COMP390

2019/20

Vegetarian Radar

Mobile app to make social vegan friends

and share ideas

Student Name:	Linshan Li
Student ID:	201376016
Primary Supervisor:	Dr Bakhtiar Amen
Secondary Supervisor:	Dr Olga Anosova

DEPARTMENT OF

COMPUTER SCIENCE

University of Liverpool
Liverpool L69 3B

Table of Contents

Abstract

The name of the project is "Vegetarian Radar", which is an Android App project began on October 13rd, 2019 and ended the code part of the entire program on April 14th, 2020, the testing and improvement parts started from April 15th, 2020 until May 6th, 2020. The entire development process lasted for seven months.

This Android app "Vegetarian Radar" committed to providing convenience for vegetarians, locating where the users are and showing vegetarian restaurants around them in two forms, the app also provide users with detail information about the nearby restaurants, as well as route and solutions to the target restaurants. Since the app is also a platform to share good restaurants with others, a screenshot function is provided, the picture of the interface can be sent to their friends. In addition, customers can mark a restaurant and add it to favorite list, providing convenience for later search.

The project was programmed in Android Studio 3.6.2 using Java and XML, the databases stored information of registered customer, nearby vegetarian restaurants, and customer's favorite restaurants, which implemented by SQLite 3.28.0 database built in Android Studio. The data of restaurants was crawled from Google Map through Google Map Places API and Google Map Directions API which were got from Google Cloud Platform Console. The developer chose Iterative Development as the development method and BitBucket as the repository to store backups. The test of the project was conducted on Huawei P10 phone and the result of the project was generally successful.

The article consists of eight parts, the introduction chapter outlines the motivation, purpose, and objectives of the project; the background chapter describes the specific terms and related tools used in the project; the design part shows project's data source, UML diagram and corresponding primary use case, activity diagram that shows how to operate the app, and interfaces and functions design. The implementation part outlines software and hardware resources of the project, the development method, the backup of code, data structure, algorithm, and code implementation; the fifth chapter narratives the process of project testing and results while the sixth chapter is about the evaluation of the project and questionnaire and feedbacks from others, then a conclusion is given. The last part is The BCS project criteria and my self-reflection according to the process of doing this project.

1 Introduction

1.1 Project Introduction

A brief introduction of the Android mobile App named as Vegetarian Radar which aimed to providing convenience for vegetarians will be illustrated in this chapter.

According to the Oxford Dictionary, vegetarians are those who do not eat any meat, poultry, fish, shellfish, or by-products of animal slaughter (Petre, 2016). This terminology was started from the British Vegetarian Society in the mid-1800's, while vegetarianism itself can be dated back to a time before recorded history since plant food is the main meal for most early human beings (no date). Pythagoreans, were considered as the first self-proclaimed vegetarians, which is a subtitle derived from the Greek philosopher Pythagoras, the founder of the geometric Pythagorean theorem (Avey, 2014). There are many reasons for people to become vegetarians, including health, religious convictions, concerns about animal welfare or antibiotics and hormones use in livestock, and a desire to eat in a way that avoids excessive use of environmental resources (2018).

Mobile apps are software programs operated on mobile devices, they are general small, individual software units with specific functions and can provide similar services to those run on PCs (no date). Compared mobile Apps to PC applications, the former one has the advantages such as higher portability, faster running speed, and providing users with better interactive engagement. Therefore, according to numbers of studies by Oracle and MobileSmith, more than 60% users prefer mobile apps over online mobi-sites (2017). And because of the mobile technology development, millions of mobile users depending on mobile apps every day.

However, most restaurants navigations apps on the App Store such as UberEATS and Trip Advisor are not suitable for vegetarians since they cannot filtrate omnivorous restaurants to only display restaurants that meet the requirements of vegetarians and veggies, using them might cause numerous of confusion as well as waste time. A Few apps exist on the market which aims at vegetarians have some cumbersome and unnecessary functions, which makes the interfaces look unconcise and provides difficulties for user's operation.

1.2 Aims

- Create an Android App which aims to provide convenience for vegetarians
- The App can help customers find vegetarian restaurants nearby as well as navigate to there.
- Customers can share restaurants they recommend with others.
- Customers can record their favorite restaurants for future convenience

1.3 Objectives

Functions：

- Implemented a login system to verify customer’s identity, including create a new account, forget password and a login window.
- User information in register should include a username, password, and a phone number. Password should follow the rule: minimum 8 characters, maximum 16 characters, only numbers, English characters, “/” were allowed, phone number should only contain numbers
- Imported Google Map API for searching nearby (less than 5 kilometers) vegetarian restaurants and show detailed route between customer and a specific restaurant
- Showed Restaurants’ information in two ways: Map and List
- Implemented two ways of ordering restaurants: Distance and Praise
- Displayed detail information of a certain restaurant
- Recorded customer’s favorite restaurants and showed in list
- Provided a screen capture function

Interface：

- Designed user-friendly interfaces with consistent style
- Designed an icon related to the theme of the app
- Achieved smooth interface switching

Algorithm:

- Designed and implemented an algorithm for calculating distance between current location of customer and target restaurants

2 Background

2.1 Terms

API: An application programming interface is a computing interface which defines interactions between multiple software intermediaries and defines requests or calls can be made and the way to make them (2015).

2.2 Related Tools

This project aimed to make an Android App that used Android Studio as its integrated development environment (IDE) and used Java as the development language. Compared Android Studio with another IDE, Eclipse with Android Development Tools (ADT), the former one which is built on JetBrains' IntelliJ IDEA software has numerous of advantages. It has a more user-friendly interface, with more clear and concise layout, allowing first-time Android app developers get a hang of easier and quicker. In addition, the graphical user interface of Android Studio has a Drag-and Drop function, which allows users to move data from one View to another View in the current layout, reducing the time spending on drawing out the interface (Fakhrddin, 2015) (no date). Moreover, in XML files, color, images, and code can be previewed in real time with code, enables developers quickly adjusting the layout code according to the requirement.

In order to get vegetarian restaurants near the location of the user, the App stored a large amount of this kinds of restaurants details including their name, degree of praise, address, pictures and location in SQLite, a light weighted database with low cost and low complexity. The android.database.sqlite class in Android Studio provided APIs needed to use a database on Android, therefore, there was no need to install a SQLite additionally. Before storing data, the developer defined the attributes of the database at first, and implemented methods that create and maintain databases and tables, all of these should be written in classes of Java. Besides java code, the created databases can be checked and modified by SQLite Studio, which is an advanced SQLite database manager with intuitive user interface, as well as a single executable file. In addition to nearby vegetarian restaurants' information, customer's information and their favorite restaurants were also stored in a database.

The reason why choosing Google Map as the web to be crawled is that it provides wealth of information, it contains information about almost every facility, leading it became the most popular navigation app by a wide margin, with 67% as its audience support rate, while only 77% of smartphone owners use navigation app (Panko, 2018). In addition, Google Map also record restaurants with their location, name, as well as photos and rating, which provides all the information required by the "Vegetarian Radar" app. Therefore, Google Map is a relatively appropriate web for crawling compared with other navigation platforms such as Apple Maps or MapQuest.

Moreover, Google Maps API is a computing interface called by JavaScript that owned by Google and based on Google Maps, which allows programmers who interested in Google Map to develop their own services and build their own applications based on Google Map. By communicating with Google Map Service, the third-party app can get the data such as a latitude/longitude location, route information or place information according to different types of APIs (DeeDeeG, 2017). The Google Map API defines a set of function classes for displaying control and cascading information on Google Maps, placed in a package called com.google.android.maps. In order to display and use Google Map in Android Studio, an abstract class called MapsActivity was created, and a MapView instance (Map fragment) derived from android.view. viewgroup was created by using onCreate() method.

The User interface (UI) for the app is also important since it is where a user interacts with the software, a good interface which designed to display the services that the platform offers without ambiguity makes it easier to attract target audiences and keep them on the site. A good GUI Design presents a seamless conjunction of visual design, interaction design, and information architecture (Berezhnoi, 2019). In order to achieve the goal, the developers need to follow the user's mental model instead provide any counterintuitive settings, and meet user's needs, which means it is necessary for GUI developers to take user's habits into consideration, like where the habit of looking for the navigation bar. Therefore, GUI design refers to the overall design of the logic interface of human-computer interaction, a good use of GUI will attract more stakeholders and maintain their loyalty in addition.

3 Design

3.1 Data Source

Since the “Vegetarian Radar” app is a platform mainly aims at recommending vegetarians with nearby veggie restaurants, so it is necessary for developer to get detailed information of restaurants which might be needed. Because of the time limitation, the project set a fixed hunting zone, only vegetarian restaurants less than 5 kilometers near the user can be found. The detailed information included restaurants' name, degree of praise, address, whether is open, and specific route.

All the information of restaurants was captured from Google Map by its API, which is the most popular navigation app with wealth of information. In order to use its data up to maximum extend, so Google provides with numerous of APIs, including API for Google Map, since the “Vegetarian Radar” project took Places API and Directions API of Google Maps Platform into use, it was legal to crawl data form Google Map though the number of free searches through API allowed per day is limited. To use the data of Google Map, an API key to authenticate requests associated with the project for usage and billing purposes is needed, which was got on Google Cloud Platform Console and add the key to the request (AndroidManifest.xml and the end of each searching URL).

