Food Delivery App Development

By

Chen Yang, Xiaoyang Lu

Submitted to
The Faculty of the school of Information Technology
In Partial Fulfillment of the Requirements for
The Degree of Bachelor of Science
In Information Technology

Chen Yang ____________________ Date: 4/18/2016
Chen Yang

____________________ Date: 4/18/16
Xiaoyang Lu

James J Scott ____________________ Date: 4-16-2016
James Scott **Faculty Advisor**

____________________ Date: 4/14/16
Robin Carew **Faculty Advisor**

University of Cincinnati
College of
Education, Criminal Justice, and Human Services
School of Information Technology
April 2016

Table of Contents

Abstract

"Foods On Wheels" developed by UC students Xiaoyang Lu and Chen Yang was published to the Android platform to solve massive food delivery problems. Worrying about the delivery range, different passwords for different food sites, or any restaurants' delivery limitations were no longer a problem. The application lists restaurants by area, and also offered drivers a chance to earn money by delivering food. After our enrollment verification, anyone can be a driver, accept orders, and earn money! Participating restaurants are waiting for your selection. Ordering food from those restaurants that once didn't deliver just became possible. Our slogan is: Foods on Wheels and earn on wheels!

Introduction

International food and restaurant consultants “Baum + Whitman” have released their forecast of the biggest trends in 2016. Food delivery stands at the top of the list (2015). Solving problems of food delivery service becomes a potential trend among technology companies. As IT students, we have our own way of optimizing food delivery procedures.

We developed a Food Delivery Application using the UBER model (Farhad Manjoo,Jan 28, 2015). Drivers can get online at any time to receive delivery requests from customers. Every restaurant would not need to hire a delivery person. Instead, drivers would complete the order made by customers and be paid through our application. There would be three parts to our application: customer, driver and restaurant host.

Problem Statement

In recent years, with the rapid developments in **f**ood delivery services, more and more restaurants have launched their own food delivery applications, in order to attract more customers as well as make it convenient for those who do not want to go out for dinner. Therefore, these kinds of application services have brought good news to customers, especially for salaried employees, because it can provide more choices on the application’s food menu and it allows for less wait time. It brings people a lot of convenience, but at the same time, problems may arise. For the restaurants, not only are they bearing all of the delivery cost, but they also must guarantee the food's freshness and quality and whether they can deliver on time.

Furthermore, food delivery service is a big challenge for restaurant delivery persons that are without professional training. According to market research, we can draw the following conclusions; (1) The delivery capacity is very limited for normal food outlets. Generally, each person is only able to carry about two to three servings of food at one time, (2) Being unfamiliar with the geographical position, even if they are only able to carry about two to three servings of food at one time, they are unable to outline a reasonable timetable and trips, (3) They are not professional drivers, so they don't have security during the period of delivery (Massachusetts safety offices league, 2012), (4) The efficiency is not high. Each delivery person is employed by a restaurant and they are unable to work for another restaurant until their contract ends and as a result they earn very little money each day, (5) In addition, customers have to apply for different account numbers for different restaurants. Not only it is very inconvenient, but also they might often forget the account number or password. It is really, in a sense, reducing the customer's information security.

In summary, for these reasons, it is essential for a food delivery application to develop a study on how to improve production and management efficiency, and reduce agency cost in competitive markets to maximize profit.

Solution

What we want to do is to develop a mobile application that could solve these problems. We want to reduce the cost restaurants are paying to their drivers, but we are not able to establish our own delivery team at the moment. We referenced UBER and Airbnb with their sharing economy model, and we decided to adopt that mode into our concept. Matthew Crosby, the manager of the Rocky Mountain Institute, offers us clues about the advantage of doing so: “Shared economy companies unbundle existing assets and enable value exchange out of those assets, with close to zero marginal capital cost since the users themselves own the actual physical assets, whether a car or a home” (2015). This mode would allow our application to offer lower costs and better service. That is, a driver does not need to be someone employed by the restaurant. Anyone who wants to be a driver could register for our application and commit to deliver food. This solves many problems at a time; (1) The restaurant could cut down the budget of hiring many drivers, (2) There usually could be more than 10 restaurants within a few blocks. This gives drivers’ broader business opportunities. They now are not limited in a unique restaurant but they have choices to accept the delivery orders they want, (3) A restaurant usually sets a range of delivery service, which if exceeded, they would refuse to deliver that order. While a driver could select whichever order he wants to work on, even though it would cost him 20 minutes getting there — they could get more tips by the customer, (4) Working time becomes more flexible. Drivers could get online at any time when they want to accept orders and deliver food, (5) Some restaurants do not offer delivery services but only carryout. Drivers could still deliver food for customers from such restaurants using our application.

We also did market research to find out if this mode could really bring drivers the profits they imagined. A Chinese restaurant owner nearby campus told us that they have 150 - 200 orders daily on average, which brings them $5500 - $6500 in income. About 30% - 40% of the entire income comes from delivery. They hire one driver or two daily drivers to work on delivery orders, and that could bring them $100 - $180 of tips income with working 6 to 8 hours (over 80% of the tips come from rush hours). So if a driver got online during rush hours and delivered food for two to three hours daily using our application, the profit would be as much as $80 to $150 a day, or, roughly $2000 a month. This makes it extremely profitable for a driver who wants to rely on this application full time.

