Table of Content

Decision Supporting System for Subway Routes

1. Background

The metro system in London has contributed significantly to the transportation in this crowded city as it undertook more than 31 million passengers in 2015. Whereas, without an efficient decision supporting system, customers found confused or sometimes lost in the intricate underground network which contains numerous routes. This situation can be worse when a part of stations have not yet been covered by internet.

2. Introduction

To solve the problem, this report will introduce a beta version of decision supporting system for customers to use in subway stations in order to find their way. The program was constructed on three spreadsheets which have functions of presenting the general information, calculating the shortest path and holding the data. The main algorithms for finding the shortest path and drawing the instruction map were implemented in VBA. In this report, it will firstly describe the data preparation and algorithm that has been used to give a general knowledge about the project. After that, an example solution will be mentioned for explanation purpose. Then, the detailed instructions about how to use its function and the three spreadsheets will be demonstrated in detail. Finally, it will extend to a little further to the potential limitations and assumption of the spreadsheet in real-time setting.

3. Data Preparation

3.1 The Original Data Description

To start with, it will firstly present the data that will be used in this project. As a beta version, this program contains only three metro services, Line 1, Line 2, and Line 3 which run three times a day to connect five stations (Station A, Station B, ..., Station E). The running time of these three lines are shown as below,

Subway Routes

Line1	**Station**	**First Run**	**Second Run**	**Third Run**
	A	08:00	11:15	16:00
	C	08:45	12:00	16:45
	E	10:00	13:15	18:00
Line2	**Station**	**First Run**	**Second Run**	**Third Run**
	E	08:30	11:30	16:30
	D	09:15	12:15	17:15
	B	10:30	13:30	18:30
Line3	**Station**	**First Run**	**Second Run**	**Third Run**
	C	09:10	12:10	17:10
	B	10:00	13:00	18:00
	A	11:00	14:00	19:00

.

3.2 Data Processing

Expect for the three lines that are shown above, customers are also able to transfer into different lines in stations, for example, those who want to go to Station B from Station A at 08:00, they can take line 1 to Station C and wait for the departure of line 2 by which they can reach Station B at 10:00. Taking all three lines and waiting periods into consideration, the optional shortest paths are listed as below,

Table 1: Optional Shortest Paths List

Optional Shortest Pathes

Start Station	Start Time											
A	08:00	Line1 =>	C(08:45)	Line1 =>	E(10:00)	Waiting =>	E(11:30)	Line2 =>	D(12:15)			
				Waiting =>	C(09:10)	Line3 =>	B(10:00)					
	11:15	Line1 =>	C(12:00)	Line1 =>	E(13:15)	Waiting =>	E(16:30)	Line2 =>	D(17:15)			
				Waiting =>	C(12:10)	Line3 =>	B(13:00)					
	16:00	Line1 =>	C(16:45)	Line1 =>	E(18:00)							
				Waiting =>	C(17:10)	Line3 =>	B(18:00)					
B	10:00	Line3 =>	A(11:00)	Waiting =>	A(11:15)	Line1 =>	C(12:00)	Line1 =>	E(13:15)	Waiting =>	E(16:30)	Line2 => D(17:15)
	13:00	Line3 =>	A(14:00)	Waiting =>	A(16:00)	Line1 =>	C(16:45)	Line1 =>	E(18:00)			
	18:00	Line3 =>	A(19:00)									
C	08:45	Line1 =>	E(10:00)	Waiting =>	E(11:30)	Line2 =>	D(12:15)	Line2 =>	B(13:30)			
	09:10	Line3 =>	B(10:00)	Line3 =>	A(11:00)							
	12:00	Line1 =>	E(13:15)	Waiting =>	E(16:30)	Line2 =>	D(17:15)	Line2 =>	B(18:30)			
	12:10	Line3 =>	B(10:00)	Line3 =>	A(11:00)							
	16:45	Line1 =>	E(18:00)									
	17:10	Line3 =>	B(18:00)	Line3 =>	A(19:00)							
D	09:15	Line2 =>	B(10:30)	Waiting =>	B(13:00)	Line3 =>	A(14:00)					
	12:15	Line2 =>	B(13:30)	Waiting =>	B(18:00)	Line3 =>	A(19:00)					
	17:15	Line2 =>	B(18:30)									
E	08:30	Line2 =>	D(9:15)	Line2 =>	B(10:30)	Waiting =>	B(13:00)	Line3 =>	A(14:00)	Waiting =>	A(16:00)	Line1 => C(16:45)
	16:30	Line2 =>	D(17:15)	Line2 =>	B(18:30)							

. This table provides customers with an overall view of all the branches of subway services. Moreover, in order to use this data in algorithm with VBA and programmatically draw the instruction scatter plot, all the data has been further processed and stored in the spreadsheet 'Backend_data'. The five letters that represent the stations have been replaced by numbers from 1 to 5 for the purpose of being demonstrating in the plot.

