



UNIVERSITY OF
LIVERPOOL

[COMP208] Portfolio

EasyTrip

Team 22

Jamie Garvin

Henan Liu

Weihua Gao

Hao Wu

Xiao Ma

Kuan Lu

Content

1. Design Document
2. Project Report
3. Testing Document
4. Screen Shot



UNIVERSITY OF
LIVERPOOL

[COMP208] Design Document

EasyTrip

Team 22

Jamie Garvin

Henan Liu

Weihua Gao

Hao Wu

Xiao Ma

Kuan Lu

Table of Contents

1. Summary of Proposal	3
1.1 Background	3
1.2 Project Objectives.....	3
1.3 Improvement of Original Proposal	3
1.4 Research and Analysis	3
2. Design Plan	4
2.1 System Component	4
2.1.1 Class Diagram	4
2.1.2 Use-Case Diagram	5
2.1.3 Sequence Diagram	9
2.1.4 System Boundary Diagram	错误!未定义书签。
2.2 Data Structure	10
2.3 Algorithm	10
2.3.1 Description of the Problem	10
2.3.2 Background	11
2.3.3 Description of Our Algorithm	11
2.3.4 Algorithm Analysis	13
2.4 Design of Intended Interface	14
2.5 Evaluation of the System	17
2.5.1 Evaluation design	17
3. Review against Plan	28
4. Database design	29
4.1 Description	29
4.2 Detailed implementation	29
4.3 Data Dictionary.....	31
4.4 Global Logical Data Model.....	34
4.5 Physical Data Table	35
5. Business Rules	377
Bibliography	377
Team 22 Signatures	388

1. Summary of Proposal

1.1 Background

When you travel to a city where there are multiple attractions you want to visit, visitors are often confused and do not know how to arrange the order of the attractions is the best (saving time, short walking distance). Based on this situation, our project provides visitors with an easy trip app to help them plan travel routes between multiple attractions, making it easy for visitors to go through all the attractions.

1.2 Project Objectives

The overall goal is designing an app that can be used to help users plan routes between multiple points. The team need to find an algorithm that can solve multi-point shortest path planning, similar to the problem of traveling salesman. The team need to implement the basic operations of the app on Android studio, such as: login, displaying of map and list interface, connection of Google map etc. The team need to have our own database to store some information needed for the app, such as user information.

1.3 Improvement of Original Proposal

In the algorithm, the team added a function to ask the user whether to return to the starting point after the end of the trip. If the user chooses yes, the algorithm will calculate a route through the user-selected attractions and return to the starting point. Besides, because the team finds that the distance from attraction A to B may be different from the distance from B to A. So the team deals with two more NP problem (open-cycle) ATSP and revise our algorithm program. Our app doesn't ask the user how long they want to stay at the attraction because the user will not know in advance, as there are also unexpected delays (queues, etc). The application also removed the weather information because that can be displayed in other applications and the application focuses on path planning. The popular attractions of different cities are recommended to the user in list view in our app. According to the feedback from reviewer, the usability testing and specific standard of username and password are included in this testing documentation.

1.4 Research and Analysis

The research and analysis that our team have carried out so far varies between different aspects of mobile development. When designing this application, we must consider some basic building blocks for a successful application. Firstly, we realised we need to consider the industry standard method of creating an Android application. Therefore, we have researched into Android Studio development, and studied the kind of framework that we will have to deal with across implementation. One of the first things we had to think about was the graphical side of implementation, and for this we considered at first Java Swing, but decided this was not a viable option, thus we moved onto the Layout Editor provided by Android Studio which uses a mixture of XML code and Java code to bring together the user interface. The team members assigned to user interface design were required to research various layouts, widget types, and Java code that makes the application dynamic.

Secondly, other teammates needed to research how to interact with Google API regarding the Maps functionalities required by the application. This is a vital part of the

application so ensuring this is correctly implemented is very important. We also had to research and analyse how webscraping could be used effectively and what the most efficient way of implementing it is, for the attraction information.

Lastly, we had to decide what type of databases we'll be using is, so we had to research many important factors to get us a very efficient system in place. These strong foundations have been researched to give us a good start in development.

2. Design Plan

2.1 System Component

2.1.1 Class Diagram

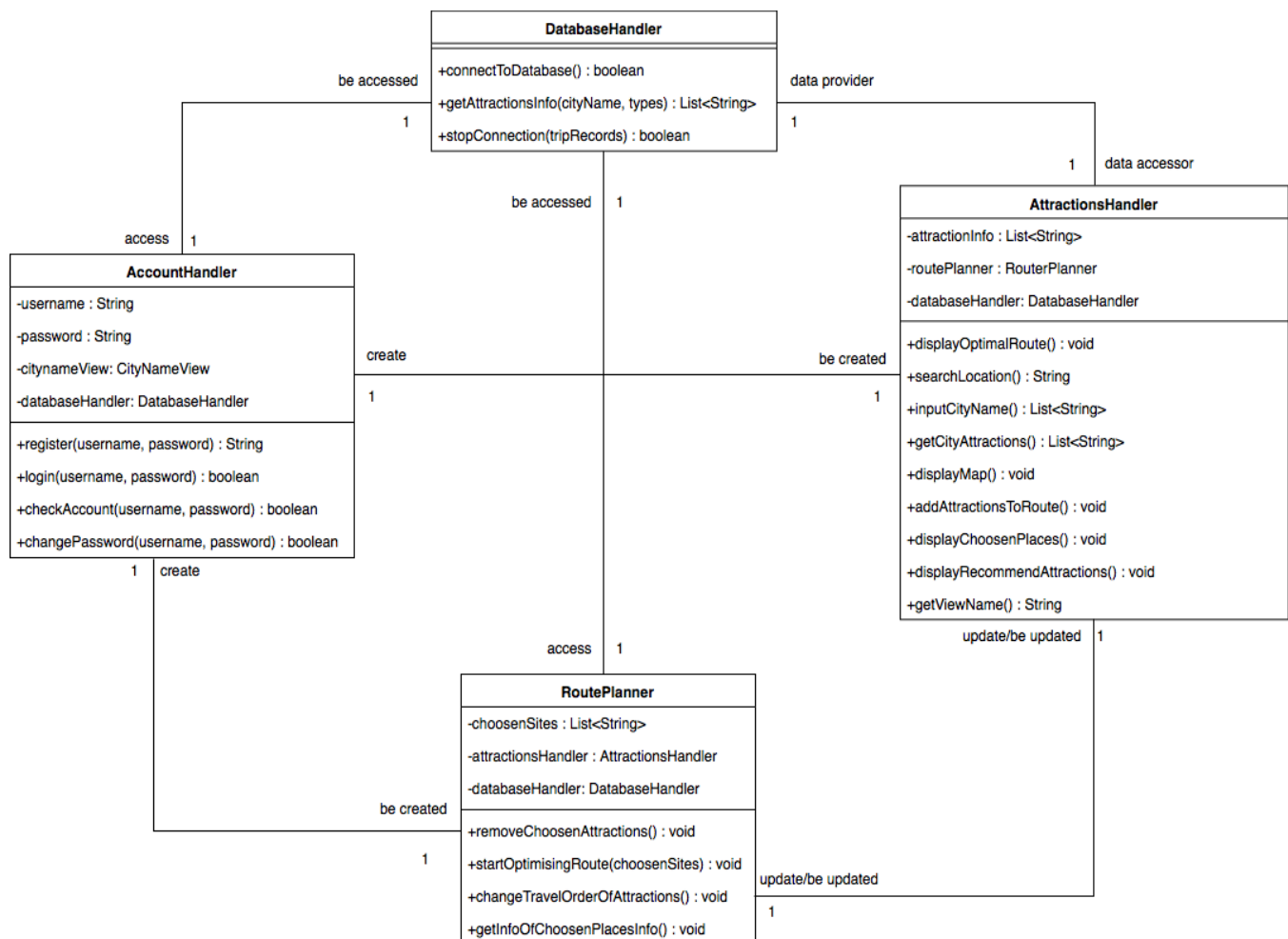


Figure.1 Class Diagram

There are four anticipated components in this system, including account handler, database handler, attractions handler and route planer, among which database handler and account

handler will work as soon as the application starts running. As is shown in class diagram, 'AccountHandler' handles login function, creates and stores account information to database with the help of 'DatabaseHandler'. After choosing city name, 'AccountHandler' will initialise both 'AttractionsHandler' and 'RoutePlanner', and handover work to 'AttractionsHandler' by passing a specific city name inputted by the user. Once receiving city name, 'AttractionsHandler' send a request to 'DatabaseHandler' to get attractions information of the chosen city. Then, attractions information will be displayed on both list view and map view, and user can add recommended attractions shown in both views to plan list. Additionally, user can also search and add a specific place with Google search engine. All attractions in plan list are updated and maintained in 'RoutePlanner' where the order of attractions implies the route on the map; Therefore, 'RoutePlanner' should update information stored in 'AttractionsHandler', especially that of map view which visually display the route. Accordingly, 'AttractionsHandler' is required to update plan list through interactions with 'RoutePlanner'.

2.1.2 Use-Case Diagram

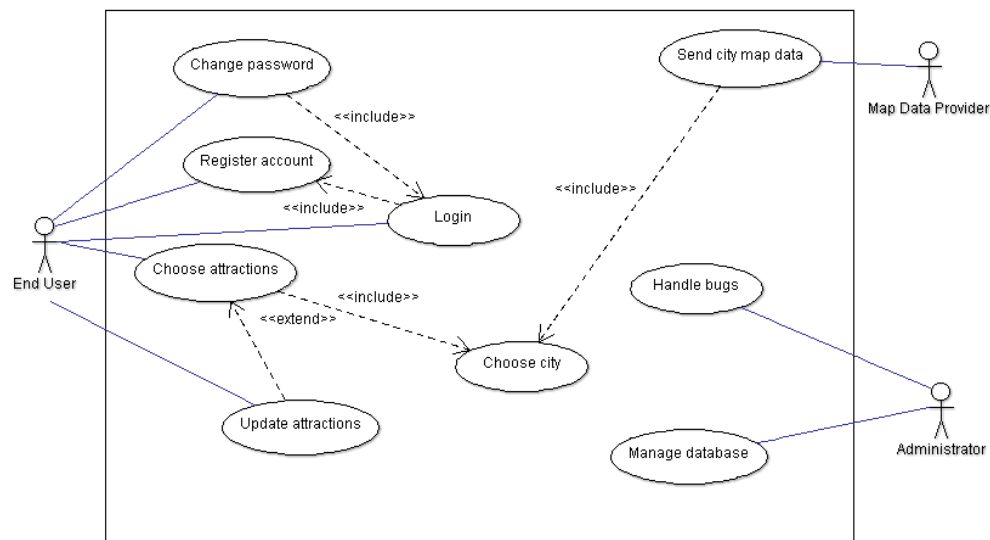


Figure 2. Use-Case Diagram

[COMP208] Design Document

ID	UC1
Name	Login
Description	Login to the system with a username and corresponding password.
Pre-conditions	Application running properly with no user account stored locally and network connection stable.
Event flow	Log in with username and password.
Includes	UC3
Extensions	
Triggers	Application opened.
Post-conditions	User has logged in.

ID	UC2
Name	Choose city.
Description	User input city name for travelling.
Pre-conditions	Application running properly with user logged in and network.
Event flow	Input city into search bar, select city from drop down suggestions.
Includes	UC1
Extensions	
Triggers	City name confirmed.
Post-conditions	Map view displayed.

ID	UC3
Name	Register account.
Description	The user registers an account onto the system, this can be used to login later.
Pre-conditions	Application running properly, and a stable network connection.
Event flow	Open application Click "register"
Includes	
Extensions	
Triggers	Account to be created
Post-conditions	User can now login, application opens login interface.

[COMP208] Design Document

ID	UC4
Name	Choose attractions
Description	User can choose attractions to travel in the chosen city, and the route between these attractions will be generated automatically through algorithms, but the user is able to modify the order of attractions they want to visit.
Pre-conditions	Application logged in and a stable connection available.
Event flow	Choose which attractions to view. View and add attractions to trip list. Delete attractions if user wish. Click "optimise" button. One appropriate route based on chosen attractions generated.
Includes	UC2
Extensions	UC7
Triggers	Attractions added to trip list.
Post-conditions	Route for chosen attractions generated.

ID	UC5
Name	Update attractions
Description	User can modify the route at any time during the trip, including reordering attractions, adding and deleting attractions. Algorithms for generating route will still work, but the scope will change to prioritise the changes made by the user.
Pre-conditions	Application running properly with user logged in, network connection stable, city name chosen and route for attractions generated.
Event flow	Reorder, add or delete attractions Click "optimise" button
Includes	
Extensions	
Triggers	
Post-conditions	User change the order of attractions in the generated route, add or delete some attractions.

ID	UC6
Name	Send city map data
Description	Map data provider supplies map information for the chosen city.

[COMP208] Design Document

Pre-conditions	Application running properly with user logged in, network connection stable, city name chosen.
Event flow	Send required map data for chosen city. Data is processed and displayed.
Includes	UC2
Extensions	
Triggers	
Post-conditions	Map information for the chosen city displayed.

ID	UC7
Name	Manage database
Description	Administrator can manage data stored in database of this application.
Pre-conditions	Database of this application running properly.
Event flow	Change the structure of database if needed. View or modify the data stored in database.
Includes	
Extensions	
Triggers	Administrator has opened the database of this application.
Post-conditions	Database modified.

ID	UC8
Name	Change password
Description	Change password for user account.
Pre-conditions	Application running properly with user account stored locally and network connection stable.
Event flow	Change password for username. Switch to login page with username and new password filled.
Includes	UC1
Extensions	
Triggers	Click "change password" button
Post-conditions	Password changed.