The other problem we solved is that we united all of the restaurants in our application so that customers would only need one application on their phone. This also brings various benefits to customers while ordering food; (1) Being able to choose from a list of restaurants. This saves time for them to figure out from where they could order food, (2) Remembering a set of account numbers and passwords for different websites would be tough for elderly people, but now, this can all be on one application, (3) Many restaurants have their own reward system, while we might develop a new one that could be applied to every restaurant listed in our application.

After solving our potential problems, the next step is to design the application and its features.

Project Description

This application helps customers to order food delivery services from any restaurant, even restaurants that do not normally offer delivery. Moreover, this application helps restaurants to reduce the cost of hiring a delivery person, but offer delivery services at the same time. We developed an operation flow to explain the whole procedure of ordering a food delivery service using our application:

Step 1: This application could let customers view all restaurants based on a map from nearest to furthest, or from highest rating to lowest. Customers could make an order online for delivery, type in payment method for further usage, and set a tip amount or percentage.

Step 2: This application uses an UBER MODE to hire a delivery person. As long as one is approved by us, he or she could become a delivery person. A driver could get online anytime to receive orders. Once the driver is online, many orders from nearby would come from his or her client. Once they choose the orders he or she wants, and sets the arrival time, the order status becomes active, which means the customer is going to be charged at this time, also, the restaurant would receive this order with the information of the customer and delivery person.

Step 3: The delivery person is going to pick up the order from the restaurant. He or she does not need to pay any money since we would pay for this order using the money we charged. They only need to show their driver’s license to prove their identity. After they change the order status to “delivering,” the customer could track his route on the application.

Step 4: When the food is delivered, the delivery person would change the order status to "finish." The tip is going to be charged to the customer. The customer has a chance to rate this driver and a chance to draw rewards, including coupons, gift cards or etc.

Here is the system diagram of the application, listing all the features we are offering to our users:

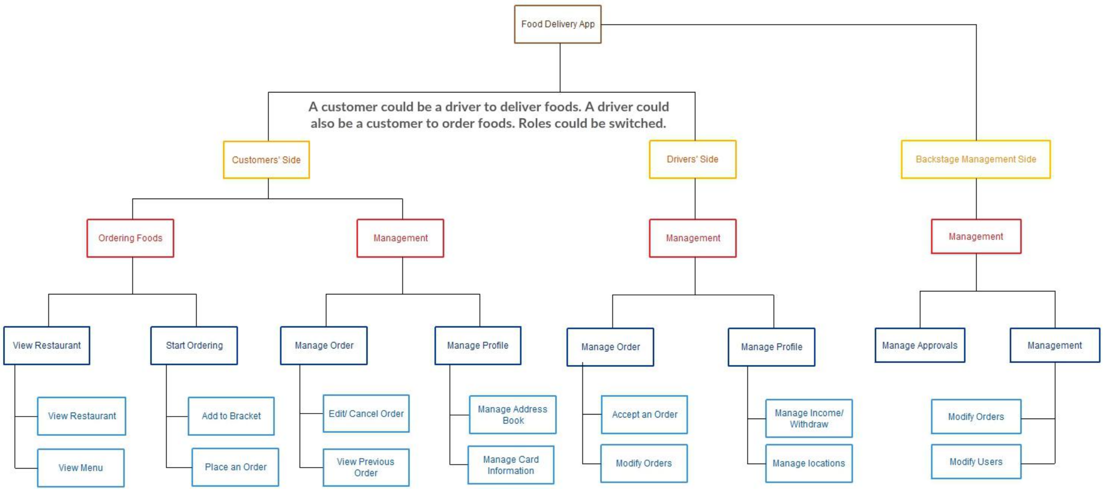


Figure1. Schematic of application.

User Profile

Potential Users:

Catering service industry, customers of all ages who needs a food delivery service.

Software and Interface Experience:

1. Customer's Side:

   Allow customers to browse restaurants by distance

   Allow customers to order food from a single restaurant

   Allow customers to complete payment via the app

   Allow customers to track order status

Allow customers to give feedback and see order history

2. Driver's Side

   Allow Driver to receive order requests

   Allow Driver to accept order requests and see detailed information

   Allow Driver to receive tips

3. Restaurant's Side

   Allow restaurant to accept order requests

   Allow restaurant to update menu

   Allow restaurant to receive payment

Experience with Similar Applications:

1. UBER – A taxi service app with users being both drivers and customers. (Farhad Manjoo, Jan 28, 2015)
2. GrubHub – An application for food delivery and carry out orders that allows users to browse all restaurants in a single page view.

Task Experience:

Customer's Side: No requirements

Driver's Side: Driving experience over 2 years

Restaurant's Side: Experience of managing orders / preparing for settlements through mobile devices.

Frequency of Use:

1. Drivers accepting orders (up to 8h/day)
2. Customers ordering food delivery (2 – 5 times/week)
3. Restaurants updating menu (1 time/month)
4. Restaurants receiving orders (up to 8h/day)

<u>Key Interface Design Requirements that the Profile Suggests:</u>

1. User Profile Data:
    a) Real Name
    b) Delivery Address
    c) Credit Card Information
    d) Billing Address
    e) Saved Restaurant
    f) Order History
2. Restaurant Profile Data:
    a) Menu Data
    b) Restaurant Name
    c) Restaurant Address
    d) Restaurant Phone Number
3. Verification System
    a) Driver's License
    b) Employer Identification Number

Technical Elements

Eclipse would be the main development tool, using JAVA. The application would generate a local SQLite database and external online MYSQL database as well. The application should be able to run under any Android client with version 4.3 or higher offering a GPS model and data transfer model.

