Design Specification

Zhixuan Xu 201377035

Hongyi Wu 201376889

Zhuo Wang 201376856

Xiaoyu Qin 201376441

Yifan Hu 201375824

Weiqi Chen 201375504

Department of Computer Science

15 March 2019

Author: Group 15

Monitor: Sebastian Coope

Reviewer: Rida Laraki

Module: COMP208

Lecturer: Michele Zito

Group Member Signature:

Contents

1 Summary of Proposal

The Travel Counsellor System is a web application consists of a database with travel preferences of registered customers, information on travel destinations and tourist attractions. The inspiration of the project is to combine the travelling recommendation function and navigation function into one system to provide an optimal solutions for short period travelling. The preference feature supported by a questionnaire to the registered users, which may involve questions of preferred weather, means of travel, preference of cultural landscapes or natural landscapes and lifestyles during the trip. There are multiple means for the users to acquire their personalized travelling route. The standard recommendation is solely based on the destination and starting point via user input, providing the shortest path available to visit as many tourist attractions as possible. The preference recommendation is based on the preferences of the users. The system will recommend several candidate travel destinations all around the world, and provides a suggestion route and days of travel. The advanced recommendation will take both the preference and actual travelling destination into account, potentially includes real-life situation like transportation fares and opening hours of tourist attractions. The final result of the project will be displayed on a user-friendly website and the travel route might be displayed by animation with recommendations of accommodations, restaurants and plane or train ticket booking by links to external website to provide convenience for the users. The ultimate goal of this project is to assist inexperienced travellers in handling every decision of planning a complete tour to ensure a pleasant and hassle-free trip experience.

2 Design

2.1 System Components and user views

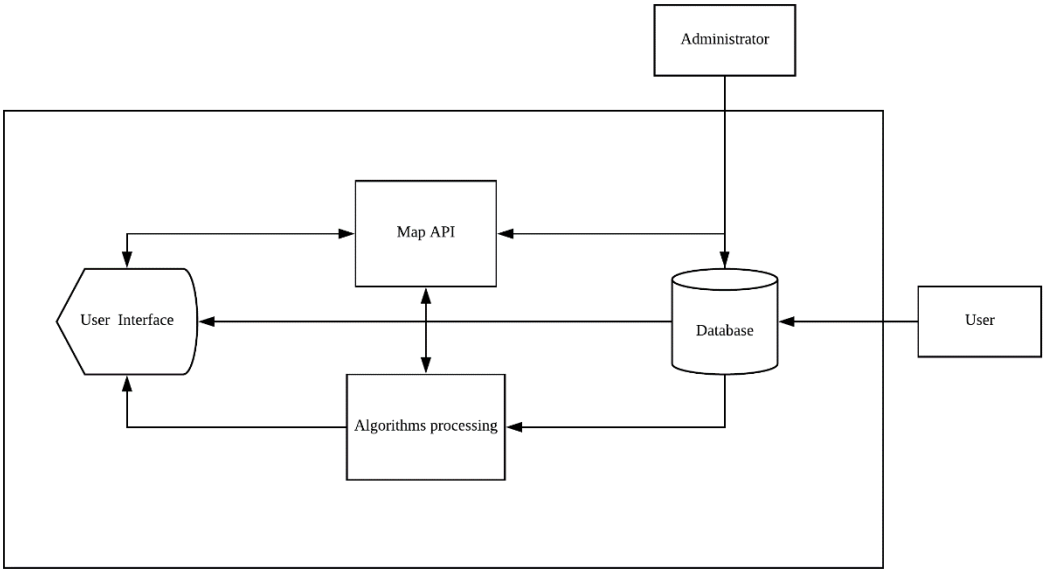


Figure 1 System Component Diagram

This system consists of user interface, map API, database and processing algorithm. Only the administrator can access and maintain these system components. The user chooses their travel destinations from the database, which stored the information of user preferences and history queries, basic information of travel destinations and the tourist attractions are labelled referring to user preference questionnaires to filter the destination for future recommendation. The processing algorithm combined the location information provided by the external Map API and the preference stored in the database to design the travel route. The result will be displayed to the user interface to the user.

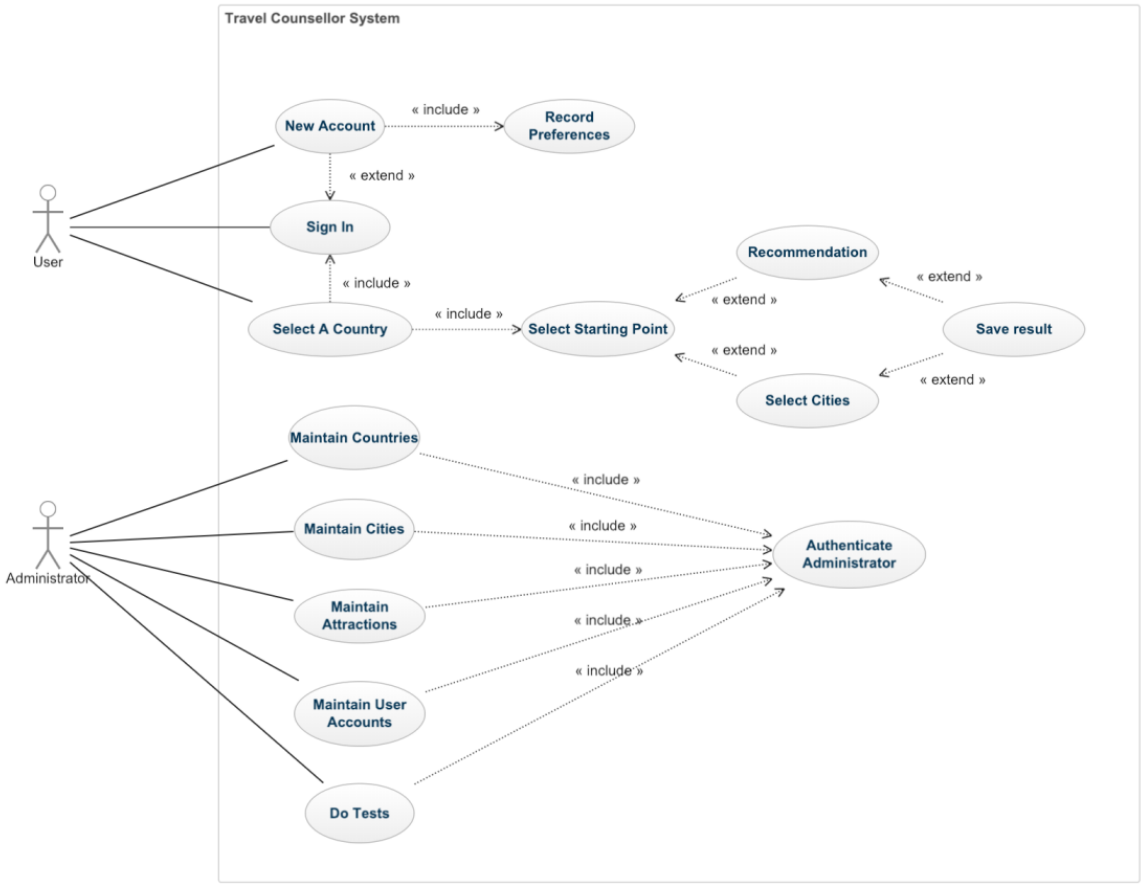


Figure 2 Use Case Diagram

ID	UC1
Name	New Account
Description	The user signs up a new account
Pre-condition	1. Travel Counsellor System in service (internet connected) 2. The database is not full
Event flow	The user clicks the Sign-Up button and set username and password
Post-condition	The account created
Includes	Use case 2 “Record Preference”
Extensions	
Triggers	User clicking the Sign-up button