2.1.3 Sequence Diagram

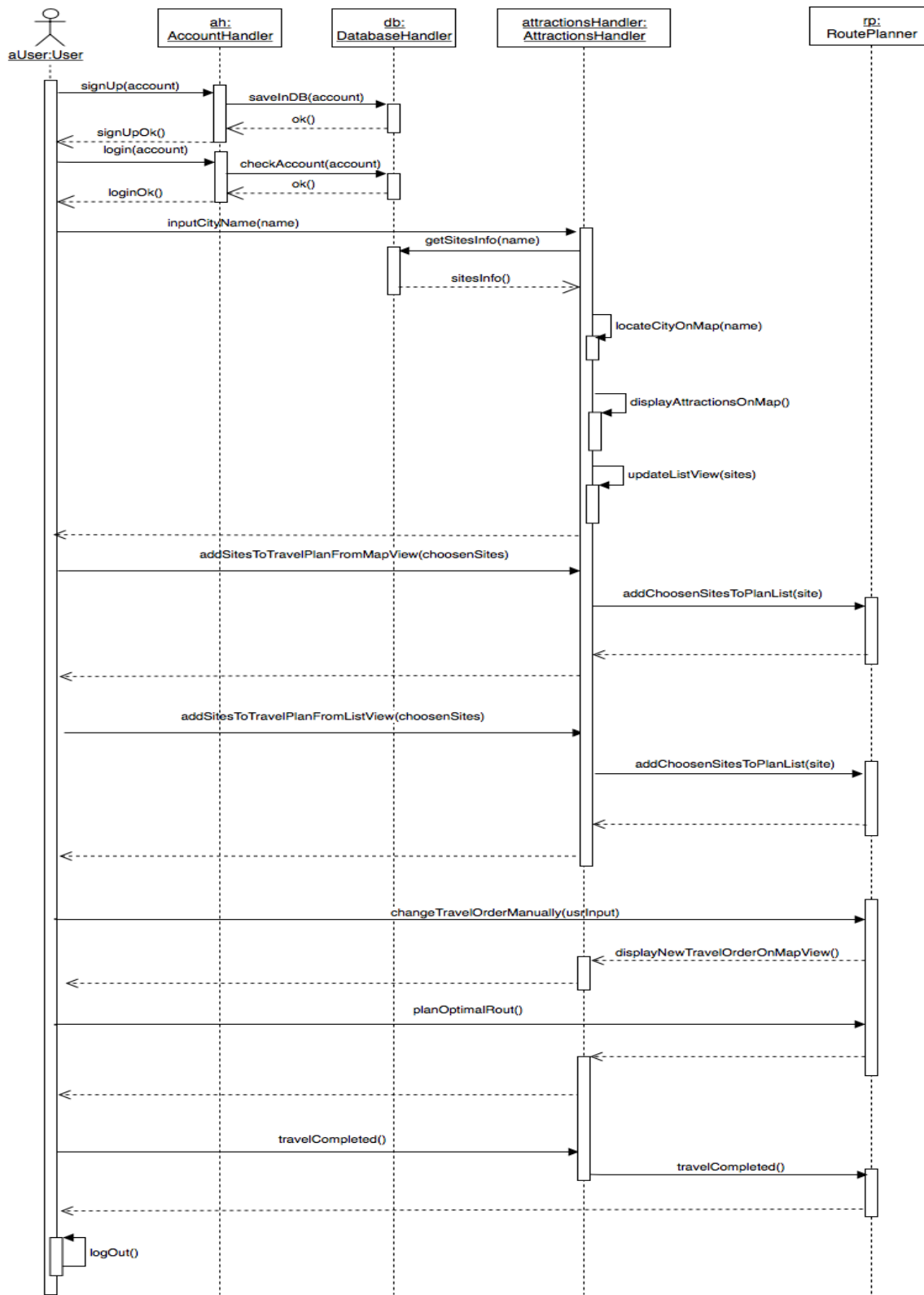


Figure 3. Sequence Diagram

2.1.4 System Boundary Diagram

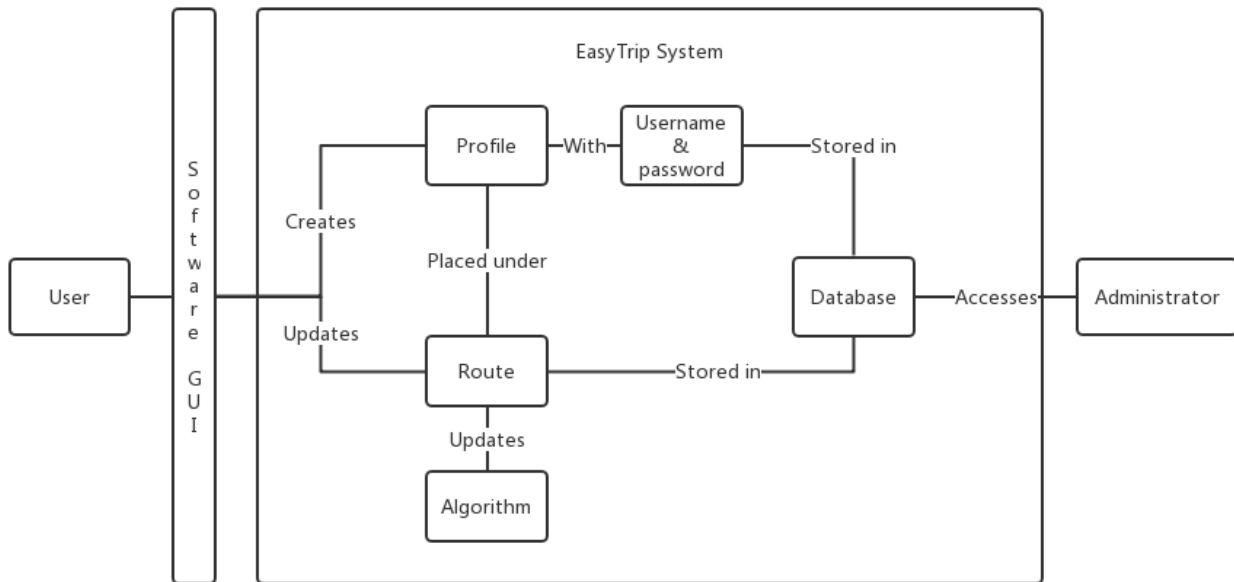


Figure 4. System boundary diagram

2.2 Data Structure

In this system the team mainly use HashMap, one-dimensional array and two-dimensional array to achieve our algorithm. The team uses a `HashMap<Integer,String>` to represent attraction that has not been added to the sequence of path. Each time the attraction has been added to sequence, this attraction will be deleted from HashMap. One-dimensional array is mainly for record attraction order in path. In addition, two-dimensional array is to store two chosen edges for each attraction and lower bound.

2.3 Algorithm

2.3.1 Description of the Problem

This project requires an algorithm to plan the path between different attractions. There are three ways for users to generate routes.

First of all, the user should choose transport method of the whole journey driving or walking. The first, the user can select the attraction and drag the attraction in the plan view to select the order to go to the attraction. The map will be connected attractions according to the order selected by the user. Second user can choose attractions (the attraction is not repeated) and choose to return to the starting point in the end. Third user can choose the attractions (the attraction is not repeated) and choose not to return to the starting point. Allowing user to return to the starting point is to ensure that perhaps the user is departing from a hotel, so they will want to return to the hotel at the end of the journey. The algorithm for the second and the third is almost the same. The algorithm will calculate a shortest path from the starting point and go through the other attractions only once. The shortest path is based on the distance between the attractions. Therefore the

second is a typical TSP/ATSP problem and the third is an open-cycle TSP/ATSP problem. The difference between (open-cycle) TSP and (open-cycle) ATSP problem is that for (open-cycle) TSP the distance from attraction A to B is same as the distance from B to A but for (open-cycle) ATSP it is different. In algorithm, the only difference is that the second does not use a dummy point, and the third uses a dummy point.

Some definition of our problem:

TSP (travelling salesman problem)

Given the distances between each pair of cities, and the distance from attraction A to B is same as the distance from B to A. To find the shortest possible route that visits each city once and returns to the origin city

Open cycle TSP problem

Given the distances between each pair of cities, and the distance from attraction A to B is same as the distance from B to A. To find the shortest possible route that visits each city once but not returns to the origin city

ATSP (Asymmetric travelling salesman problem)

Given the distances between each pair of cities, and the distance from attraction A to B is different from the distance from B to A. To find the shortest possible route that visits each city once and returns to the origin city

Open cycle ATSP problem

Given the distances between each pair of cities, and the distance from attraction A to B is different from the distance from B to A. To find the shortest possible route that visits each city once and returns to the origin city

2.3.2 Background

There are lots of algorithms in this area, but some algorithms have fatal shortcomings. For example, greedy algorithms only obtain local optimal result and ant colony algorithm has high time complexity and space complexity. More detailed analysis is in the following. Greedy algorithm: For example, the user gives 4 attractions A, B, C, D, and selects A as the starting point, then the algorithm needs to look for the closest point to A in BCD. If B is closest to A, then the algorithm does not need to consider the distance between A and other points, next, it looks for the closest point to B, C is closest to B. Then look for the point closest to C and get D. Finally, the team can get ABCD. Therefore, the shortest path is ABCD. This greedy algorithm takes a local optimal solution each time, but it is not a global optimal result, that is, the shortest path is not necessarily obtained.

Another algorithm for solving such problems is called ant colony algorithm, which imitates ant addressing to get the shortest path and solves the traveling salesman problem. However, there is too much iteration in this method and it needs to search for all possible paths, so the time complexity and space complexity are relatively higher than the team wants. Therefore, it is not a best choice.

2.3.3 Description of Our Algorithm

We use branch and bound algorithm to solve open-cycle TSP problem, TSP problem, ATSP problem and open-cycle ATSP problem. For open-cycle TSP/ATSP problem (not go back to starting point), the team needs to add dummy vertex v . Add edges of weight 0 from all other vertices to v and v to all other vertices. Then any Hamiltonian cycle in the resulting graph will be the path which starts at fixed starting point, go through all the attractions once and end at any attraction(except starting vertex). The team use distance matrix to represent graph G . $C(i, j)$ in a distance matrix M represents the cost(distance) between attraction i to attraction j .

Therefore, the team have a distance matrix M like

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \ddots & & & & \\ 0 & & \ddots & & & \\ 0 & & & \ddots & & \\ \vdots & & & & \ddots & \\ 0 & & & & & \ddots \end{bmatrix}.$$

For TSP/ATSP problem (go back to starting point), the only difference is the algorithm does not need dummy point. Therefore, the only difference is distance matrix. In TSP problem, the distance matrix for n

attractions are like M

$$\begin{bmatrix} d_{11} & d_{12} & \dots & \dots & \dots & d_{1n} \\ d_{12} & \ddots & & & & \\ \vdots & & \ddots & & & \\ \vdots & & & \ddots & & \\ \vdots & & & & \ddots & \\ d_{1n} & & & & \dots & d_{nn} \end{bmatrix}.$$

The pseudocode for this algorithm

//attractions: a map<Integer,String> to represent attraction that has not been added to the sequence.

//sequence[]: one-dimensional array [0..sizeTotal-1] to record attraction order in path

//lb[]: two-dimensional array to store lb, lb[i][0] for value, lb[i][1] for whether calculate this

//result: a map<Integer,ArrayList<Integer>> to store the final result

//initialize the start of the sequence

if user choose not return to starting point

 sequence[0]=dummy point

 sequence[1]=starting vertex

 attractions.remove(dummy point)

 attractions.remove(starting vertex)

else if user choose not to return to starting point

 sequence[0]= starting vertex

 attractions.remove(starting vertex)

//find nearest attractions of each attraction

checkMinLastMin(int i)

 find two nearest attractions from each attraction

 stores result in mark_row[][]

 find two nearest attractions to each attraction

 stores result in mark_column[][]

//start a recurrence

dealWithNextLayer(attractions,sequence)

 repeat

 add one remaining attraction to sequence

 calculate lower bound and store it in lb[].

 remove this attraction

 until all the remaining attraction have been added once

find the smallest lower bound in lb[][] and its corresponding sequence of attractions call it sequenceNow

```

if the number of remaining attraction is larger than 1
    remove this smallest lower bound in lb[][]
    call dealWithNextLayer(attractions,sequenceNow)
if the number of remaining attraction equal to 1
    calculate its lower bound
    if result is empty
        add lower bound and sequence to the result
    else if this lower bound is smaller than the lower bound in result
        remove the item in result
        add lower bound and sequence to the result

if there is some lower bound in lb[][] that has not been processed yet
    find its corresponding sequence of attractions call it sequenceNow
    call dealWithNextLayer(attraction,sequenceNow)

output sequence in result
    
```

2.3.4 Algorithm Analysis

Mathematical analysis: The idea of branch and bound is almost like force search. The team tries to list all the possible result and calculate their lower bound. The bounding part of the algorithm stops us from exploring a branch only if it is proven to not consist of the optimal solution. Since any potential global optimal will not be discarded, the team finally finds one optimal result if it exists.

The lower bound in (open-cycle) TSP/ATSP problem is $\sum(\text{distance of two possible nearest attractions of } v(\text{to and from}) \text{ in } G)/2$. Because each city has two adjacent edges, one in and one out. Then, if you add the two smallest elements of each row in the matrix divided by 2 (without loss of generality, you can assume that all distance weights in the graph are integers), and then round up the result. The result is a reasonable lower bound because the most ideal situation is that the in edges of the vertex and out edge of vertex are assumed to be the two smallest. In this coincidence, it is exactly lower bound lb. As a result, it is a reasonable Lower bound b2. Besides, after deciding some edges like the sequence of attraction is ab, the algorithm also decides that there's no ba in the path. So, for vertex b, one edge is ba, another is still the smallest. Therefore, the result of lb is still a reasonable lower bound.

Experiential Analysis: For example, the distance matrix is

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 2 & 7 \\ 0 & 3 & 0 & 5 & 4 \\ 0 & 2 & 5 & 0 & 6 \\ 0 & 3 & 4 & 6 & 0 \end{bmatrix}. \text{ The row and}$$

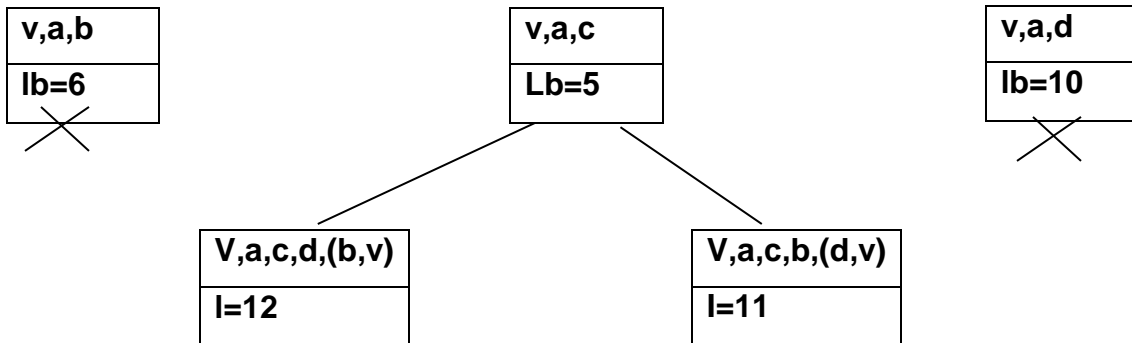
column is dummy vertex v, attraction a, b, c, d accordingly.
For brute search

abcd	14
abdc	13
acbd	11
acdb	12
adbc	16

adcb	18
------	----

So adcb is optimal result.

For branch and bound, sequence start at va.



As a result, experimental result is same as real result.

2.4 Design of Intended Interface

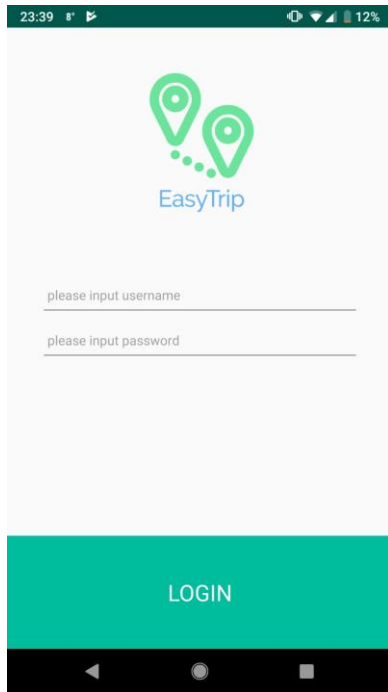


Figure5

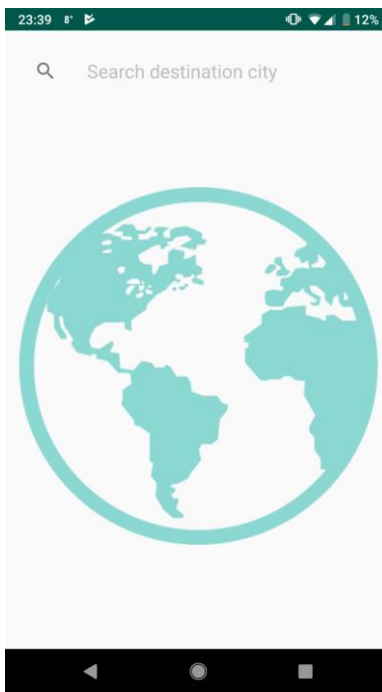


Figure6

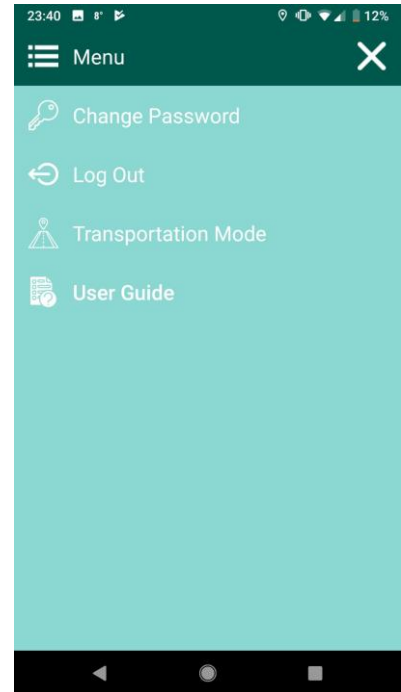


Figure7

5. When the user opens the app, the first is the login interface, the user can enter the username and password to log in to the system by pressing the large 'login' button at the bottom of the interface.

6. After the user entered a valid username and the correct password, the following interface is shown, and the user enters the city name on this interface. When the user starts typing, a dropdown search suggestion box appears.

7. User personal information interface, the function includes the following three items; 1, the user can change their password. 2, the user can logout. 3, the user can change their

transportation mode. 4, the user can logout.

