaurants, restaurants can rearrange the sequence of ordering by drag and drop the component of the process.

Figure 83 - Screen capture from FoodServe, modified workflow sequence

The changed sequence become payment after finish dinner rather than payment before notify kitchen. After changing the sequence, the effect will be taken immediately on the mobile application.
4.9 e-Coupon

![Image of e-Coupon interface]

**Figure 84 - Screen capture from FoodServe, list of e-Coupon**

e-Coupon is a feature that allows restaurants to perform promotion by posting advertisement. FoodServe provides a simple interface for restaurants to create a coupon to attract customers. Restaurant can create e-Coupon easily with title, content and a quota.
After submitting the e-Coupon, FoodServe will notify customers that installed FoodServe mobile application by push notification. Push notification is a feature of mobile device that can alert customers without opening the mobile application. Unlike e-mail or SMS, push notification advertise is light weight and acceptable for customers. It is not like traditional spam therefore will not skip by customers, thus improve the effectiveness of the advertisement.
However, the target of push notification should be chosen carefully as it will affect the effectiveness of advertisement. To tackle this problem, FoodServe will push the notification of e-Coupon to nearby customer. Therefore, before pushing the notification, FoodServe will select the target customers according to their current location, and then push the notification if they are close to the restaurant. As a result, when customers receive the e-Coupon, the chance of visiting the restaurant is higher as their location is close to the restaurant.

Figure 86 - Screen capture from FoodServe, push notification
4.10 Recommendation

To provide better service to customers, FoodServe has a feature that recommends restaurants to customers. Customers can rate on each restaurant. According to customers' rating history, FoodServe can recommend similar restaurant that customers are interested.

Figure 87 - Screen capture from FoodServe, restaurant list with recommendation
To implement this feature, SlopeOne algorithm is adapted. SlopeOne algorithm is introduced by Daniel Lemire and Anna Maclachlan. SlopeOne algorithm is designed for item-based collaborative filtering based on rating.

There are two parts in SlopeOne algorithm, namely deviation calculation and score prediction.

For deviation calculation, it can be denoted as following:

\[
\text{dev}_{i,j} = \sum_{r \in P_{i,j}(U)} \frac{r_i - r_j}{\text{COUNT}(P_{i,j}(U))}
\]

For prediction calculation, it can be denoted as following:

\[
\text{PS}(u)_i = \frac{\sum_{j \in R(u) \backslash \{i\}} (\text{dev}_{i,j} + u_j) \text{COUNT}(P_{i,j}(U))}{\sum_{j \in R(u) \backslash \{i\}} \text{COUNT}(P_{i,j}(U))}
\]
To illustrate the calculation of Slope one algorithm, an example is provided as following.

Following is the table of rating:

<table>
<thead>
<tr>
<th></th>
<th>Wah Hing</th>
<th>Shing Ling</th>
<th>Perfect Lemon Cafe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>Betty</td>
<td>3</td>
<td>4</td>
<td>N/A</td>
</tr>
<tr>
<td>Candy</td>
<td>N/A</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

To calculate the deviation, use the equation:

\[
dev_{i,j} = \sum_{r \in P_{i,j}(U)} \frac{r_i - r_j}{\text{COUNT}(P_{i,j}(U))}
\]

For example, the deviation of Shing Ling Restaurant with respect to Wah Hing Restaurant is:

\[
\frac{2-3}{2} + \frac{3-4}{2} = -1.
\]
After calculated all pair of restaurant, a table of deviation can be shown as:

<table>
<thead>
<tr>
<th></th>
<th>Wah Hing</th>
<th>Shing Ling</th>
<th>Perfect Lemon Cafe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wah Hing</td>
<td>0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Shing Ling</td>
<td>-1</td>
<td>0</td>
<td>0.5</td>
</tr>
<tr>
<td>Perfect Lemon Cafe</td>
<td>-3</td>
<td>0.5</td>
<td>0</td>
</tr>
</tbody>
</table>

With the help of deviation, we can predict an item for a user. For example, the predict score of Perfect Lemon Café for Betty will be:

\[
PS(u)_i = \frac{\sum_{j \in R(u)-\{i\}}(dev_{i,j} + u_j)\text{COUNT}(P_{i,j}(U))}{\sum_{j \in R(u)-\{i\}}\text{COUNT}(P_{i,j}(U))}
\]

\[
= \frac{((1 + 3) \times 1) + ((0.5 + 4) \times 2)}{3} = \frac{4}{3} = 1.3
\]