ID	UC2
Name	Record Preference
Description	The user chooses their

	preferences
Pre-condition	1. Travel Counsellor System in service (internet connected) 2. The database is not full 3. A new account is created just now
Event flow	The user chooses their preference for travelling
Post-condition	The account preference recorded
Includes	
Extensions	
Triggers	User clicking the personalized preference button

ID	**UC3**
Name	Sign In
Description	The user signs in the system
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	The user signs in the system with username and password
Post-condition	Log in successfully or wrong password
Includes	
Extensions	User case 2 “Record Preference”
Triggers	User clicking the login button

ID	**UC4**
Name	Select A Country
Description	The user chooses a country to start
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	1. The user signs in successfully 2. The user chooses a country to travel
Post-condition	Country selected
Includes	User case 3 “Sign In”
Extensions	
Triggers	User clicking the countries button

ID	UC5
Name	Select Starting Point
Description	The user chooses a point to start
Pre-condition	The user chooses a country to travel
Event flow	1. The user signs in successfully 2. The user chooses a country to travel 3. The user chooses a point to start
Post-condition	Starting point selected
Includes	
Extensions	User case 6 “Recommendation” User case 7 “Select Cities”
Triggers	User clicking the starting point

ID	UC6
Name	Select Starting Point
Description	The user chooses a point to start
Pre-condition	The user chooses a country to travel
Event flow	1. The user signs in successfully 2. The user chooses a country to travel 3. The user chooses a point to start
Post-condition	Starting point selected
Includes	
Extensions	User case 6 “Recommendation” User case 7 “Select Cities”
Triggers	User clicking the starting point

ID	UC7
Name	Recommendation
Description	The user chooses the recommended cities from the system
Pre-condition	The user chooses a point to start
Event flow	1. The user signs in successfully 2. The user chooses a country to travel

	3. The user chooses a point to start 4. The user chooses the recommended cities from the system
Post-condition	The cities selected and the route presented
Includes	
Extensions	
Triggers	User clicking the recommendation button

ID	**UC8**
Name	Select Cities
Description	The user chooses one's preferred cities from the system
Pre-condition	The user chooses a point to start
Event flow	5. The user signs in successfully 6. The user chooses a country to travel 7. The user chooses a point to start 8. The user chooses one's preferred cities from the system
Post-condition	The cities selected and the route presented
Includes	
Extensions	
Triggers	User clicking the cities and Done button

ID	**UC9**
Name	Maintain Countries
Description	The administrator maintains (enter, update, and delete) data from countries
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	1. Include Use Case 14 "Authenticate Administrator"

	2. Maintain countries
Post-condition	
Includes	Use Case 14 “Authenticate Administrator”
Extensions	
Triggers	Maintain Countries requested

ID	**UC10**
Name	Maintain Cities
Description	The administrator maintains (enter, update, and delete) data from cities
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	3. Include Use Case 14 “Authenticate Administrator” 4. Maintain cities
Post-condition	
Includes	Use Case 14 “Authenticate Administrator”
Extensions	
Triggers	Maintain cities requested

ID	**UC11**
Name	Maintain Attractions
Description	The administrator maintains (enter, update, and delete) data from tourist attractions
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	5. Include Use Case 14 “Authenticate Administrator” 6. Maintain attractions
Post-condition	
Includes	Use Case 14 “Authenticate Administrator”
Extensions	
Triggers	Maintain attractions requested

ID	**UC12**

Name	Maintain User Account
Description	The administrator maintains (enter, update, and delete) data from user accounts
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	7. Include Use Case 14 "Authenticate Administrator" 8. Maintain user account
Post-condition	
Includes	Use Case 14 "Authenticate Administrator"
Extensions	
Triggers	Maintain user account requested

ID	UC13
Name	Do Test
Description	The administrator does a series of test
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	9. Include Use Case 14 "Authenticate Administrator" 10. Do test
Post-condition	
Includes	Use Case 14 "Authenticate Administrator"
Extensions	
Triggers	Test requested

ID	UC14
Name	Authenticate Administrator
Description	The administrator proves the identity to the system
Pre-condition	Travel Counsellor System in service (internet connected)
Event flow	The administrator enters a username and passcode and is authenticated
Post-condition	Permitted to the system if the identity is administrator

Includes	
Extensions	
Triggers	There is someone who intends to do test or maintain data

Figure 3 Use Case Description

There are two actors exist in the system: Users and administrator, which have access to different function in this system. The user can create a new account and record travel preferences, and then select a country and a starting point. The user either can select the city manually or opts to receive recommendations if signed in; the result should be provided to the user after the operation.

The administrator can maintain each database in the system and test to evaluate the system performance after authenticated by the system to log in.

Data	Access Type	User	Administrator
Countries	**Maintain**		**X**
	Search	**X**	**X**
	Report		**X**
Cities	**Maintain**		**X**
	Search	**X**	**X**
	Report		**X**
Attractions	**Maintain**		**X**
	Search	**X**	**X**
	Report		**X**
Preference	**Maintain**	**X(registered)**	**X**
	Search		**X**
	Report		**X**

Figure 4 User Views

The whole system involves two different user views: users and administrators. The user has access to read destinations from the database to provide queries and to write the preferences into the database. The administrator can maintain the database to update information, report any alterations to the data and provide searches for testing. The preference feature should be available to users after the user is registers and authenticated.

2.2 Data Structure Used in the System

For the data structures to be used by the system, as the main language of the program is chosen to be Python, the data structure of dataframe, supplied by the powerful data analysis library—pandas, is chosen to be used as the basic data structure. Dataframe is a two-

dimensional size-mutable, potentially heterogeneous tabular data structure, whose arithmetic operations align on both row and column labels. It can be thought of as a dict-like container for Series objects, which means that there is no need to consider the size or space it occupies as the size of it is flexible. Besides, as the Pandas data analysis library is considered to be the main tool to manipulate data and tackle the issue of using AI algorithm to extrapolate the optimization of tourism attraction sets, dataframe is a more convenient data structure than any other data structures to realize the functions like traversing, sorting, indexing, deleting and updating. For the data of user registrations and preferences, besides tourism attractions, dataframe is applied as well out of integrity.

In terms of tourism attractions information data stored in the database, there are five tables to store them. The first table of first level is a country table, which has attributes {Country Name, Continent, Cities, Nearby Country, Official Website, Travel Label}. Besides, in term of the specification of cities, the second table of the second level is city is created, which has attributes selected as {City name, Latitude, Longitude, Attractions, Accommodation, Restaurant, Transport Information, Nearby cities, Population}. For the specification of Attractions, Accommodation and Restaurant, three tables of third level are created correspondingly. For Attractions table, it has the attributes of {Attraction name, ID of attraction, Latitude, Longitude, Address, Category, Website, Abstract Stay Time}. For the table of Accommodation, it has the attributes of {Name, ID, Latitude, Longitude, Address, Price Level, Category, Contact Number, Website, Rate}. For the table of Restaurant, it has the attributes of {Name, ID, Latitude, Longitude, Address, Category, Price Level, Rate, Contact number}. As for the user account table, the attributes of it are {Name, Email Address, Password, ID, Preference, City, Accommodations, History}. The history attribute of user account is specified as an individual table with the attributes of {Time, Duration, Route, Result URL}.