4. Main Algorithm

The main algorithm that has been implemented to find the shortest path is called "FastestSubwayPath". This algorithm contains generally three steps which can be concluded as preparation, route calculation and output. In the preparation stage, the program firstly reads data from "Backend_data" sheet and stores arrival information as vertices and connection information as arcs. It then sets initial values for all the vertices and finds the original one and redefine its values.

In the calculation period, the algorithm firstly traverses all the vertices and arcs to finds every possible path and redefine the cost of reaching that destination. This process is restricted by a vertex attribute, permanent, which indicates whether the current vertex and its predecessor have been used. By doing that, the process of exploring new paths will not be repeated. Then it traverses all the vertices again and chooses the vertex with minimal cost from those that have not been used and sets it as used.

In the third step, where the result is outputted, it uses a loop to extract the vertices from the destination to the original vertex and then revert the result list. This program was also designed with an output function which prints a text instruction and a list of stations in the shortest path that will also be used as the data for the "drawScatterPlot" program to draw the scatter plot.

5. Solution

The "FastestSubwayPath" algorithm prints out a list of stations together with the line code and the departure time as shown below,

Table 2: the solution for Station E to Station C

Station List of Recommended Path							
Depature Time	08:30:00	09:15:00	10:30:00	13:00:00	14:00:00	16:00:00	16:45:00
Line1(Your Path)						1	3
Line2(Your Path)	5	4	2				
Line3(Your Path)				2	1		

this indicates the solution for starting from Station E at 08:30 to Station C. These numbers, as mentioned above, represents different stations. The adjacent same numbers indicate that customers need to wait in the station for the departure of next line, for example, at 10:30 to 13:00, customers are supposed to get off line 2 and wait for 90 minutes for line 3 in the station 2(B).

This solution is also demonstrated in a scatter plot so that customers can understand more easily (see the scatter plot in Appendix 1.1).

6. User Instructions of the Decision Supporting System

In this part, it will introduce the process of using this system and also show several results of the representative normal and special examples as well as the error handling measures.

6.1 The Usage of three Spreadsheets and User Form

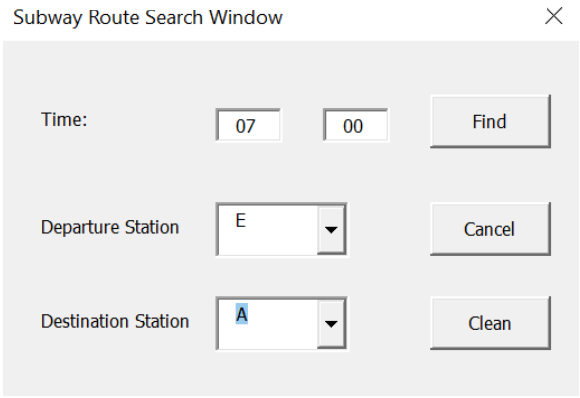
This section will demonstrate how customer should use the three spreadsheets together with the user form to find the way to their destination.

6.1.1 The "Connections" Sheet

This spreadsheet is the main interface where customers should normally start with. Customers can get a general knowledge about the metro service here. In the "Connections" sheet, it provides with the departure time and corresponding stations for all the three lines and a list of optional shortest paths in two tables that have been mentioned above. A scatter plot, which contains the detail of routes, is introduced to give customers a clearer vision of the

overall underground map. By reading this information, customers may be able to find their way. However, if they may not, they can use the search function which is launched by a button called “Search”. By pressing this button, a user form (See as Picture 1 below) will be popped up.

Picture 1: The user form Window



6.1.2 The User Form

This window contains two textboxes for entering the departure hour and minute and two dropdown lists for original station and destination, as well as three buttons that have functions of finding the shortest path, closing the pop-up window and cleaning all the entered information and results respectively. For the textboxes, customers are supposed to enter 0 to 24 in the hour box and 0 to 59 in the minute box while in dropdown lists, only the situation with two same stations is not acceptable since it indicates either the customer does not need the subway service or the information has been entered inappropriately. After filling in the correct information, customer should click the “Find” button which then leads to the result page.