According to our work done this semester, a brief technical diagram could be given as follows, which generally introduces the relationship between the software part and the server part. Logic codes are connected with PHP pages through JSON (JSON is called JavaScript Object Notation. It is the primary data format used for asynchronous browser/server communication, largely replacing XML (Wikipedia, 2001 introductions)), this is how local data is being sent and retrieved by the server. JSON controls all the interfaces located in *.php files. By using this method to communicate with the server-side database, passwords and payment information can be parsed more safely.

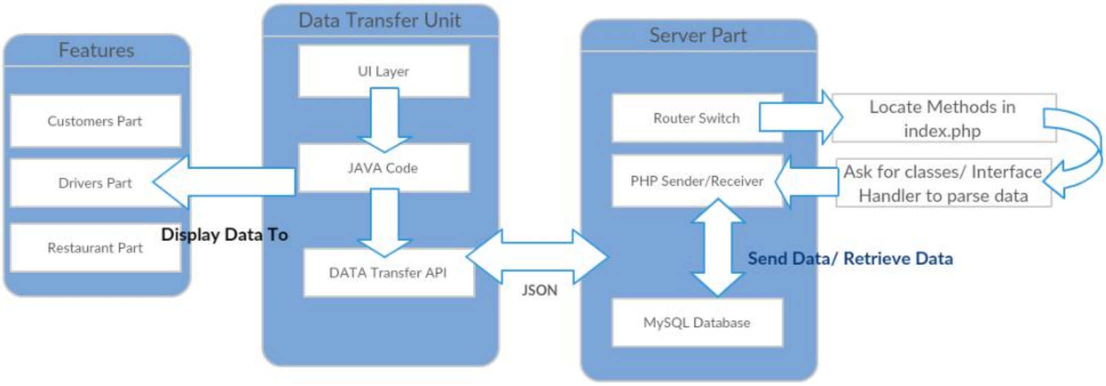
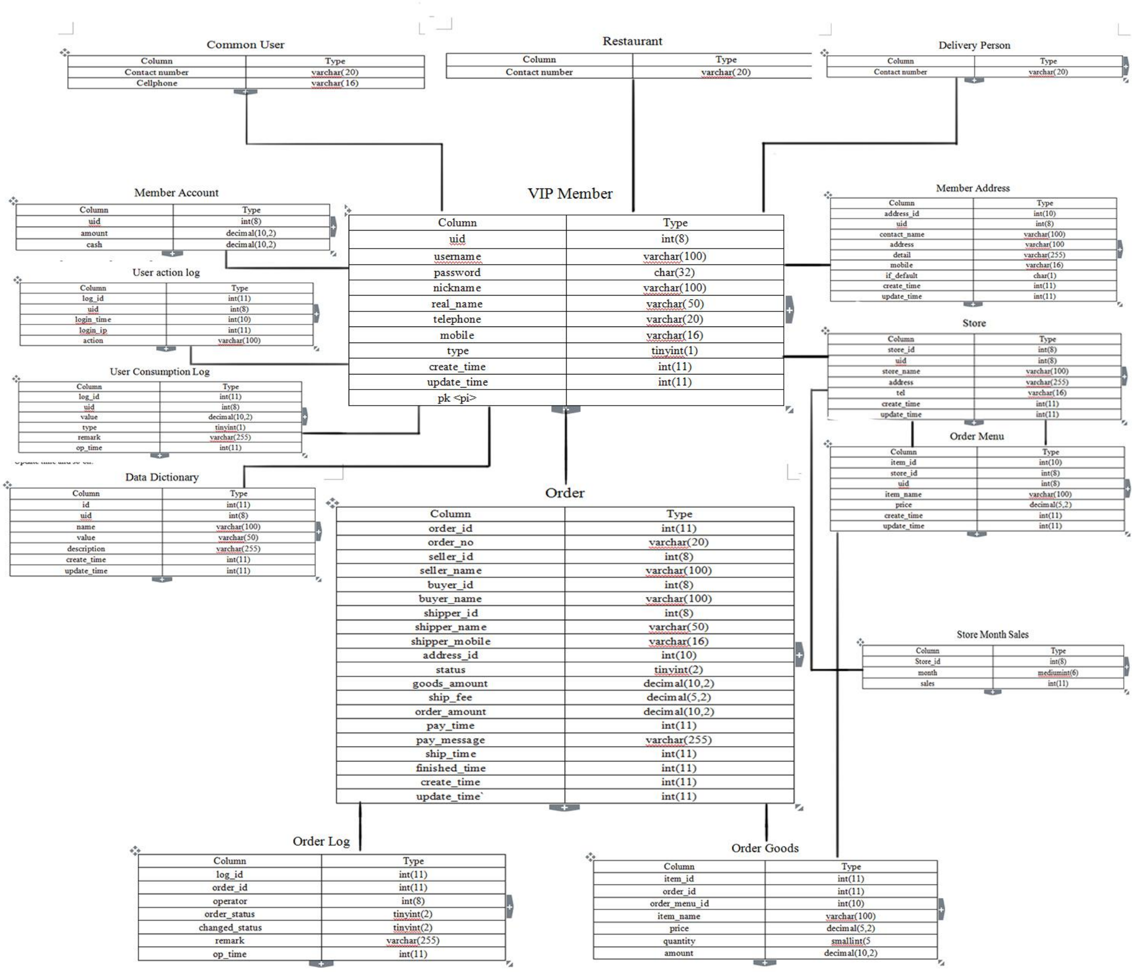


Figure 2. The Server Architecture Design Diagram

Also, the database management system we intend to use is MYSQL, which houses each restaurant, customer and delivery person’s information in the database.