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyDJrAUAdMX80XmvBru6WbZ43KynzF1P-cE" />
```

Although users are required to login or create a new account before starting using other functions in “Vegetarian Radar” app, however, they do not need to use their true name as username, only a nick name is enough, which protects not leaking user’s privacy in a maximum degree. And there is a disclaimer states that users should be responsible for all the data they enter, including their password and phone number used for registration, though meanwhile, all the information will only be used in this app, and won’t apply to other software, users do not need to worry about the problem of information disclosure.

3.2 UML Diagram

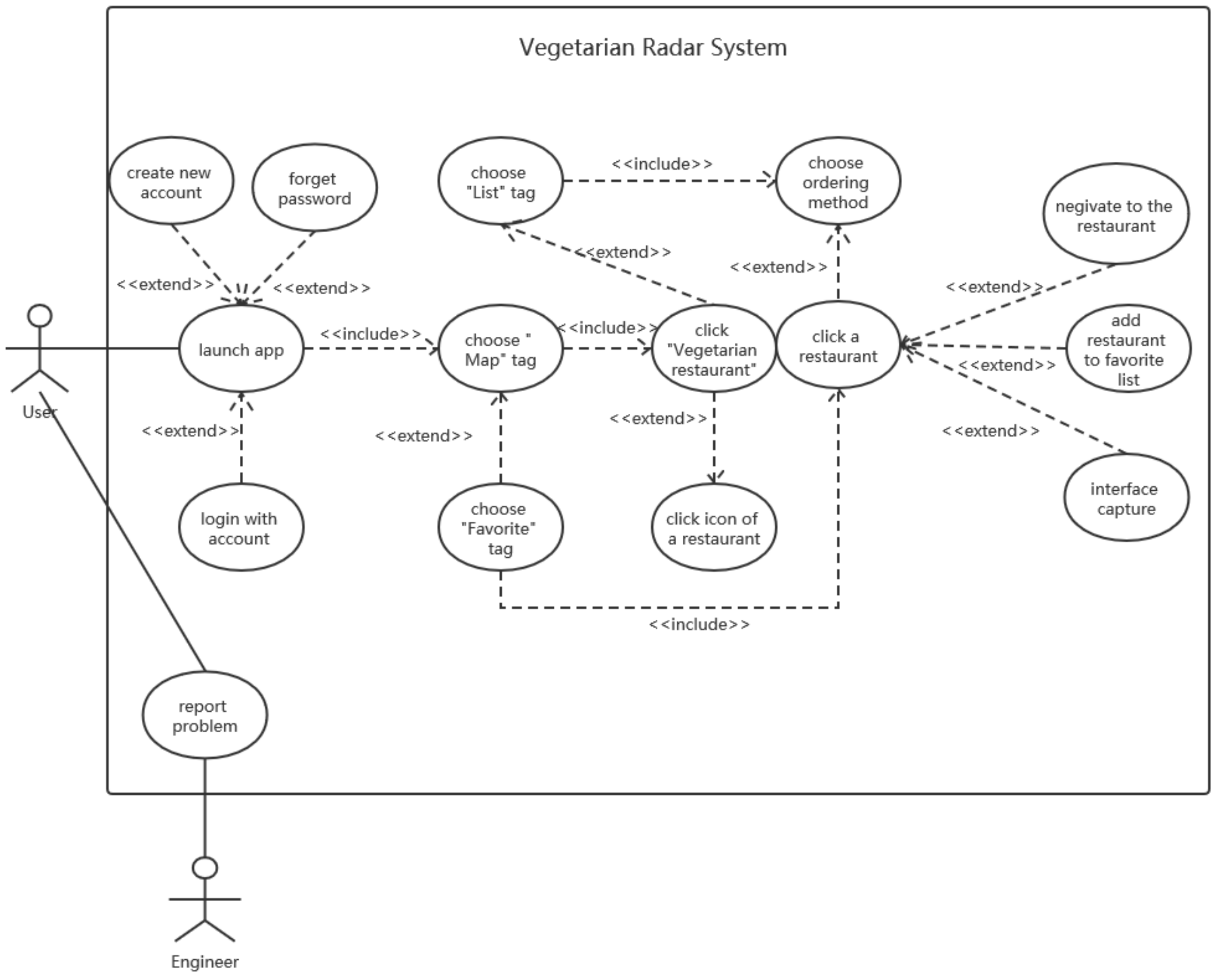


Fig 1. Use-Case Diagram

3.3 Primary Use Case

ID	UC1
NAME	Launch App
DESCRIPTION	Player will enter the login interface.
PRE-CONDITION	App is running
EVENT FLOW	
POST CONDITION	
INCLUDES	UC5
EXTENSION POINTS	UC2, UC3, UC4
TRIGGER	User open this app

ID	UC2
NAME	Create New Account
DESCRIPTION	New user registers a new account
PRE-CONDITION	App is running
EVENT FLOW	User enters his username, password, phone number, admits the disclaimer
POST CONDITION	Phone GPS open
INCLUDES	UC5
EXTENSION POINTS	
TRIGGER	User press “New Customer” button

ID	UC3
NAME	Forget Password
DESCRIPTION	User already registered but forget the password
PRE-CONDITION	App is running
EVENT FLOW	User enters his registered phone number and new password
POST CONDITION	Phone GPS open
INCLUDES	UC5
EXTENSION POINTS	
TRIGGER	User press “Forget Password” button

ID	UC4
NAME	Login with account
DESCRIPTION	User has already registered in the system
PRE-CONDITION	App is running
EVENT FLOW	
POST CONDITION	GPS open, app jump to the map page
INCLUDES	UC5
EXTENSION POINTS	
TRIGGER	Username and password are matched

ID	UC5
NAME	Choose “Map” tag
DESCRIPTION	User clicks the “Map” tag in the bottom navigation bar
PRE-CONDITION	User successfully login
EVENT FLOW	User clicks the “VEGETARIAN RESTAURANT” button
POST CONDITION	Phone GPS open, app jump to the map page
INCLUDES	UC8
EXTENSION POINTS	UC7
TRIGGER	User login in the app

ID	UC6
NAME	Choose “List” tag
DESCRIPTION	User clicks the “List” tag in the bottom navigation bar
PRE-CONDITION	User successfully login
EVENT FLOW	User chooses a listing method
POST CONDITION	Phone GPS open, app jump to the list page
INCLUDES	UC9
EXTENSION POINTS	
TRIGGER	User clicks the “List” tag

ID	UC7
NAME	Choose “Favorite” tag
DESCRIPTION	User clicks the “Favorite” tag in the bottom navigation bar
PRE-CONDITION	User successfully login
EVENT FLOW	Click a restaurant in favorite list
POST CONDITION	
INCLUDES	Click a restaurant
EXTENSION POINTS	
TRIGGER	User clicks the “Favorite” tag

ID	UC8
NAME	Click “VEGETARIAN RESTAURANT”
DESCRIPTION	User clicks the “VEGETARIAN RESTAURANT” button above
PRE-CONDITION	User in the map page
EVENT FLOW	Click the button
POST CONDITION	Phone GPS open
INCLUDES	
EXTENSION POINTS	Click icon of a restaurant, UC6
TRIGGER	

ID	UC9
NAME	Choose Ordering method
DESCRIPTION	User chooses one of the two buttons (Rating and Distance) at the top of the list page
PRE-CONDITION	User in the list page
EVENT FLOW	Choose a specific restaurant
POST CONDITION	
INCLUDES	
EXTENSION POINTS	Click a restaurant
TRIGGER	User want to list restaurants by another method

ID	UC10
NAME	Navigate to a restaurant
DESCRIPTION	User clicks the icon next to the address in restaurant detail information page
PRE-CONDITION	User in restaurant detail information page
EVENT FLOW	Choose a specific restaurant
POST CONDITION	User’s phone has installed Google Map
INCLUDES	Navigate to the restaurant
EXTENSION POINTS	
TRIGGER	User click the navigation icon next to the address

ID	UC11
NAME	Add restaurant to favorite list
DESCRIPTION	User clicks the icon next to the headline in restaurant detail information page
PRE-CONDITION	User in restaurant detail information page
EVENT FLOW	
POST CONDITION	
INCLUDES	
EXTENSION POINTS	
TRIGGER	

ID	UC12
NAME	Interface capture
DESCRIPTION	User clicks the camera icon next to the headline in restaurant detail information page
PRE-CONDITION	User in restaurant detail information page
EVENT FLOW	Send the picture to friends
POST CONDITION	
INCLUDES	
EXTENSION POINTS	
TRIGGER	

3.4 Activity Diagram

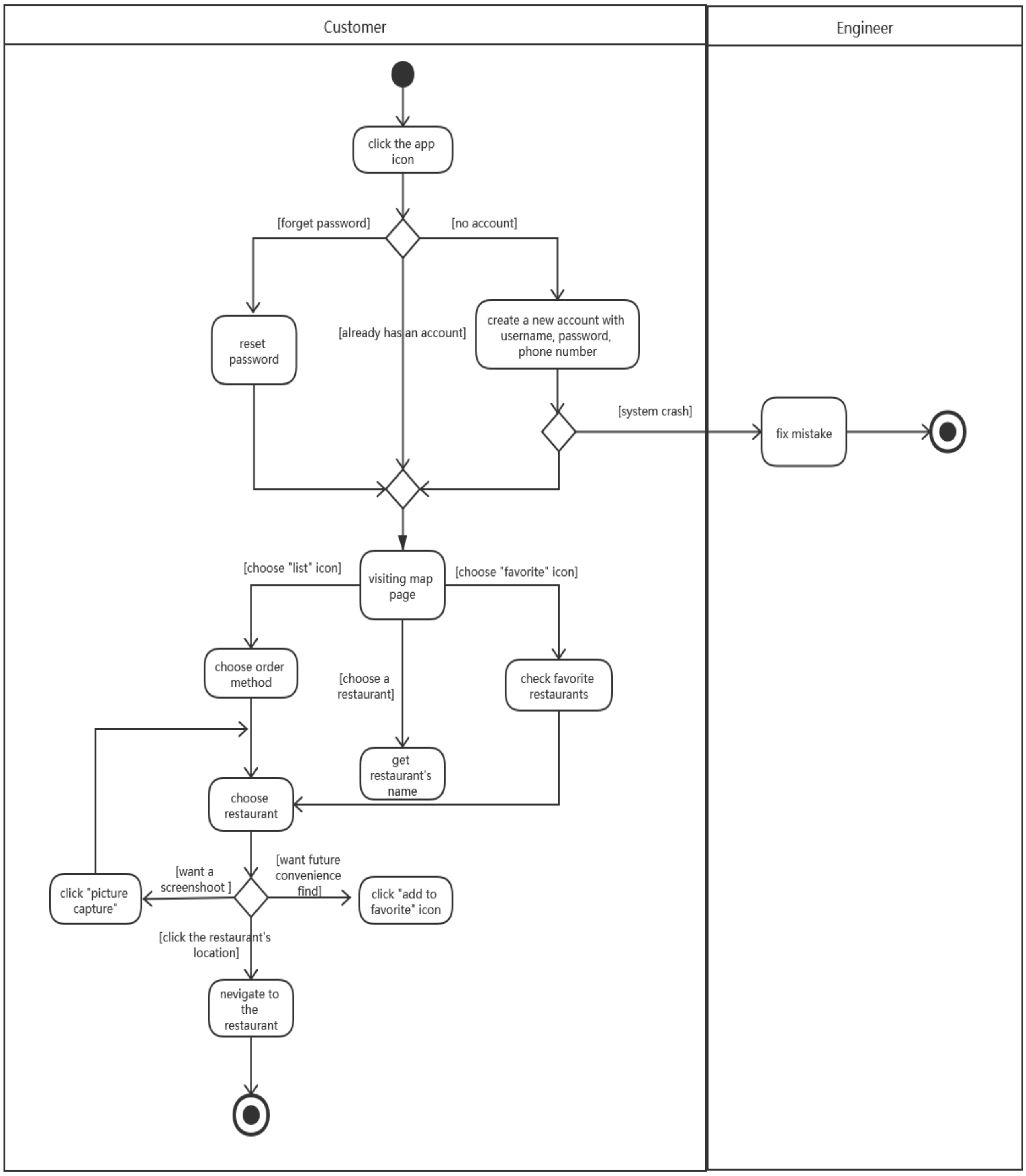


Fig 2. Activity Diagram

3.4 Components and Functions

The “Vegetarian Radar” project consists of five parts: login system, map system, list system, navigation system, and favorite system.

● Logo / App Icon

Fig 3. App icon

The app used a colza as the app icon as well as its logo, the logo satisfied the theme of the App

Before customer enters the app, they should click the app icon at first

● Login System

Once the user first clicks the app icon, they will enter the login page, and according to whether they already have an account, they can choose “New Customer”, directly type in their username and password to login, or “Forget Password” to get back the account.

■ Login Interface

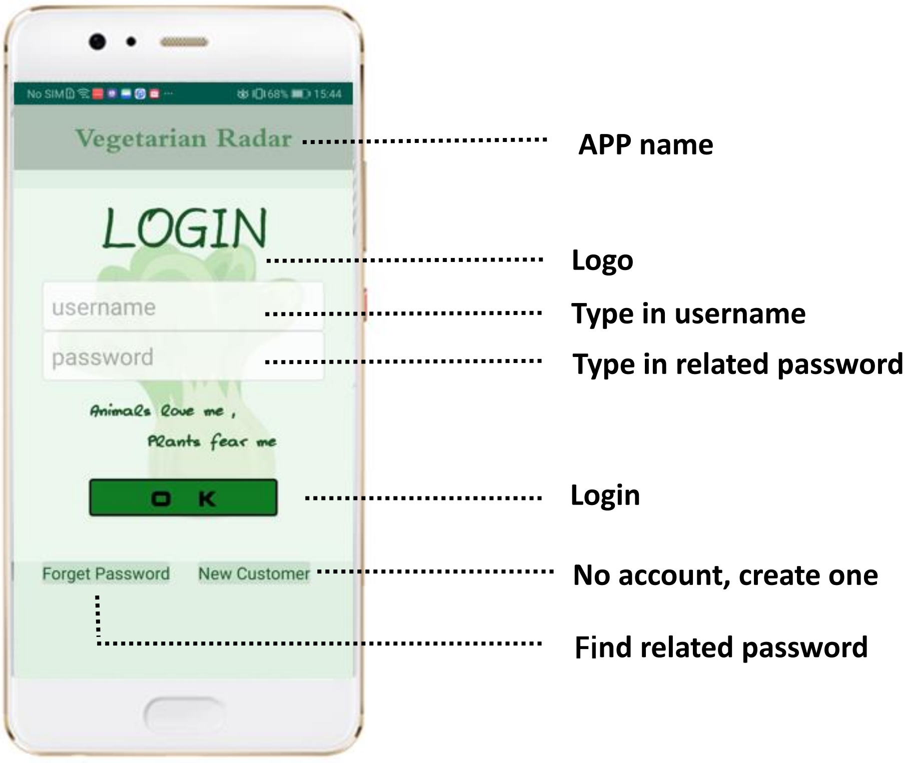


Fig 4. Login Interface

■ Forget Password Interface

If customer have already registered in the app before but forget his username or password, they can click the “Forget Password” icon in the login page, then the page will jump to the “Reset Password” Page. Since all the phone numbers which have been registered in the system are stored in a database, once the customer enters the phone number, the system will check the database whether the phone number has already been registered, if yes, the new entered password will replace the old one stored in the database and the page will jump to the map interface, else the system will remind the customer the “User does not exist”.

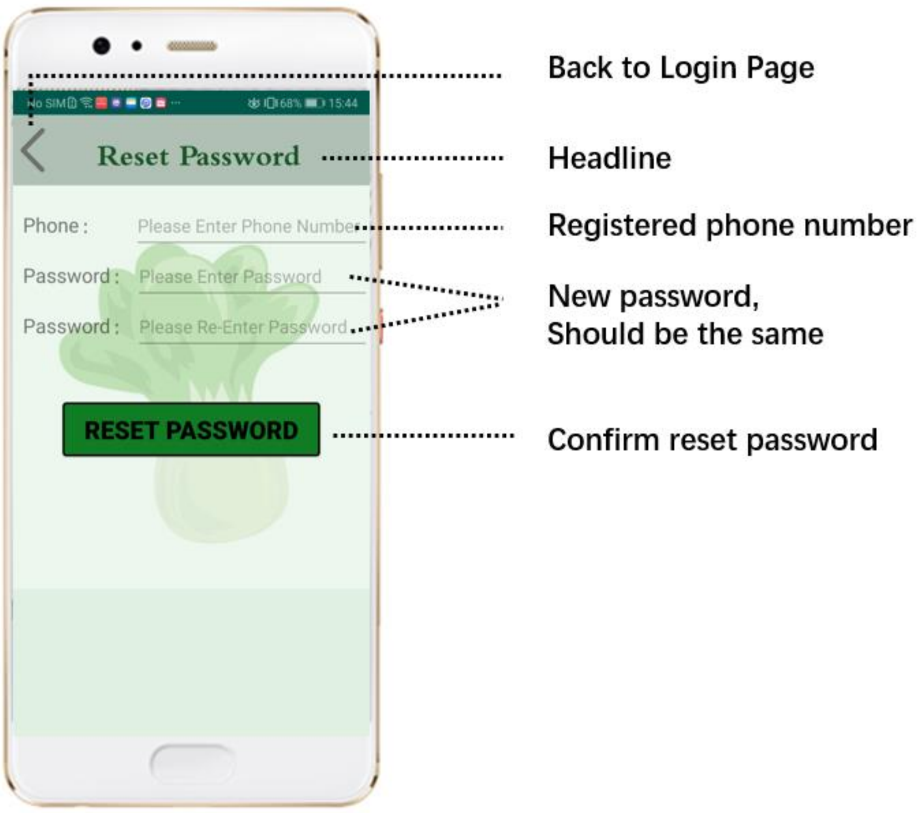


Fig 5. Forget Password Interface

■ **Register Interface**

If the customer does not have an account, he should select the "New customer" icon, the app will switch to the registration interface, and users need to type in a username(no format restrictions and no need to use their real name) and a password (minimum 8 characters, maximum 16 characters, only numbers, English characters, "/" are allowed) as well as a valid phone number, also there is a disclaimer states that customers are responsible for the information they enter should be agreed by customer, then the account will be created successfully. After successfully registered, the page will jump to the map interface.