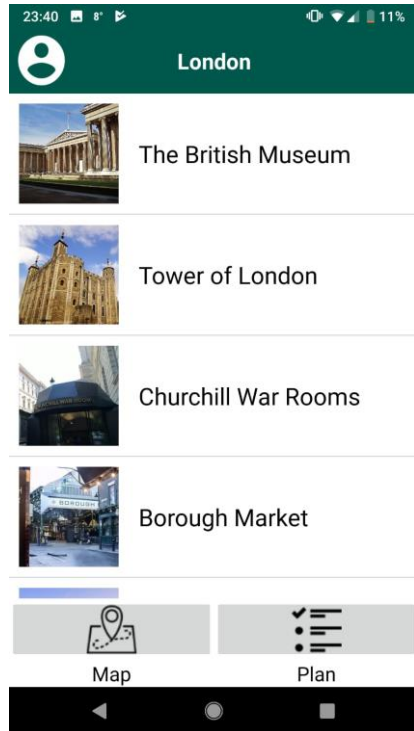


Figure8

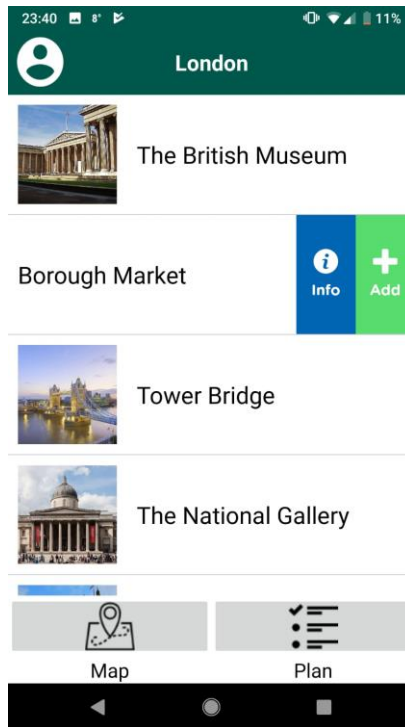


Figure9

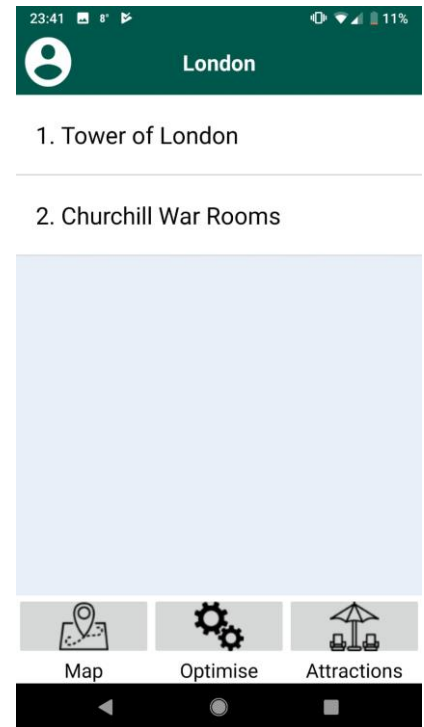


Figure10

8. List view: This shows the city's most famous attractions; users can view the introduction of the attraction and add the attraction to their plan list. The user can also return to the previous interface and display options.

9. When user adds an attraction to the plan list, there is a plus symbol on the bottom left. This can be pressed to add location to the plan.

10. This interface shows the plan list after the selection of the user. The first attraction is to be recommended as the start point.

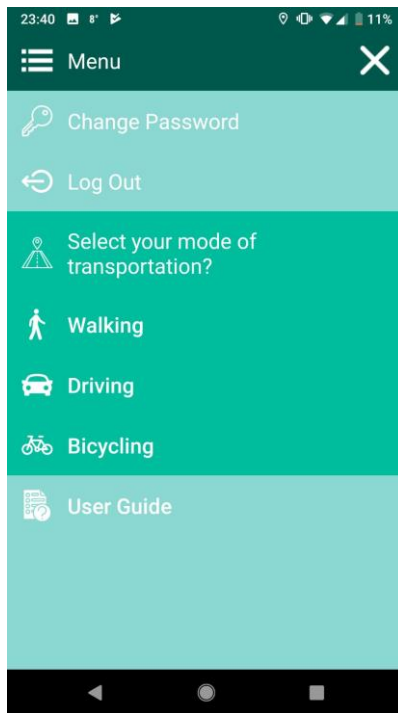


Figure11

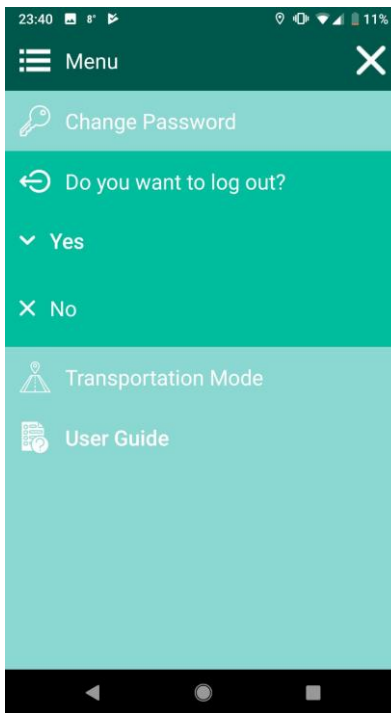


Figure12



Figure13

11-12. These interfaces includes the previously shown options that the user can select.

13-14. This is a map view interface, the top ten attractions are also shown in the map view with an orange symbol, and user can also add these attractions into plan list. When the attraction is in the plan list, the symbol will become green. Before user finish selecting, the route will appear in the map directly (this route is not the best way to visit).

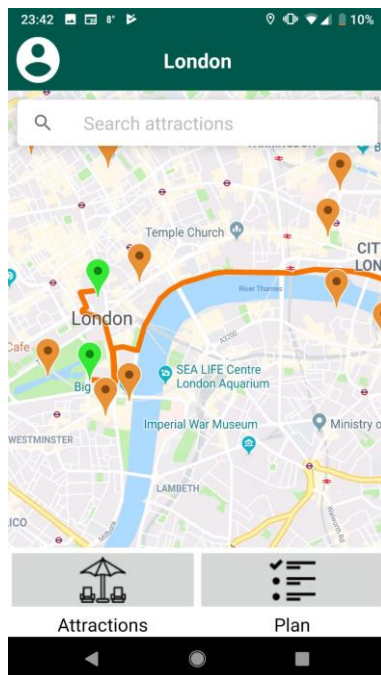


Figure14

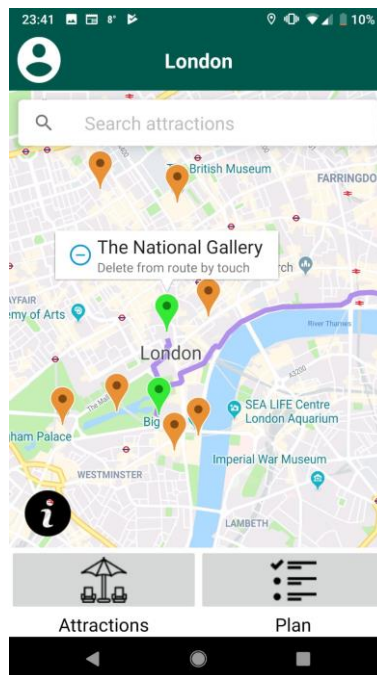


Figure15

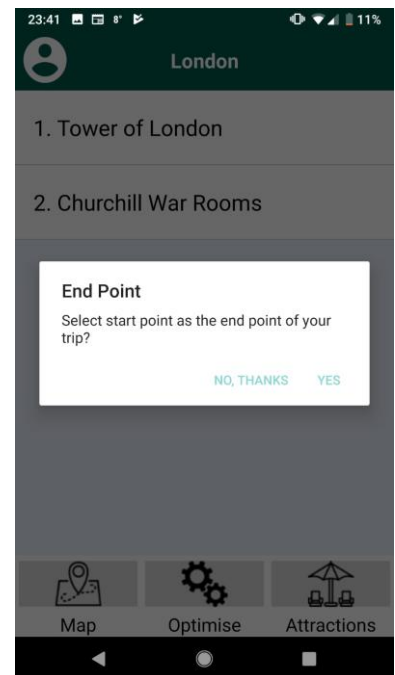


Figure16

15-16. When user press the 'optimise', it'll ask the user if they want to make the trip a loop, so they end where they began.

17-18. When the user has selected all their options, the algorithm will give a best order of these attractions and user is also able to change the order of the them. Map view will be update at the same time.

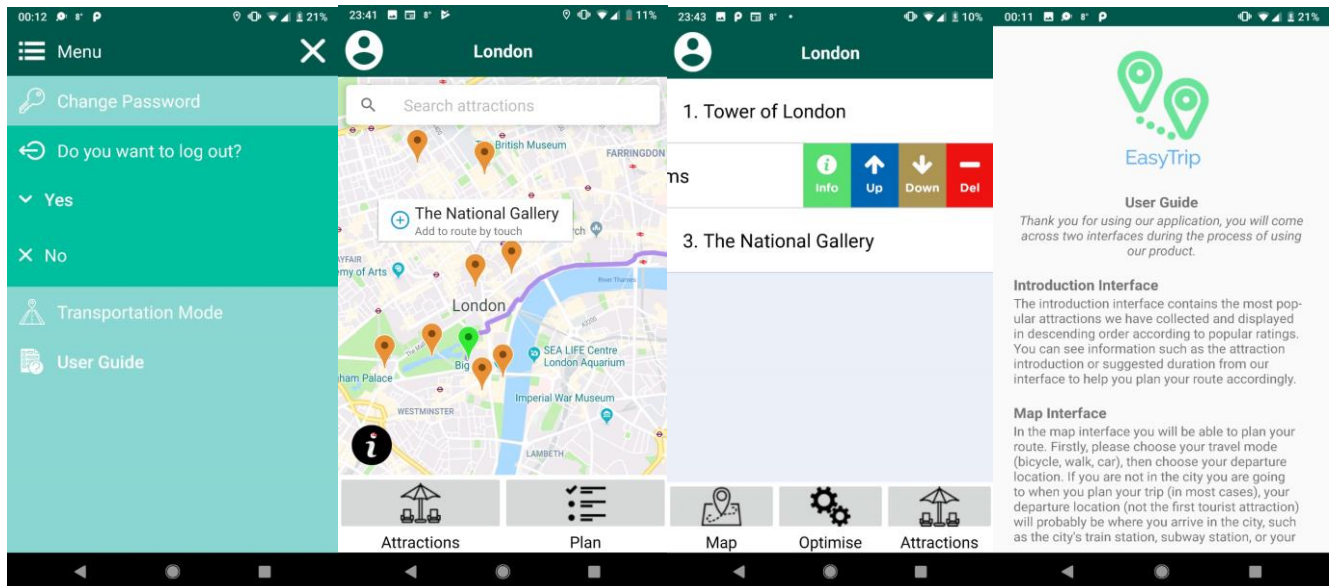


Figure.17

Figure.18

Figure.19

Figure.20

19. This interface shows the manipulation options when dealing with the plan locations. You're able to move priority of a location, view information, and delete from plan.

20. This interface shows the user a guide to the application and what each view represents. This is important in case some people do not understand something.

2.5 Evaluation of the System

2.5.1 Evaluation design

The team plan on evaluating the application through using the previously talked about testing methods and criteria for the functional side of making sure the application is fit for purpose. After each individual function is checked using **black box testing**, the team will evaluate the entire system using integration testing and smoke testing, which will validate whether the build of the application will work as expected for general use, as individual functions working fine on its own may not mean they'll work well together.

For the interface tests, we'll evaluate using **GUI testing** to ensure the user will see what is intended by the application on multiple devices as a seal of approval for the functionality of the interface. As mentioned, the response times will be evaluated using testing and will be compared to what is acceptable/satisfactory when considering load times of an application, while taking in mind reasonable measures that are taken in development to make it as fast as possible. All members will be taking evaluation on their work during development to reduce the amount of tinkering after functionality development is complete. Once the software is finished, the team will employ several usability inspection methods [1], such as formal usability inspections [1].

Smoke testing will be used after each part is implemented by the tester to check basic functionality is working correctly rather than missing a vital piece of logic. This is a very basic

and informal method of testing and will be used to ensure the function can be used correctly. At these stages, the team won't test every scenario, but to make sure the implementation is valid. Deeper testing will be employed later in development.

After the team (the developers) have finished testing using black-box, the developers will move onto **white-box testing** to ensure the team executes each path in the program. These developers will go through the code finding the possible defects which cannot be located via black-box testing, such as duplicate code and such.

Integration testing will be used within development to make sure different functionalities work together as expected while implementing various solutions. This is important when the team test the interface against the core components required in the application. Ensuring the software is all integrated properly will mitigate complications happening later in the development cycle.

2.5.2 Test Plan

1. Introduction

According to the feedback from reviewer, the **usability testing** and **specific standard** of username and password are included in this testing documentation.

In the following content, the team have test plans laid out which the tester will strictly follow to make sure the main functions work correctly. In the actual tests, flaws may be exposed which are preventing a system which works correctly, and then fixed.

The following testing are split into three parts, the login system, the map system and response times of the system. They include description of the test data, experiment design and result analysis

2. Login system

Our general basis for testing the login system will be split into validating several key functionalities which are used by the login system. This includes testing the username input and password input namely. The system will need to check both conditions to be true (both valid inputs) to proceed, any incorrect attempts must not let the user through. The team will also inspect the change password and log out functionalities to make sure they achieve the desired effect, and without compromising security. The team aims to use black-box testing to ensure these works correctly.

A: Login

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
A1.	Test login function with the	The dialog pop up to inform	As expected

[COMP208] Design Document

	test cases provided in List I.	tester login failed	
A2.	Enter the correct username and password, then push "Login" button	Tester log in system successfully and GUI turns to city selection	As expected
A3.	Push "Return" button and push "Sign Up" button	GUI turns to sign up part	As expected
A4.	Test sign-up function with the test cases provided in List II.	The dialog pop up to inform tester sign up failed	As expected
A5.	Input unused username and password with correct format	The dialog pop up to inform tester sign up successfully	As expected
A6.	Push "Return" button and input username and password registered before, then push "Login" button	Tester log in system successfully and GUI turns to city selection	As expected
A7.	Push "Menu" button then push "Account" button	The GUI turn to account interface	As expected
A8.	Push "Logout" button	The GUI back to login interface, and the username box and password box are both empty	As expected
A9.	Login again and enter city selection interface	/	As expected
A10.	Push "Menu" button then push "Account" button	The GUI turn to account interface	As expected
A11.	Push "Change Password"	The GUI turn to interface to change password	As expected
A12.	Input new password and push "Confirm" button	The dialog pop up to inform tester if the change is valid	As expected

◆ **Standard of username and password**

Standard of Username:

- Username must contain no space or tab
- Username should use a minimum of 5 characters
- Username must include at least three of the four following types of characters:
 - English upper-case letters (A-Z).
 - English lower-case letters (a-z).
 - Numbers (0 through 9).
 - Underline ("_").

Standard of Password:

- Password must contain no space or tab
- Password should use a minimum of 10 characters
- Password must include at least three of the four following types of characters:
 - English upper-case letters (A-Z).

[COMP208] Design Document

- English lower-case letters (a-z).
- Numbers (0 through 9).
- Special characters and punctuation symbols (E.g. `_`, `-`, `+`, `=`, `!`, `@`).

◆ List I. Test case of login

Tester selects the combination of username and password from the Table I. and Table II.

1. Correct username and password
Anticipated result: login successfully
2. Incorrect username but correct password
Anticipated result: The dialog pop up to inform tester login failed
3. Incorrect password but correct username
Anticipated result: The dialog pop up to inform tester login failed
4. Incorrect username and password
Anticipated result: The dialog pop up to inform tester login failed

➤ Test data of incorrect password

Assume the correct password is "comp208"

Equivalence Partitions	Input Example
Empty input	<code>""</code>
Over-limit length password	<code>"comp208comp208comp208 comp208"</code> (length over 12)
Case-sensitive	<code>"CoMp208"</code>
Redundant characters	<code>"comp208abcd"</code>
Missing characters	<code>"cm208"</code>
Special characters	<code>"@#\$comp208"</code>

Table I. Test data of incorrect login password

➤ Test data of incorrect username

Assume the correct username is "group22"

Equivalence Partitions	Input Example
Empty input	<code>""</code>
Over-limit length username	<code>"group22group22group22"</code> (length over 12)
Case-sensitive	<code>"gRoup22"</code>
Redundant characters	<code>"group22group22"</code>
Missing characters	<code>"grp22"</code>
Special characters	<code>"@#\$group22"</code>

Table II. Test data of incorrect login username

◆ List II. Test case of sign-up

Tester selects the combination of username and password from the Table III. and Table IV.