As a result, the predict score is 1.3. To recommend restaurants, we need to calculate for each restaurant, and then recommend restaurants with higher predict score.
## 5. Evaluation

### Test cases:

<table>
<thead>
<tr>
<th>Test case ID</th>
<th>Test case description</th>
<th>Test data</th>
<th>Expected output</th>
<th>Actual output</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>Restaurant login with valid data</td>
<td>user name: <a href="mailto:deliWine@food-serve.com">deliWine@food-serve.com</a> password: 123123</td>
<td>Login success with restaurant role</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC2</td>
<td>Restaurant login with invalid password</td>
<td>user name: <a href="mailto:deliWine@food-serve.com">deliWine@food-serve.com</a> password: abcabc</td>
<td>Login fail, error invalid username or password</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC3</td>
<td>Admin login with valid data</td>
<td>user name: <a href="mailto:admin@food-serve.com">admin@food-serve.com</a> password:123123</td>
<td>Login success with admin role</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC4</td>
<td>Admin login with invalid password</td>
<td>user name:<a href="mailto:admin@food-serve.com">admin@food-serve.com</a> password:abcabc</td>
<td>Login fail, error invalid username or password</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC5</td>
<td>Login with invalid email</td>
<td><a href="mailto:abc@gmail.com">abc@gmail.com</a> password: 123123</td>
<td>Login fail, error invalid username or password</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC6</td>
<td>Add an item: Category, proceed with an existing category</td>
<td>category: Dessert</td>
<td>Continue to next step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC7</td>
<td>Add an item: Category, proceed with adding a new category with valid value</td>
<td>category: Pasta</td>
<td>Continue to next step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC8</td>
<td>Add an item: Category, proceed with adding a new category with empty value</td>
<td>category: null</td>
<td>Error: invalid category</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC9</td>
<td>Add an item: Basic Information, proceed with valid data</td>
<td>item name: Chocolate cake item description: Sweet and tasty price: 25</td>
<td>Continue to next step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC10</td>
<td>Add an item: Basic Information, proceed with invalid price</td>
<td>item name: Chocolate cake item description: Sweet and tasty price: abc</td>
<td>Error: invalid price</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC</td>
<td>Description</td>
<td>Details</td>
<td>Expected Outcome</td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
<td>---------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>TC11</td>
<td>Add an item: Basic Information, proceed with empty item name</td>
<td>Item name: Sweet and tasty, price: 25</td>
<td>Error: invalid item name</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC12</td>
<td>Add an item: Basic Information, proceed with empty item description</td>
<td>Item name: Chocolate cake, item description: price: 25</td>
<td>Error: invalid item description</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC13</td>
<td>Add an item: Basic Information, proceed with empty price</td>
<td>Item name: Chocolate cake, item description: Sweet and tasty, price:</td>
<td>Error: invalid price</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC14</td>
<td>Add an item: Advanced Setup with all advanced setup set to No</td>
<td>Multiple size: No, Options: No, Schedule: No</td>
<td>Continue to last step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC15</td>
<td>Add an item: Advanced Setup, with all advanced setup set to Yes</td>
<td>Multiple size: Yes, Options: Yes, Schedule: Yes</td>
<td>Continue to multiple size, and then options, schedule</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC16</td>
<td>Add an item: Size, add an existing size to the item</td>
<td>Size: Large</td>
<td>Size is added to item</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC17</td>
<td>Add an item: Size, create a new size for the item</td>
<td>Size: Small</td>
<td>Size is added to item</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC18</td>
<td>Add an item: Size, create a new size for the item with empty input</td>
<td>Size: null</td>
<td>Error: invalid size</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC19</td>
<td>Add an item: Size, set a valid price for the item's size</td>
<td>Price of size: 30</td>
<td>Set price successful</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC20</td>
<td>Add an item: Size, set an invalid price for the item's size</td>
<td>Price of size: abc</td>
<td>Error: invalid price</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC21</td>
<td>Add an item: Size, proceed without assigning size to the item</td>
<td>Item's size: null</td>
<td>Error: No size is assigned</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC22</td>
<td>Add an item: Option, add an existing option to the item</td>
<td>Option: Peanuts</td>
<td>Option is added to item</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC23</td>
<td>Add an item: Option, create a new option for the item</td>
<td>Option name: Sugar Choice: - More sugar, 1$, - Less sugar, 0$</td>
<td>Option is added to item</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC24</td>
<td>Add an item: Option, proceed without assigning option to the item</td>
<td>Item's options: null</td>
<td>Error: No option is assigned</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC25</td>
<td>Add an item: Schedule, choose an existing schedule for the item</td>
<td>Schedule: Lunch</td>
<td>Continue to next step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC26</td>
<td>Add an item: Schedule, create a new schedule for the item</td>
<td>Schedule name: Breakfast Day: - Monday, 10:00 - 12:00</td>
<td>Continue to next step</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC</td>