2.3 Algorithm to manipulate the data structures

1. Retrieval and Screen data from Data Frame

The pandas method—DataFrame.ix[row, column] is used to retrieve the especial data the program need. The row is the default index of each row in the data frame, and the column is the label of each column. For example, if the program needs the data of Hotel in London, for the city data frame, the command '*city.ix[[1],['Hotel']]* ' will return the needed value and 1 is the index of the city London.

To screen data from the data frame according to some restriction, the pandas method—*DataFrame[condition][[column]]* could do this. Condition argument in this method is the Boolean expression about the value of the data frame, and the column argument in the method declares what columns value will be returned. For example, if the program needs all the cities which have airports, for the city data frame, the command ' *city[city['airport']!=None][['cityname']]* ' will return all the city names which have airports.

2. Update the value in the Data Frame

If the program wants to update one value in the data frame, the first step is to retrieve this value. And the second step is to assign the new value to the retrieved data. Therefore, the update method will be ' *DataFrame.ix[row, column]=Newvalue* '.

3. Delete data in the Data Frame

Use the pandas method—*drop([rows or columns], axis=0 or 1)* to delete the especial row or column in the data frame. Axis in the method indicates that the command is to delete rows or columns(0 means row, one means column). For example, if the f=program need to delete all the data about Liverpool, for the city data frame, the command ' *city.drop([city[cityname]==Liverpool], axis=0)* ' will delete the Liverpool data.

4. Insert new data into the Data Frame

If the program needs to insert a new row to the data frame, the pandas function—*concat([DataFrame, newRow])* can do this. For example, if the program needs to insert a new city—Liverpool to the city data frame, the command ‘ *concat([city, LiverpoolDic])* ’ will return a new data frame with Liverpool, and the LiverpoolDic is a dictionary with the value of Liverpool.

5. Traverse the data

In some situation, the program may need to traverse all the values of some columns, the pandas method—*itertuples()* can traverse all the values in a shorter time than retrieving them one by one.

6. Sort

If the program needs to sort all the records in the data frame by a certain value, the pandas method—*sort_values(by=[column])* can do this. This method could return a new data frame which sorted by the value of column argument in the expression, and it can sort by more than one column. For example, if the program needs the data frame sort by its population, for the city data frame, the command ‘ *city.sort_values(by=[‘population’])* ’ will return the sorted data frame.