1. Unused and correct username and correctly formatted password
Anticipated result: sign up successfully
2. Unused and correct username and incorrectly formatted password
Anticipated result: sign up failed
3. Unused and incorrectly formatted username and correctly formatted password
Anticipated result: sign up failed

4. Unused and incorrectly formatted username and incorrectly formatted password
Anticipated result: sign up failed
5. Used username and correctly formatted password
Anticipated result: sign up failed
6. Used username and incorrectly formatted password
Anticipated result: sign up failed

➤ **Test data of incorrect username**

Assume the only used username in the database is "group22"

Equivalence Partitions	Input Example
Empty input	""
Over-limit length username	"group88abcdefghijkl" (length over 12)
Illegal characters	"group88%&#"

Table III. Test data of sign-up username regarding unused and wrong-formatted ones

➤ **Test data of unused username**

Equivalence Partitions	Input Example
Unused username	"group88"

Table IV. Test data of sign-up username regarding unused and right-formatted ones

◆ **Result Analysis**

For other tests such as testing the login function, we'll do a large batch of test cases and have an anticipated failure quota the team cannot exceed. For example, if the team were testing login authentication and had 100 test cases, the team would want a 100:0 pass-fail ratio of expected outcomes. On the other hand, if the team wanted to test crash likelihood across 100 test cases on various devices each with 5 attempts of making a route (so effectively 500 test cases), the team would want no more than 5% failure.

3. Map function

In this test, the team plans on testing route planning, route optimization, and route saving. These are a vital core functionality of our software so ensuring it's up to specification is top priority. The team will be using black-box testing for this test.

➤ **B: Test Positioning**

Testing description	System version	v 1.00	
	Target function	Positioning	
Step	Operation	Anticipated result	Actual result
B1.	Input invalid city name in Table V	The dialog pop up to inform tester the city is not in database	As expected
B2.	Input valid city name (e.g. Liverpool)	Map of the city is displayed	As expected
B3.	Push "Map View"	Enter Map interface	As expected
B4.	Input customized	Correctly locate input place on	As expected

	start location in search box and select the place	the map and mark it	
B5.	Push “Return” button	The GUI goes back to city selection interface	As expected
B6.	Enter the same city name	Map of the same city is displayed, and the start location is empty, and the mark is removed	As expected
B7.	Enter another city name	Map of the city is displayed	As expected
B8.	Enter customized start location and select the place from search suggestion	Correctly locate input place on the map and mark it as start location	As expected

➤ **Test data of input city name**

Equivalence Partitions	Input Example
Invalid city name	ABC, 123
Out-of-domain* city	Pyongyang

Table V. Test data of invalid city name

Out-of-domain city: In the demo, only several cities (e.g. Liverpool, London and etc.) will be stored in our database. These cities store their attractions information and other details in the database so that our program can invoke them to make the route.

➤ **C: Test Attraction Selection**

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
C1.	Input an invalid or out-of-domain city name	Map of the city is displayed but no recommended attractions	As expected
C2.	Enter valid city name and select city	Map of the city is displayed, and all recommended attractions are marked on map	As expected
C3.	Select travelling method by swiping the menu and click the button	Enter an interface ask user to choose “Map View” or “List View”	As expected
C4.	Push “List View”	The list of recommended	As expected

[COMP208] Design Document

	button	attractions is displayed	
C5.	Swipe out the buttons under the attraction names, and push “Plus” button	The corresponding attraction is added to user’s list and the attraction is removed in recommended attraction list	As expected
C6.	Select random number of attractions	The system should run as usual, and the attractions are shown in the user’s list	As expected
C7.	Select travelling method by swiping the menu and clicking the button	Three methods buttons are displayed	As expected
C8.	Click them one by one	The route on map will change correspondingly	As expected

➤ **D: Route Planning, Route Optimization and Route Saving**

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
D1.	Enter a valid city name and select the city	Map of the city is displayed, and all recommended attractions are marked on map	As expected
D2.	Push “List View” button	The list of top attractions is displayed	As expected
D3.	Select random number of attractions	The system should run as usual, and the attractions are shown in the user’s list	As expected
D4.	Go back to map and search a location in search box	Correctly locate input place on the map and mark it	As expected
D5.	Push the mark	The attraction name pops up with a “Plus” character	As expected
D6.	Push the “Plus” button	The attraction will be added in user’ list	Unexpected outcome: To close the “Plus” button user must firstly tap on a different location

			on the screen. The user cannot tap on the marker again to make it disappear.
D7.	Push the “Attractions” button again	The attraction list should pop up,	As expected
D8.	Change the order of selected attractions and return to “Map View”	The new shortest route is displayed on the map	As expected
D9.	Push the “Plan List” button and delete one or more attractions and return to “Map View”	The new shortest route corresponding to altered list is displayed on the map	Unexpected outcome: The use of “back button” causes parameter loss
D10.	Push the “Plan List” button again and click “Optimization” button	Dialog pop up to ask if user want to choose the start point as the end point of the trip	As expected
D11.	Press “Yes” or “No, thanks”	The order of selected attractions will change to correspond to the new shortest route displayed on the map	As expected

◆ Result analysis

The analysis for this test category will be based off getting a correct/incorrect value on each test across 5 attempts (if one test in each category fails, the test category fails as it is a software fault) and the team shall look at the number of correct tests see how many have failed and how many have passed. Ideally, all tests should be passed, and if any fail they should be resolved after all tests are complete.

4. Response times

We will test the response times primarily through implementing an internal timer to measure the amount of time a function takes to finish. This testing will store times and create the average using an external script (excel, etc.) for calculating the average time across tests.

➤ E: Test Response Time

Testing description	System version	v 1.00	
	Target function	Response times	
Test	Operation	Anticipated result	Actual result

E1.	Time taken for application to open completely	The application should open within an average time of 5000ms (5 seconds) across testing devices.	
E2.	Time taken for the application to generate a route between chosen attractions	The application (with reasonable connection to the internet) should build a suitable route within an average of 3000ms (3 seconds) across testing devices.	
E3.	Time taken to generate list of recommended attractions	The application should gather data and display attractions from a chosen city within an average time of 8500ms (8.5 seconds) across testing devices.	

◆ Result Analysis

The results shall be analysed primarily by taking the average time and comparing them to expected/desired results, when testing performance related experiments concerning login times, route buildings times, etc. Ideally, most processes would want to be achieved within their target times on average, and any which take much longer than norm will be examined to see if it's an application-side error/problem. Given that the team does not have a large number of devices to test on, we've given more liberal targets to account for it.

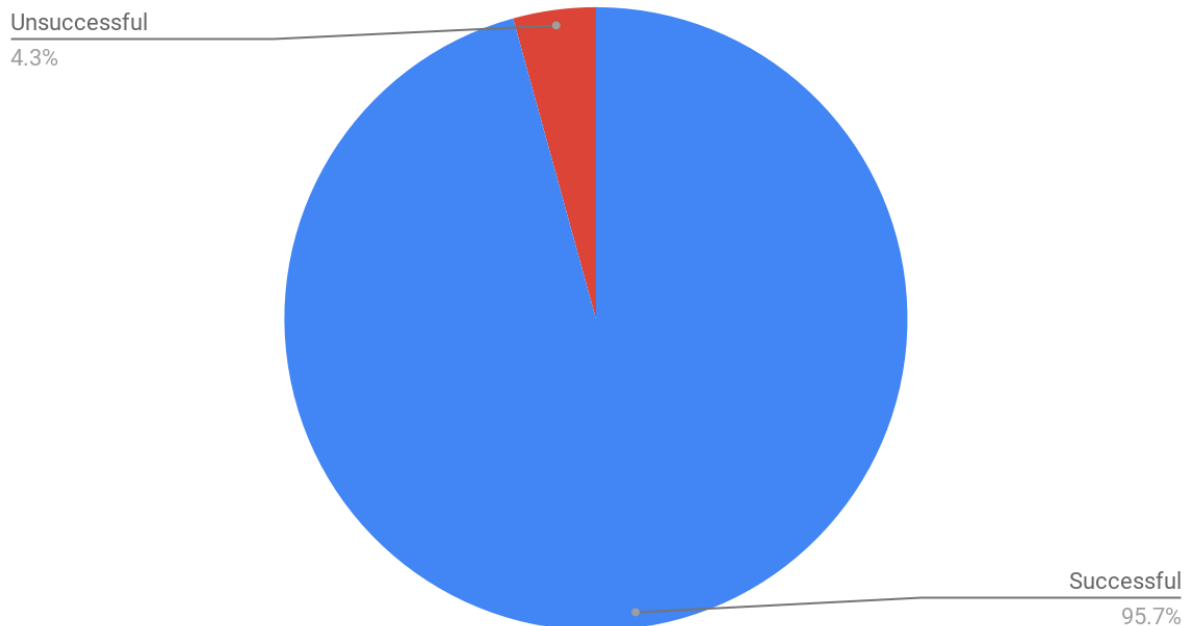
5. Usability testing

For the summative testing[2], the team had a pool of 10 people to try our application. The age range varied, along with technical knowledge. 3 people aged between 16-17, 5 people aged between 18-25, and the remaining 2 people aged between 43-51. Two of the participants study/studied computer science or software development.

We have broken down analysis into several groups;

- Successful actions (of legal attempts).
- Ability to create a route around London, Bath, or Edinburgh without developer help or hints.
- General understandability of user interface.

Success rate of actions performed



The success rate of actions performed was based off each user completing each possible action once on the application. An action is denoted by either clicking an application button (button to enter register screen, or to see list view, etc.) or providing an input type to the application (such as inputting a username, adding route to list, etc.). There is 28 clear actions to be performed on our application, meaning there is a total of 280 actions to analyze (10 people carrying out all 28 actions each).

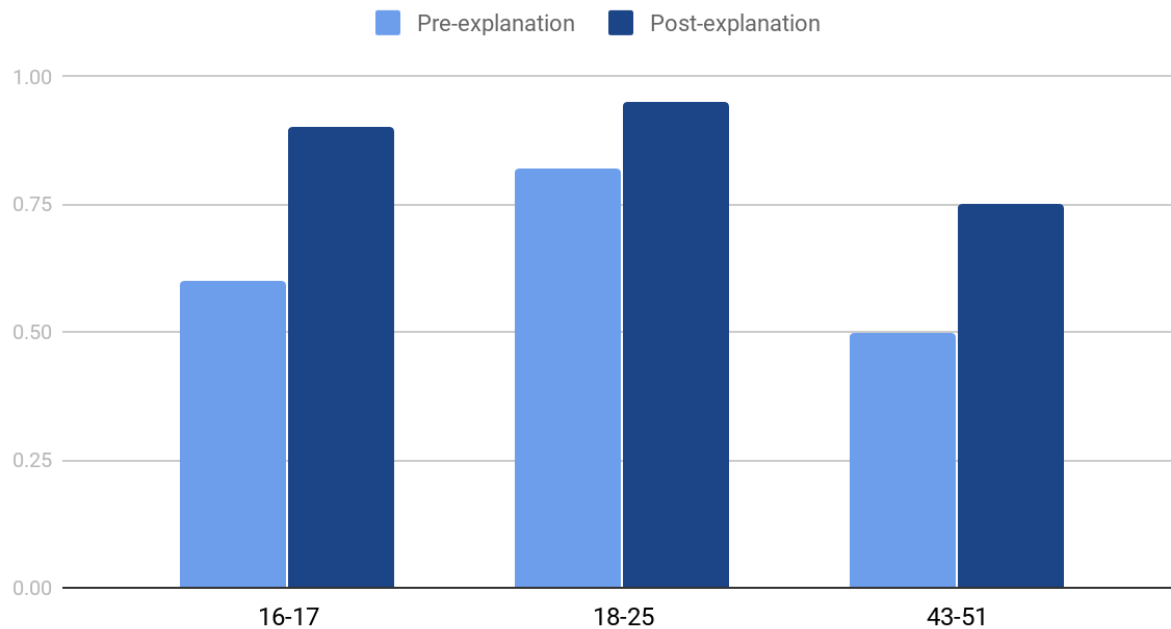
Out of the 280 tests carried out, 95.7% of tests were successful. The only issue from these tests was that returning to the previous screen sometimes caused an activity duplication bug, from the user selecting the Android 'return' button rather than selecting a button from the bottom of the screen (such as the map view or list view). This is a known bug and is down to Google's own code, and in further development the team would aim to create a workaround for this. All the successful attempts were legal actions (using correct registration criteria, rather than not meeting password requirements, for example).

Route creation clarity

When it came to testing the usability of whether it is clear and succinct to understand how to create the route without supervision from a developer, results were more varied to that of the last test. The team found that some users did not understand the ability to add multiple locations onto the same route was available, meaning it took them a little longer to work out the scope of the application. The longest amount of time it took was 34 seconds to figure out how to create a tangible route and optimise it, which was one of the participants in the 43-51 age range, with the second longest being 26 seconds, also belonging to the 43-51. The other participants had an average time of 18 seconds. This data suggests that the older generations may not find this application as straightforward as younger generations. Although, it is worth noting the sample size may not be reflective of a public release. Regardless, using this data the team can deduct that it may seem ideal to introduce a tutorial in future releases. All of these tests were based off the first usage of the end user.

User interface understandability

UI understandability



X Axis - age ranges used during testing

Let Y Axis numbers denote:

All numbers are average of the respective age range.

0 - Absolutely no understanding

0.25 - Disagree that it is understandable

0.50 - Feel that UI can cause confusion

0.75 - Agree that the UI is understandable

1 - Completely understand the meaning of the UI

In this test, the team conducted two types of questioning for the same issue. One test was before the team had explained what each component of the UI achieved/signalled, and the next was after the team had explained what each UI component achieved/signalled. This opens more data to be interpreted and gives the user to provide more opinion. From getting a before/after data set, we're able to analyse if the UI at least makes sense to a user after a short amount of time. For example, if a user does not understand something (while being new to the application) before it being explained, and understands it after the team had explained the general idea, it signifies the problems lies more with their understanding of what the application achieves functionally rather than a problem with the user interface itself.

3. Review against Plan

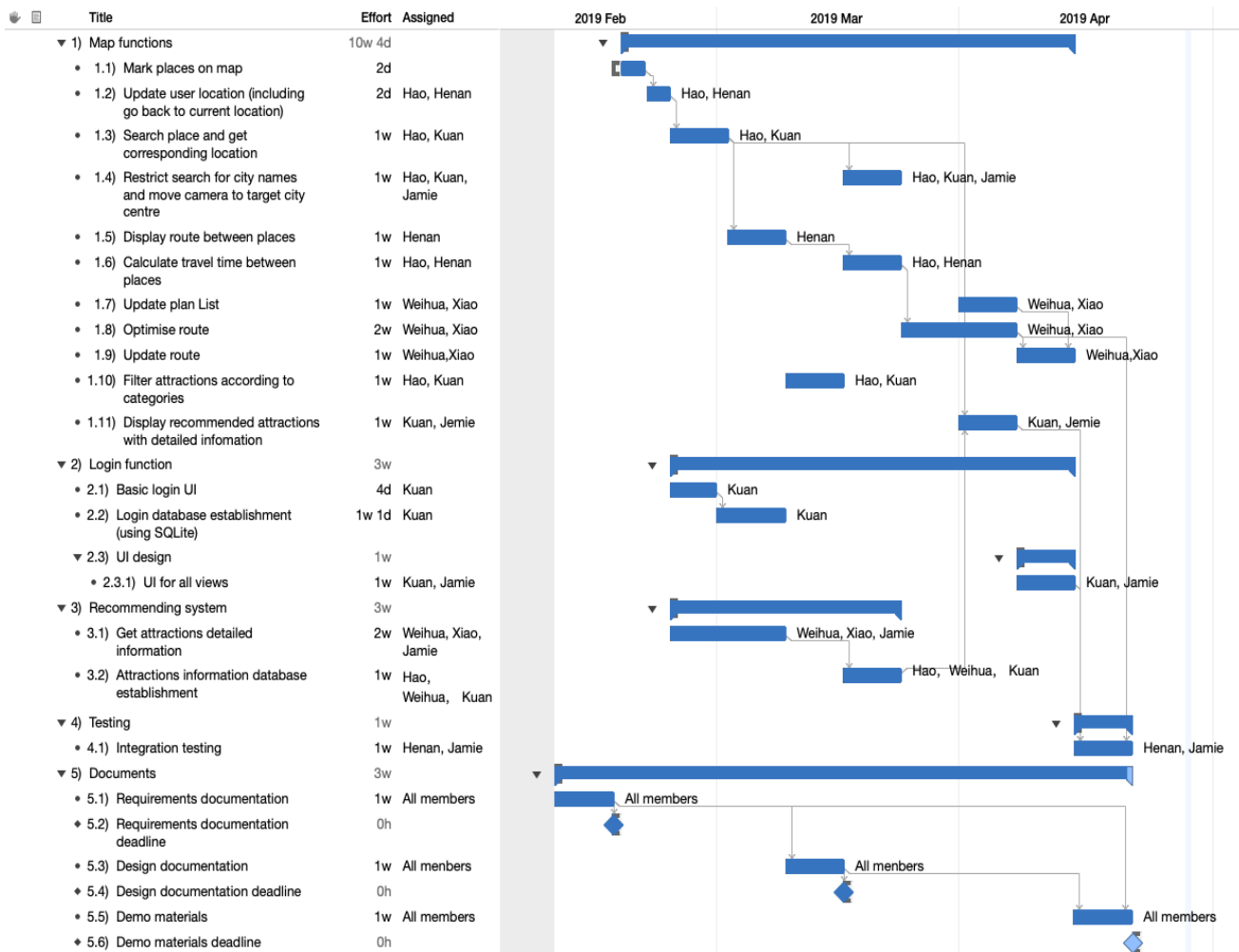


Figure 21: Gantt chart

After learning basic Android programming concepts in Android Studio, login function was implemented with SQLite database engine, where basic User Interface design techniques and SQL statements played important roles. Meanwhile, basic map view for this mobile application had been built on which users' real-time location can be traced and displayed. Additionally, a route between two chosen places can be displayed with the help of Google Directions API and several modes of transportation, including transit, driving, walking or cycling, are proven possible to implement. However, Google Places API for picking a searched place is not stable, solution to this problem may be explored in the following stage. As to information for attractions, which will be used in recommendation system of this application, a web crawler was designed to analyze popularity of worldwide attractions in some famous cities and to gain detailed descriptions of popular attractions. Further works such as calculating travel time between places and updating plan list, are temporarily assigned. Since there are only three persons in charge of map functions, which is the most challenge part, one more person will be assigned to implement some map functions, and all of us will spend more time to familiar

ourselves with features of android mobile developing, which are essential or helpful to our application.