Common User

Column	Type
Contact number	varchar(20)
Cellphone	varchar(16)

Restaurant

Column	Type
Contact number	varchar(20)

Delivery Person

Column	Type
Contact number	varchar(20)

Member Account

Column	Type
uid	int(8)
amount	decimal(10,2)
cash	decimal(10,2)

VIP Member

Column	Type
uid	int(8)
username	varchar(100)
password	char(32)
nickname	varchar(100)
real_name	varchar(50)
telephone	varchar(20)
mobile	varchar(16)
type	tinyint(1)
create_time	int(11)
update_time	int(11)
pk <pi>	

Member Address

Column	Type
address_id	int(10)
uid	int(8)
contact_name	varchar(100)
address	varchar(100
detail	varchar(255)
mobile	varchar(16)
if_default	char(1)
create_time	int(11)
update_time	int(11)

User action log

Column	Type
log_id	int(11)
uid	int(8)
login_time	int(10)
login_ip	int(11)
action	varchar(100)

Store

Column	Type
store_id	int(8)
uid	int(8)
store_name	varchar(100)
address	varchar(255)
tel	varchar(16)
create_time	int(11)
update_time	int(11)

User Consumption Log

Column	Type
log_id	int(11)
uid	int(8)
value	decimal(10,2)
type	tinyint(1)
remark	varchar(255)
op_time	int(11)

Order Menu

Column	Type
item_id	int(10)
store_id	int(8)
uid	int(8)
item_name	varchar(100)
price	decimal(5,2)
create_time	int(11)
update_time	int(11)

Data Dictionary

Column	Type
id	int(11)
uid	int(8)
name	varchar(100)
value	varchar(50)
description	varchar(255)
create_time	int(11)
update_time	int(11)

Order

Column	Type
order_id	int(11)
order_no	varchar(20)
seller_id	int(8)
seller_name	varchar(100)
buyer_id	int(8)
buyer_name	varchar(100)
shipper_id	int(8)
shipper_name	varchar(50)
shipper_mobile	varchar(16)
address_id	int(10)
status	tinyint(2)
goods_amount	decimal(10,2)
ship_fee	decimal(5,2)
order_amount	decimal(10,2)
pay_time	int(11)
pay_message	varchar(255)
ship_time	int(11)
finished_time	int(11)
create_time	int(11)
update_time`	int(11)

Store Month Sales

Column	Type
Store_id	int(8)
month	mediumint(6)
sales	int(11)

Order Log

Column	Type
log_id	int(11)
order_id	int(11)
operator	int(8)
order_status	tinyint(2)
changed_status	tinyint(2)
remark	varchar(255)
op_time	int(11)

Order Goods

Column	Type
item_id	int(11)
order_id	int(11)
order_menu_id	int(10)
item_name	varchar(100)
price	decimal(5,2)
quantity	smallint(5
amount	decimal(10,2)

Figure 3. The Database E-R Diagram.

In addition, we have designed the application architecture diagram in order to make the whole application's structure easy and clear. In our diagram, the HTTP client can get database information through the HTTP API interface, where the transmission tool is PHP and JSON protocol. Then, HTTP Client can transfer all database information to Android GUI and display. For the convenience of the API interface development that we supported the API Debug Backstage between the API interface and MYSQL database layer. As a result, we can test and check that all of the server interfaces are connected to HTTP client successfully through the API Debug Backstage.