<td>Description</td>
<td>Expected Outcome</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC27</td>
<td>Add an item: Schedule, create a new schedule for the item with invalid time</td>
<td>Error: invalid time</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC28</td>
<td>Add an item: Summary, proceed with confirm</td>
<td>Insert item record to database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC29</td>
<td>Rearrange workflow sequence of seat-in order</td>
<td>Take effect immediately on mobile application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC30</td>
<td>Workflow setting: change minimum charge to 0</td>
<td>Save successful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC31</td>
<td>Workflow setting: change minimum charge to 1</td>
<td>Save successful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC32</td>
<td>Workflow setting: change minimum charge to -1</td>
<td>Error: invalid minimum charge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC33</td>
<td>Workflow setting: change minimum charge to abc</td>
<td>Error: invalid minimum charge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC34</td>
<td>Workflow setting: add a new row of schedule</td>
<td>a new row will be shown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC35</td>
<td>Workflow setting: remove a row of schedule</td>
<td>a row will be disappeared</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC36</td>
<td>Workflow setting: remove the only row of schedule</td>
<td>nothing happen</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC37</td>
<td>Delivery Area: draw a square on map and save</td>
<td>display a square on map</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC38</td>
<td>Delivery Area: draw a triangle on map and save</td>
<td>display a triangle on map</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC39</td>
<td>Delivery Area: draw a rectangle on map and save</td>
<td>display a rectangle on map</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC40</td>
<td>Delivery Area: draw a polygon on map and save</td>
<td>display a polygon on map</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC41</td>
<td>Delivery Area: remove delivery area</td>
<td>remove successful</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC42</td>
<td>Delivery Area: search a valid location</td>
<td>focus the map on search result</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC43</td>
<td>Delivery Area: search an invalid location</td>
<td>Error: invalid address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC44</td>
<td>QR Code: generate QR code with 1 table</td>
<td>generate 1 QR code image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC45</td>
<td>QR Code: generate QR code with 0 table</td>
<td>Error: invalid table number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC46</td>
<td>QR Code: generate QR code with -1 table</td>
<td>Error: invalid table number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC47</td>
<td>QR Code: generate QR code with non-integer table</td>
<td>Error: invalid table number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC48</td>
<td>QR Code: download a QR code</td>
<td>Download a QR code image</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC49</td>
<td>QR Code: download all QR code</td>
<td>Download a zip file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC50</td>
<td>QR Code: print QR code</td>
<td>print setting popup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC51</td>
<td>Item menu: enable an item</td>
<td>press enable on an item</td>
<td>item hide in menu on mobile app</td>
<td>As Expected</td>
</tr>
<tr>
<td>------</td>
<td>-------------------------</td>
<td>------------------------</td>
<td>---------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>TC52</td>
<td>Item menu: disable an item</td>
<td>press disable on an item</td>
<td>item shown in menu on mobile app</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC53</td>
<td>Combo menu: enable an combo</td>
<td>press enable on a combo</td>
<td>combo hide in menu on mobile app</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC54</td>
<td>Combo menu: disable an combo</td>
<td>press disable on a combo</td>
<td>combo shown in menu on mobile app</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC55</td>
<td>mobile app: place an order</td>
<td>restaurant: deliWine item: hot coffee order type: dine in</td>
<td>place order successful, backend receive order</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC56</td>
<td>mobile app: scan a QR code for table number</td>
<td>scan QR code</td>
<td>capture the table number successful</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC57</td>
<td>mobile app: user registration</td>
<td>email: <a href="mailto:ccn223@gmail.com">ccn223@gmail.com</a> password: 123</td>
<td>SMS confirmation code sent to the phone</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC58</td>
<td>mobile app: valid credit card payment</td>
<td>credit card no: 4111111111111111 cvv: 123 month: 12 year: 2014</td>
<td>payment successful with reference</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC59</td>
<td>mobile app: invalid credit card payment</td>
<td>credit card no: 4111111111111111 cvv: 000 month: 12 year: 1999</td>
<td>Error: invalid credit card</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC60</td>
<td>mobile app: view order history</td>
<td>press order history</td>
<td>display a list of order history</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC61</td>
<td>mobile app: search nearby restaurant</td>
<td>press nearby restaurant</td>
<td>display nearby restaurant</td>
<td>As Expected</td>
</tr>
<tr>
<td>TC62</td>
<td>mobile app: recommendation</td>
<td>press recommendation</td>
<td>display recommended restaurants</td>
<td>As Expected</td>
</tr>
</tbody>
</table>
6. Future Improvements

As time for the project is limited, there are attractive features that didn’t implement in FoodServe. In this section, we would discuss the improvements in future if possible.