4. Database design

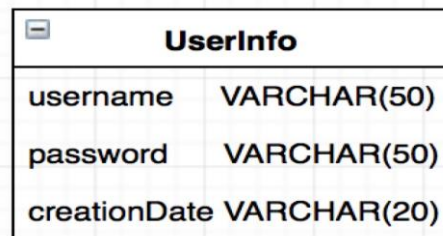
4.1 Description

SQLite is used in our application rather than SQL as SQLite is a light-weight SQL but is faster and smaller than SQL. Such a feature of SQLite is more adaptable for mobile applications. There are two local databases established in our application. The one is named "UserInfomationDB.db" which stores all data related to user account. The other is called "RecommendedPlacesDB.db" which contains all data displayed on the attraction information interface.

4.2 Detailed implementation

UserInfomationDB:

The first database used is called user information database which contains one table named user information. This table has three attributes which are username, password and account creation date. The following image shows the structure of the table. Username is set as the primary key of the table since it is unique.

A screenshot of a database table structure for 'UserInfo'. The table has three columns: 'username' with data type 'VARCHAR(50)', 'password' with data type 'VARCHAR(50)', and 'creationDate' with data type 'VARCHAR(20)'. The table is displayed in a window with a title bar and a close button.

UserInfo	
username	VARCHAR(50)
password	VARCHAR(50)
creationDate	VARCHAR(20)

Figure 22: Account Database

This database will not be created until a user has downloaded the application to his/her mobile devices and registered an account. Each time the user creates an account, program will check the existence of this database.

The application will interact with this database when the following events happen:

When a user registers an account, the program sends a query to the database to check whether the input username has existed in the database.

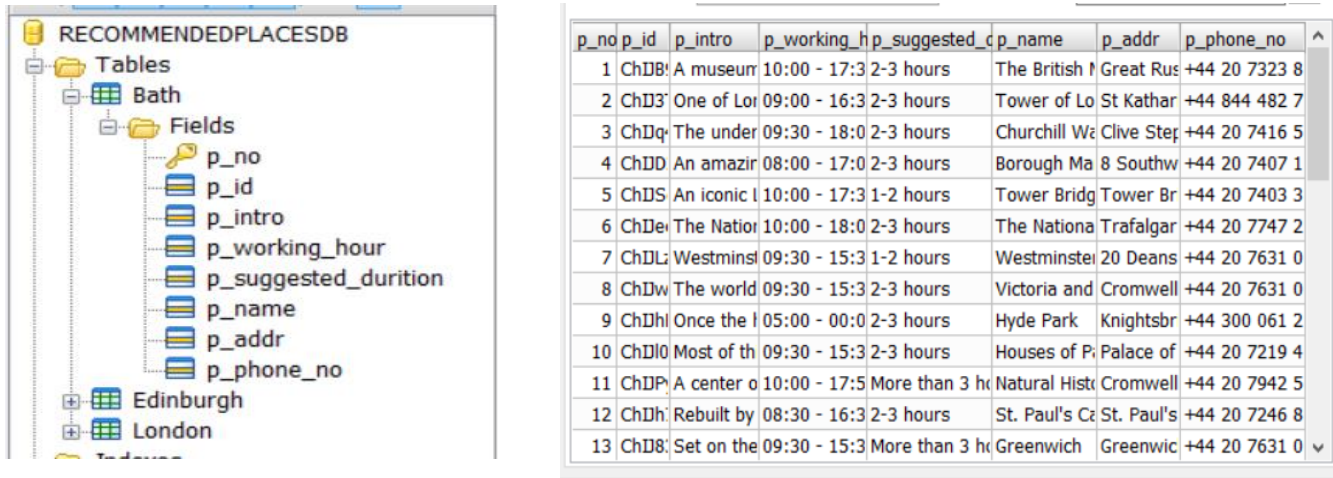
When a user logs in using his/her account, the program sends query to database to check whether the input username and password match the username and password stored in the database.

When a user resets his/her password, the program will send a query to the database to check whether the input old password matches to the password stored in the database.

RecommendedPlacesDB

The second database used is named recommended places database which contains three tables named Bath, London and Edinburgh respectively. Each table has eight attributes which are place number, place ID, place introduction, place working hour, place suggested duration,

place name, place address and place phone number. The following images show the structure of the database and tables. The place number which represents the popular ranking of scenic spots is set as the primary key of all three tables since it is unique. This database is different from the first database since it is established in the development stage.



p_no	p_id	p_intro	p_working_h	p_suggested_d	p_name	p_addr	p_phone_no
1	ChDB	A museum	10:00 - 17:3	2-3 hours	The British M	Great Rus	+44 20 7323 8
2	ChD3	One of Lor	09:00 - 16:3	2-3 hours	Tower of Lo	St Kathar	+44 844 482 7
3	ChDq	The under	09:30 - 18:0	2-3 hours	Churchill Wa	Clive Steg	+44 20 7416 5
4	ChDD	An amazin	08:00 - 17:0	2-3 hours	Borough Ma	8 Southw	+44 20 7407 1
5	ChDS	An iconic l	10:00 - 17:3	1-2 hours	Tower Bridg	Tower Br	+44 20 7403 3
6	ChDe	The Nation	10:00 - 18:0	2-3 hours	The Nationa	Trafalgar	+44 20 7747 2
7	ChDL	Westminst	09:30 - 15:3	1-2 hours	Westminster	20 Deans	+44 20 7631 0
8	ChDw	The world	09:30 - 15:3	2-3 hours	Victoria and	Cromwell	+44 20 7631 0
9	ChDh	Once the l	05:00 - 00:0	2-3 hours	Hyde Park	Knightsbr	+44 300 061 2
10	ChDl0	Most of th	09:30 - 15:3	2-3 hours	Houses of P	Palace of	+44 20 7219 4
11	ChDP	A center o	10:00 - 17:5	More than 3 h	Natural Histi	Cromwell	+44 20 7942 5
12	ChDh	Rebuilt by	08:30 - 16:3	2-3 hours	St. Paul's Ce	St. Paul's	+44 20 7246 8
13	ChD8	Set on the	09:30 - 15:3	More than 3 h	Greenwich	Greenwic	+44 20 7631 0

Figure 23: Information Database

The application will interact with this database when the following event happen: When a user clicks the attraction information button which would be shown as the user clicks an attraction. Program will send query to the database to retrieve all data related to the chosen attraction.

Figure 24 shows the query to get data and figure 25 shows the attraction information interface with all information related to this attraction displayed.

```
Cursor cursor=recommendedPlacesDB.query(targetCityName,
    new String[] {PLACE_INTRODUCTION,
        PLACE_WORKING_HOURS,
        PLACE_SUGGESTED_DURATION,
        PLACE_NAME,
        PLACE_ADDRESS,
        PLACE_PHONE_NUM},
    selection: PLACE_ID+ "=?",
    new String[] {place.getId()},
    null,
    null,
    null,
    null,
    null);
```

Figure 24: Retrieve Data from database



Figure 25: display Retrieved Data

4.3 Data Dictionary

Entity	Attributes	Description	Data type and length	Field constraints
UserInfo	username	Uniquely identifies a user	50 variable characters	Not null primary key
	password	The password is used to authenticate the user	50 variable characters	not null
	creationDate	The time of the account created	20 variable characters	not null
Bath	p_no	Uniquely represents the popular ranking of scenic spots	integer	not null primary key
	p_id	Place ID got from Google Places API	200 variable characters	not null

[COMP208] Design Document

	p_intro	Attraction's introduction	500 variable characters	
	p_working_hour	Attraction's working hour	50 variable characters	
	p_suggested_duration	Suggested play time	50 variable characters	
	p_name	Attraction's name	50 variable characters	not null
	p_addr	Attraction's address	100 variable characters	not null
	p_phone_no	Attraction's phone number	30 variable characters	
London	p_no	Uniquely represents the popular ranking of scenic spots	integer	not null, primary key
	p_id	Place ID got from Google Places API	200 variable characters	not null

[COMP208] Design Document

	p_intro	Attraction's introduction	500 variable characters	
	p_working_hour	Attraction's working hour	50 variable characters	
	p_suggested_duration	Suggested play time	50 variable characters	
	p_name	Attraction's name	50 variable characters	not null
	p_addr	Attraction's address	100 variable characters	not null
	p_phone_no	Attraction's phone number	30 variable characters	
Edinburgh	p_no	Uniquely represents the popular ranking of scenic spots	integer	not null, primary key
	p_id	Place ID got from Google Places API	200 variable characters	not null
	p_intro	Attraction's introduction	500 variable characters	

[COMP208] Design Document

	p_working_hour	Attraction's working hour	50 variable characters	
	p_suggested_duration	Suggested play time	50 variable characters	
	p_phone_no	Attraction's phone number	30 variable characters	
	p_name	Attraction's name	50 variable characters	not null
	p_addr	Attraction's address	100 variable characters	not null

Figure 26: Data Dictionary

4.4 Global Logical Data Model

UserInfo(username, password, creationDate) Primary Key username
Bath(p_no, p_id, p_intro, p_working_hour, p_suggested_duration, p_name, p_addr, p_phone_no) Primary Key p_no
London(p_no, p_id, p_intro, p_working_hour, p_suggested_duration, p_name, p_addr, p_phone_no) Primary Key p_no
Edinburgh(p_no, p_id, p_intro, p_working_hour, p_suggested_duration, p_name, p_addr, p_phone_no) Primary Key p_no

4.5 Physical Data Table

Table UserInfo

domain Username	variable length character string maximum length 50
domain Password	variable length character string maximum length 50
domain Creation Date	variable length character string maximum length 20

UserInfo(username	Username	NOT NULL,
password	Password	NOT NULL,
creationDate	Creation_Date	NOT NULL)
Primary key username		

Table Bath

domain Place_Number	Integer
domain Place_ID	variable length character string maximum length 200
domain Place_Introduction	variable length character string maximum length 500
domain Place_Working_Hour	variable length character string maximum length 50
domain Place_Suggested_Duration	variable length character string maximum length 50
domain Place_Name	variable length character string maximum length 50
domain Place_Address	variable length character string maximum length 100
domain Place_Phone_Number	variable length character string maximum length 30

Bath(p_no	Place_Number	NOT NULL,
p_id	Place_ID	NOT NULL,
p_intro	Place_Introduction	
p_working_hour	Place_Working_Hour	
p_suggested_duration	Place_Suggested_Duration	
p_name	Place_Name	NOT NULL,
p_addr	Place_Address	NOT NULL
p_phone_no	Place_Phone_Number)	
Primary Key p_no		

Table London

domain Place_Number	Integer
domain Place_ID	variable length character string maximum length 200
domain Place_Introduction	variable length character string maximum length 500
domain Place_Working_Hour	variable length character string maximum length 50
domain Place_Suggested_Duration	variable length character string maximum length 50
domain Place_Name	variable length character string maximum length 50
domain Place_Address	variable length character string maximum length 100
domain Place_Phone_Number	variable length character string maximum length 30

Bath(p_no	Place_Number	NOT NULL,
p_id	Place_ID	NOT NULL,
p_intro	Place_Introduction	

[COMP208] Design Document

p_working_hour	Place_Working_Hour	
p_suggested_duration	Place_Suggested_Duration	
p_name	Place_Name	NOT NULL,
p_addr	Place_Address	NOT NULL
p_phone_no	Place_Phone_Number)	
Primary Key p_no		

Table Edinburgh:

domain Place_Number	Integer
domain Place_ID	variable length character string maximum length 200
domain Place_Introduction	variable length character string maximum length 500
domain Place_Working_Hour	variable length character string maximum length 50
domain Place_Suggested_Duration	variable length character string maximum length 50
domain Place_Name	variable length character string maximum length 50
domain Place_Address	variable length character string maximum length 100
domain Place_Phone_Number	variable length character string maximum length 30

Bath(p_no	Place_Number	NOT NULL,
p_id	Place_ID	NOT NULL,
p_intro	Place_Introduction	
p_working_hour	Place_Working_Hour	
p_suggested_duration	Place_Suggested_Duration	
p_name	Place_Name	NOT NULL,
p_addr	Place_Address	NOT NULL
p_phone_no	Place_Phone_Number)	
Primary Key p_no		

5. Business Rules

Our software is free to download. As a travel route plan app, because the team does not have function to help user book hotel, sell tickets and provide group travel service, it is a little bit difficult to make profit. However, with the accumulation of users, the team still have some viable profitable methods.

- I. Advertising revenue will probably be an important part of our product's profitability. At the beginning, the team should increase the number of users as soon as possible by posting advertisement on social media. After that, you can attract some advertisement to make a profit.
- II. We expect that when our users break through 100,000, the team can provide VIP services, users can buy our membership to get more privileges. For example, omitting advertisements and customizing application style.
- III. When the number of our users has exceeded 1 million, the team will open the shopping module and target selling travelling equipment to our user, such as trekking poles, tents, etc. VIP users will be able to redeem points and give 20% off on some items.
- IV. We can also try to attract different type of people (not only tourists). It is possible to transplant our function that plans the shortest route in multiple locations to open up new sections to help the courier and other workers who need to visit multiple places a day to plan the shortest route, thus attracting more types of opportunity (other than tourists) using our product. This will increase the number of our app users, thereby increasing advertising revenue, etc.

Bibliography

- [1] Kahn, M. J., and Prail, A. (1994). Formal usability inspections. In Nielsen, J., and Mack, R.L. (Eds.), Usability Inspection Methods, John Wiley & Sons, New York, 141–172.

Team 22 Signatures

Jamie Garvin -

A stylized, cursive signature consisting of several overlapping loops and a long horizontal stroke at the bottom.

Henan Liu -

A cursive signature in Chinese characters, appearing to read '刘赫南' (Liu Hènnán).

Kuan Lu -

A cursive signature in Chinese characters, appearing to read '路宽' (Lù Kuān).

Xiao Ma -

A cursive signature in Chinese characters, appearing to read '马骁' (Mǎ Xiāo).

Weihua Gao -

A cursive signature in Chinese characters, appearing to read '高伟华' (Gāo Wěihuá).

Hao Wu -

A cursive signature in Chinese characters, appearing to read '吴昊' (Wú Hào).



UNIVERSITY OF
LIVERPOOL

Project Report



Team 22

Jamie Garvin

Henan Liu

Weihua Gao

Hao Wu

Xiao Ma

Kuan Lu

Contents

People and Roles	3
Application Overview	4
Project outcome	5
Object Achieved	5
Object Not Achieved	5
Project Strengths	6
Project Weaknesses	7
Future Developments	7
BCS Code of Conduct – Professional Issues	8
Bibliography	9
Team Signatures	10

I - People and Roles

Team 22 consists of 6 members; Hao Wu, Weihua Gao, Xiao Ma, Henan Liu, Kuan Lu, and Jamie Garvin. The team have a clear division of work so that everyone understands our project clearly and complete the project. The team mainly divided the tasks into three parts: the algorithm, Google Maps and the user interface design.

The following is a chart describes our main tasks and some additional contribution.