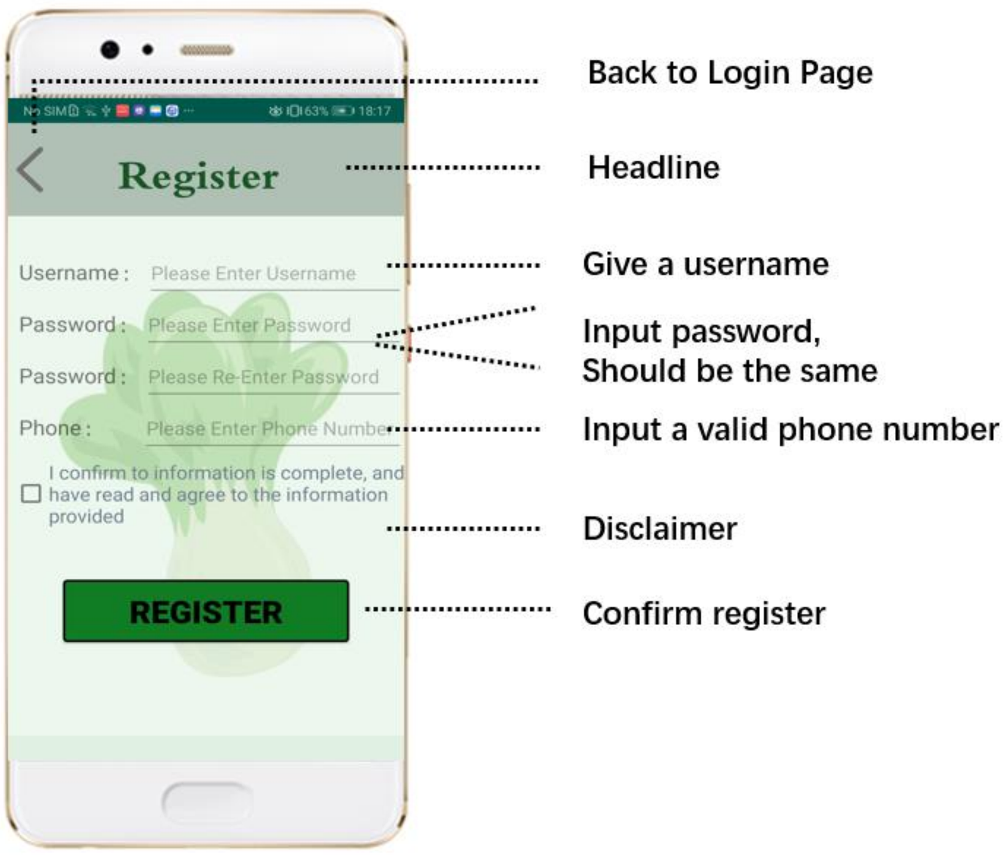

Register
Username : Please Enter Username
Password : Please Enter Password
Password : Please Re-Enter Password
Phone : Please Enter Phone Number
I confirm to information is complete, and have read and agree to the information provided
REGISTER
Back to Login Page
Headline
Give a username
Input password, Should be the same
Input a valid phone number
Disclaimer
Confirm register

Fig 6. Register Interface

- ## Map System

Once the user successfully login the system, the interface automatically will jump to map interface, which displays the current location of user and nearby information on government, transportation, and business. When user click the "Vegetarian Restaurant" button above the map fragment, all the vegetarian restaurants less than 5 kilometers away from the user will be displayed on the map. When click the red icon of each restaurant, a title will show up and show the name of the restaurant, the center of map will follow the user's click to move as well.

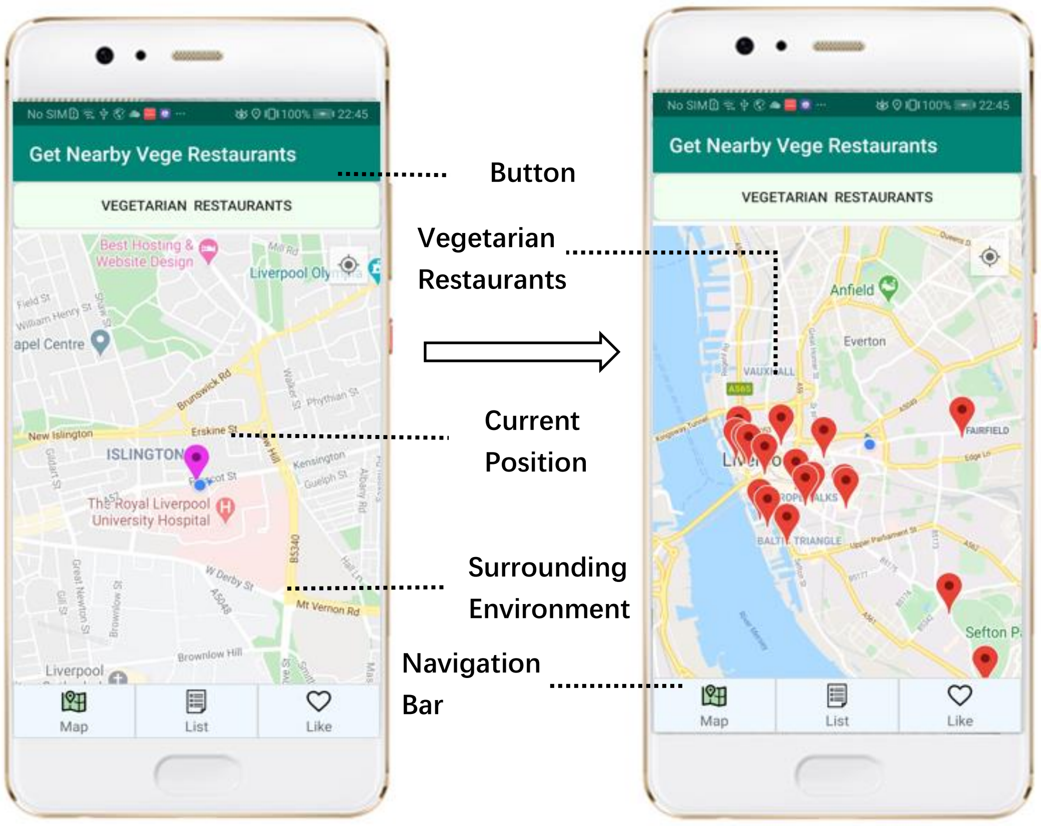


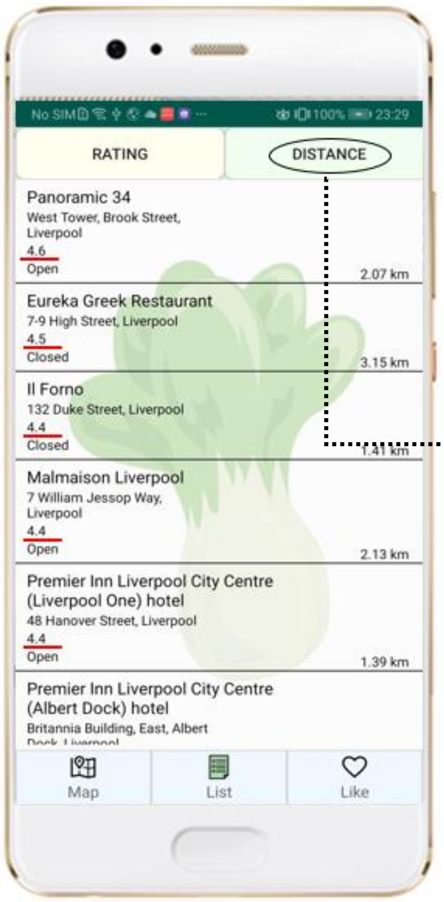
Fig 7. Map Interface before searching

Fig 8. Map Interface after searching

● List System

The bottom bar is the navigation bar, the one in the middle can lead users to the List Interface. This system has two listing methods, the default one is ranking by rating, and the other one is ranking by distance. Users can switch the ranking method by choosing one of the two buttons at the top of the list interface.

From these two interfaces, information about the nearby vegetarian restaurants are displayed, including their name, location address, rating by users, whether is open or not at current time, and distance between user and restaurants.



No SIM 100% 23:29

RATING | DISTANCE

Panoramic 34
West Tower, Brook Street, Liverpool
4.6
Open
2.07 km

Eureka Greek Restaurant
7-9 High Street, Liverpool
4.5
Closed
3.15 km

Il Forno
132 Duke Street, Liverpool
4.4
Closed
1.41 km

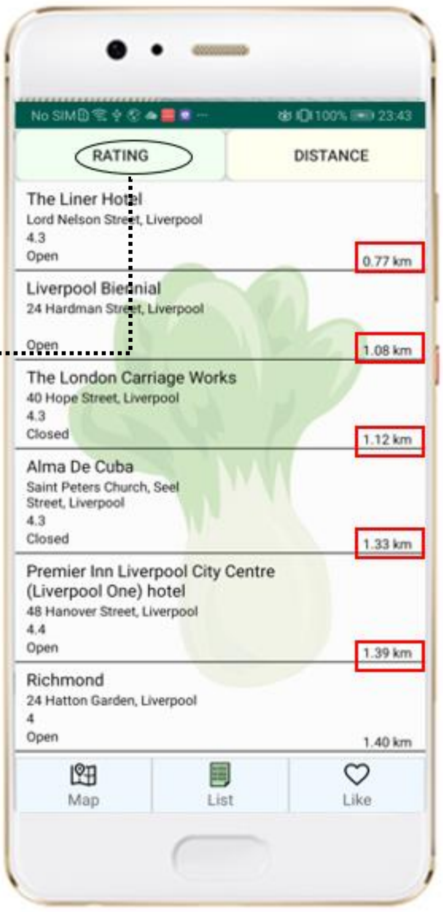
Malmaison Liverpool
7 William Jessop Way, Liverpool
4.4
Open
2.13 km

Premier Inn Liverpool City Centre (Liverpool One) hotel
48 Hanover Street, Liverpool
4.4
Open
1.39 km

Premier Inn Liverpool City Centre (Albert Dock) hotel
Britannia Building, East, Albert

Map | List | Like

Fig 9. Ranking by Parise



No SIM 100% 23:43

RATING | DISTANCE

The Liner Hotel
Lord Nelson Street, Liverpool
4.3
Open
0.77 km

Liverpool Biennial
24 Hardman Street, Liverpool
Open
1.08 km

The London Carriage Works
40 Hope Street, Liverpool
4.3
Closed
1.12 km

Alma De Cuba
Saint Peters Church, Seel Street, Liverpool
4.3
Closed
1.33 km

Premier Inn Liverpool City Centre (Liverpool One) hotel
48 Hanover Street, Liverpool
4.4
Open
1.39 km

Richmond
24 Hatton Garden, Liverpool
4
Open
1.40 km

Map | List | Like

Fig 10. Ranking by Distance

● Restaurant Detail Information Page

If a user wants to find more detailed information about a certain restaurant, he can click the item of the restaurant in the List Interface, then the page will jump to the detail information page. User can not only see the information mentioned in the list interface, but also see the restaurant's photo and the route to get to the restaurant.

At the middle of the page is the address of the restaurant, next to the word, there is an icon, if the user clicks this, the app will check whether the user's phone has already downloaded and installed the Google Map and jump to the third-party navigation software.

Next to the headline of the page, there are two icons: heart and camera, which can achieve the function of adding restaurant to user's favorite list and get the screenshot of the interface (will be mentioned later). User can send the picture to their friends through a third-party communication software.

The map at the bottom of the interface is a bit same like the one in the map interface. However, only two icons are displayed: user's current location and the target restaurant, the center of map cannot be moved, but the size of map can be changed.

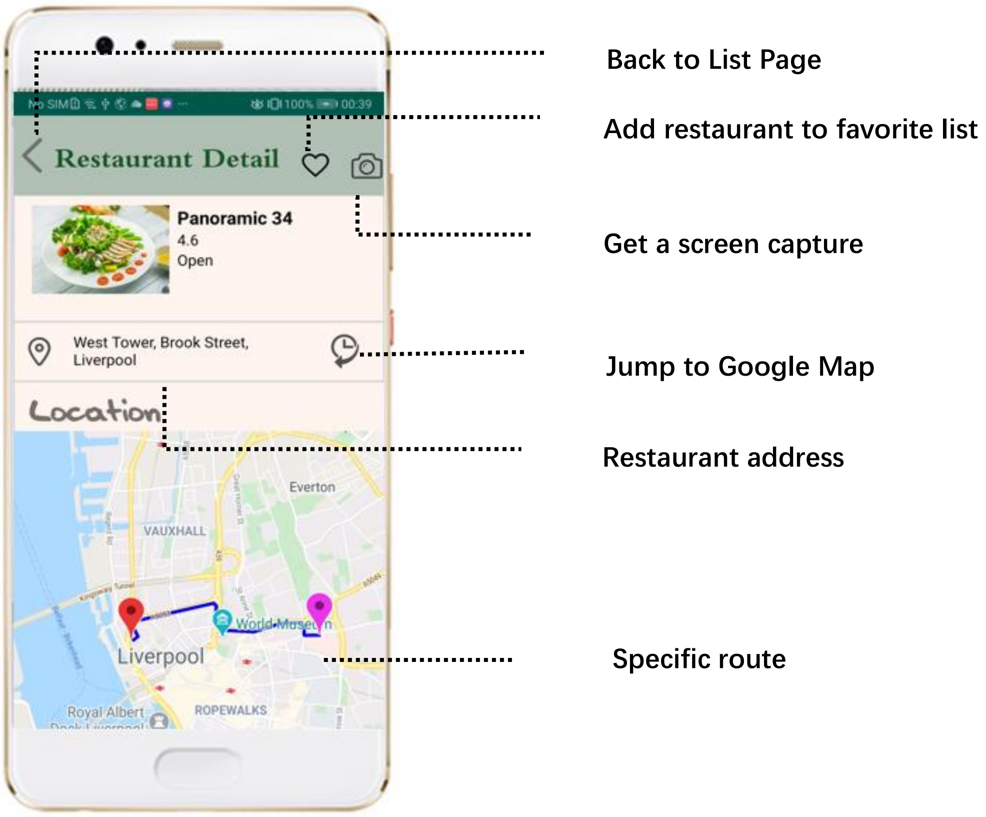
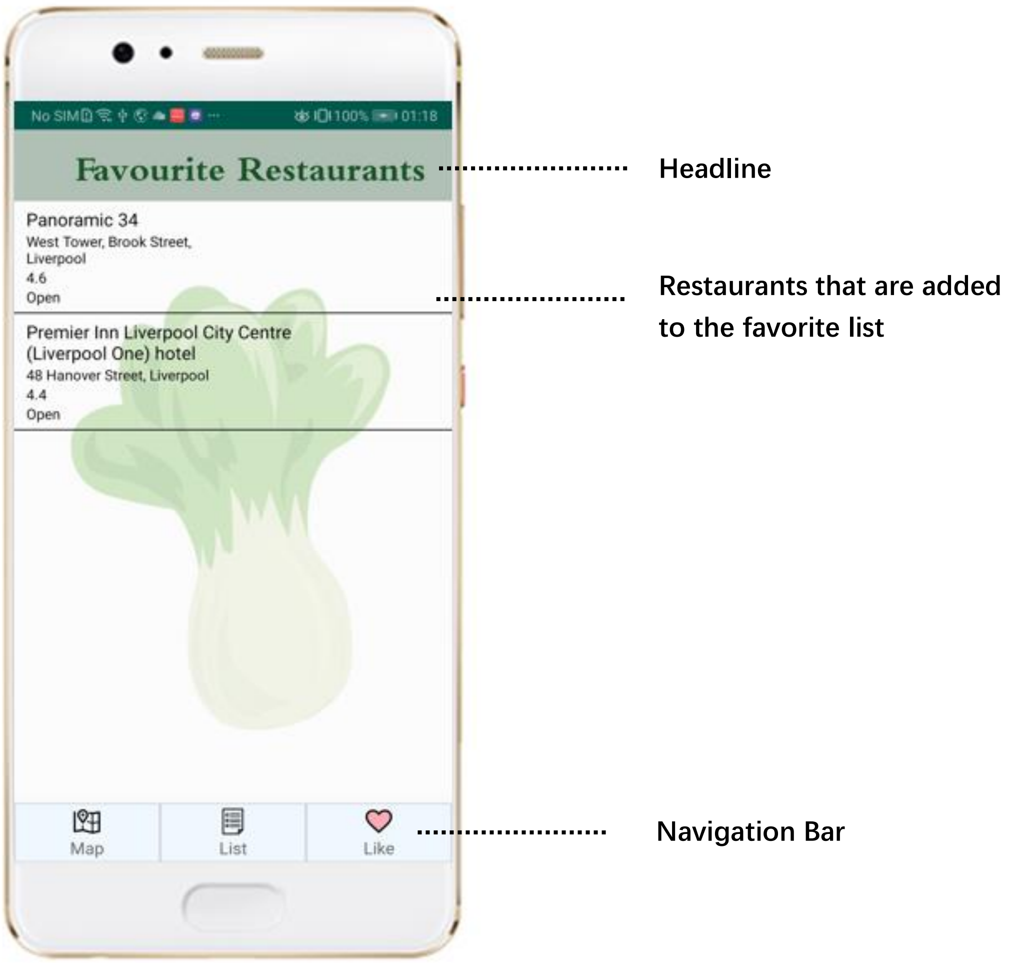


Fig 11. Detail Information Interface of "Panoramic 34"

Favorite List

If the user likes the restaurant and want to quickly find them in the future, he can click the "heart" sign next to the headline in the restaurant detail information page, the restaurant will be added to the user's favorite list, which can be found on the right side of the bottom navigation bar. The favorite List can display all the restaurants (without distance) which have been added as the favorite and ranking by praise. Restaurant Detail Information Page can also be arrived from this interface.



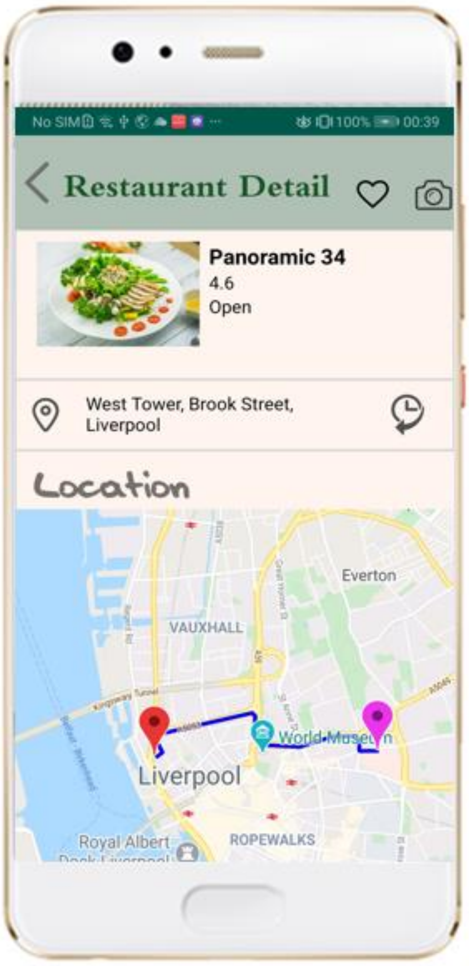
Panoramic 34
West Tower, Brook Street, Liverpool
4.6
Open

Premier Inn Liverpool City Centre (Liverpool One) hotel
48 Hanover Street, Liverpool
4.4
Open

Fig 12. Favorite Restaurant Interface

● **Navigation System**

In the detail information page, next to the word of address, there is an icon, if clicks this, the app will check whether the user's phone has already downloaded and installed the Google Map. If Google Map has already been installed, the "Vegetarian Radar" app will jump to the third-party navigation software and start the navigation, else user will be reminded that no navigation software is available.



⇒

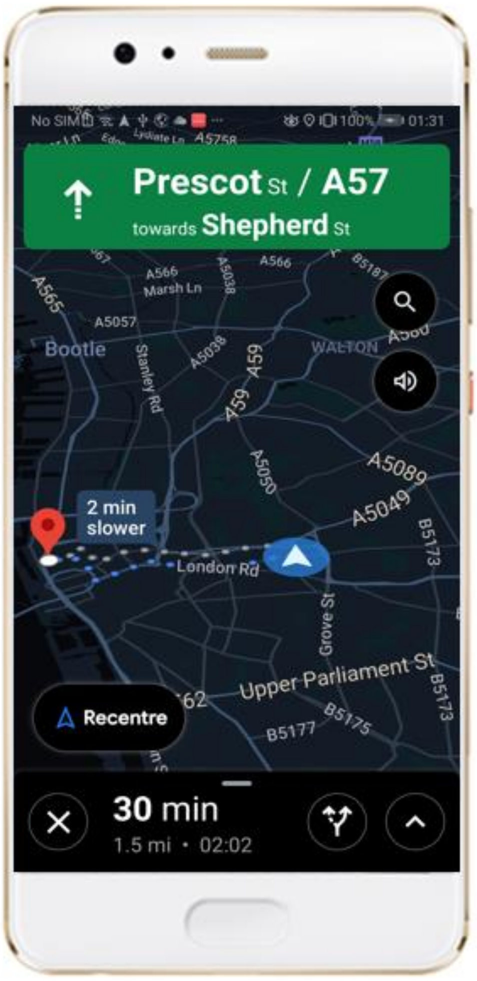


Fig 13. Vegetarian Radar App

Fig 14. Google Map App

4 Implementation

4.1 Software & Hardware Resources

DEVICE	VERSION	FUNCTION
LAPTOP	LENOVO Windows 64-bit	programming
PHONE	HUAWEI P10	testing
ANDROID STUDIO	3.6.2	Interface design & code implement
SQLITE	3.28.0 (build-in Android Studio)	create database & Store data

Table 1. Software & Hardware Resource

4.2 Development Method

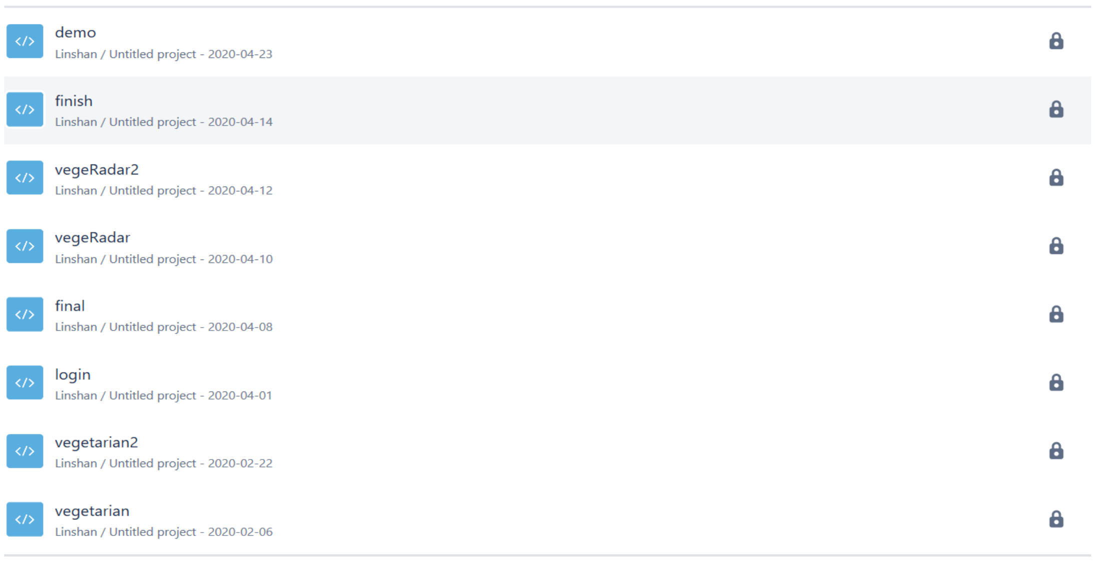
The project chose Iterative Development as the development method, since it is a powerful methodology that designs and implements only a part of the product at a time, and every iterative process contained requirements analysis, system design, implementation and testing, enabling reducing the risk of development and getting higher success rate and productivity.