6.1.3 The “Result” Sheet

In the “Result” sheet, it presents the output of the main algorithm, or in other words, the shortest path. The result is shown in three versions, a text instruction, a map instruction and a list of vertices which make up the optimal solution. The text instruction gives detailed information about the route, including how much time customers need to wait and the specific arrive time of the subway about to take. Additionally, a map instruction which shows the shortest path in a more directly way is programmatically generated (the pseudocode for this can be seen in Appendix 2). On the map, it illustrates the departure times, the passed stations, running directions and the line codes of the route that is recommended. The three lines are marked as purple, yellow and blue respectively while the waiting periods are marked as grey for distinguishing. At last, another alternative option is the list of stations. Customer can instantly get an idea of when and which subway to take by reading this list.

6.1.4 The “Backend_data” Sheet

Finally, in this sheet, all the data has been processed in certain format that is more suitable for VBA program to read. In the real-time setting, this sheet should be encapsulated or encrypted so that customer cannot see or access.

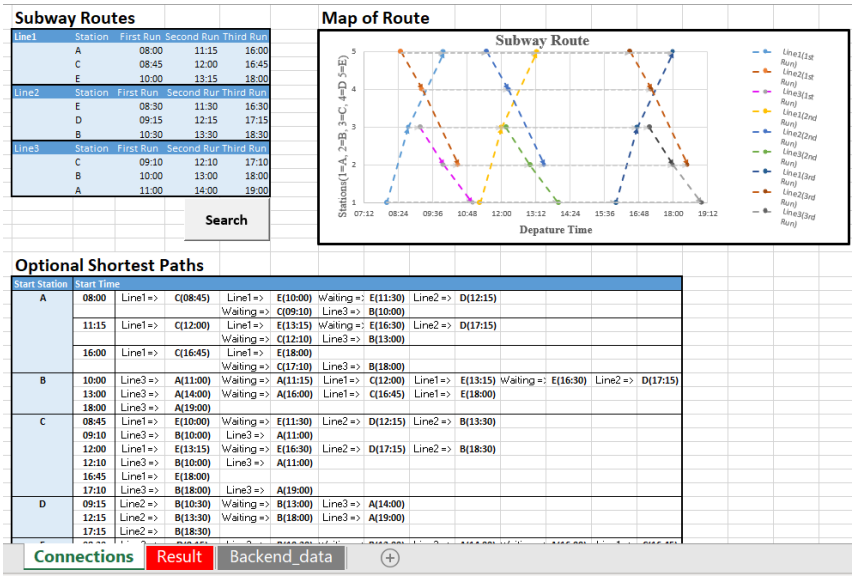
6.2 Sample Usage

In this part, it will illustrate several normal examples of using the system to find the shortest path and also some instances for the special cases.

6.2.1 Normal Cases

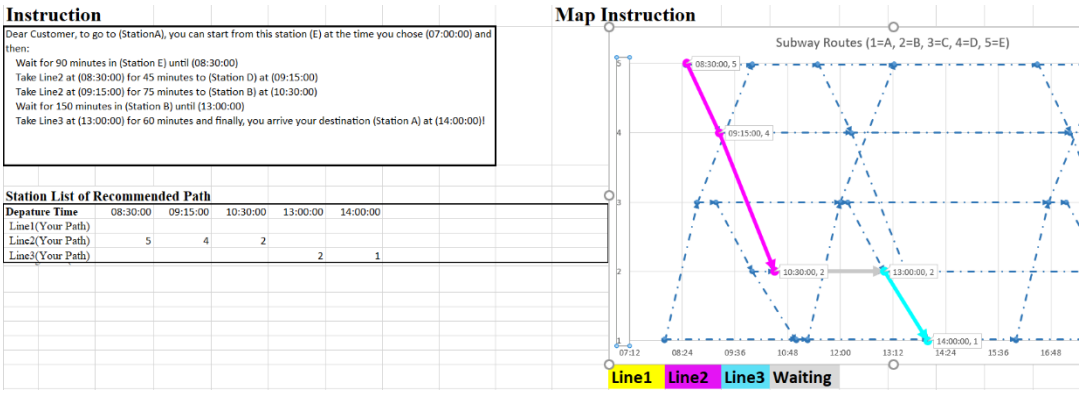
In normal cases, customers can get a recommended route from the system either by reading the route information in the "Connections" sheet or using the search function. For example, Customer J (assume J is a lady) wants to travel from Station E at 07:00 to Station A. She opens the system, and sees the main interface (the "Connection" sheet) as below,