Group Member	Main Tasks	Additional contribution
Jamie Garvin	Requirements elicitation and analysis Study UI principle and UI design Study web crawler and find suitable travel website	Documentation grammar and error checking Documentation revision
Hao Wu	Requirements elicitation and analysis Study Google API and finish map view Algorithm design Program testing and debug	Project planning and management Integration of all functions in application
Weihua Gao	Requirements elicitation and analysis Data collection via web crawler and coding Study algorithm and achieve route optimization function	Project planning and management Documentation revision
Henan Liu	Requirements elicitation and analysis Study Google API and finish map view Interface design and program testing and debugging	Integration of all functions in application
Xiao Ma	Requirements elicitation and analysis Data collection via web crawler Study UI principle and UI design	Documentation revision
Kuan Lu	Requirements elicitation and analysis	Integration of all functions in applications

	Study Google API and finish map view Database design and implementation Interface design and Program testing and debugging	
--	---	--

From the perspective of the completion of the project, the part of learning Google Maps API has been completed by Kuan, Hao, and Henan; the data collected by the web crawler has been completed by Weihua, Xiao, and Jamie; and the algorithm was participated by Weihua, Xiao, and Hao and it is finished mainly by Weihua; UI design was mainly carried out by Jamie and Xiao; database was participated by Weihua, Kuan, Hao, and was finished by Kuan.

II - Application Overview

EasyTrip is a travel application which main purpose is to provide travellers with an introduction to the attractions and path planning between the attractions. Its audience should be visitors, as well as those who love to travel.

The use of this application has a related background. When visitors travel to a city where there are multiple attractions they want to visit, visitors are often confused and do not know how to arrange the order of the attractions considering (saving time, short walking distance). Based on this situation, this application provides visitors a way to plan travel routes between multiple attractions, making it easy for them to go through all the attractions.

The in-city trip route planning system is used to recommend attractions in a given city with a list view, it shows the city's famous attractions, travellers can click on each attraction to see the introduction of the attraction, address, opening and closing time. Another view is map view, famous sights have been marked on the map so that travellers can clearly see the sights they want to choose. In both views, users can add their favourite attractions to the list of plans. The attractions in the plan list are used for route planning. If the user selects 4 attractions, EasyTrip application will show the best travel route to three other points from the starting point. Users can switch travel modes at any time, walk, ride a bicycle, or drive, and update routes for different modes of transportation. In this case, the EasyTrip application is better than Google Maps. Google Maps cannot achieve multi-point shortest path planning, and EasyTrip application implements this function.

III - Project outcome

Overall, the team finished the main objectives stated in the specification and design document and create a functional Android multi-attraction trip planning application with detailed information of attractions.

Objectives Achieved

1. Completed the design and implementation of the login system, including sign in, sign out and log out function.
2. Completed the design and implementation of the list view, recommended 20 to 30 famous attractions in the city to our users and provided information on the attractions, prices, opening hours, photos, etc.
3. Completed the algorithm design of route planning and solved the problem of returning to the starting point or not returning to the starting point TSP (travelling salesman problem) and ATSP (Asymmetric travelling salesman problem).
4. Completed the design and implementation of the map view, enabling the user to select the travel mode (bicycle, walk, or car) and whether they want to return to the starting point (such as the starting point being a hotel) and the user can select multiple attractions before planning the route. Besides, the user could choose attractions, plan routes, optimise routes, and track their locations in real time through GPS.
5. Designed a plan list interface to help users clearly view the attractions they selected and change the order of the attractions (plan the route in the order they prefer rather than in the optimal order) and remove attractions if they want.
6. Completed user account design, implemented reset password, log out and some other functions.
7. Added user guide to introduce users the use of our programs so that users could use our program more efficiently.

Objectives Not Achieved

1. At the beginning of the project design, the team would have liked to add a function that allows the user to choose a restaurant and plan a route between user and restaurant; but this was later removed due to the fact that only a route was extended from the user to the restaurant and the main object of our application was travel route plan. This functionality is also possible without the need for this separate segment being implemented.
2. At the beginning of the design, the team planned to add a function allowing users storing the route planned by them as a saved journey; but in the end, the team removed this feature due to time constraints.
3. At the beginning of the design, the weather function was planning to be added. However, because previous feedback suggested that this feature does not make much sense, in the end the team removed this feature.
4. At the beginning of the design, the team decided to make a category system so a user can sift out attractions that do not appeal to them, but given the data the team can and cannot get from the web, this is a little implausible given the time that the team have.

IV - Project Strengths

Function

EasyTrip application is characterized by its functions, which not only provides users with an introduction to the popular attractions of the selected city, but also plans the route of the attractions they want to go. The focus is on multi-point path planning, which saves travel time and provides users with the best route and sequence of multiple attractions.

Usability

Usability reflects the friendliness of the application. A large part of the user experience comes down to how usable the software is by the user.

The user interface design of the application is clear, and the icons used are easy to understand, the user can understand how to use the application for the first time. The menu bar also introduces the usage and functions of the application, which increase the usability. The user interface design of the application follows the UI design principles, button functions, jumps and other functions are simple and easy to understand, so that the entire application runs harmoniously.

Efficiency

Efficiency refers to the degree to which computer resources (including time) are required for software to perform a certain function under specified conditions. Efficiency reflects the waste of resources when implementing effective functions. The efficiency of an application is important, and it determines the space and time savings.

EasyTrip application does a good job of improving efficiency. All resources are called once and are not called repeatedly, which saves a lot of time. For example, the map view will only be loaded once, not every time the user switches views.

Dependability

Reliability refers to the extent to which software maintains its performance levels under specified time and conditions. Reliability is a very important quality requirement for an application. It responds to the extent to which the application meets the user's needs for normal operation and reflects the extent to which it can continue to operate in the event of a failure.

EasyTrip application adds error handling, such as limiting user names and passwords, it also set up a local database for each user to store their information, increasing the reliability of the application.

Maintainability

Maintainability reflects the ease with which software systems can be modified as changes in user requirements or software environments.

The EasyTrip application program has an obvious distinction and connection between each modules when the code is written. The code is independent and easy to modify. Add comments at the same time as the code is written to make the code easier to understand. Because the code is independent, the code is easier to test.

V - Project Weaknesses

Cross platform support

Due to time and technical reasons, the application has not yet implemented cross-platform. At present, our application can only be used on the Android platform; iOS and Windows phones cannot be achieved at this time.

Security

There is still a lack of security in our application. The team have only limited the username and password so that the user's username and password are not too simple. The team have also implemented the password change function, and the password can be changed only when the user has logged in. Other security measures have not been done thoroughly, and this can be improved.

VI - Future Developments

1. Show order of in-route places

Travel route could be displayed with extra information to show the order of in-route places, making it easier for the user to recognise the starting point and the order of places to visit.

To achieve this, numbers could be added to route markers to indicate the visiting order. For example, starting point can be labelled as “1” and the next location should be labelled as “2” and so on. Alternatively, numbers could be displayed when user clicks a specific route segment. Take this as an example, the first segment of the route which connects the first and the second place should be labelled with “1”, the next segment which starts at the second place and ends in the third place should be labelled with “2” and so on.

2. Display and update travel time

Travel time for each route segment and for the whole route could be displayed and updated in real-time. Application could automatically decide whether it is possible to visit the next place depending on travel time to next place and opening hours of next attraction.

This could be achieved by sending a request for travel time between two places to Google every five minutes or by requesting travel time for a specific route segment once user clicks it. On receiving a response, simple calculation is performed, and application compares the calculation result with the opening hours of next attraction stored in database to decide whether to inform user that there is no time to visit the next attraction.

3. Enrich information for the recommendation system

Additional information for recommended attractions, including 3D panoramic view, 2D plan view and audio tours, could be added to enhance user experience.

To enrich information for attractions, more functions provided by current content providers or new content providers should be introduced. For instance, 3D panoramic view from Google could be added since this function had been implemented in the early stage of this application but was deprecated since not necessary. In addition, this application can cooperate with VoiceMap or similar content providers to provide audio tours for tourist attractions or events to users.

4. Save route for further retrieval

In order to improve the convenience of using this application, user-defined route could be stored under user's account, user can restore stored routes, add or delete routes under his/her account.

This function requires expanding database for each user. NoSQL is more appropriate for storing user information for Android application due to its flexibility, scalability and efficiency. Firebase real-time database could be used to store information for users. Data would be synced across all clients in real-time and remains available when the application goes offline [1].

5. Recommend popular travel routes

Some recommended routes for the user new to a specific city save the user's time for choosing attractions and planning routes. Feedback for recommended routes should be collected and the one with highest rate will be recommended for future uses.

Many features, including travel time, route feedback, traffic and weather conditions, should be taken into account with some specific mathematical models, like Principal Component Analysis (PCA), to generate appropriate recommended route for a user.

6. Filter recommended attractions based on categories

User could choose to display recommended attractions with specific categories on map.

To fulfill this, attractions had been divided into several categories in the final stage at development. Further operations on markers on map should be programmed to make this function work properly.

VII - BCS Code of Conduct – Professional Issues

The British Computer Society (BCS) Code of Practice [2] describes standards of practice relating to information technology (IT). It is of importance for a group project to follow these guidelines due to real development environment requiring specific industry standards to help produce practical IT products. To fit into this framework of guidance, this group project is finished in strict with relevant professional standards.

In requirements specification phase, all functional and non-functional requirements are fully in line with interests of target customers. Concise functions, including displaying travel route with attractions information, are provided by this easy-to-control application. Security is guaranteed by introducing formal and complex rules for setting passwords [3] and by not recording any personal information. Meanwhile, confidential information for any account shall not be disclosed to any third party. To efficiently finish this project with high quality, work is separated into three parts in general and assigned to different members in this group. Considering quality of the application, reasonable workload is defined and some functionalities which are not necessary or time-consuming, such as weather function, are regarded as optional unless time allows.

In the design phase, to maintain technical competence, each group member improved developing skills on Android application through attending google developers training for Android [4] and official guidelines for Android developers [5]. Since many Google APIs, including Places API, Directions API,

are used in EasyTrip, regulations provided by Google are adhered to fulfill required functions correctly. When encountering programming problems, group members tried to act professionally as a specialist. For example, in the early development process, place name autocomplete widget with a search dialog, was considered to implement search function for a specific place. However, google decided to deprecate this widget [6] and some map functions in EasyTrip are implemented by old APIs. Google recommended us implementing new Google Maps APIs, but migration of some key functions for EasyTrip were not finished and there are very few tutorials on new APIs. Another solution provided by Google is using 'compat APIs' which support old version of Google Maps Library. With the help of a Github project provided by a professional Google Android Developer and many suggestions from Stack Overflow, this problem was solved efficiently.

As to group management, this project has a clear distinction between different parts. The whole program of work is separated into implementing algorithm, showing map widgets, displaying route, designing user interface (UI) and collecting attractions information, each with a reasonable deadline. Additionally, all team members were given written instructions for each task to perform. During the process of the project, each member gives critical suggestions on his/her area of expertise, and all team members help with each other to overcome difficulties.

Since BCS code is related to supply of expertise to a customer, many guidelines are not relevant to EasyTrip application due to its non-commerciality.

VIII - Project Bibliography

Websites

- [1] Google. (2019). Firebase Available from <https://firebase.google.com/docs/database> (25 April 2019).
- [2] The British Computer Society. (2011). Code of Good Practice. Available from <https://www.bcs.org/upload/pdf/cop.pdf> (25 April 2019).
- [3] University of Georgia. (2018). Password Standard. Available from https://eits.uga.edu/access_and_security/infosec/pols_regs/policies/passwords/password_standard/ (17 April 2019).
- [4] Kuan, K. (2014). Build your Android development skills. Available from <https://developers.google.com/training/android> (17 April 2019).
- [5] Kuan, K. (2017). build anything on android. Available from <https://developer.android.com/> (6 May 2019).
- [6] Chawla, K. (2015). Place Autocomplete Available from <https://developers.google.com/places/android-sdk/autocomplete> (6 May 2019).
- [7] The Code City (2018). Draw route between two locations in Android – Google Map directions API. Available from <https://www.youtube.com/watch?v=wRDLjUK8nyU6> (9 May 2019).
- [8] TripAdvisor (2000). Things to do in Bath Available from https://www.tripadvisor.co.uk/Attractions-g186370-Activities-Bath_Somerset_England.html.html (11 April 2019).
- [9] SmashIcon (2019). FlatIcon Available from <https://www.flaticon.com/> (9 May 2019).
- [10] Google (2019). Google Docs API – Using the API Available from <https://developer.android.com/docs> (9 May 2019).

IX – Acknowledgment

This work was supported by the project “Draw route between two locations in Android” belongs to The Code City, project “Methods in Alert Dialog” belongs to Chandu Yadav.

IX - Team 22 Signatures

Jamie Garvin –

A stylized, cursive signature consisting of several overlapping loops and a long horizontal stroke at the bottom.

Henan Liu –

A signature in Chinese characters, written in a cursive style. The characters are '刘' (Liu) and '赫南' (Henan).

Kuan Lu –

A signature in Chinese characters, written in a cursive style. The characters are '路' (Lu) and '宽' (Kuan).

Xiao Ma –

A signature in Chinese characters, written in a cursive style. The characters are '马' (Ma) and '晓' (Xiao).

Weihua Gao –

A signature in Chinese characters, written in a cursive style. The characters are '高' (Gao) and '伟华' (Weihua).

Hao Wu –

A signature in Chinese characters, written in a cursive style. The characters are '吴' (Wu) and '昊' (Hao).



UNIVERSITY OF
LIVERPOOL

Testing Report

Team 22

EasyTrip

Hao Wu

Jamie Garvin

Weihua Gao

Xiao Ma

Kuan Lu

Henan Liu

1 - Testing Strategy

According to the feedback from reviewer, the **usability testing** and **specific standard** of username and password are included in this testing documentation.

1.1 - Method Used

- **Summative (usability) testing**

Summative testing is a form of usability testing which focuses on the user experience of the fully developed product (rather than in-development usability testing, formative testing). This will be the primary method of ensuring our application hits the criteria we set out to achieve when defining our project. This is important, as although everything may “functionally” work, it may not work in a coherent manner and achieve our targets. [2]

- **Static testing**

In this test, the tester does not need to run the application executable but to analyse the code or operations to find internal defects. For example, the tester will test the subsystems by using use cases provided in the use case diagrams or follow the user manual to examine if all the functional requirements are satisfied.

- **Black box testing**

The tester mainly uses black box testing for the login function and search function of our application. Large amounts of testing cases and edge values are offered in tables and the examiner can follow them to test the functions thoroughly.

- **Graphical User Interface testing**

After the integration of subsystems, the tester conducts GUI testing to make sure the user can understand all of the interfaces and each of them can be operated normally.

- **Smoke testing**

This testing is incomplete and not detailed, tester utilizes it to ensure that the most important functions work, so we (the developers) use it at the state of sub-system accomplished. The result of this testing is used to decide if the current version is stable enough to proceed with further testing [1].

- **White-box testing**

After we (the developers) have finished testing using black-box, the developers will move onto white-box testing to ensure we execute each path in the program. These developers will go through the code finding the possible defects which cannot be located via black-box testing, such as duplicate code and such.

1.2 - Testing Plan

The testing plan as follows covers the testing of every function of EasyTrip.

The tables used in testing plans are retrieved from design documentation, which have been updated to reflect the latest development since the design. Tester can find them in appendix A the tables mainly focus on Graphical User Interface testing (GUI testing) and black box testing. Additionally, the blanks of the actual results are filled with real testing outcomes.

The method of using the table is simple; the tester will carry out each test written down and provide the outcome of said test. This achieves that each function is modularly tested in order to ensure the subsystems are all working together.

1. Login Function and Change Password
 - Firstly, the tester uses Graphical User Interface and black box testing with the testing cases provided in Table A and check the if output obeys the anticipated results.
2. Map View
 - Firstly, the tester uses Graphical User Interface and black box testing with the testing cases provided in Table B, C and D and check the if output obeys the anticipated results.
3. Attractions List
 - Firstly, the tester uses Graphical User Interface and black box testing with the testing cases provided in Table C and D and check the if output obeys the anticipated results..
 - Test if the details about the specific attraction is matched and correct when tester enters the introduction interface.
4. Plan List
 - Firstly, the tester uses Graphical User Interface and black box testing with the testing cases provided in Table C and D and check the if output obeys the anticipated results.
 - Secondly, use Graphical User Interface to examine the function of swipe view, if the button below the appearance can be seen after swipe and if the button provides correct effects after they are pressed.
5. City and Attraction Search
 - Firstly, the tester uses Graphical User Interface and black box testing with the testing cases provided in Table C and check the if output obeys the anticipated results.
6. Recommended attractions
 - Manually read samples of the database and compare various entries against each other to find discrepancies in the database concerning consistency. This is important, as the data from professional website must be appropriately categorized.
7. Route
 - Firstly, tester uses Graphical User Interface and black box testing with the testing cases provided in Table D and check the if output obeys the anticipated results
8. Alternate User Interfaces
 - Test all the buttons and view with tapping to see if they provide correct function and interface switch

2 - Analysis of Testing Results

2.1 – Summative (usability) testing results

For the summative tests[2], we had a pool of 10 people to try our application. The age range varied, along with technical knowledge. 3 people aged between 16-17, 5 people aged between 18-25, and the remaining 2 people aged between 43-51. Two of the participants study/studied computer science or software development.