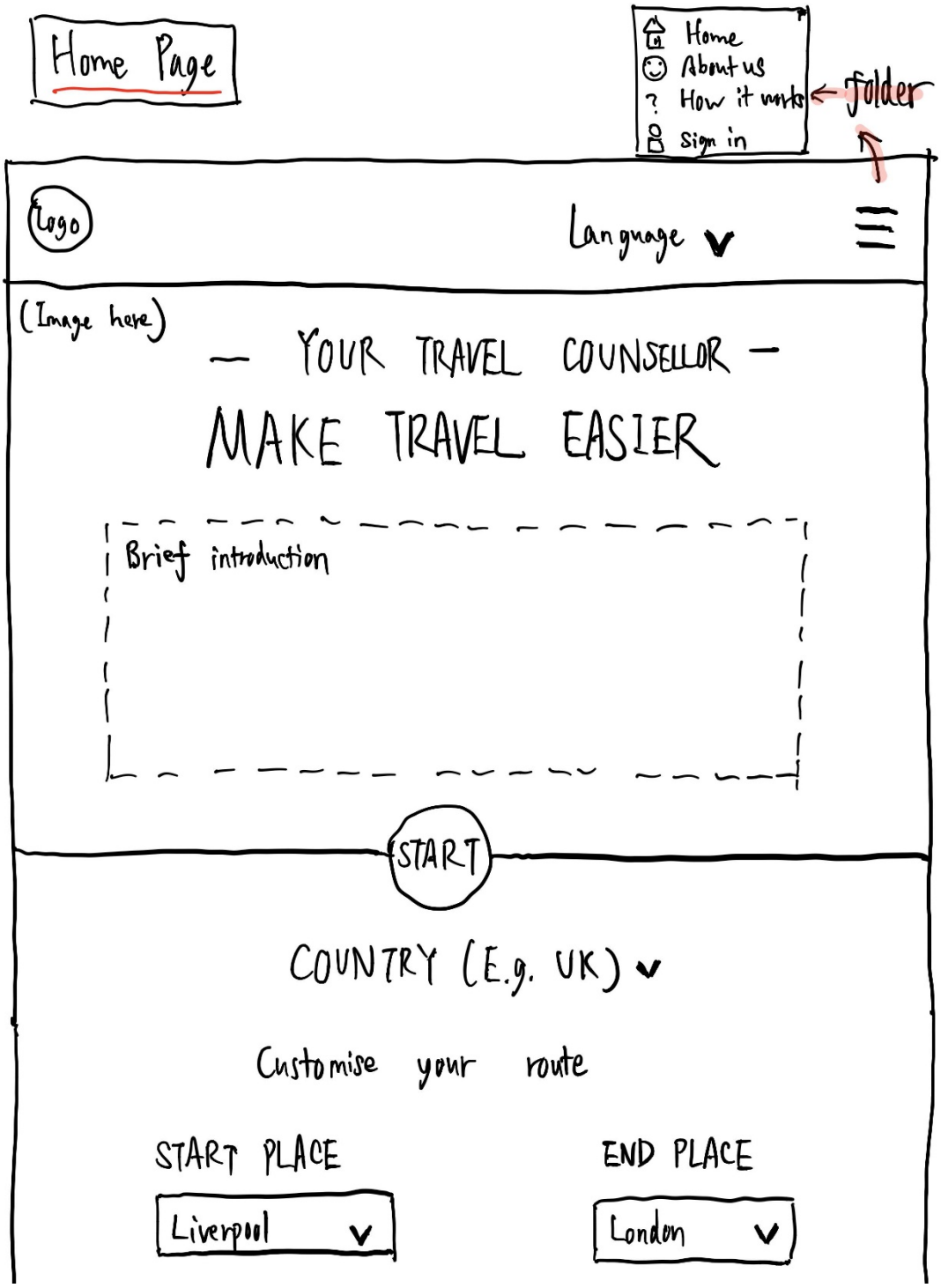
7. Null value handling

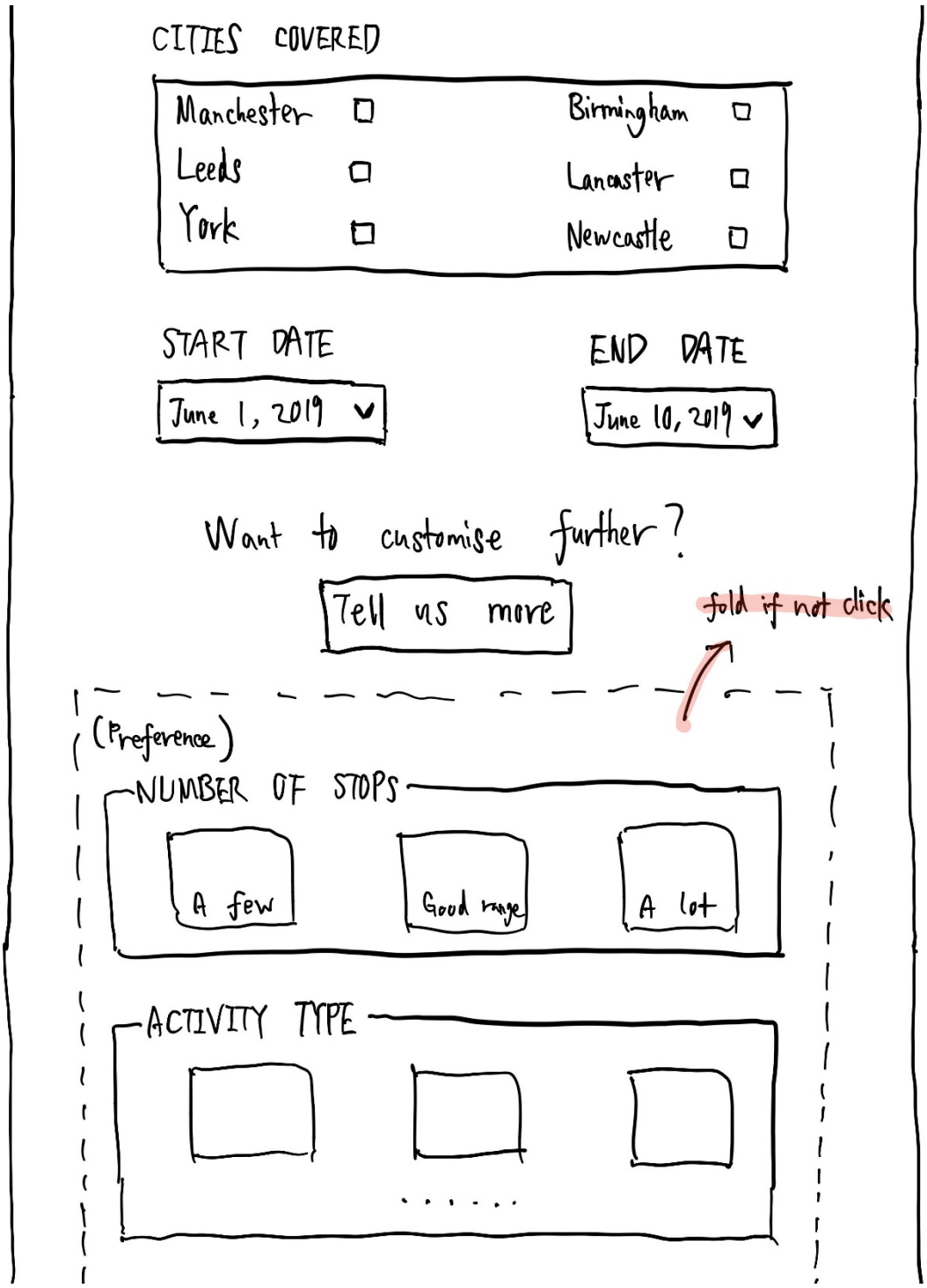
If there were some NaN values in the data frame when the source data was inputted from the database, the program needs to handle these null values to prevent the program from running failure. Use the pandas function—*DataFrame.dropna()* to handle this problem. This method will delete all the records in the data frame that have null values.

2.4 Design of intend interfaces

2.4.1. Design sketches

In the travel counselor system, webpages will be the user interface. The drawings of the design webpages are shown below.


Home Page
Home
About us
How it works
Sign in
folder
Logo
Language
(Image here)
— YOUR TRAVEL COUNSELLOR —
MAKE TRAVEL EASIER
Brief introduction
START
COUNTRY (E.g. UK)
Customise your route
START PLACE
Liverpool
END PLACE
London


CITIES COVERED
Manchester
Birmingham
Leeds
Lancaster
York
Newcastle
START DATE
June 1, 2019
END DATE
June 10, 2019
Want to customise further?
Tell us more
fold if not click
(Preference)
NUMBER OF STOPS
A few
Good range
A lot
ACTIVITY TYPE