Picture 2: The interface (Connection Sheet)



where she may find the way from either the map or the optional paths table which has the information about how to get to Station A from E at 08:30. However, if J may not, then she can also click the "Search" button and the user form will be popped up. After filling in the form as shown in Picture 1 and clicking the "Find" button, J is leaded to the "Result" sheet in which she sees the three types of instructions shown as below,

Picture 3: J's Route



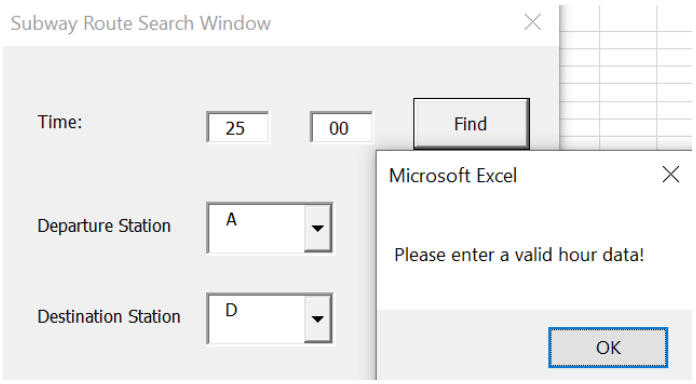
J can find from any one of the instructions that she needs to take line 2 at 08:30 to Station B and then transfer to line 3 which is able to take her to Station A.

Two more examples will be demonstrated in the Appendix 1.

6.2.2 Special Cases

In this section, it will mainly introduce two special cases. The first one occurs when inappropriate data has been entered in the user form. As mentioned above, in order to improve the search function, the user form has been restricted so that it only accepts correct data. If the two textboxes, for instance, are entered with letters, symbols or numbers out of the range, the system will then send a specific error message in a pop-up window, see in Picture 4,

Picture 4: The Error Message



and clean the inappropriate data. According to different types of error, it will send three different error messages. The other two error message will be presented in Appendix 1.2.1.

Another special case happens when customers are later for the last subway. The system will then print out a message in the text instruction telling them that they are too late for getting there. An example of this will be shown in the Appendix 1.2.2.

7. Limitation and Assumption

This system, as a beta version of a decision supporting system for metro services, is fully functional and direct for customers to use to find their routes. However, there are also several limitations in the system which affect its quality of service.

7.1 Limitations

I. As mentioned above, the data sheet in this program has not been encapsulated which means when it is put into practice, customers can also see or even change the original data.

II. This beta version is only designed with a small set of data (3 lines, 3 runs and 5 stations). In practice, there may have more than 15 lines and each of them can run with a higher frequency covering hundreds of stations. The current version is likely to be unsuitable for that condition.

III. In the design stage of this program, some situation has been neglected which may lower its practicability. In some cases, for example, customers may have special requirements for the passing stations. This cannot be handled by the current version of this program.

7.2 Assumption

This system assumes that there is no gap between arrival and departure, which means there is no time for customers to get off or get on the subway. In reality, the pick-up time should also be in consideration.

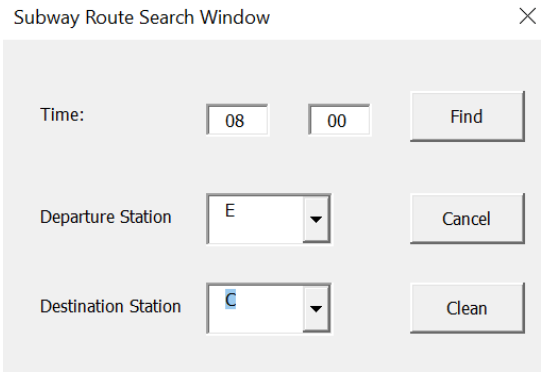
8. Conclusion

This report has introduced a decision supporting system for guiding customers in subway stations. It has presented the data preparation process and discussed about the algorithm being used. The instructions of using this system are also clearly illustrated with figures and tables. In order to improve the program's practicability, it has also raised three limitations that may affect the quality of services. For the encapsulation problem, in later versions, it is suggested that the data sheet should be isolated from the other two sheets so that customers are not able to access the original data. In addition, it is crucial for this program to be tested with real-size data sets that contains more lines, stations and operations. Also, adding more functions is likely to make this program more comprehensively functional in order to fit in all the requirements from customers.