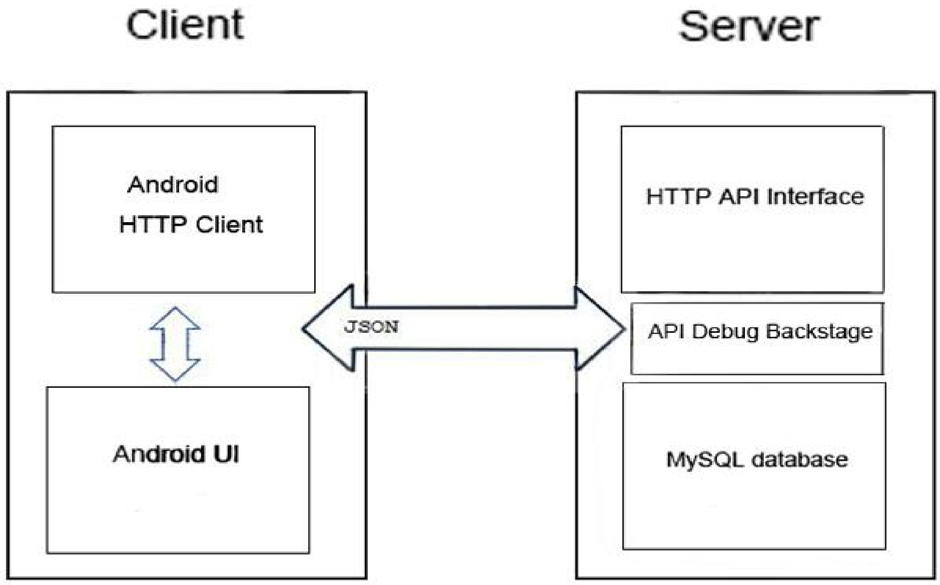


Figure 4. The Application Architecture Design Chart

Moreover, we will set up the application back-end server on the VM (Application-V Management Server, Application-V Client, and Application-V Sequencer Serve). App-V Management Server is the core of the Application virtualization server (Wilco Van Bragt, 18 Dec.2013). Application virtualization does so by Application-V Management Server applied to the Active Directory environment or in conjunction with SQL server database as well as client authentication.

Objective/Deliverables

Fall semester (8/24/2015-12/06/2015)

I. User Interface

   a. Sign in/ up page

   b. Menu Management

   c. Profile Management

   d. Menu List View

   e. Payment Preview

II. Features and Functions

   a. Connections to database

   b. Fetch from database

   c. User Control Model

   d. Profile Handler

   e. Payment Simulator

   f. List View

III. Web Server Interface

   a. Backup Server Management

   b. Database Management

   c. Database List View

IV. Network Interface

   a. Server Interface Connection

   b. Database Connection

V. User Experience

   a. Menu Debug

   b. Payment Procedure Debug

   c. Database Connection Debug

   d. Optimizing process

Spring Semester (1/11/2016-4/15/2016)

I. User Interface

    a. Order Management
    b. Drivers' Page

II. Features and Functions

    a. Accept Order Model
    b. Order Status Model
    c. Payment Confirm Model
    d. Interaction Model

III. Test

    a. Test Plan Determination
    b. Application Functionality Test
    c. API Test
    d. Database Test
    e. Backup Server Test
    f. Finalizing and Debugging

Proposed Budget

The project now has a budget under $100 for the server fee and construction fee of the network environment. This may increase as a low cost server might not have the ability to offer enough bandwidth for our testing purposes.

Timeline

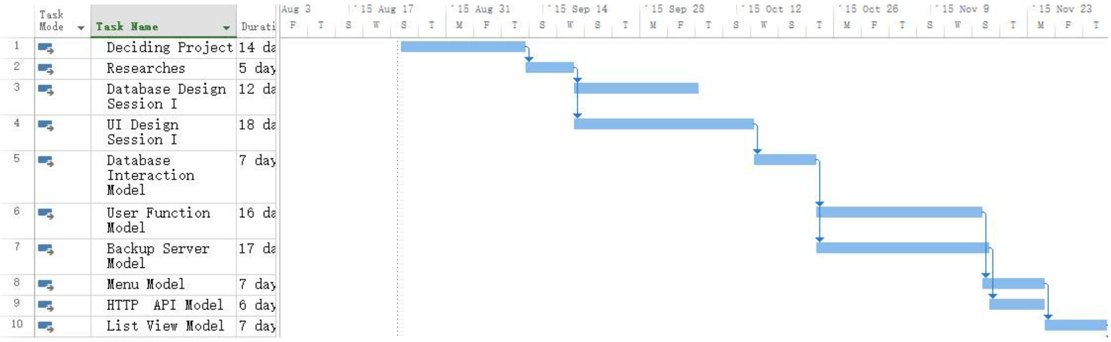
We set up our timeline of developing this application, and it is shown below:

Name	Duration	Start	Finish
Deciding Project	14 days	2015 August 24	2015 September 11
Researches	5 days	2015 September 11	2015 September 18
Database Design Session I	12 days	2015 September 18	2015 October 6
UI Design Session I	18 days	2015 September 18	2015 October 14
Database Interaction Model	7 days	2015 October 14	2015 October 23

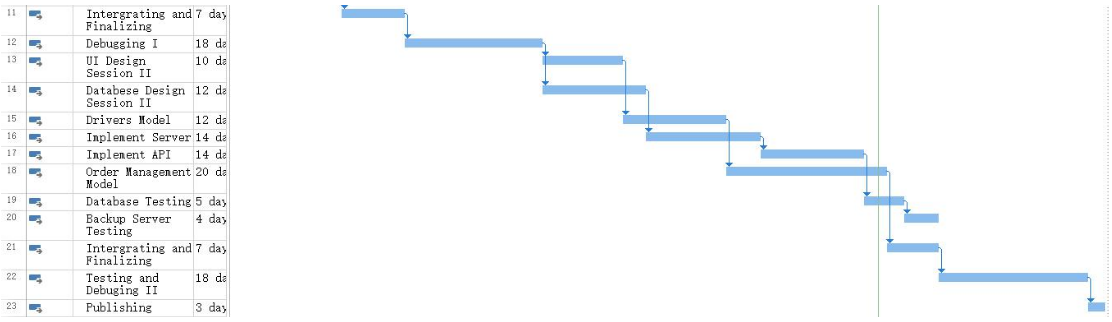
User Function Model	16 days	2015 October 23	2015 November 16
Backup Server Model	17 days	2015 October 23	2015 November 17
Menu Model	7 days	2015 November 16	2015 November 25
HTTP API Model	6 days	2015 November 17	2015 November 25
List View Model	7 days	2015 November 25	2015 December 4
Integrating and Finalizing	7 days	2015 December 4	2015 December 15
Debugging I	18 days	2015 December 15	2016 January 8
UI Design Session II	10 days	2016 January 8	2016 January 22
Database Design Session II	12 days	2016 January 8	2016 January 26
Drivers Model	12 days	2016 January 22	2016 February 9
Implement Server	14 days	2016 January 26	2016 February 15
Implement API	14 days	2016 February 15	2016 March 4
Order Management Model	20 days	2016 February 9	2016 March 8
Database Testing	5 days	2016 March 4	2016 March 11
Backup Server Testing	4 days	2016 March 11	2016 March 17
Integrating and Finalizing	7 days	2016 March 8	2016 March 17
Testing and Debugging II	18 days	2016 March 17	2016 April 12
Publishing	3 days	2016 April 12	2016 April 15