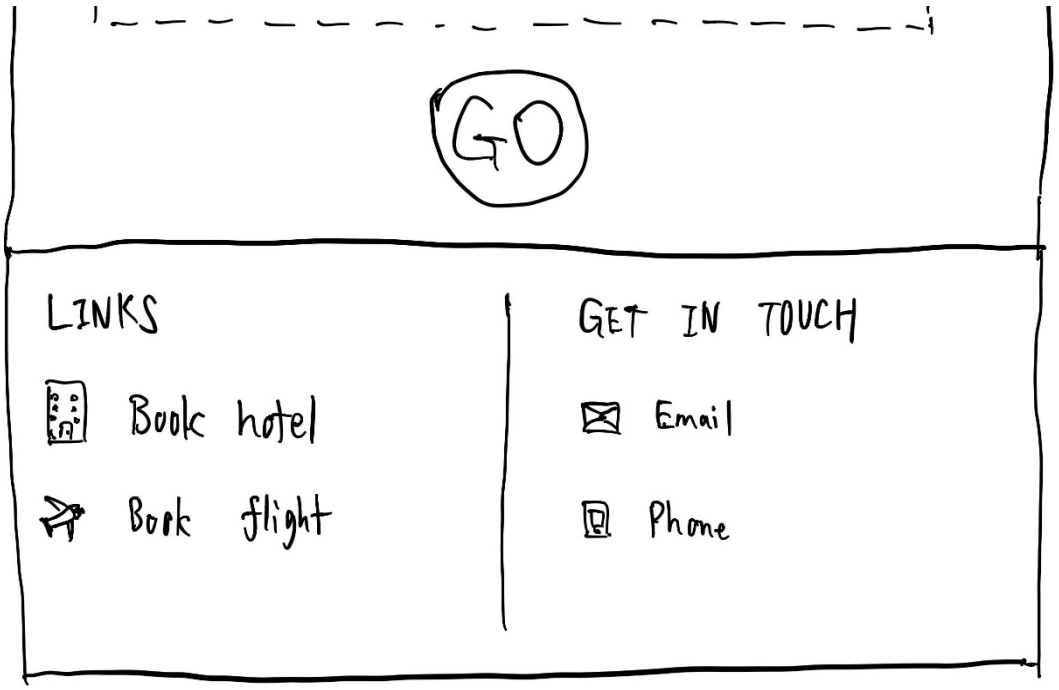


Figure 5 Sketch of the Index Page

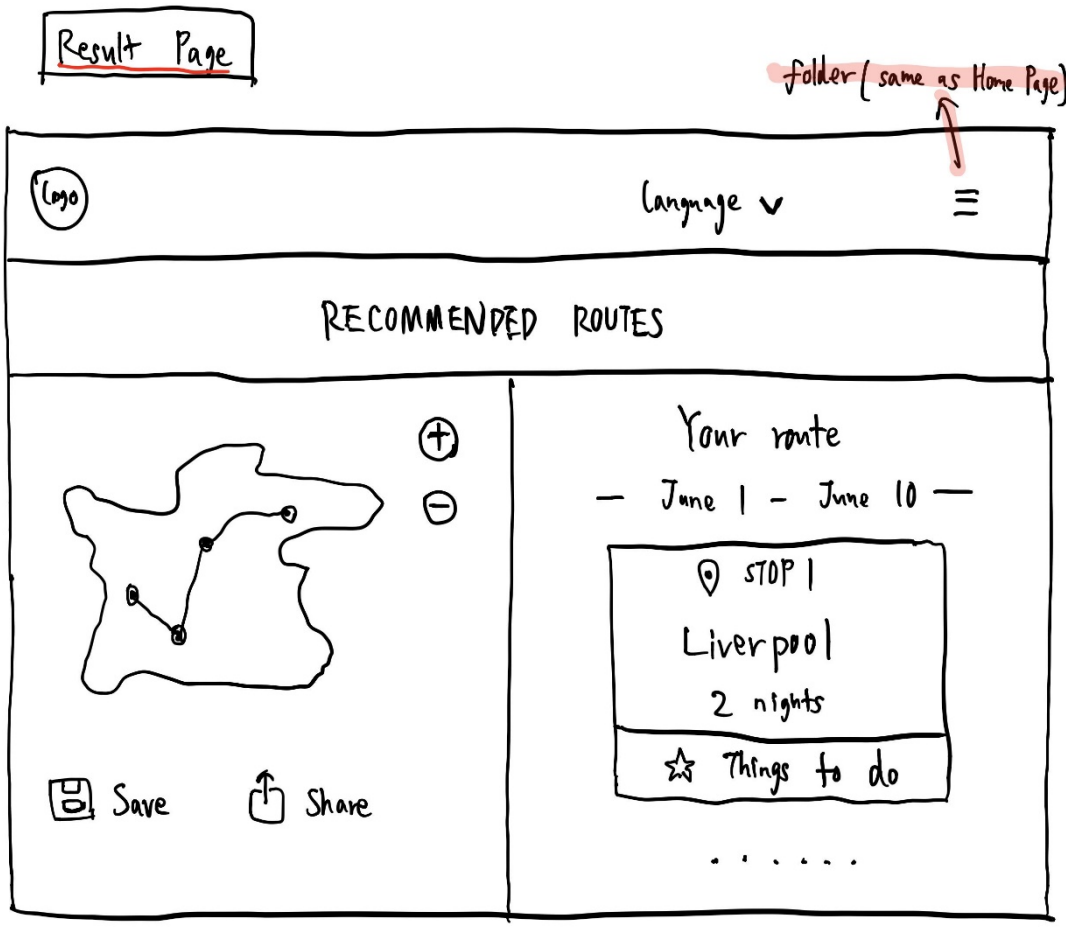


Figure 6 Sketch of the Result page

The user will see two main webpages that are the home page and result in a page during operation. The home page consists of 4 parts which are a navigation bar, title, selection, and links. Navigation bar provides some basic functionality such as change language, return to the home page, and register & login. A folder is applied to keep these basic functions in order. The navigation bar is always on the top of webpages while page scrolling. Therefore, the user can easily change the language or return to the home page at any time. The second part shows the title and the use of our product. Some brief introduction will guide the user, which enhances the usability of the website. The third part is the core of the home page that gives the user some selections. The route recommendation will be given based on these selections range from place to date, as well as preference selected. The last part of the home page links, which contains some useful hyperlinks. These hyperlinks could jump to an external booking website and the team contact information for those who want to cooperate or send feedback to our team.

The result page consists of two parts which are navigation bar and recommended route. The navigation bar is the same as the home page. The left half of the recommended route part is Google Map while the right half shows personalized routes. Each stop (city) with an assigned number of days are shown clearly in the route. The user could click "Things to do" to see the information on tourist attractions and activities of this city. Moreover, the user could save the route to their account for later browsing.

2.4.2. Major scenarios of user-system interactions

Two major scenarios of interactions will be demonstrated in this report. The first scenario is registration and the second scenario is "login -> make selections -> see the recommendations." Flow charts of these two scenarios are shown below.