We have broken down analysis into several groups;

- Successful actions (of legal attempts).
- Ability to create a route around London, Bath, or Edinburgh without developer help or hints.
- General understandability of user interface.

Success rate of actions performed

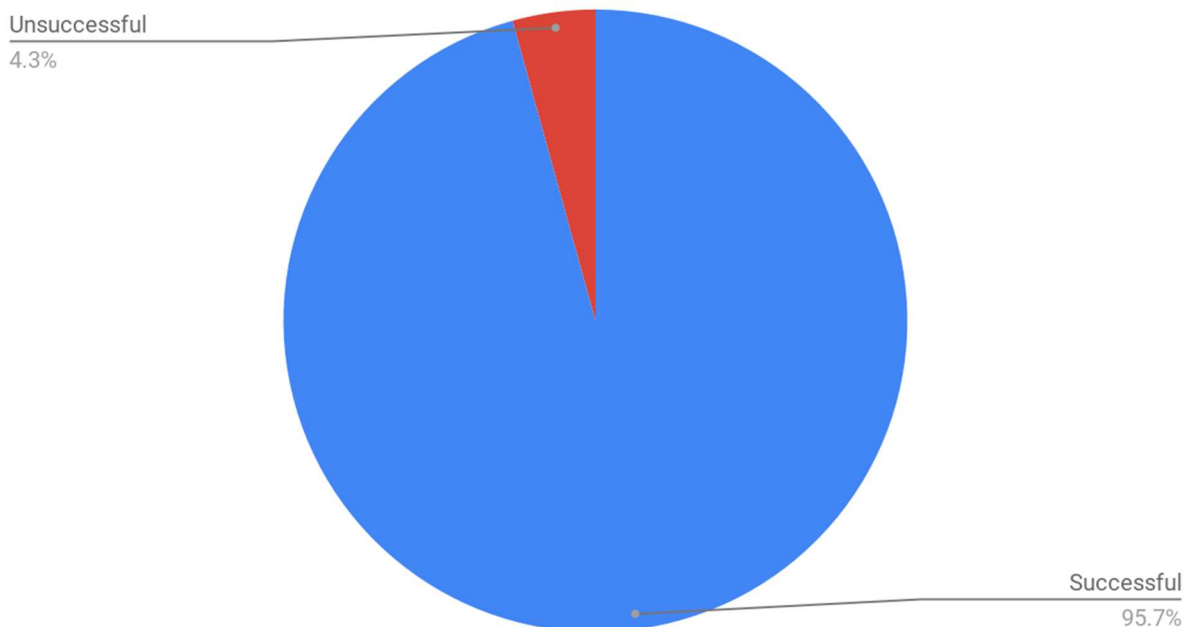


Figure 2.1: A pie chart depicting ratio of successful and unsuccessful actions amongst various uses.

The success rate of actions performed was based off each user completing each possible action once on the application. An action is denoted by either clicking an application button (button to enter register screen, or to see list view, etc.) or providing an input type to the application (such as inputting a username, adding route to list, etc.). There is 28 clear actions to be performed on our application, meaning there is a total of 280 actions to analyze (10 people carrying out all 28 actions each).

Out of the 280 tests carried out, 95.7% of tests were successful. The only issue from these tests was that returning to the previous screen sometimes caused an activity duplication bug, from the user selecting the Android 'return' button rather than selecting a button from the bottom of the screen (such as the map view or list view). This is a known bug and is down to Google's own code, and in further development we would aim

to create a workaround for this. All the successful attempts were legal actions (using correct registration criteria, rather than not meeting password requirements, for example).

Route creation clarity

When it came to testing the usability of whether it is clear and succinct to understand how to create the route without supervision from a developer, results were more varied to that of the last test. We found that some users did not understand the ability to add multiple locations onto the same route was available, meaning it took them a little longer to work out the scope of the application. The longest amount of time it took was 34 seconds to figure out how to create a tangible route and optimize it, which was one of the participants in the 43-51 age range, with the second longest being 26 seconds, also belonging to the 43-51. The other participants had an average time of 18 seconds. This data suggests that the older generations may not find this application as straightforward as younger generations. Although, it is worth noting the sample size may not be reflective of a public release. Regardless, using this data we can deduct that it may seem ideal to introduce a tutorial in future releases. All of these tests were based off the first usage of the end user.

User interface understandability

UI understandability

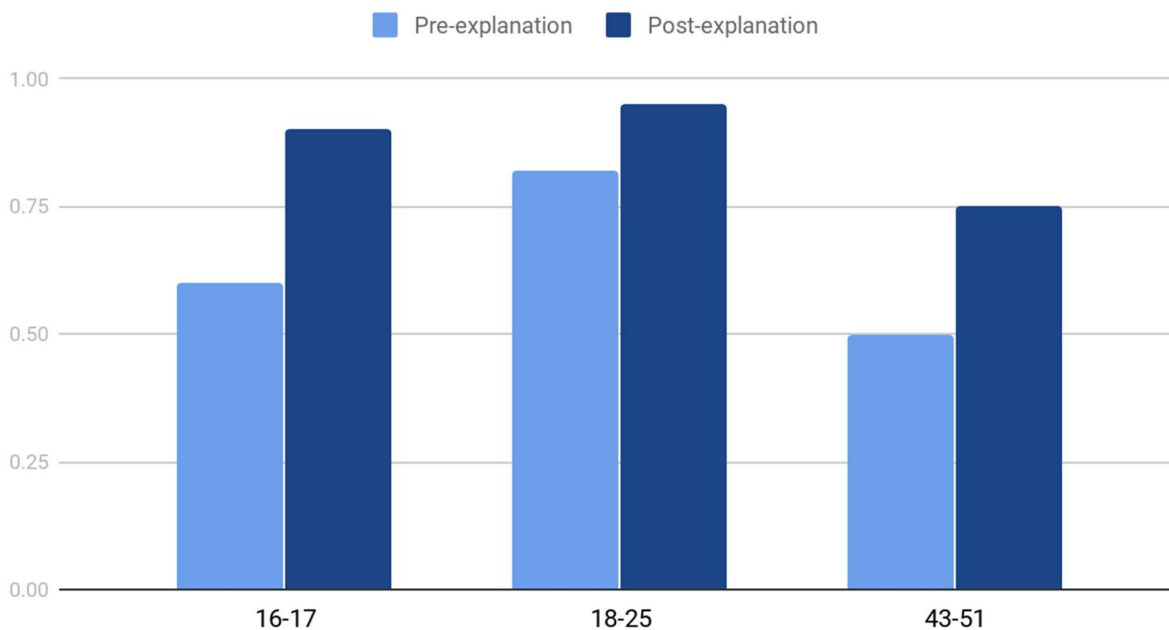


Figure 2.2: A bar chart depicting user feeling about whether user interface is clear or not.

X Axis - age ranges used during testing

Let Y Axis numbers denote:

All numbers are average of the respective age range.

0 - Absolutely no understanding

0.25 - Disagree that it is understandable

0.50 - Feel that UI can cause confusion

0.75 - Agree that the UI is understandable

1 - Completely understand the meaning of the UI

In this test, we conducted two types of questioning for the same issue. One test was before we had explained what each component of the UI achieved/signaled, and the next was after we had explained what each UI component achieved/signaled. This opens more data to be interpreted and gives the user to provide more opinion. From getting a before/after data set, we're able to analyze if the UI at least makes sense to a user after a short amount of time. For example, if a user does not understand something (while being new to the application) before it being explained, and understands it after we had explained the general idea, it signifies the problems lies more with their understanding of what the application achieves functionally rather than a problem with the user interface itself.

2.2 - Login Function and Change Password Testing Results

The function runs correctly and smoothly.

2.3 - Map View Testing Results

Unexpected outcome: To close the "Plus" button user must firstly tap on a different location on the screen. The user cannot tap on the marker again to make it disappear.

Analysis: this bug is generated due to API provided by google reacting to click activity automatically, and we have tried to rewrite corresponding listeners, but still conflict. We will consider it in future improvement of this application.

2.4 - Attractions List Testing Results

The function runs correctly and smoothly.

2.5 - Plan List Testing Results

The function runs correctly and smoothly.

2.6 - City and Attraction Search Testing Results

The function runs correctly and smoothly.

2.7 - Recommended attractions Testing Results

The function runs correctly and smoothly.

2.8 - Route Testing Results

The function runs correctly and smoothly.

2.9 - Alternate User Interfaces Testing Results

Unexpected outcome: After the operation of the plan list (e.g. adjust the order of plan, delete the attraction, optimization), if user want to go back to map view by using the Android "return" button provided by cell phone, the route cannot update correspondingly, as the Android "return" button will obstruct the parameter passing.

Analysis: the "Return" button will obstruct the parameter passing. The problems may be resolved by disabling the "Return" button provided by cell phone or change the way of passing data.

Appendix A

Login system

In this table we use JUnit to test each of the following, to ensure it is formally verified. The screenshots of the JUnit code and results are included in the appendix.

Table A: Login

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
A1.	Test login function with the test cases provided in List I.	The dialog pop up to inform tester login failed	As expected
A2.	Enter the correct username and password, then push “Login” button	Tester log in system successfully and GUI turns to city selection	As expected
A3.	Push “Return” button and push “Sign Up” button	GUI turns to sign up part	As expected
A4.	Test sign-up function with the test cases provided in List II.	The dialog pop up to inform tester sign up failed	As expected
A5.	Input unused username and password with correct format	The dialog pop up to inform tester sign up successfully	As expected
A6.	Push “Return” button and input username and password registered before, then push “Login” button	Tester log in system successfully and GUI turns to city selection	As expected
A7.	Push “Menu” button then push “Account” button	The GUI turn to account interface	As expected
A8.	Push “Logout” button	The GUI back to login interface, and the username box and password box are both empty	As expected
A9.	Login again and enter city selection interface	/	As expected
A10.	Push “Menu” button then push “Account” button	The GUI turn to account interface	As expected
A11.	Push “Change Password”	The GUI turn to interface to change password	As expected

A12.	Input new password and push “Confirm” button	The dialog pop up to inform tester if the change is valid	As expected
------	--	---	-------------

◆ Standard of username and password

Standard of Username:

- Username must contain no space or tab
- Username should use a minimum of 5 characters
- Username must include at least three of the four following types of characters:
 - English upper-case letters (A-Z).
 - English lower-case letters (a-z).
 - Numbers (0 through 9).
 - Underline (“_”).

Standard of Password:

- Password must contain no space or tab
- Password should use a minimum of 10 characters
- Password must include at least three of the four following types of characters:
 - English upper-case letters (A-Z).
 - English lower-case letters (a-z).
 - Numbers (0 through 9).
 - Special characters and punctuation symbols (E.g. _, -, +, =, !, @).

◆ List I.

Test case of login

Tester selects the combination of username and password from the Table I. and Table II.

1. Correct username and password

Anticipated result: login successfully

2. Incorrect username but correct password

Anticipated result: The dialog pop up to inform tester login failed

3. Incorrect password but correct username

Anticipated result: The dialog pop up to inform tester login failed

4. Incorrect username and password

Anticipated result: The dialog pop up to inform tester login failed

◆ Test data of incorrect password

Assume the correct password is "Comp208_victory"

Equivalence Partitions	Input Example
Empty input	""
Disobey the standard	"111222"
Case-sensitive	"CoMp208_victory"
Redundant characters	"comp208abcd_victory"
Missing characters	"cm208_victory"
Special characters	" comp 208 victory"

Table I. Test data of incorrect login password

◆ Test data of incorrect username

Assume the correct username is "group22"

Equivalence Partitions	Input Example
Empty input	""
Disobey the standard	"\$#@&*!"
Case-sensitive	"gRoup22"
Redundant characters	""group22group22"
Missing characters	"grp22"
Special characters	"@#\$group22"

Table II. Test data of incorrect login username

◆ List II.

Test case of sign-up

Tester selects the combination of username and password from the Table III. and Table IV.

- Unused and right-formatted username and right-formatted password
Anticipated result: sign up successfully
- Unused and right-formatted username and wrong-formatted password
Anticipated result: sign up failed
- Unused and wrong-formatted username and right-formatted password
Anticipated result: sign up failed

4. Unused and wrong-formatted username and wrong-formatted password

Anticipated result: sign up failed

5. Used username and right-formatted password

Anticipated result: sign up failed

6. Used username and wrong-formatted password

Anticipated result: sign up failed

◆ Test data of incorrect username

Assume the only used username in the database is "group22"

Equivalence Partitions	Input Example
Empty input	" "
Over-limit length username	"group88abcdefghijk" (length over 12)
Illegal characters	"group88%&#"

Table III. Test data of sign-up username regarding unused and wrong-formatted ones

◆ Test data of unused username

Equivalence Partitions	Input Example
Unused username	"group88"

Table IV. Test data of sign-up username regarding unused and right-formatted ones

◆ Test data of wrong-formatted password

Assume the right-formatted password is "Comp208_victory"

Equivalence Partitions	Input Example
Empty input	" "
Disobey the standard	"111222"
Less than the standard length	"CoMp208"
Missing standard characters	"comp208victory" (Only two types of required characters)
Special characters	" comp 208_victory"

Table V. Test data of incorrect login password

Map function

➤ Table B: Test Positioning

Testing description	System version	v 1.00	
	Target function	Positioning	
Step	Operation	Anticipated result	Actual result
B1.	Input invalid city name in Table V	The dialog pop up to inform tester the city is not in database	As expected
B2.	Input valid city name (e.g. Liverpool)	Map of the city is displayed	As expected
B3.	Push “Map View”	Enter Map interface	As expected
B4.	Input customized start location in search box and select the place	Correctly locate input place on the map and mark it	As expected
B5.	Push “Return” button	The GUI goes back to city selection interface	As expected
B6.	Enter the same city name	Map of the same city is displayed, and the start location is empty, and the mark is removed	As expected
B7.	Enter another city name	Map of the city is displayed	As expected
B8.	Enter customized start location and select the place from search suggestion	Correctly locate input place on the map and mark it as start location	As expected

◆ Test data of input city name

Equivalence Partitions	Input Example
Invalid city name	ABC, 123
Out-of-domain* city	Pyongyang

Table VI. Test data of invalid city name

Out-of-domain city: In the demo, only several cities (e.g. Bath, London and etc.) will be stored in our database. These cities store their attractions information and other details in the database so that our program can invoke them to make the route.