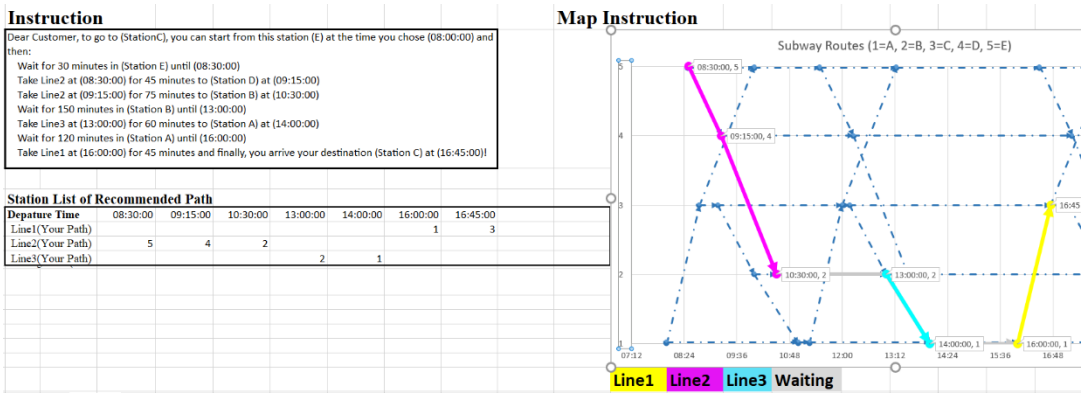
Appendix 1

1.1 Two Examples of Using the System:

Customer K wants to go to Station C from Station E at 08:00. By entering the user form as below,

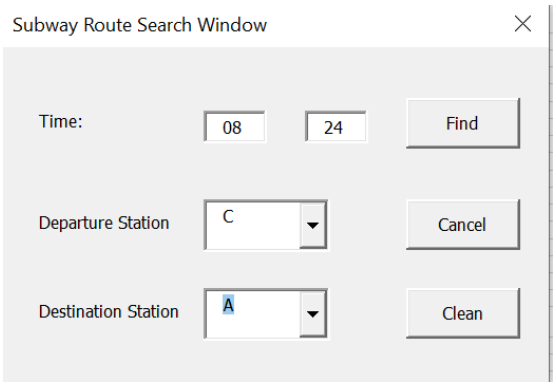


K is able to get a result page like,

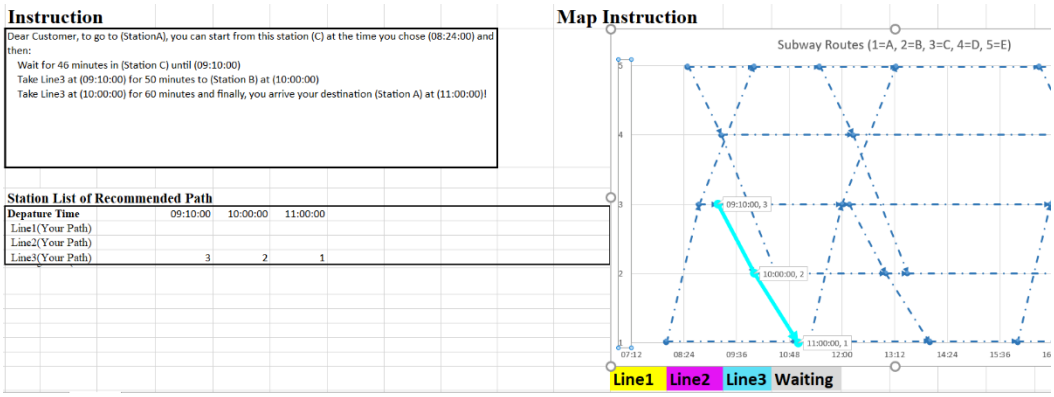


.

There is another customer L. She wants to start from Station C to Station A, and the time when she arrived the Station C is 08:24. Thus, she entered her form as below,



and by clicking the “Find” button, she got an instruction as below,



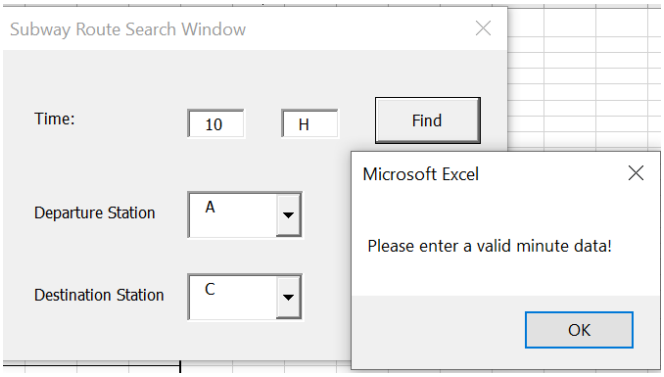
.