The reason why choosing this method as the project development method was that the code development time for this project was less than half of a year, and only one developer is responsible for the whole project, including choose the project, GUI design, content design, get data, implement and test, therefore, there might be a possibility of designing all the functions at the begin of the project but fail to finish some of them at the end, or implementing the whole project without testing but nothing display at the testing phase, the time for finding the existence of the bug will be huge under this situation.

In addition to the advantages mentioned above, the project consisted of five parts, each of them are separated expect data sharing, using the Iterative Development method and setting the part that was currently being completed as the main activity interface in AndroidManifest.xml can omit the rest of the project while testing this part, which can save the time of development. Also, since one part were after another was finished, problems in one part did not affect the others, reducing rollback costs as well as lower risk of error.

4.3 Data Backup

In order to minimize the cost caused by improper modifications and prevent the losing of work due to system failures, the developer updated the code stored in Bitbucket after one part was finished or before big changes. Store codes there as a backup can enhance the privacy of code as well as prevent the loss of code or data. After registering an account in Bitbucket, an empty Git repository was required to be created and SourceTree should be downloaded as a client to clone the code to the repository, when cloning a repository, a connection was created between the Bitbucket server and the local system. To make it clear and convenient to roll back, the developer created a new repository to store all the completed code rather update the old repository.



demo
Linshan / Untitled project - 2020-04-23

finish
Linshan / Untitled project - 2020-04-14

vegeRadar2
Linshan / Untitled project - 2020-04-12

vegeRadar
Linshan / Untitled project - 2020-04-10

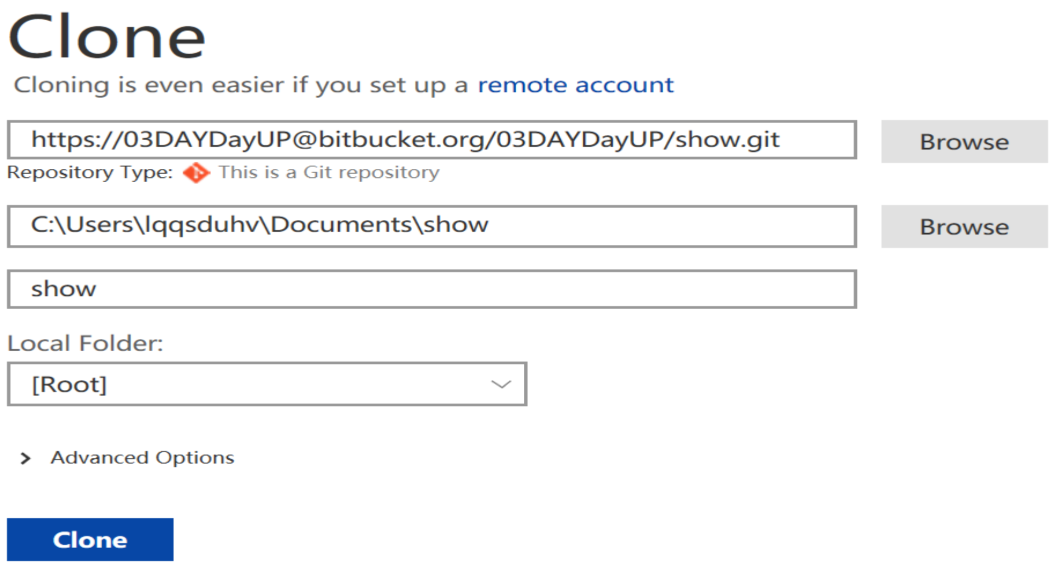
final
Linshan / Untitled project - 2020-04-08

login
Linshan / Untitled project - 2020-04-01

vegetarian2
Linshan / Untitled project - 2020-02-22

vegetarian
Linshan / Untitled project - 2020-02-06

Fig 15. All the repositories



Clone
Cloning is even easier if you set up a remote account

https://03DAYDayUP@bitbucket.org/03DAYDayUP/show.git
Repository Type: This is a Git repository
C:\Users\lqqsduhv\Documents\show
show
Local Folder: [Root]

Fig 16. Clone code to a new repository

4.4 Data Structure

The main data structures used in this project were ArrayList and HashMap.

HashMap:

HashMap is used as a temporary data structure to read data from json object (through specific URL and API key, the required data were got and showed in browser in JSON format), the key was String format, as well as the value. In order to split the information between different restaurants, jsonObejct were switched to jsonArray, and stored in a List one by one.

As for a restaurant, the most important information is its name and address, therefore before adding the information to the HashMap, the restaurant's name and address were checked whether is null or not, if neither of the two values was null, the information of the restaurant would be crawled from the browser and stored under corresponding key, keys for a restaurants including name, address, latitude of restaurant, longitude of restaurant, distance between user and restaurant (should be calculated with algorithm), rating, and whether open. Since json data was switched to String while splitting different restaurant's information, and in order to unify the format of value, all the data crawled from browser were stored in String format as the value of the HashMap, including latitude and longitude.

ArrayList:

ArrayList was used in different ways in this project, one used to store all the HashMap which contains detail information of a specific restaurant.

The relationship between this ArrayList and HashMap is showed as below:

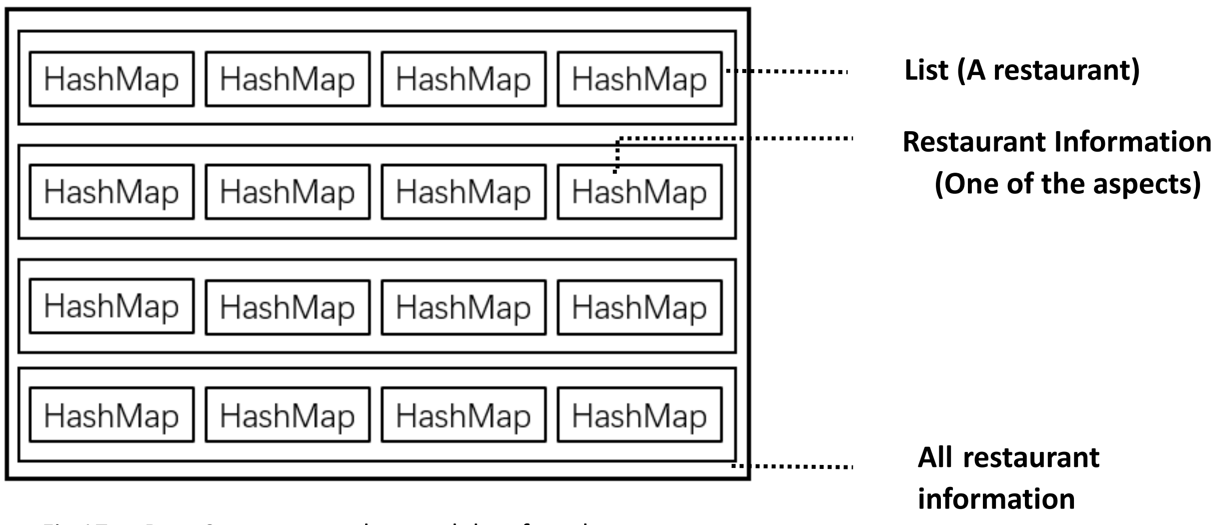

HashMap
List (A restaurant)
Restaurant Information (One of the aspects)
All restaurant information

Fig 17. Data Structure used to read data from browser

This ArrayList was useful when storing all restaurant information to database and show markers on map. When user clicks “VEGETARIAN RESTAURANT” button in the Map Page, the app got the information of nearby restaurants through Google Map Places API, and stored them in the List<HashMap<String, String>> nearbyPlacesList mentioned above. After the preparatory work, the app would run in the background and read data from the nearbyPlaceList one by one and get the required information. For showing nearby restaurants as icons in map fragment, only latitude, longitude and name were needed, they were stored in a new HashMap and displayed the markers. At the meanwhile, information of each restaurant would be read one at a time from the List and added to the specified database.

Expect read and store restaurant information, List and HashMap were also used to get the route from current position to target restaurant and showed route on map in restaurant detail information page. The usage was similar to the one mentioned above, the differences were in the first layer of the nest, only latitude and longitude of the inflection point were needed and stored in a HashMap; the second layer stored one complete route which contains all the points in the first nested layer; the third layer stored the walking route of all routes since the app only considered the means of getting there on foot. Therefore, the nest of HashMap and List looks like this:

List<List<HashMap<String, String>>>

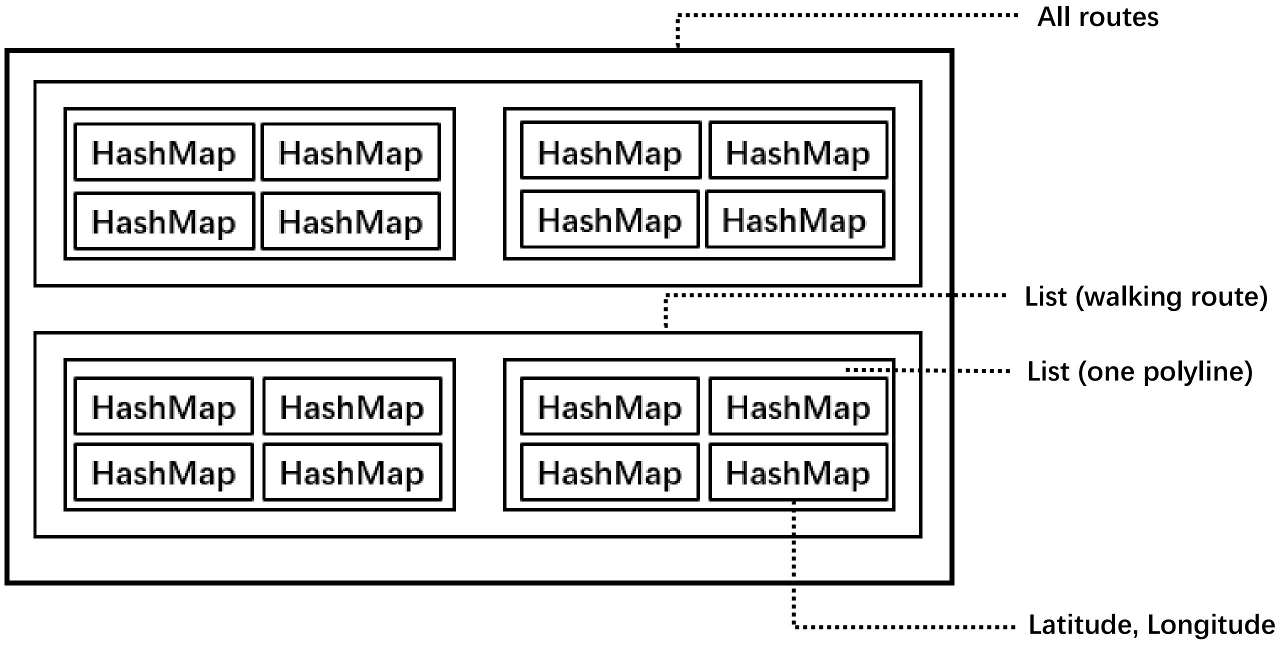


Fig 18. Data Structure used to store routes

The data type used in the project included int, String, and double. And all the data stored in database were TEXT type.

4.5 Algorithm

The main algorithm used in the project was calculating the distance (straight line distance) between user's current location and target restaurant since restaurant would be listing by distance in list interface but only latitude and longitude of restaurant could be got through Google Map API.

The Pseudo Code is shown as below:

```
function CalculationByDistance (curLat, curLon, resLat, resLon)
    Radius ← 6371
    PI ← Math.Pi
    LatAngle ← resLat – curLat
    LonAngle ← resLon – curLon
    LatRadians ← LatAngle * PI / 180
    LonRadians ← LonAngle * PI / 180

    curLat ← curLat * PI / 180
    resLat ← resLat * PI / 180

    distanceL ← sin (LatRadians / 2) * sin (LatRadians)
                + cos (curLat) * cos (resLat) * sin (LonRadians / 2)
                * sin (LonRadians / 2)

    Temp ← sqrt (distanceL)
    C ← 2 * asin (temp)

    km ← radius * C

    return km

    end function
```

The latitude and longitude of the two places needed to be switched to 3D rectangular coordinates (only consider X-axis and Y-axis in this project):

$X = R \times \cos\alpha \times \cos\beta$ (α: latitude β: longitude)
$Y = R \times \cos\alpha \times \sin\beta$

Though Pythagorean theorem, straight line distance = ((x1-x2) ^ 2 + (y1-y2) ^2) ^ 0.5
Then change the distance between two points according to the chord length (arc length)

4.6 Code Implementation

Some of the code were got from CSDN, Android Developer and Android Tutorial Point.

4.6.1 Database Implementation

The developer created three databases in this project, each database has a table. Table "customer" was used to store customers' information, including their name, password, and phone number; table "restaurant" was created to store information (name, address, latitude, longitude, rating, distance, open now) on nearby restaurants; table "favorite" was used to connect user and their favorite restaurants, which stored phone number of a user as the primary key and name, address, rating, distance, open now of the restaurants he marked as favorite. Phone was both the primary key of "customer" table and the foreign key of "favorite" table.

Databases were created through classes which extends SQLiteOpenHelper, though implementing several subclass onCreate (SQLiteDatase), onUpgrade (SQLiteDatabase, version1, version2), the class achieved the functions of creating the target database if it did not exist, upgrading the database as necessary.

"customer" table: onCreate() method and onUpgrade() method

```
@Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS customer(" +
                    "name TEXT," +
                    "password TEXT," +
                    "phone TEXT   PRIMARY KEY)");
    }
@Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
            db.execSQL("DROP TABLE IF EXISTS customer");
            onCreate(db);
    }
```

The databases were created through onCreate() method when the app was first run and the databases were required. Every time the column of the databases modified, the old version would be deleted and replaced by a new one. New items would be added to the database through the SQL statement INSERT INTO, and updated by

```
            UPDATE table SET column = value WHERE [condition]
```

In some conditions the program should traverse all the data in a table, the API of a cursor would read the columns that were returned from the query and iterate over the rows of the result set (Adam, 2016). In addition to this, an ArrayList would be created as a container to store data in a certain database, the collection of the ArrayList was defined by another Java class, and after getting the whole information of an item, the values would be added to the ArrayList. The process repeated constantly until the cursor moved to the last row (item) of the database and the List would be transferred to the method where data were needed.

Method to get all data in "customer" table

```
public ArrayList<User> getAllData() {
    ArrayList<User> list = new ArrayList<User>(); //create a container
    Cursor cursor = db.query("customer", null, null, null, null, null, "name
DESC");                                    //descending
    while (cursor.moveToNext()) {
        String name = cursor.getString(cursor.getColumnIndex("name"));
        String password = cursor.getString(cursor.getColumnIndex("password"));
        String phone = cursor.getString(cursor.getColumnIndex("phone"));
        list.add(new User(name, password, phone));   //add to list
    }
    return list;
}
```

While passing parameters through cursor, program can set the ranking function, the parameter could replace the ORDER BY statement in SQLITE, the data in the table would be sorted, data would also be added in the ArrayList in this order.

4.6.2 Function Implementation

The initial interface was set to the Login interface through AndroidManifest.xml:

```
<activity
    android:name=".loginSystem.LoginActivity">
```

Other interfaces were also been declared as activities in this file.

Since the project is an App which needed to achieve the display of the interface, therefore, every class was necessary to extend AppCompatActivity class, which acts as a super class for activities to take advantage of support library action bar features and should be imported from android.appcompat.app package (Oclemy, 2019). An onCreate(Bundele savedInstanceState) method would be override when the java class extends AppCompatActivity class, interface which needs to be rendered including action bar of windows, components set in XML, and required database (if not exist) would be created here.

● Login System

The Login System were consisted of three parts: login directly with existing account, create a new account, retrieve the account password. The latter two interfaces needed to be reached from the first interface by setting the text (New Customer and Forget Password) as clickable (similar to a button). Once the user clicks one of the buttons, the onClick (View v) method in LoginActivity. class which implemented View. OnClickListener interface would use “switch case” to check which button has been clicked. If “New Customer” or “Forget Password” button was clicked, the interface would jump to the corresponding new interface through Intent object.

Use switch case and Intent to achieve convert between pages

```
switch (v.getId()) {
        case R.id.Login_newCustomer: // register
            startActivity(new Intent(this, RegistrationActivity.class));
            finish();
            break;
        case R.id.Login_forget:
            startActivity(new Intent(this, ResetPassword.class));
            finish();
            break;
}
```

However, if the user clicked the “OK” button to directly login, the data entered as name and password would be get and compared with the “Customer” table whether the account has already existed. All data would be read by a cursor and stored in an ArrayList. Then in the switch case, data in this ArrayList would be read one by one to check the match. If the account matched one in the database, the interface would jump to the Map interface, else an error message would show up through Toast.

Retrieve the database and check whether user existed

```
ArrayList<User> data = mDBOpenHelper.getAllData();
    boolean match = false;
    for (int i = 0; i < data.size(); i++) {
        User user = data.get(i);
        if (phone.equals(user.getPhone())) {
            match = true;
            break;
        } else {
            match = false;
        }
    }
```

If the interface jumped to the registration interface, the user needed to enter a username, two identical passwords, and a unique phone number. After checking whether the user has entered all the information needed, the system will check in database whether the phone number has been registered, and give user corresponding suggestions. Phone number and password have their own formats, which are implemented through regular expression.

```
    public static boolean checkPassword(String pwd) {
    String regExp = "^[\\w_]{8,16}$";
    if(pwd.matches(regExp)) {
        return true;
    }
    return false;
}
```

Password

```
public static boolean checkPhone(String phone_number){
    String regExp = "[0-9]*";
    if(phone_number.matches(regExp)){
        return true;
    }
    return false;
}
```

Phone Number

In addition to the above, a disclaimer with a check box above the "REGISTER" button should be agreed by user before finish registration, in order to check whether the check box was ticked, the system used a Boolean value in onClick (View v) method in Java class and CheckBox in xml.

● Map System

■ Check if the App can run normally

If the user successfully logged in the login system, the interface would jump the map interface which initially showed the user's current location and his surrounding streets, the map was set as the normal type. In order not to affect the functions, the project would check the user permission and phone version before showing the map, if the version >= 23, and phone GPS was opened, location request would be created to start receiving updates, which means nearby restaurants can be searched.

Since the position of user might move, the map and information would update every two minutes. Else if the version of Android phone less than the requirement or the phone does not give the App or GPS permission, corresponding suggestions would be given in the format of alert.

Check version and permission (Learned from Android Tutorial Point)

```
if (android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
    if(ContextCompat.checkSelfPermission(this,
                                          Manifest.permission.ACCESS_FINE_LOCATION)
                                        == PackageManager.PERMISSION_GRANTED) {

        mFusedLocationClient.requestLocationUpdates(mLocationRequest,
        mLocationCallback, Looper.myLooper());

        mGoogleMap.setMyLocationEnabled(true);
}
```

Set update interval (Learned from Android Tutorial Point)

```
mLocationRequest = new LocationRequest();
mLocationRequest.setInterval(120000); // two minute interval
mLocationRequest.setFastestInterval(120000);
mLocationRequest.setPriority(
                    LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY)
```

■ **Display nearby restaurants**

After checking the permission, a map with user’s current location and surrounding streets was displayed. By clicking the “VEGETARIAN RESTAURANT” button to get location of nearby vegetarian restaurants, since the Google Maps API was imported through JavaScript, the URL for nearby search would be create by append(). The integral HTTP URL contained two parts: path and parameter. Google Map Places API has specific path for searching nearby facilities, which took a text input and returns places:

```
https://maps.googleapis.com/maps/api/place/nearbysearch/json?
```

output format of return

Required parameters contains

1. API key: Specific application’s API key, which the developer got from Google Cloud Platform Console, the key was also added in AndroidManifest.xml

```
<meta-data
        android:name="com.google.android.geo.API_KEY"
        android:value="AIzaSyDJrAUAdMX80XmvBru6WbZ43KynzF1P-cE" />
```

2. Current position of user: Position includes latitude and longitude, these parameters was got by the onLocationResult() methods which implements OnMapReadyCallback interface, this method was called when device location information is available.

3. Searching radius: A searching radius was set in the project to provide users with search results within a specific range, the radius in this project was 5 kilometers.

4. Type of search content: Text input that identifies the search target, which must be a String. The type input for this project was set as “vege”, as the abbreviation of vegetarian restaurant.

Construct URL for searching vegetarian restaurants less than 5 kilometers

```
private String getUrl(double latitude, double longitude, String nearbyPlace) {
    StringBuilder googlePlacesUrl = new StringBuilder
    ("https://maps.googleapis.com/maps/api/place/nearbysearch/json?")
        googlePlacesUrl.append("location=" + latitude + "," + longitude);
        googlePlacesUrl.append("&radius=" + radius);
        googlePlacesUrl.append("&type=" + nearbyPlace);
        googlePlacesUrl.append("&sensor=true");
        googlePlacesUrl.append("&key=" +
    "AIzaSyDJrAUAdMX80XmvBru6WbZ43KynzF1P-cE");
        Log.d("getUrl", googlePlacesUrl.toString());
        return (googlePlacesUrl.toString());
    }
```

The integral URL would be executed by creating a http connection to communicate with URL and reading data from URL through BufferedReader. The URL would be passes in an object, the map fragement (mGoogleMap) would also be passed as another parameter.

After the BufferedReader reading the content in URL, the project parsed the content, and stored them in the List<HashMap<String, String>> nearbyPlacesList. HashMap stored the information of a restaurant. Information of the restaurants would be crawled from the browser and stored under corresponding key, keys for a restaurant including name, address, latitude of restaurant, longitude of restaurant, distance between user and restaurant, rating, and whether open. Since json data were switched to String while splitting different restaurant’s information, and in order to unify the format of value, all the data crawled from browser were stored in String format as the value of the HashMap, including latitude and longitude.

Get if a restaurant is open and store in HashMap

```
String openTime = "";
if (!googlePlaceJson.getJSONObject("opening_hours").isNull("open_now")) {
    openTime =
googlePlaceJson.getJSONObject("opening_hours").getString("open_now");
    if (openTime.equalsIgnoreCase("true")) {
        openTime = "Open";
    } else {
        openTime = "Closed";
    }
}
```

After the preparatory work, the app read data from the nearbyPlaceList one by one and got the required information. For showing nearby restaurants as icons in map fragment, only latitude, longitude and name are needed, they would be stored in a new HashMap. At the meanwhile, information of each restaurant would be read one at a time from the List and added to the specified database.

These tasks were done in background as asynchronous tasks and sent the result to UI after finish. Another method called onPostExecute(String result) which also extended from AsyncTask class would execute the tasks (display markers and add restaurants to “restaurant” database) after the background finished tasks.

- **Bottom Navigation Bar**

The bottom navigation bar was achieved by XML file and onClick(View v) method in Java class. In xml files, both text and icons of the bar were set as clickable, when they were clicked, the onClick(View v) method which implemented View. OnClickListener interface would be triggered and use “switch case” to check which area has been clicked.

```
android:clickable="true"
android:onClick="onClick"
```

The app would jump to the corresponding interface through creating a new Intent activity between the current Java class and the target class and start the activity.

```
case R.id.bottom_bar_exercises_like:
case R.id.bottom_bar_image_like:
    Intent intent3 = new Intent(this, MyFavourite.class);
    startActivity(intent3);
    finish();
    break;
```

● List System

■ Show List

A list that can display all data should be achieved by two XML files, one set the format for the whole page, except two buttons for listing method, a bottom navigation bar, a ListView that displayed a list of vertically scrollable items was created, which specified the width and height of the entire list; another XML file was set to regulate the format of a line in the list, including the width and height of a line, the relative position between attributes.

For the onCreate(Bundle savedInstanceState) in ListActivity.java, not only interface needed to be rendered, components set in XML, required database were created here, but also the ListView was stated and set a Adapter which extends from BaseAdapter, the adapter was used to connect to the ArrayList that stored all the data of the "restaurant" table, the methods in BaseAdepter getCount(), getItemId() could return total of items in the list and the position of a specified item. The values were helpful in getView() method when creating the layout for each list row, each item was converted in to a row in the ListView by using .setText(). Therefore, one row was created after the previous line was generated, all the lines would be displayed at the same time with a scroll bar.

BaseAdapter to display the list (Learned from CSDN)

```
lv = (ListView) findViewById(android.R.id.list);
lv.setAdapter(new BaseAdapter() {

    @Override
    public int getCount() {
        return resList.size();
    }

    @Override
    public long getItemId(int position) {
         return position;
    }

    @Override
    public View getView(final int position, View convertView, ViewGroup
parent) {
        View v;
        if (convertView == null) {
            LayoutInflater inflater = ListActivity.this.getLayoutInflater();
            v = inflater.inflate(R.layout.list_place, null);
                } else {
            v = convertView;
```

One row of the ListView

■ Detail Information Interface: Transfer parameters

In order to achieve jump from the whole list to a specific restaurant detail information interface, an interface AdapterView. OnItemClickListener was set for a callback to be invoked when an item in the AdapterView had been clicked. Since through the methods extended from BaseAdepter, the position of the clicked item was got, information of the restaurant corresponding to the position were transferred to the restaurant detail information interface by Intent and displayed in a format set with restaurant_info.xml. Data were added to the intent by putExtra(String name, value) in ListActivity.java or ListActivityDis.java and retrieved from the same intent by getStringArrayExtra(String name) in DeatailedInfo.java.

Add data to Intent (only an example)

```
lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
        VegeRestaurant vr = resList.get(position);
        Intent intentClike = new Intent(parent.getContext(), DetailedInfo.class);
        intentClike.putExtra("name",vr.getName());
        startActivity(intentClike);
        finish();
    }
```

Same

Retrieve data from Intent

```
    Intent intent = getIntent();
    name = intent.getStringExtra("name");
```

Since while creating the intent, it was set to pass the data in intent to DetailedInfo.class, so in this class, an intent was stated to receive the data by getIntent().

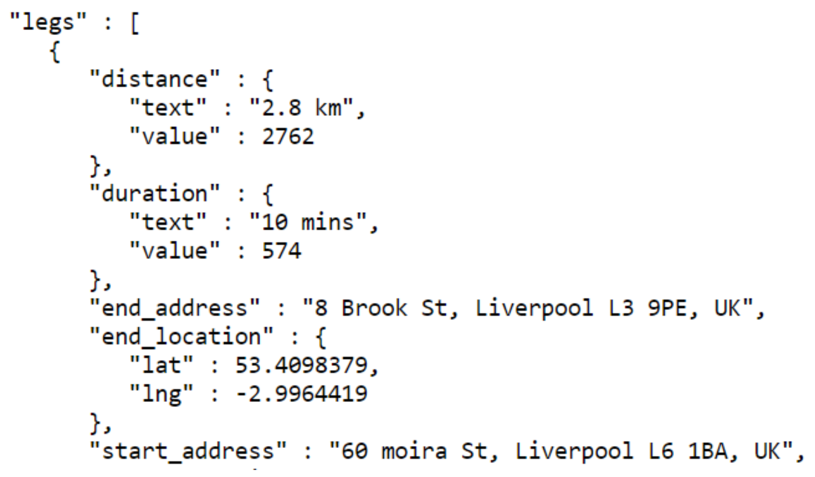
■ Detail Information Interface: Display route

The implementation of the map at the bottom of the page is a bit same like the one in the map interface. However, the HTTP URL used here replaced the Google Map Places API to Google Map Directions API, and the required parameters were latitude and longitude of current position of user and target restaurant (the one corresponding to the position), API key, but the two URLs were generated in a same way

Google Map Directions API:

```
https://maps.googleapis.com/maps/api/directions/json?
```

The integral URL would be executed by creating a HTTP connection to communicate with URL and reading data from URL through BufferedReader. The URL would be passed in an object, and the map fragement (mGoogleMap) would also be passed as another parameter.