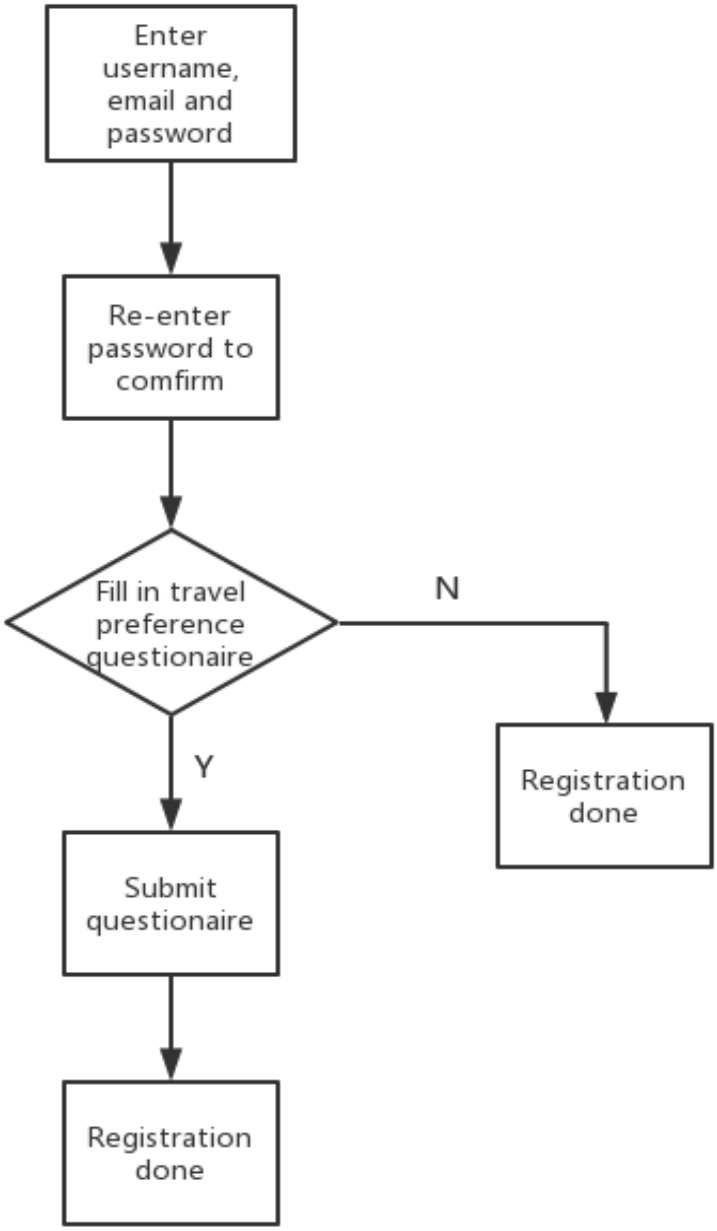


Figure 7 Flow Chart of Registration

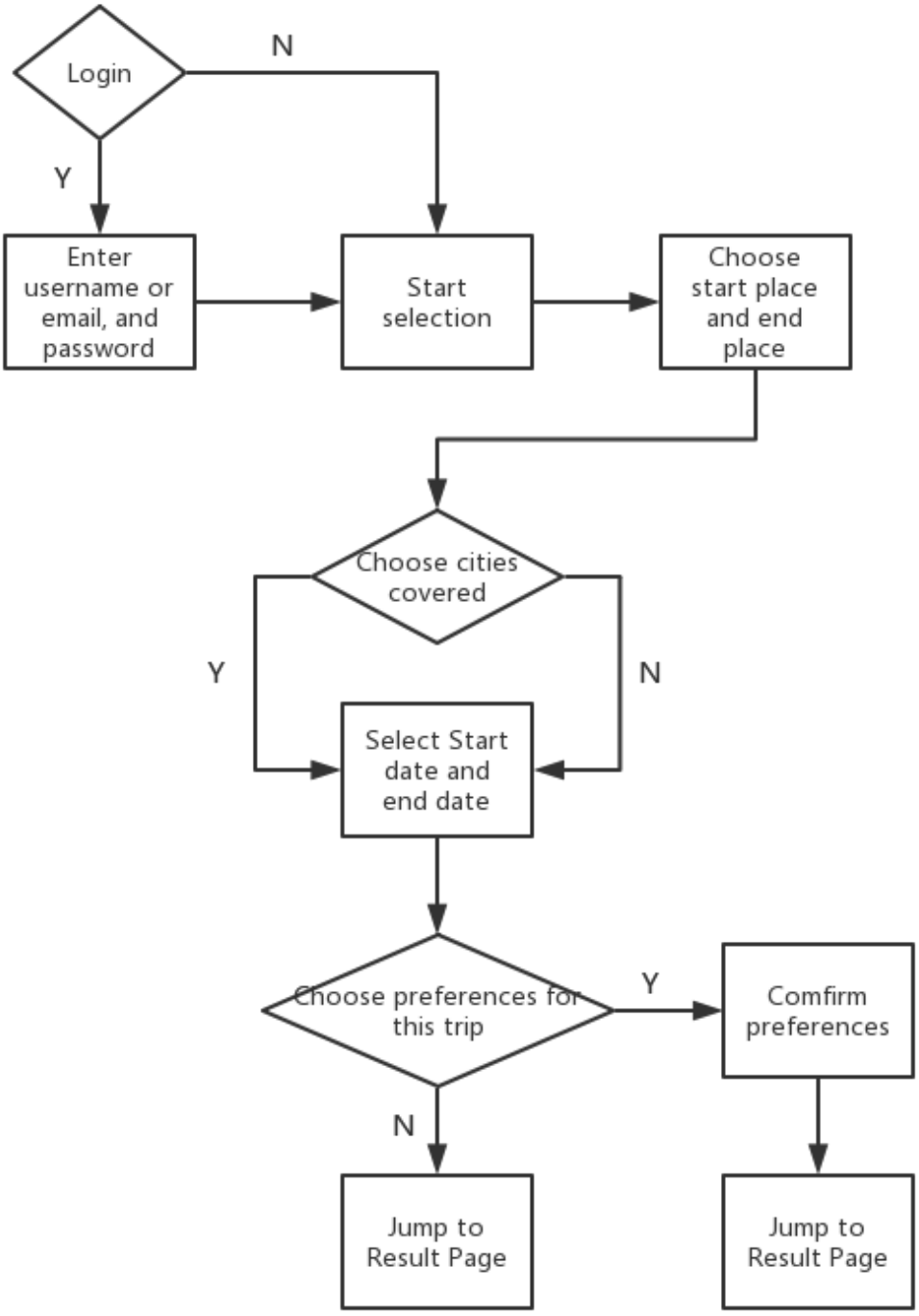


Figure 8 Flow Chart of the System Process

During registration, the user needs to enter a username, email, and password (two times for confirmation), and then the user may decide whether to do the travelling preference questionnaire or not. If the user does not intend to do the questionnaire immediately, they

could do it at another time. Finally, registration is successful, and the user will log in to the system automatically.

The second scenario simulates the whole process of obtaining the recommended route. Firstly, the user will choose to login or not since the user can be recommended route without login. However, more features will be provided such as route saving and personal preference analysis. Then, the user clicks the "start" button to start making selections. The user is required to choose the start and end cities. After that, the user may tick cities that would like to be covered. Next, the user needs to enter the intended start and end date of travel. Then, the user may choose preferences of this trip, which is an option. If the user does not choose preferences here, the system will give a recommendation based on the traveling preferenwaquestionnaire that user was done before. Finally, user could see recommended route on result webpage.

2.5 Evaluation of the System

Functional and non-functional requirements are basic metrics of evaluation of system. The basic functionality of the system should first be completed while evaluating this system. Black box testing and white-box testing are applied specifically on these two parts. Black box testing is to test the software interface and the function for semantic errors, termination and initialization errors. The white box testing is to exercise all program statement. The system should pass all the black box and white box tests to prevent fatal errors occurred during the demonstration. The Cyclomatic complexity equals to the number of conditions in a program; it is applied to describe the effect of optimization. Stress testing is adopted to evaluate system performance when beyond its maximum design load to evaluate the reliability and stability of the system. Integration and cluster testing tests complete systems or subsystems composed of integrated components, which aims to evaluate system reparability. For evaluating the usability of the system, scenario-based testing is an appropriate method based on use cases. These measures ensure the system to be stable and reliable, which satisfies the basic requirements to realize the idea of the project.

Apart from those basic functionalities, the accuracy of route recommendation and preference filters is the most vital factor of the system performance. This requires human resources to test and actual application in real life situation to evaluate whether the functionality achieves the original aspirations of this project. Some volunteers with real traveling intentions could provide valuable testing data and feedbacks for the accuracy review. Feedback questionnaires will focus on user experience, the feasibility of travel plans and the correlation between preferences and actual travel plans provided by the system. The correlation maybe hard to quantify and is subjective due to the actual user preferences and travel destinations, the development team will attempt to adjust the weighting in machine learning process according to tester's verbal feedback to achieve the ideal balance presumed in the design stage.

2.6 Pseudo-code for the key methods

2.6.1. Ant Colony Optimization

Considering about solving shortest path solution, prim algorithm, Kruskal algorithm, and Dijkstra algorithm are common use algorithm. However, these algorithms cannot be used in the project because each of mentioned algorithm has a distinct weakness. For the prim algorithm, it selects the shortest path between two points iteratively, adds shortest path to set and finally get a path which one point in the graph can get to any other point by selected path. As for the Kruskal algorithm, it is similar to the prim algorithm. Dijkstra algorithm is little different with prim and Kruskal algorithm, it first accepts one state as an original state, and find the shortest path to other states. Although these algorithms are specialized in solving specific shortest path problem, they cannot solve project problems because project aims to find the shortest path, which through all selected cities and these above-mentioned algorithms cannot guarantee that shortest path will go through all selected cities.

Traveling salesman problem is often associated with shortest path problem; traveling salesman problem describes that for a given city set and distance between each city, finds the shortest path from origin city, goes by other given cities and finally goes back to origin city. Traveling salesman problem is similar with the method of finding the shortest path in the project. However, traveling salesman problem is intended for solving the shortest circle path, which means the origin point must be the same with the destination. Therefore, it is essential and important to mine algorithm which traveling salesman problem inherits.

It is essential to mention the ant colony optimization algorithm because the ant colony optimization algorithm solves many kinds of problem, which includes scheduling problem, vehicle routing problem, and traveling salesman problem. Colony optimization algorithm

modeled on the action of an ant colony. Ant locates optimal solutions by moving through a parameter space representing all possible solutions. Ant releases pheromones directing each other to resources while exploring path. Another ant simulates this action so that in later iterations of this method can find optimal solutions of the shortest path, which is a global optimization algorithm.

The second figure can explain kernel characteristic of ant colony optimization algorithm. “1" represents an original path, which connects each point. ”2” represents all possible path connects each point, which corresponds to ant colony optimization algorithm that one ant releases pheromones on the searching path while other ants search pheromones. "3" and "4" represents that after many times the iteration method can get optimization solution of shortest paths

ANTCOLONYOPTIMIZATION(*S, E, G, M*)

▷input: Location of start city *S*, location of end city *E*, *G*(*V*, *E*) of all the cities selected and Set *M* of ants.