Figure 5. The Application Timeline.

Gantt Chart



	Task Mode	Task Name	Durati
1		Deciding Project	14 da
2		Researches	5 day
3		Database Design Session I	12 da
4		UI Design Session I	18 da
5		Database Interaction Model	7 day
6		User Function Model	16 da
7		Backup Server Model	17 da
8		Menu Model	7 day
9		HTTP API Model	6 day
10		List View Model	7 day



11		Intergrating and Finalizing	7 day
12		Debugging I	18 da
13		UI Design Session II	10 da
14		Databese Design Session II	12 da
15		Drivers Model	12 da
16		Implement Server	14 da
17		Implement API	14 da
18		Order Management Model	20 da
19		Database Testing	5 day
20		Backup Server Testing	4 day
21		Intergrating and Finalizing	7 day
22		Testing and Debuging II	18 da
23		Publishing	3 day

Testing Plan/Report

Overview

This section will explain our testing methodology for the Food Delivery Application and should be used as a guide. The following individuals should use this section:

- Developers
- Project Managers
- Software Testers
- Website Testers
- Database Testers
- Q & A Personnel

Scope

The scope of testing report is to test that the Food Delivery Application can run successfully on Android systems, and that the Database and Backup server can run as well. The test will be organized based on the requirements of the application.

Objective

Our Application testing is to verify that each technical part is working as expected. These tests are simply to give you (as a developer or tester) more insight into the areas of code that are being exercised by a set of test cases.

Entry and Exit Criteria

Entry Criteria:

- Whole system complete
- Software testing environment complete
- Network testing environment complete

Exit Criteria:

- Debug server is on
- Server' website is on
- Database are running
- API connection is good
- Debug Server are running

Logging Test and Reporting

If the Debug Server is running successfully, then the tester can find and check for all testing API interface lists. All API interface lists have been classified by the Debug server controller, which includes interface specification and action information. The tester can go into the specified interface test page if they click the "test" link. Each interface test page supports the action, test data and method information. After clicking the “test submit” button, then they can get the test report as soon as possible.

Testing Procedures

The following are steps that are needed for testing which consist of:

- Create all test systems and test cases
- Recording the test problems and expected results
- Modify the bugs in the final Application design

Below are the tests that will be performed:

1. Stability Test- This test is focused on whether the Food Delivery Application is running successfully on each device.

2. Launch Test- This test is focused on whether the Food Delivery Application is able to start and stop correctly on the device.

3. User Interface Test- This test is focused on the user interface connection.

4. Functionality Test- This test is focused on that the Food Delivery Application can effectively achieve all functions as expected.

Schedule of Team Member Testing

The schedule of the team member testing is listed in the table below.

Team Member	**Timeline to be Completed**	**How often?**
Developer	12/06/2015 to 04/16/2016	Weekly
Project Manager	12/06/2015 to 04/16/2016	Weekly

Figure 6. Table: Team Member Testing

Schedule of Round Table Testing

The schedule of the round table testing is summarized in the table below.

Round Table	**Timeline to be Completed**	**How often?**
Xiaoyang Lu	01/22/2016 to 03/17/2016	Once peer week (2 times)
Chen Yang	01/22/2016 to 03/17/2016	Once peer week (2 times)

Figure 7. Table: Team Member Testing

Testing Plan

We also designed and performed a test plan on both software logic layers and network interface layers. Plans for logic layers are designed to test if functions are acting as desired and that for network interface layers are designed to test if data could be passed through correct interface and method into MYSQL database. Here are the test results for both categories of layers:

Software Logic Layer:

Designed Testing Item	Designed Result	Actual Result	
User/ Restaurant Sign in	Sign in Successfully	Sign in Successfully	PASS
User/ Restaurant Sign up	Sign up Successfully	Sign up Successfully	PASS
Load Restaurant List	Load Successfully	Load Successfully	PASS
Load Menu List	Load Successfully	Load Successfully	PASS
Update Profile	Data updated to DB	Data updated to DB	PASS
Update Price in Bracket	Price Shown Correctly	Price Shown Correctly	PASS

Update Menu	Able to write in DB	Able to write in DB	PASS

Figure 8. Table: Software Logic Layer

Network Interface Layer:

Designed Testing Item	Designed Result	Actual Result	
Login Action	Data Recorded	Data Recorded	PASS
Register Action	Data Created	Data Created	PASS
Update Address Action	Data Recorded	Data Recorded	PASS
Load Restaurant List	Data passed back	Data passed back	PASS
Load Menu List	Data passed back	Data passed back	PASS

Figure 9. Table: Network Interface Layer

Testing Reports

Stability Test **-** This test will focus on the Food Delivery Application running successfully on the Android device.