6.1 Social Network

Social network is popular nowadays. Feature like sharing in Facebook / Twitter are important for a platform because it can bring positive effect on advertising the restaurant. Account linking with Facebook helps customers to access FoodServe without registering a new account on FoodServe. These social network feature function should be implemented in the future.

6.2 e-Ticket Queue

Long queue may appear often in restaurant during rush hour. Ticket queue is helpful for managing customers and thus save time. If the basic function of issue ticket is implemented, advanced feature like time estimation on table turn should be considered. Time estimation can benefits both restaurants and customers in many ways. For example, customers can find a nearest restaurant with available seat, or with least waiting time. Therefore, e-Ticket queue should be implemented if time is available.
7. Conclusion

As there are lots of restaurants in Hong Kong, the needs of application platform for restaurants is very high. FoodServe provides a self-service platform for restaurants to solve the problems that restaurants are facing.

Menu creation helps restaurants to manage their menu easier. FoodServe provides basic functionality of menu creation. Meanwhile, FoodServe also provides advanced functionality of menu creation for those who want extra information of items. Digital menu allows restaurants to provide better services on ordering.

Workflow customization allows restaurants to rearrange the process flow of ordering. Since different restaurants may have different behavior and culture on order handling, the functionality of workflow customization can satisfy user.

e-Coupon can advertise restaurants with push notification feature on smartphone. The push notification will be sent to customers, who are close to the restaurants, thus improving the effectiveness of advertisement.

Cross-platform mobile application allows customers to place order in a convenient way with QR code scanning, delivery address detection. Customers also can search nearby restaurants or recommended restaurants with their smartphone.

However, as time constraint, there is lots of improvement that had not been implemented. With those improvements, FoodServe can be shared in easier way and help restaurants to solve the queuing problem.

To conclude, FoodServe is a platform that provides complete and useful services to both restaurants and customers. Hence, bring the benefits of technology to users.
References

1. Tsang, C.Y. (2003). The food culture in Hong Kong and Taiwan. The University of Hong Kong


## Appendices

### Monthly Log

<table>
<thead>
<tr>
<th>Month</th>
<th>Completed Task:</th>
<th>Task in Progress:</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2013</td>
<td>- Food Ordering</td>
<td>- Workflow implementation</td>
</tr>
<tr>
<td></td>
<td>- Detection of delivery area</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- QR code scanning</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- BrainTree Payment function</td>
<td></td>
</tr>
<tr>
<td>November 2013</td>
<td>- Workflow implementation (Take-out ordering)</td>
<td>- Workflow implementation</td>
</tr>
<tr>
<td></td>
<td>- Bug fix</td>
<td>- Revise UI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Code refactoring</td>
</tr>
<tr>
<td>Month</td>
<td>Completed Task</td>
<td>Task in Progress</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------</td>
<td>----------------------------</td>
</tr>
</tbody>
</table>
| December 2013 | - QR Code Generation  
- Bug Fix  
Task in Progress:  
- Revise UI  
- Code Refactoring  
- Menu App Preview |                                                   |
| January 2014  | - Bug Fix  
- Revise UI  
- Workflow Setting  
Task in Progress:  
- Queuing System |                                                   |
<table>
<thead>
<tr>
<th>Month</th>
<th>Complete Task</th>
<th>Task in Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>February 2014</td>
<td>- Interim Report</td>
<td>- Push notification on mobile device</td>
</tr>
<tr>
<td></td>
<td>- Bug Fix</td>
<td>- Location based service</td>
</tr>
<tr>
<td></td>
<td>- Revise UI</td>
<td>- Recommendation function</td>
</tr>
<tr>
<td>March 2014</td>
<td><strong>Completed Task:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Push notification on mobile device</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Location based advertisement</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Recommendation function</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Task in Progress:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Final report</td>
<td></td>
</tr>
</tbody>
</table>