▷Output: Path route *P* of the result.

```
1     for i ← 1 to |C| do
2             distmat[i][j] ← distance between city i and city j
3     procedure SETINITINFORMATION
4         for ∀k ∈ M do
5                 Let r_k1 be the starting city for ant k
6                 J_k(r_k1) ← V – {r_k1}
7                 /* the set to be visited for ant k in city r */
8                 r_k ← r_k1
9                 /* city the ant k located in */
10    end procedure

11    procedure ROUTES
12        for i ← 1 to |V| -1 do
13            for ∀k ∈ M do
```

```
14                          Select next city s_k from the formula mentioned
15                          add edge(r^k, s^k) to Tour_k
16 end procedure

17      procedure UPDATE
18              Compute L_k ∀k ∈ M
19              /* the length of tour of ant k */
20              Update τ_{r,s} from the formula mentioned
21      end procedure

22      procedure MAIN
23              for ∀edge(r,s) ∈ E do
24                      τ_{r,s} ← τ_0
25                      η_{r,s} ← 1/distmat[i][j]
26              while Not End_Condition do
27                      SETINITINFORMATION
28                      ROUTES
29                      UPDATE
30      end procedure
```

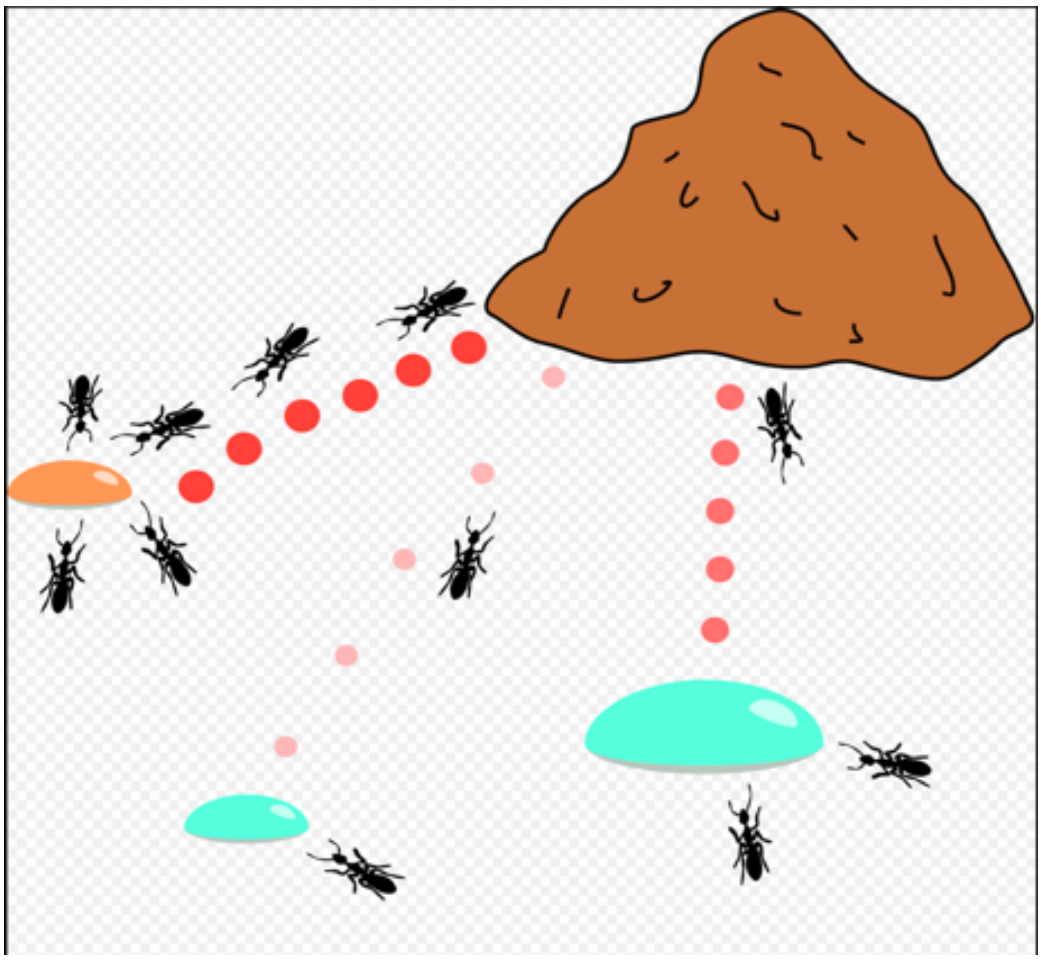


Figure 9 Illustration of the idea ant colony optimization

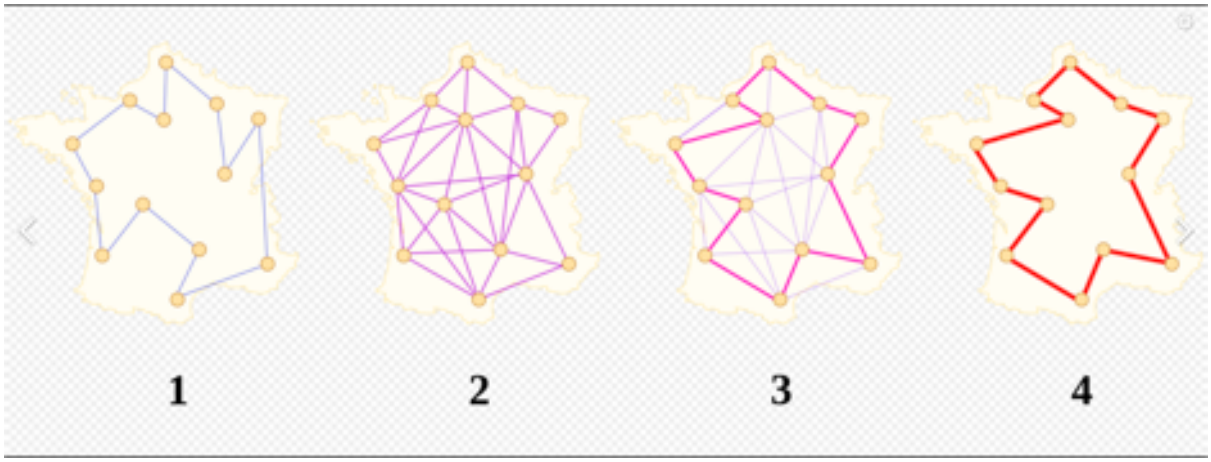


Figure 10 Illustration of the route selection

2.6.2. City Sort

City sort and recommend route function are core parts of the system. Each country has many cities, which has a different characteristic and feature. For example, Edinburgh is equipped with enriching tourist attractions of nature and cultural, Liverpool is a coastal city which is famous for football culture. Moreover, one city can be equipped with different characteristics, London is the city, which possesses abundant nature and cultural tourist attractions, besides, London is famous for entertainment. Our system will sort cities based on a different criterion, for instance, weather, temperature, human landscape, natural landscape, and others. For each city, the system will mark for each criterion based on data or comment of main travel platform. The higher comment of criterion is, the higher mark of criterion will be marked. For the user, the system will ask the user to complete questionnaire, includes travel preference of user and mark for each criterion based on the answer of questionnaire.