Tester	**Date**	**Item #**	**Expected**	**Actual**	**Pass/Fail**	**Bug**
Xiaoyang Lu	1/18/2016	Android Stability	APP runs on different version of Android systems	Ran fine	Pass	
Chen Yang	01/22/2016	Database Stability	Database runs on phpmyadmin	Ran fine	Pass	

			webpage			
Chen Yang	02/01/2016	Webpage Stability	Server' webpage runs on Apache device	Ran fine	Pass	
Chen Yang	02/07/2016	Interface Stability	Hush Framework, Zend Framework and JQuery Mobile' connection is working correctly.	Working correctly	Pass	
Chen Yang	02/22/2016	Debuge Server Stability	Able to check all API interface lists information	Pass		

Figure 10. Table: Stability Test Layer

Launch Test- This test will focus on the Food Delivery Application being able to start and stop correctly on the device.

Tester	**Date**	**Item #**	**Expected**	**Actual**	**Pass/ Fail**	**Bug**

Xiaoyang Lu	1/22/2016	Android Launches	Be able to launch	Launched successfully	Pass	
Xiaoyang Lu	1/22/2016	Android Stops	Be able to quit the activity	Quitted successfully	Pass	
Xiaoyang Lu	1/22/2016	Android Resumes	Activity could resume after quitting	Resumed successfully	Pass	

Figure 10. Table: Launch Test Layer

User Interface Test- This test focused on the user Interface connection.

Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
Xiaoyang Lu	1/22/2016	User Layout	No items originally, but listview, tab and slide menu shows correctly	Layout shows properly	Pass	
Xiaoyang Lu	1/22/2016	Driver Layout	No items originally, but listview, tab and slide menu shows	Layout shows properly	Pass	

			correctly			
		Restaurant Layout	Slide menu shows correctly and menu modifier shows correctly	Layout shows properly	Pass	

Figure 11. Table: User Interface Test

Functionality Test- This test will focus on the Food Delivery Application can effective achieved all functions as expected.

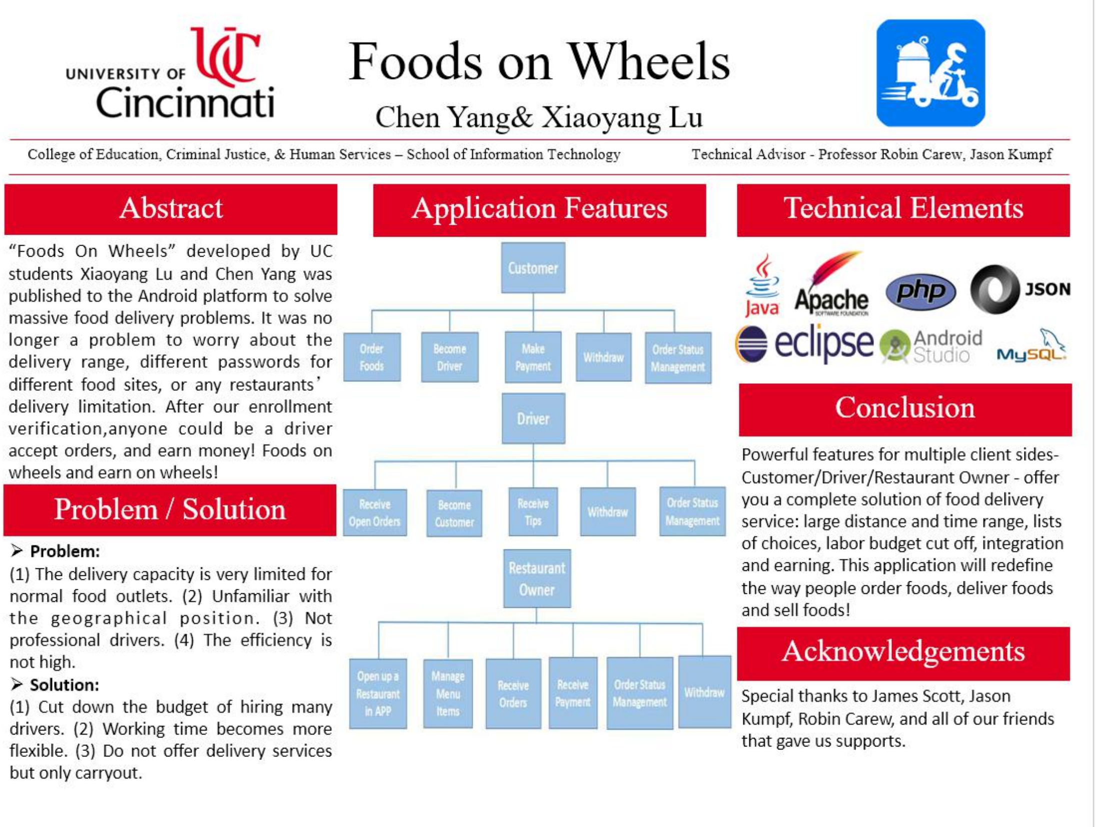
Tester	Date	Item #	Expected	Actual	Pass/Fail	Bug
Xiaoyang Lu	1/28/2016	App Buttons	Every button works well and could guide users to the destination page	Buttons work well	Pass	
Xiaoyang Lu	1/28/2016	Account Control	Users, drivers and restaurants could be able to register/ login and logout. Data could be stored	Account control works successfull y	Pass	

			into database			
Xiaoyang Lu	1/28/2016	Cash flow	Users, drivers and restaurant could be able to make payments, receive money and withdraw money	Cash flow works properly	Pass	
Xiaoyang Lu	1/28/2016	Order Management and Placement	Users, drivers and restaurant could be able to place, view and change order status	Order management works successfully	Pass	