```
"legs" : [
   {
      "distance" : {
         "text" : "2.8 km",
         "value" : 2762
      },
      "duration" : {
         "text" : "10 mins",
         "value" : 574
      },
      "end_address" : "8 Brook St, Liverpool L3 9PE, UK",
      "end_location" : {
         "lat" : 53.4098379,
         "lng" : -2.9964419
      },
      "start_address" : "60 moira St, Liverpool L6 1BA, UK",
```

Fig 19. One route from current position to a restaurant

After the BufferedReader read the content in URL, the project parsed the content, and store them in the List<List<HashMap<String, String>>>route. HashMap stored the latitude and longitude of the inflection point. The second layer stored one complete route which contains all the points in the first nested layer; the third layer stores the walking route of all routes. These tasks were done in background as asynchronous tasks and sent the result to UI after finished. Another method also extended from AsyncTask class onPostExecute(String result) would execute the tasks of drawing the route on map after the background finish tasks. The inflection points and line between two points were added to the map fragment one by one until all points were drawn.

- **Detail Information Interface: Screen capture**

The screen capture function in detail information interface was achieved by the app called the screenshot function that came with the device, only the area of the app would be captured (no Android notification bar). The width and height of the app would be got and picture was generated and saved as a BitMap. The file name of the picture was set as the current time (number of milliseconds since 0:00 on January 1st, 1970). The path would check whether the picture already existed, and create one if not exist, since the milliseconds were different, therefore no two identical file names, so the previous capture would not be overwritten by the back. All pictures would be stored in SD card through the path got by Environment.getExternalStorageDirectory().

Call screen capture function of the device

```
View dView = getWindow().getDecorView();
dView.setDrawingCacheEnabled(true);
dView.buildDrawingCache();
```

● Favorite System

If clicking the “heart” sign next to the headline in the restaurant detail information page, the background color of the sign would switch to pink, the change was realized by onClick(View v), once the command met the corresponding case, the sign would set to another color by v.setBackgroundColor(Color. rgb()), and information of the restaurant would be added to “favorite” table.

The favorite restaurants could be found in user’s favorite list. The implementation of the favorite list was similar to the list which shows nearby vegetarian restaurants. Data would be read from the “favorite” table a cursor and showed on device through an adapter.

● Navigation System

One of the aims of this app was navigating user to the restaurant whey wants to visit. If clicking the icon next to the word of address in the detail information page, the app will check whether the user’s phone has already download and install the Google Map. It is not necessary for the app to check the version of the phone and GPS permission since these requirements have already been checked in the map interface. The program gets all the package names installed in the phone through PackageManager and stored them in a List, if one of the package name equals to “com.google.android.apps.maps”, which means Google Map has been installed in the phone. Therefore, latitude and longitude of user and target restaurant would be passed through an intent to Google Map, and the “Vegetarian Radar” app would jump to this third-party navigation software and start the navigation.

Method to achieve jump between apps