After getting the mark of each criterion of cities, the system will calculate similarity to make judgment whether characteristics of city meet with the preference of the user. The method of calculating similarity is calculating Euler distance between the user vector and city vector. For each city, it has n marked features, and each marked feature is saved to an array. For each user, it also has n marked features, and each marked feature is saved to an array. For first feature[1,1] represents first feature mark of user, C[1,1] represents first feature mark of city, calculate (U[1,1]-C[1,1])^2.Using this method to calculate other features by using for loop, accumulate loss value and calculate the square root of the sum of loss value. The lower value of similarity, the system will sort cities priority. The system will show a list of cities which based on similarity value to let the user select appropriate city.

CITYSORT(*U, C)*

▷Input: Preference of the user U which stores an unidimensional array, and C[i][j] array stores the degree of factors for all the cities in a country.

▷Output: Sorted list S of cities

```
1       difference[] = 0
2       for i ← 1 to |C| do
3                   sum ← 0
4                   for j ← 1 to |C[1]| do
5                           sum ← sum + (U[j] – C[i][j])^2
6                   sum ← sum^1/2
7                   /* L2 norm of the difference between user preference
8                   and the city */
```

```
9                   difference[i] ← sum

10      procedure MERGESORT(difference[0..n-1])
11      if n > 1 then begin
12              copy difference[0..⌊n/2⌋-1] to B[0..⌊n/2⌋-1]
13      copy difference[⌊n/2⌋..n-1] to C[0..⌈n/2⌉-1]
14      MERGESORT(B[0..⌊n/2⌋-1])
15      MERGESORT(C[0..⌈n/2⌉-1])
16      MERGE(B, C, difference)
17 end procedure

18      procedure MERGE(B[0..p-1], C[0..q-1], A[0..p+q-1])
19              Set i ← 0, j ← 0, k ← 0
20              while i < p and j < q do
21              begin
22                      if B[i] <= C[j] then set A[k]=B[i] and increase i
23                      else set A[k] ← C[j] and increase j
24              k ← k+1
25              end
26              if I ← p then copy C[j..q-1] to A[k..p+q-1]
27              else copy B[i..p-1] to A[k..p+q-1]
```

2.6.3. City Recommendation

For recommend route function, the system will recommend an appropriate route based on two important criteria. One is a user preference, and the other is the distance between different cities. For recommend the route, the system will confirm the starting point which user selects; then the system will list first 2*n cities, the value of n is decided by the system. To appropriate n, the system will make lots of test until getting an ideal result. For first 2*n cities

of the list, the system will calculate the distance between origin place and cities, a select city which has the shortest path to origin place. After selecting the first city, the system will select the second city from the list, which removed previously selected city, algorithm based on distance to previously selected city. Based on this method, the system will conclude about recommend route.

CITYRECOMMENDATION (*N, L, S, P)*

▷Input: *N*: Number of cities N selected. *L*: Location of all the cities. *S*: sorted list of cities according to user preference. P: Starting point of the route

▷Output: R: route path

1 *city* ← Select first *N**2 cities from array *S*
2 *present* ← P
3 Add *present* to *R*
4 **while** *city* ≠ ∅
5 **do**
6 Remove the closest city *c* with present
Add *c* to *R*
7 Return the set *R*

3 Review against Plan

3.1 Gantt chart

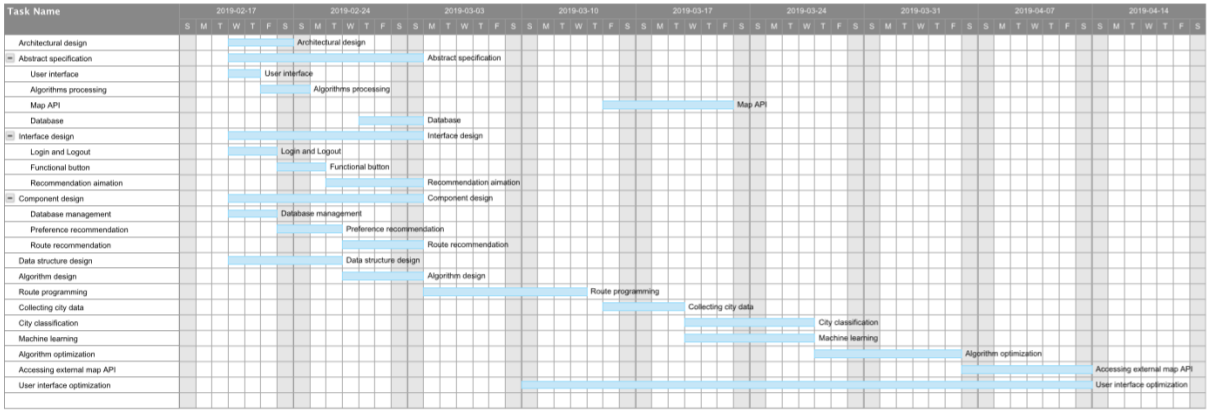


Figure 11 Gantt chart (Updated)

3.2 Current Progress

The current progress of this project is mostly proceeded according to the initial plan. The architectural design was completed from February 20th to 23rd. The Abstract specification was scheduled between 20th. Feb to 3rd.Mar. The user interface was specified on 18th.Feb, Algorithms were processed on 19th. Feb. Due to the complexity of Google API, the specification of Map API was falling behind the initial plan and the teams were trying to master the application of the API and seeking for potential simplified alternatives. The database specification was finished ahead of plan.

The Interface design was initiated on 25th.Feb, which was behind of plan and finished in 10th. Mar, which includes login interface and functional buttons. Recommendation animations has not designed yet due to the difficulty of this section required more time. Component design was managed during 3rd.Mar and 10th.Mar.Three components were design separately in that time span. Data structure and algorithm design were started on 10th. Mar and finished before this document was submitted.

Due to the difficulty of acquiring existed city data and collecting data for web crawling. This part of design process would be postponed after the submission of this document.

3.3 Outlook of the Implementation Stage

In the implementation stage, Queries of countries, cities and tourist attractions will be completed. Tables for history and user account will also be built. User interface with basic login and query action will be fulfilled in this stage. Main algorithms and methods: ant colony optimization for route design, Citysort function for sorting and city recommendation for recommendation function will also be realised in this stage. The system will be designed in Python with Pandas library, the database will be maintained via the SQL interface in Pandas library. The website will be designed by HTML and CSS and JavaScript will also be applied in this website application in this particular stage.

4 References

[Websites]

[1] http://softwaretestingfundamentals.com/black-box-testing/

[2] http://softwaretestingfundamentals.com/white-box-testing/

[3] https://www.guru99.com/stress-testing-tutorial.html

[4] http://softwaretestingfundamentals.com/integration-testing/

[5] https://www.tutorialspoint.com/software_testing_dictionary/cyclomatic_complexity.htm

[6] http://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html

[Book]

[7] Ian SommerVille, Software Engineering 10th ed., Pearson, Harlow, Essex, England, 2016

[8] Connolly and Begg “Database Systems”

[Lecture Slides]

[9] Sebastian Coope “Software Engineering I,” https://vital.liv.ac.uk/, February 2018

[Online Conference]

[10] Russell, R. Andrew. "Ant trails-an example for robots to follow?." Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. Vol. 4. IEEE, 1999.

[11] Y. C. Liang and A. E. Smith, "An ant colony optimization algorithm for the redundancy allocation problem (RAP)," IEEE Transactions on Reliability, vol.53, no.3, pp.417-423, 2004.