Figure 12. Functionality Test Layer

Tech Expo at a Glance

Tech Expo was held on April 12,2016 at the UC at the 9 am to 1 pm. We also prepared the poster for Tech Expo and would be going to demonstrate our application.



UNIVERSITY OF Cincinnati

Foods on Wheels

Chen Yang& Xiaoyang Lu

College of Education, Criminal Justice, & Human Services – School of Information Technology Technical Advisor - Professor Robin Carew, Jason Kumpf

Abstract

"Foods On Wheels" developed by UC students Xiaoyang Lu and Chen Yang was published to the Android platform to solve massive food delivery problems. It was no longer a problem to worry about the delivery range, different passwords for different food sites, or any restaurants' delivery limitation. After our enrollment verification,anyone could be a driver accept orders, and earn money! Foods on wheels and earn on wheels!

Problem / Solution

➢ **Problem:**

(1) The delivery capacity is very limited for normal food outlets. (2) Unfamiliar with the geographical position. (3) Not professional drivers. (4) The efficiency is not high.

➢ **Solution:**

(1) Cut down the budget of hiring many drivers. (2) Working time becomes more flexible. (3) Do not offer delivery services but only carryout.

Application Features

Customer

Order Foods

Become Driver

Make Payment

Withdraw

Order Status Management

Driver

Receive Open Orders

Become Customer

Receive Tips

Withdraw

Order Status Management

Restaurant Owner

Open up a Restaurant in APP

Manage Menu Items

Receive Orders

Receive Payment

Order Status Management

Withdraw

Technical Elements

Conclusion

Powerful features for multiple client sides-Customer/Driver/Restaurant Owner - offer you a complete solution of food delivery service: large distance and time range, lists of choices, labor budget cut off, integration and earning. This application will redefine the way people order foods, deliver foods and sell foods!

Acknowledgements

Special thanks to James Scott, Jason Kumpf, Robin Carew, and all of our friends that gave us supports.

Figure 13. Tech Expo Poster

During the Tech Expo, we demonstrated our application to many of the attendances, and meanwhile received many feedback as listed below:

- Quality Control: A suggestion was raised about quality control. Customers, drivers and restaurant owners should have more flexibilities to rate, comment or communicate with each other. Each user should also be able to report problems to us if necessary.

- Payment: We still need to implement some features to secure our payment process. When sending credit card information, we should be able to encrypt those sensitive data.
- Mapping: We could offer a feature that let restaurant owner to see driver's real time location in order to control time for preparing hot foods, not having them to become cold.

Conclusion

Our purpose is to focus on how to build a complete and orderly distribution platform for the whole food service delivery system. We want to achieve a timely, fully systematized, and highly efficient distribution agent that is safe and supplies a variety of choices. We know it is no big deal for large catering enterprises to form a distribution team, but for many small and medium-sized restaurant businesses, the distribution team's labor costs and team operation costs are a huge financial burden for them and limits future development. Therefore, our food delivery application design concept is to try to re-install an order system, route planning algorithm, staffing management system, intelligent distribution system and so on, in order to reasonably and accurately give the delivery order to a suitable delivery person without any dispatching center, consequently, to improve the efficiency of delivery and customer satisfaction.

References

1. *11 HOTTEST FOOD & BEVERAGE DINING TRENDS IN RESTAURANT & HOTELS*, 2016. (n.d.). Retrieved November 2, 2015, from http://www.baumwhiteman.com/2016Trends.pdf

2. Crosby, M. (2014, September 2). *An Airbnb or Uber for the Electricity Grid?* Retrieved November 2, 2015, from http://blog.rmi.org/blog_2014_09_02_an_airbnb_or_uber_for_the_electricity_grid

3. Farhad Manjoo, *Uber's Business Model Could Change Your Work*. Retrieved Jan 28, 2015, from

http://www.nytimes.com/2015/01/29/technology/personaltech/uber-a-rising-business-model.html?_r=0

4. Massachusetts safety offices league. *Safety Tips for fast food delivery drivers and their employers*. Retrieved 2012, from

http://www.saif.com/_files/SafetyHealthGuides/msol_deliverydriver.pdf

5. Wilco Van Bragt. *APP-V Basics: Installing and Using the App-v 5 Infrastructure.*

Retrieved December 18,2013, from

http://www.virtualizationadmin.com/articles-tutorials/application-virtualization-articles/app-v-basics-installing-and-configuring-app-v-5-infrastructure-part1.html

6. Wikipedia, JSON , Retrieved 2001, from https://en.wikipedia.org/wiki/JSON