➤ Table C: Test Attraction Selection

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
C1.	Input an invalid or out-of-domain city name	Map of the city is displayed but no recommended attractions	As expected
C2.	Enter valid city name and select city	Map of the city is displayed, and all recommended attractions are marked on map	As expected
C3.	Select travelling method by swiping the menu and click the button	Enter an interface ask user to choose “Map View” or “List View”	As expected
C4.	Push “List View” button	The list of recommended attractions is displayed	As expected
C5.	Swipe out the buttons under the attraction names, and push “Plus” button	The corresponding attraction is added to user’s list and the attraction is removed in recommended attraction list	As expected
C6.	Select random number of attractions	The system should run as usual, and the attractions are shown in the user’s list	As expected
C7.	Select travelling method by swiping the menu and clicking the button	Three methods buttons are displayed	As expected
C8.	Click them one by one	The route on map will change correspondingly	As expected

Table D: Route Planning, Route Optimization

Testing description	System version	v 1.00	
	Target function	Attraction Selection and Route Planning	
Step	Operation	Anticipated result	Actual result
D1.	Enter a valid city name and select the city	Map of the city is displayed, and all recommended attractions are marked on map	As expected
D2.	Push “List View” button	The list of top attractions is displayed	As expected
D3.	Select random number of attractions	The system should run as usual, and the attractions are shown in the user’s list	As expected
D4.	Go back to map and search a location in search box	Correctly locate input place on the map and mark it	As expected
D5.	Push the mark	The attraction name pops up with a “Plus” character	As expected
D6.	Push the “Plus” button	The attraction will be added in user’ list	Unexpected outcome: To close the “Plus” button user must firstly tap on a different location on the screen. The user cannot tap on the marker again to make it disappear.
D7.	Push the “Attractions” button again	The attraction list should pop up,	As expected
D8.	Change the order of selected attractions and return to “Map View”	The new shortest route is displayed on the map	As expected
D9.	Push the “Plan List” button and delete one or more attractions and return to “Map View”	The new shortest route corresponding to altered list is displayed on the map	Unexpected outcome: The use of “back button” causes parameter loss

D10.	Push the “Plan List” button again and click “Optimization” button	Dialog pop up to ask if user want to choose the start point as the end point of the trip	As expected
D11.	Press “Yes” or “No, thanks”	The order of selected attractions will change to correspond to the new shortest route displayed on the map	As expected

Response times

Table E: Average response time

Testing description	System version	v 1.00	
	Target function	Response times	
Test	Operation	Anticipated result	Actual result
E1.	Time taken for application to open completely	The application should open within an average time of 5000ms (5 seconds) across testing devices.	As expected
E2.	Time taken for the application to generate a route between chosen attractions	The application (with reasonable connection to the internet) should build a suitable route within an average of 3000ms (3 seconds) across testing devices.	As expected
E3.	Time taken to generate list of recommended attractions	The application should gather data and display attractions from a chosen city within an average time of 8500ms (8.5 seconds) across testing devices.	As expected

Appendix B

B1. Screenshot of Junit Code used for black box testing

```

260 ▾ /**
261    * Password is null
262    */
263    @Test
264    public void test19() {
265        username = "My06_";
266        password = "";
267        assertFalse("PASSWORD IS NULL!!!", login.checkUsernamePassword(username, password));
268    }
269
270
271 ▾ /**
272    * Password only has uppercase and lowercase alphabet
273    */
274    @Test
275    public void test20() {
276        username = "My06_";
277        password = "IkissMeeeee";
278        assertFalse("PASSWORD ONLY HAS UPPERCASE AND LOWERCASE ALPHABET!!!", login.checkUsernamePa
ssword(username, password));
279    }
280
281
282 ▾ /**
283    * Password only has uppercase alphabet and digit
284    */
285    @Test
286    public void test21() {
287        username = "My06_";
288        password = "IKISS09090";
289        assertFalse("PASSWORD ONLY HAS UPPERCASE ALPHABET AND DIGIT!!!", login.checkUsernamePasswo
rd(username, password));
290    }
291

```

Figure B1 1 Code of testing username

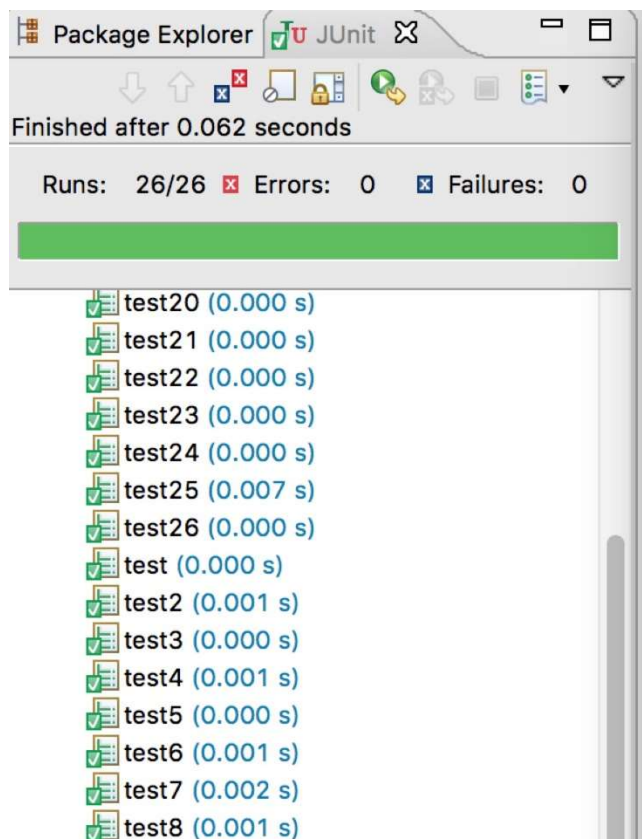
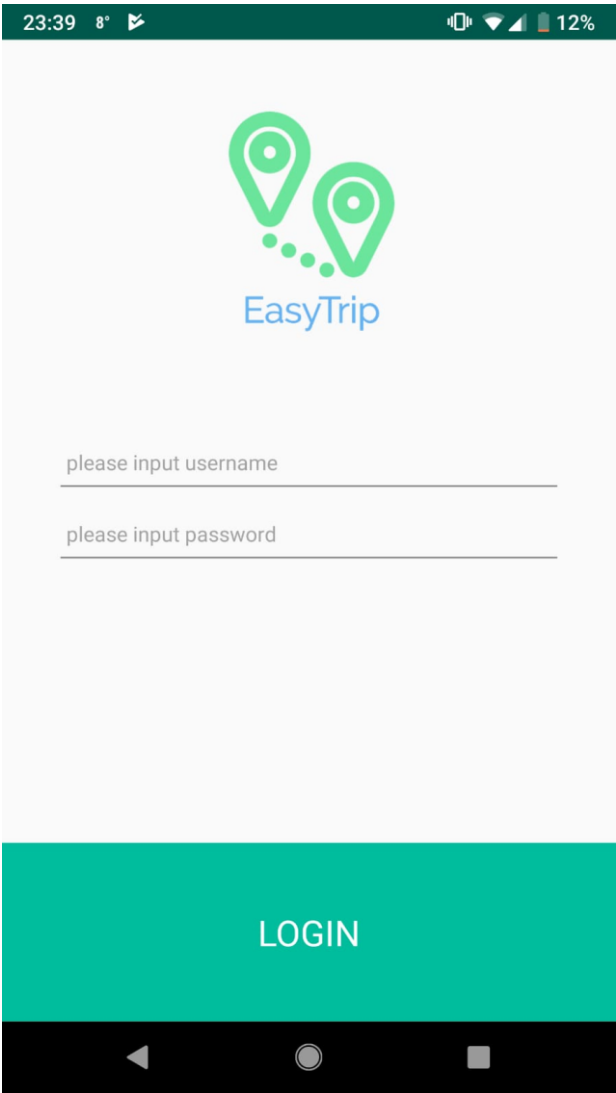
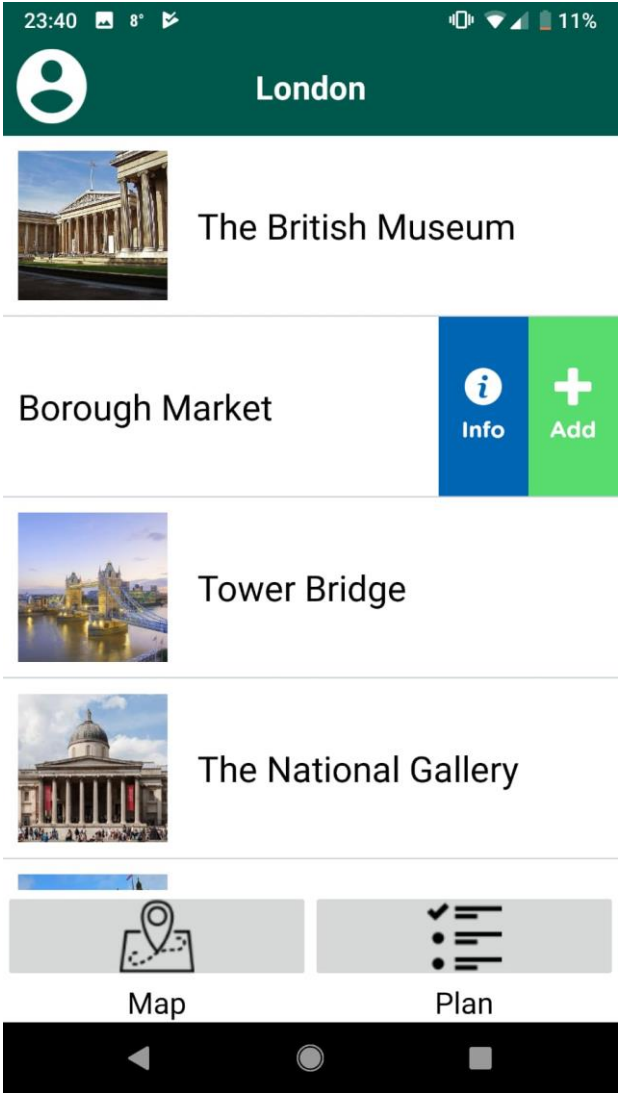


Figure B1 2 Testing result of username and password

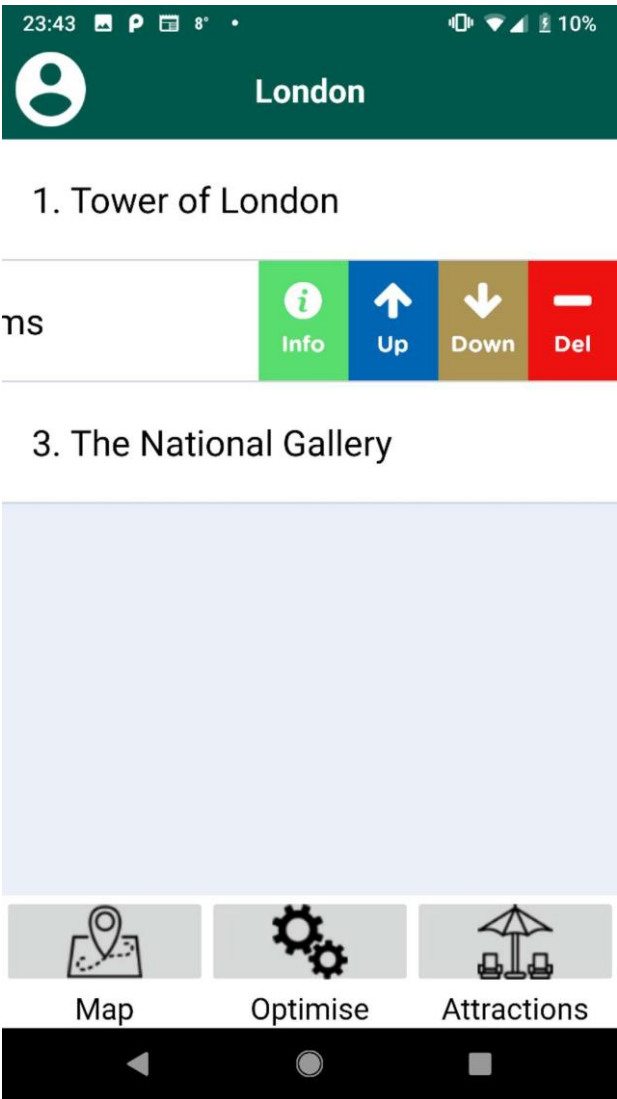
1. Login interface



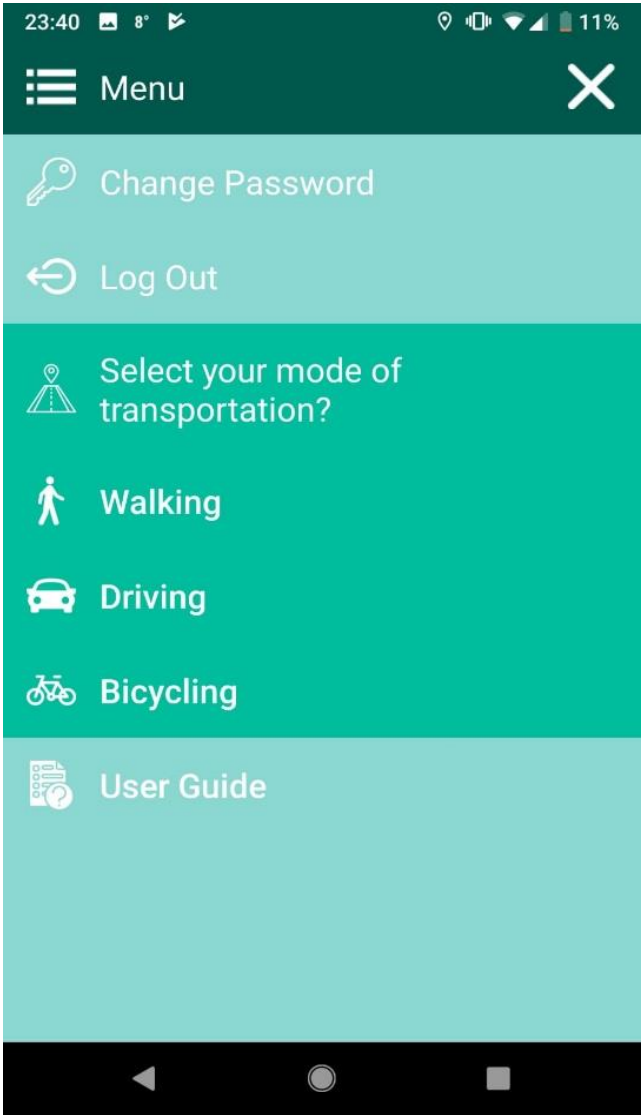
2. Attractions list view



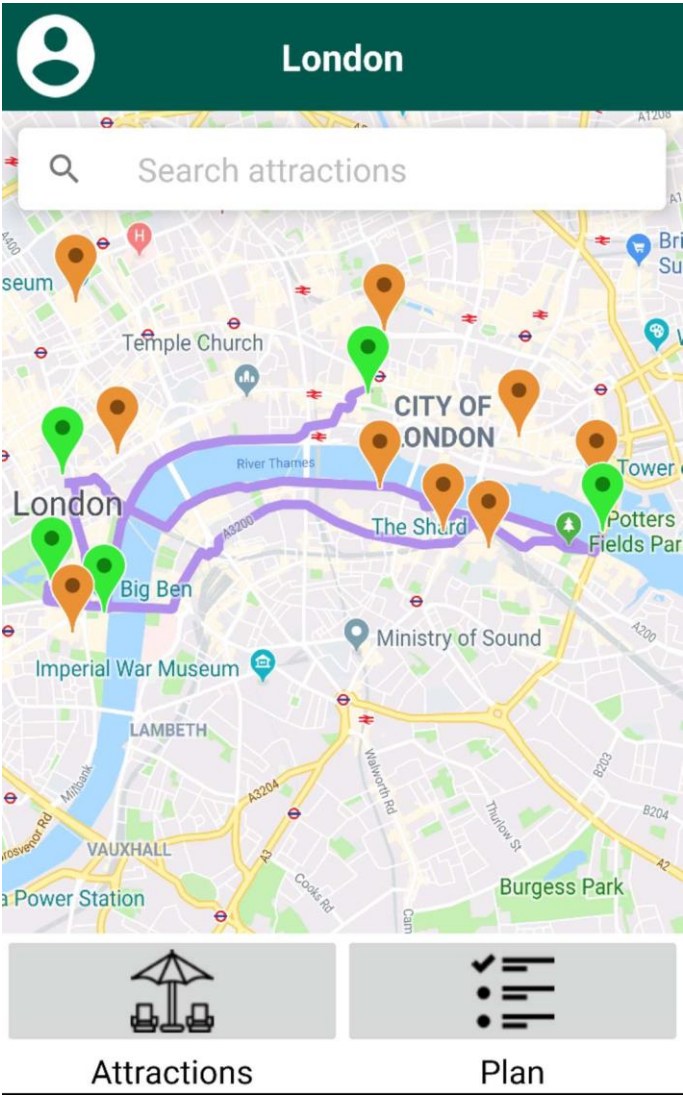
3. Plan list view



4. Menu



5. Route before and after optimization



GROUP PROJECT DECLARATION OF ACADEMIC INTEGRITY

NAME (Print)	Henan.Liu
STUDENT NUMBER	201376148
GROUP NAME	Team 22
MODULE TITLE/CODE	COMP 208 GROUP SOFTWARE PROJECT
TITLE OF WORK	EasyTrip

This form should be completed by one student on behalf of the group and appended to any piece of work that is submitted for summative assessment. Submission of the form by electronic means by a student constitutes their confirmation of the terms of the declaration.

Students should familiarise themselves with Section 9 of the Code of Practice on Assessment and Appendix L of the University's Code of Practice on Assessment which provide the definitions of academic malpractice and the policies and procedures that apply to the investigation of alleged incidents.

Students found to have committed academic malpractice are liable to receive a mark of zero for the assessment or the module concerned. Unfair and dishonest academic practice will attract more severe penalties, including possible suspension or termination of studies.

STUDENT DECLARATION

I confirm that the group has read and understood the University's Academic Integrity Policy.

I confirm that the group has acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that no member of the group has copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work.

I confirm that no member of the group has previously presented the work or part thereof for assessment for another University of Liverpool module.

I confirm that no member of the group has copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that no member of the group has incorporated into this assignment material that has been submitted by any other person in support of a successful application for a degree of this or any other University or degree awarding body.



SIGNATURE.....

DATE..... 9/5/2019