```
public void NaviGoogle(Context context, String latitude, String longitude) {
    if (isAvilible(context, "com.google.android.apps.maps")) {
        Uri gmmIntentUri = Uri.parse("google.navigation:q=" + latitude + "," +
longitude + "&mode=w");
        Intent mapIntent = new Intent(Intent.ACTION_VIEW, gmmIntentUri);
        mapIntent.setPackage("com.google.android.apps.maps");
        mapIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK );
        context.startActivity(mapIntent);
    }
```

5 Testing

Premise:

Name	Password	Phone
abc	123456	074567891

Table 2. Data in "customer" table

Test Case:

■ Login System

Test Case Description	Test Steps	Test Data	Expected Result	Pass/Fail
Directly Login				

Check customer login with valid data	1. Clicked the icon of the app
	2. Entered Username
	3. Entered Password
	4. Click "OK"
		Username = abc
		Password = 12345678
			User should login into the app	Pass

Check customer login with invalid data	1. Clicked the icon of the app
	2. Entered Username
	3. Entered Password
	4. Click "OK"
		Username = efg
		Password = 12345678
			Login failed, and alert "The user does not exist" shows up	Pass

Check customer login with empty data	1. Clicked the icon of the app
	2. Click "OK"
		Username = ""
		Password = ""
			Login failed, alert "Name or Password cannot be empty"	Pass

Register a new account				

Check register with valid data	1. Clicked "New Customer"
	2. Entered Username
	3. Entered Passwords
	4. Entered phone
	5. Clicked to agree to the disclaimer
	6. Clicked "REGISTER"
		Username = "efg"
		Password1 = 123456/a
		Passwords2 = 123456/a
		Phone = 076241639
			User should login into the app, the information would be added to the "customer" table	Pass

Table 3. Testing table 1

Test Case Description	Test Steps	Test Data	Expected Result	Pass/Fail

Check register with invalid data (two passwords different)	1. Clicked “New Customer”
	2. Entered Username
	3. Entered Passwords
	4. Entered phone
	5. Clicked to agree to the disclaimer
	6. Clicked “REGISTER”
		Username = “hhh”
		Password1 = 123456/b
		Passwords2 = 123456/a
		Phone = 076228375
			Register failed, alert “Passwords not match” shows up	Pass

Check register with invalid data (Did not agree to the disclaimer)	1. Clicked “New Customer”
	2. Entered Username
	3. Entered Passwords (same)
	4. Entered phone
	5. Clicked “REGISTER”
		Username = “hhh”
		Password1 = 123456/a
		Passwords2 = 123456/a
		Phone = 076228375
			Login failed, and alert “Please check the disclaimer” shows up	Pass

Check register with invalid data (phone number already registered)	1. Clicked “New Customer”
	2. Entered Username
	3. Entered Passwords
	4. Entered phone
	5. Clicked to agree to the disclaimer
	6. Clicked “REGISTER”
		Username = “hhh”
		Password1 = 123456/a
		Passwords2 = 123456/a
		Phone = 074567891
			Login failed, alert “This Phone Number has already registered”	Pass

Check register with invalid data (password format incorrect)	1. Clicked “New Customer”
	2. Entered Username
	3. Entered Passwords
	4. Entered phone
	5. Clicked to agree to the disclaimer
	6. Click “REGISTER”
		Username = “efg”
		Password1 = 123456??
		Passwords2 = 123456??
		Phone = 076241639
			Login failed, alert “PASSWORDS must be 8-16 digits of English, numbers, / "	Pass

Table 4. Table of Testing 2

Test Case Description	Test Steps	Test Data	Expected Result	Pass/Fail
Check register with invalid data (phone format incorrect)	1. Clicked “New Customer” 2. Entered Username 3. Enter Passwords 4. Entered phone 5. Clicked to agree to the disclaimer 6. Clicked “REGISTER”	Username = “hhh” Password1 = 123456/a Passwords2 = 123456/a Phone = 076….23.ds	Register failed, alert “This Phone Number has already register” shows up	Pass
Reset Password				
Check reset password with valid data	1. Clicked “Forget Password” 2. Entered Phone 3. Entered Passwords 4. Clicked “RESET PASSWORD”	Phone = 074567891 Password1 = 123456/a Passwords2 = 123456/a	User should login into app, the information would be updated in table	Pass
Check reset password with invalid data (phone is not registered)	1. Clicked “Forget Password” 2. Entered Phone 3. Entered Passwords 4. Clicked “RESET PASSWORD”	Phone = 1 Password1 = 123456/a Passwords2 = 123456/a	Login failed, alert “The user does not exist” shows up	Pass
Check register with invalid data (phone or password empty)	1. Clicked “Forget Password” 2. Entered Passwords 3. Clicked “RESET PASSWORD”	Phone = “ ” Password1 = 123456/a Passwords2 = 123456/a	Login failed, alert “Please complete your information" shows up	Pass
Check register with invalid data (passwords different)	1. Clicked “Forget Password” 2. Entered Phone 3. Entered Password 4. Clicked “RESET PASSWORD”	Phone = 074567891 Password1 = 123456/a Passwords2 = 123456/b	Login failed, alert “Passwords not match " shows up	Pass

Table 5. Table of Testing 3

■ Map System

Test Case Description	Test Steps	Expected Result	Pass/Fail
Check the effect of GPS permission	1. Closed device GPS 2. Login the app	Map shows up but current location does not show	Pass
Check the function of finding and displaying nearby restaurants	1. Opened device GPS 2. Login the app 3. Clicked "VEGETARIAN RESTAURANT" button above the map fragment	Vegetarian restaurants less than 5 kilometers showed up on map by red icons, their name would show up and camera would move when click the icon	Pass

Table 6. Table of Testing 4

■ List System

Test Case Description	Test Steps	Expected Result	Pass/Fail
List interface			
Check the display of the list	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface	Vegetarian restaurants within 5 kilometers will be sorted by praise from high to low	Pass
Check the switch of the two sorting methods	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface 4. Clicked the button of "DISTANCE" 5. Clicked the button of "RATING"	The restaurant firstly sorted by praise from high to low and change to sorted by distance form high to low, the change to sorted by praise from high to low	Pass

Test Case Description	Test Steps	Result Expected	Pass / Fail
Detail Information Interface			
Check the display of a specific restaurant detail information interface	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface 4. Clicked a specific restaurant in the list	The detail information of the viewed restaurant will show on screen in the fixed format	Pass
Check the screen capture function	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface 4. Clicked a specific restaurant in the list 5. Clicked the "camera" icon	The screen shortcut is got and saved in SD card, "save successfully" displayed on the screen	Pass
Check the detail information interface without clicking "Vegetarian Restaurant" button	1. Login the app 2. Jumped to the "List" interface	No restaurants shown in the interface	Pass

Table 6. Table of Testing 4

■ Favorite System

Test Case Description	Test Steps	Expected Result	Pass/Fail
Check the favorite icon in detail information page	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface 4. Clicked a specific restaurant in the list 5. Clicked the "heart" icon	The background color of the "heart" icon changes to pink, "Good Restaurant" displayed on the screen	Pass
Check the function of finding and displaying nearby restaurants	1. Login the app 2. Click "vegetarian restaurant" button in map interface 3. Jumped to the "List" interface 4. Clicked a specific restaurant in the list 5. Clicked the "heart" icon 6. Jumped to the "favorite" interface	The restaurants marked as "Good Restaurant" are displayed in list and sorted by praise from high to low	Pass
Check the restaurants in favorite interface	1. Login the app 2. Jumped to the "favorite" interface 3. Clicked one of the items	The interface jumps to the specific restaurant detail information interface	Pass

Table 7. Table of Testing 5

■ Navigation System

Test Case Description	Test Steps	Expected Result	Pass/Fail
Check the route display in detail information page	1. Login the app 2. Jumped to the “List” interface 3. Clicked a specific restaurant in the list	Current Position and target restaurant are displayed in the form of icons, a blue fold line connects them	Pass
Check the navigation icon in detail information page	1. Login the app 2. Jumped to the “List” interface 3. Clicked a specific restaurant in the list 4. Clicked the “navigate” icon next to the word of the address	The app will jump to the Google Map App, navigating customer from his current position to the target restaurant, an estimated time spent will also be showed	Pass
Check whether Google Map is installed on the device	1. Uninstalled Google Map App 2. Login the app 3. Jumped to the “List” interface 4. Clicked a specific restaurant in the list 5. Clicked the “navigate” icon next to the word of the address	The app stays in the current interface and “Google Map has not been installed” shows up	Pass

Table 8. Table of Testing 6

■ Whole Project

Test Case Description	Test Steps	Expected Result	Pass/Fail
Check the "back" button in login system	1. Login the App 2. Jumped to "New Customer" or "Forget Password" 3. Clicked the "Back" button	The interface jumps back to directly login page smoothly	Pass
Check the switch between Login System to Map System	1. Login the app 2. Entered Username 3. Entered Password 4. Clicked "OK"	The interface jumps to the map interface which displays customer's current location and surrounding streets, also "Map" will be displayed on screen	Pass
Check the switch between Map System to List System	1. Login the app 2. Entered Username 3. Entered Password 4. Clicked "OK" 5. Clicked "List" button in the bottom navigation bar	The interface jumps to the list interface that vegetarian restaurants within 5 kilometers will be sorted by praise from high to low also "List" will be displayed on screen	Pass
Check the switch between Map or List System to Favorite System	1. Login the app 2. Entered Username 3. Entered Password 4. Clicked "OK" 5. Clicked "Favorite" button in the navigation bar	The interface jumps to the favorite interface smoothly that displays all the restaurants that the customer marked as "Good"	Pass

Table 9. Table of Testing 7

Name	Password	Phone
abc	123456/a	074567891
efg	123456/a	076241639

Table 10. Data in "customer" table after test

6 Evaluation

6.1 Compare with Original Objectives

The original aims of the project were to provide convenience for vegetarians, let customers share the restaurants they like and provide customers with routes and solutions to the destination they chose. The final forming app did follow the original design and aims, however, compared the finished product with the original objectives, there are some modifications because of the time limitation and developer's inspiration while developing.

The mainly modification was that in the original objectives plan, the app had a comment system, however, considering a rating value crawled from the Google Map that could indicate customers' evaluations of the restaurants, the comment of user might seem cumbersome, therefore it was replaced by a specific route display. In the meanwhile, because the comment system was removed, the customer can only see their own information in the app, one of the desirable functions in the original objectives: allowing private chat with other users were deleted.

In addition to the modifications mentioned above, the setting page was deleted and substituted with a favorite list to store vegetarian restaurants the user like, since in the communication with testers, the developer realized that providing user an easier way to find the restaurant they like was a convenient and necessary function. And if user want to reset his password, the "Forget Password" button in the Login system might help, there was no need to implement an interface specifically for users to browse their basic information.

6.2 Ethical Consideration

All the information of restaurants can be captured by web crawler technology from Google Map. In order to use its data up to maximum extend so Google provides with numerous of APIs, including API for Google Map, since the "Vegetarian Radar" project take Places API of Google Maps Platform into use, so it is legal to crawl data form Google Map. All the data crawled were stored in three SQLite databases.

Since the "Vegetarian Radar" is an app navigation, which should be allowed to get the location of the user, access to memory and screenshot function of the phone, customer's phone number was also needed in registration stage. However, these requests only aim at providing a better user experience, and will not be disclosed for other commercial purposes, information of user would not be used in other software.

6.3 Questionnaire and Feedbacks

Questionnaire:

In order to receive others' opinions on the app, the developer invited 10 people from 16 to 55 years old (friends and family members, not only focus on vegetarians) to do the acceptance testing and recorded their comments. A questionnaire is designed for developers to check the functions and their feelings of the functions. The questionnaire also committed to protecting testers' privacy.

The questionnaire is below:

1. What's your age?

2. Can you operate the app well though there is no instruction?

3. Do you like the GUI design of the app?

4. Do you think the style of each interfaces is consistent?

5. What do you think the advantages of this app?

6. Are there any function you think is useless?

7. Could you give some advice for improvement?

Combining the feedback, the developer analyzed some points:
1. The GUI design and implementation are good since the background color set as green, which meets the theme of the app. The design of the icon is also vivid that testers can find the functions they need according to icons through no instruction.

2. In the "New Customer" interface, the declaimer is not detailed enough, which might cause misunderstanding. To solve this problem, the developer added a detailed description of the disclaimer in a new interface, this interface would be jumped to when the users click on the text of the declaimer.

3. The functions met demand and work correctly.

4. Some testers thought the advantages of this app were the rich restaurant information, the implementation of the favorite list, and the smooth cohesion of various interfaces.

5. One of the advices was that the app did not adapted to all Android models, only phone with 5.1 in. (1080p) was suitable.

6. Another advice was that in Map interface, only when users click the "VEGETARIAN RESTAURANT" button the nearby vegetarian restaurants would display.

7 Conclusion

Aims

- Create an Android App which aims to provide convenience for vegetarians
- The App can help customers find vegetarian restaurants nearby as well as navigate to there.
- Customers can share restaurants they recommend with others.
- Customers can record their favorite restaurants for future convenience

Objectives

Functions：

- Implement a login system to verify customer’s identity, including create a new account, forget password and a login window
- User information in register should include a username, password, and a phone number. Password should follow the rule: minimum 8 characters, maximum 16 characters, only numbers, English characters, “/” are allowed, phone number should only contain numbers

The transfer of the interfaces was achieved by switch…case… and intent, the restriction of the passwords and phone number were implemented by regular expression.

- Import Google Map API for searching nearby (less than 5 kilometers) vegetarian restaurants and show detailed route between customer and a certain restaurant

This objective was successfully implemented. The API keys were got from Google Cloud Platform Console and corresponding HTTP URLs were created according to demands.

- Show Restaurants’ information in two ways: Map and List

Data got by URLs was read by BufferedReader and stored in database, after the running of background programs, they were presented in different forms.

- Implement two ways of ordering restaurants: Distance and Praise

This objective was successfully implemented. When reading data from database, two cursors with different ordering method would be applied according to different listing ways. The transfer of the interfaces was achieved by intent.

- Display detail information of a certain restaurant

This objective was successfully implemented. Set a BaseAdapter with the ListView, got the information of the restaurant by its position in the ArrayList.

- Record customer’s favorite restaurants and show in list

This objective was successfully implemented. Set a change of the background color when user clicked the “heart” icon, and added the restaurant to the database. The implementation of the favorite list was achieved by a BaseAdapter with ListView.

- Provide a screen capture function

The screen capture function in detail information interface was achieved by the app called the screenshot function that came with the device.

Interface：

- Design user-friendly interfaces with consistent style
- Design an icon related to the theme of the app
- Achieve smooth interface switching

The app used a colza as the app icon as well as its logo, and the logo satisfied the theme of the App, the theme color was set as light green. The transfer of the interfaces was achieved by intent and switch…case….

Algorithm:

- Design and implement an algorithm for calculating distance between current location of customer and target restaurants

The latitude and longitude of the two places needed to be switched to 3D rectangular coordinates, and the distance was calculated by Pythagorean theorem.

Future Improvement:

- The private chat system can be implemented
- Animation effect can be added when screenshotting
- Nearby vegetarian restaurants could be searched without clicking the “VEGETARIN RESTAUTANT” button in map interface
- Complete model adaptation, let the app displays successfully on all Android models

8 BCS Project Criteria & Self-Reflection

- **An ability to apply practical and analytical skills gained during the degree programme**
  This app combined the using of Android Studio, and XML, and implements the application of SQLite and even API of Google Map. It was the first time for me to design and complete a program by myself, and combined the front end and back end. Though I have learned Java during my postgraduate stage, however, some classes I extended such as AppCompatActivity and interfaces I implemented such as View.OnClickListener and OnMapReadyCallback were the first time I contacted, and the packages applied to Android were different from those I used in previous project. In addition, I also learned how to create SQLite database and table in Java command and read data from them. Through implementing the mentioned above to the app, I improved my practical and analytical ability of being a developer.

- **Innovation and/or creativity**
  The "Vegetarian Radar" aimed at providing a more convenient approach to find vegetarian restaurants which meet their requirements. The operations were easy and even accessible for the elderly. Moreover, since the app also aimed at sharing ideas with friends, users can share screen shortcut of restaurants to friends. Compared with other similar apps, these functions are original.

- **Synthesis of information, ideas, and practices to provide a quality solution together with an evaluation of that solution.**
  Most restaurants navigations on the market are not suitable for vegetarians since they cannot filter omnivorous restaurants and only display vegetarian restaurants, using them might cause numerous of confusion as well as waste time. A Few apps exist on the market which aims at vegetarians have some cumbersome and unnecessary functions, which makes the interface look unconcise and provides difficulties for user's operation. I installed some of the apps with similar functions and summed up their advantages and shortcomings, then made the design for my own app. The knowledge I needed were learned from the Internet.

- **That your project meets a real need in a wider context.**
  According to the source, there were estimated ten percent of the population in Europe, and total 375 million vegetarians in the worldwide in 2014 (Cinzia, 2014). Though vegetarianism is not mainstream in society, but their needs should also be paid attention to. This app was aimed at providing convenience to vegetarians, I although the project is not face to the whole public, still there are many people in need of this, so I protected the rights of Third Parties, no personal information will be leaked or used for other commercial purposes.

- **An ability to self-manage a significant piece of work**

  To develop an Android app with complete functions and highly praised regarded by users, it was necessary for me to manage time well, and all functions were finished in seven months. The initial design stage was finished before October 31st, and study stage was started in November 1st and finished in January 15th, during the period, I self-studied skills which were needed in the app on the Internet (CSDN, Android Developers, and Android tutorial points). Since the final app consisted of five parts, I originally planned to allocate 15 days for each part, however, because I was not familiar with AppCompatActivity, I wasted some time while implementing the list function, which caused me to runout of time in the later implementation, leading the delete of private chat function. Through designing and implementing the project, my self-learning ability, analytical skills, and ability of code writing have improved, also I realized that I need to improve my ability of making plans and time management.

- **Critical self-evaluation of the process.**

  Though I have learned Java during my postgraduate stage, however, there were still many things that I contacted for the first time, at the beginning I designed the app, I felt a lot of pressure and afraid could not implementation the app well. However, after seven months of hard work, I finished the whole project and it basically met all the function and non-function requirements I originally designed, except some modifications. Therefore, I thought I have the conditions for becoming a software developer, including the ability of self-learning and code writing.

  However, there are still some places for me to improve in the future. For example, protecting users' privacy in apps. When I first designed and implemented the app, I did not consider the situation of the leakage of user information. There was no disclaimer to tell the users that the information they entered including their username, password, and phone number could only be stored in the database of the app, and would not be used in another third-party software. After being remined by the supervisor, a new interface and a checkbox was implemented to remind the customers and protect their privacy. Therefore, when designing the program in the future, I should specifically consider from the user's perspective.

  Apart from the above, since there was a function that I originally designed that was implemented because of time, though the development method of this project followed the iterative development method, the delete of the unachieved function did not affect other functions, however I realized that the ability of time management should be improved in the future. In the future implementation, I need to have a detailed understanding of the knowledge required to implement all the functions during the design stage.

Reference

Petre, A. (2016) 'Vegan vs Vegetarian - What's The Difference?' Available at:
https://www.healthline.com/nutrition/vegan-vs-vegetarian
(Accessed:13 October 2019)

'History of Vegetarianism' (no date) Available at:
http://www.edu.pe.ca/sourishigh/Pages/Cmp6-03/Beth/Homepage/history_of_vegetarianism.htm
(Accessed:13 October 2019)

Avey, T. (2014) 'From Pythagorean to Pescatarian – The Evolution of Vegetarianism' Available at:
http://www.pbs.org/food/the-history-kitchen/evolution-vegetarianism/
(Accessed: 13 October 2019)

'Becoming a vegetarian' (2018) Available at:
https://www.health.harvard.edu/staying-healthy/becoming-a-vegetarian
(Accessed: 13 October 2019)

'*Mobile Application (Mobile App)*' Available at:
https://www.techopedia.com/definition/2953/mobile-application-mobile-app
(Accessed: 14 October 2019)

'9 Advantages of Mobile Apps over responsive eCommerce websites' (2017) Available at:
https://medium.com/@KNOWARTH/9-advantages-of-mobile-apps-over-responsive-ecommerce-websites-6aed1e6db0d8
(Accessed: 25 April 2020)

'What is an API? (Application Programming Interface)' (2015) Available at:
https://www.mulesoft.com/resources/api/what-is-an-api
(Accessed: 27 April 2020)

Fakhruddin, H. *(2*015) 'Eclipse vs Android Studio: Which IDE Is Better For Android Developers?*'* Available at:
https://teks.co.in/site/blog/eclipse-vs-android-studio-ide-better-android-developers/
(Accessed: 14 October 2019)

'Android - Drag and Drop' (no date) Available at:
https://www.tutorialspoint.com/android/android_drag_and_drop.htm
(Accessed: 14 October 2019)

Panko, R. (2018) ‘The Popularity of Google Maps: Trends in Navigation Apps in 2018’ Available at:
https://themanifest.com/app-development/popularity-google-maps-trends-navigation-apps-2018
(Accessed: 15 October 2019)
DeeDeeG. (2017) ‘What is the Google Maps API? How is it used?’ Available at:
https://github.com/RefugeRestrooms/refugerestrooms/wiki/What-is-the-Google-Maps-API%3F-How-is-it-used%3F
(Accessed: 28 April 2020)

Berezhnoi, R. (2019) ‘What is UI Design and why is it important?’ Available at:
https://f5-studio.com/articles/what-is-user-interface-design-and-why-is-it-important/
(Accessed: 15 October 2019)

Oclemy. (2019) ‘Android AppCompatActivity’ Available at:
https://camposha.info/android-appcompatactivity/
(Accessed: 1 May 2020)

Adam, S. (2016) ‘Working with Databases in Android’ Available at:
https://www.informit.com/articles/article.aspx?p=2731932&seqNum=4
(Accessed: 2 May 2020)

Cinzia, F. (2014) ‘375 million vegetarians worldwide. All the reasons for a green lifestyle’ Available at:
http://www.expo2015.org/magazine/en/lifestyle/375-million-vegetarians-worldwide.html
(Accessed: 4 May 2020)

Appendices

URL for searching vegetarian restaurants less than 5 kilometers to the customer
https://maps.googleapis.com/maps/api/place/nearbysearch/json?location=53.4106443,-29651428&radius=5000&type=restaurant&sensor=true&key=AIzaSyDJrAUAdMX80XmvBru6WbZ43KynzF1P-cE

URL for searching route between L6 1BA to L3 9PE
https://maps.googleapis.com/maps/api/directions/json?origin=53.410471,-2.9651572&destination=53.409966,-2.996603&&key=AIzaSyDJrAUAdMX80XmvBru6WbZ43KynzF1P-cE

Questionnaire for testing
1. What's your age?
2. Can you operate the app well though there is no instruction?
3. Do you like the GUI design of the app?
4. Do you think the style of each interfaces is consistent?
5. What do you think the advantages of this app?
6. Are there any function you think is useless?
7. Could you give some advice for improvement?

Some of the Feedbacks:

Question	Person 1	Person 2	Person 3
What's your age?	22	30	44
Can you operate the app well though there is no instruction?	Yes	Yes	Generally yes, excepts "Vegetarian Restaurant" button
Do you like the GUI design of the app?	Yes	Yes	Yes
Do you think the style of each interfaces is consistent?	Yes	Generally Yes	Yes
What do you think the advantages of this app?	GUI	Easy operating	Functions
Are there any function you think is useless?	Screenshot	No	"Vegetarian Restaurant" button
Could you give some advice for improvement?	Model Adaptation	Model Adaptation	No

Table 12. Feedbacks

Read Me:

1. Unzip the "Vegetarian Radar.zip"
2. Open the project with Android Studio
3. Set up an Android simulator or connect with Android mobile phones in Developer Mode with 1080p resolution and run