Both of them are satisfied with the recommended paths.

1.2 Example of Special Cases:

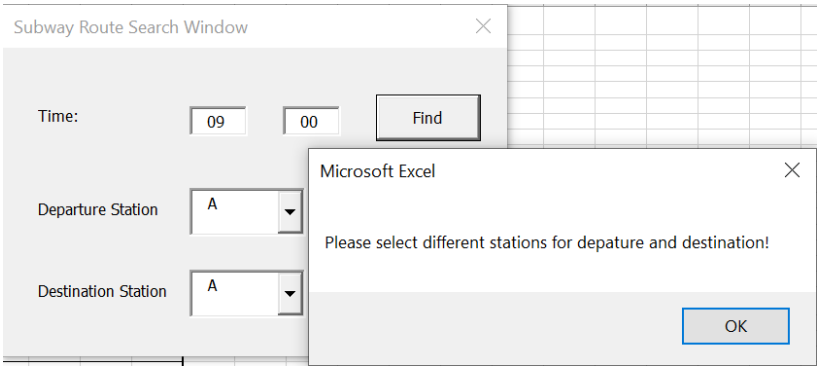
1.2.1 The Other Two Error Messages:

Customer Q entered the letter “H” into the minute box by mistake, the system returned him as,



and he suddenly realised his mistake.

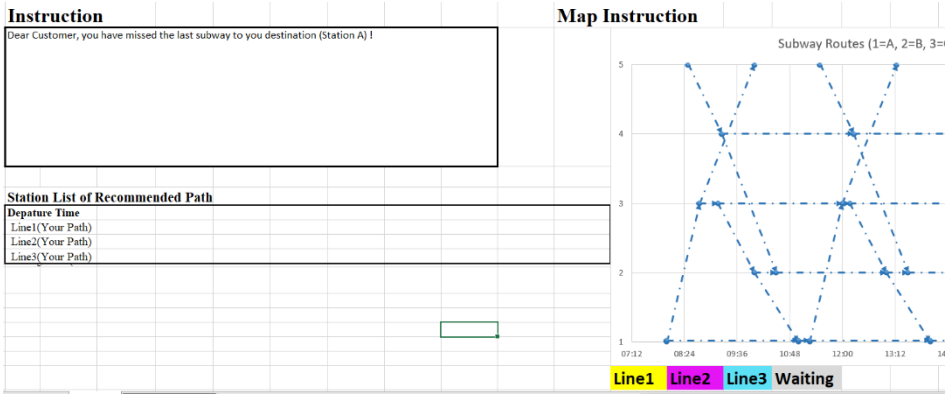
Customer W wanted to choose Station A as her departure station and Station D as her destination station. However, instead, she chose both A for departure and destination stations. Therefore, system send her an error message as below,



she successfully reached her destination by selecting the correct station afterward.

1.2.2 The Case for missed Last Subway:

Mr E had some extra work today, when he arrived Station C, it was already 18:00. He wondered whether there was still a subway to Station A, so he searched. The system told him that,



so, he had to take a taxi to go home.

Appendix 2

2.1 drawScatterPlot Sub

Sub drawScatterPlot()

Active “Result” worksheet;

Add a scatter plot chart;

Call Function drawLines () to draw the 9 running lines;

Call function drawLines () to draw the 5 waiting lines;

Draw the lines that customer that needs to take (don’t call the drawlines() function, because of setting for these lines are different)

Change the plot name

Change the size and location of the plot

Change the scale of axes of the plot

End Sub

2.2 Private Function drawlines ()

Private Function drawLines (lineNum1 As Long, pointNum1 As Long, dataStartPoint As String, dataEndPoint As String, seriseNamePoint As String, startPoint As Long, counter As Long) As Long

```
    For i = startPoint to startPoint + lineNum1 - 1
        Create a new series;
        Get the data from “Backend_data” sheet;
        For j = 1 to pointNum1
            Set values for every points of the series
        Next j
    Next i
End Function
```