COMP 1828 - Designing, developing and testing a journey planner model for the London Underground system

Team members:

Name	ID
Aisha Scacchi	001067045
Abdullah Albashed	001063351
Gabriele Qazolli	001037593
Nayem Miah	001081392

1. "HOWTO guide" for a customer and any assumptions made (max 3 pages) [10 marks]

Once we run the file 'MainGUI', the initial and principal window will be showed; this window contains most of the buttons that will be used for interaction. The main window contains the buttons to show the map for each line, the map button, the search button, time arrows, the quit button and the information button. Each button or fields of input will be highlighted and labelled, below on image 1.

Image 1

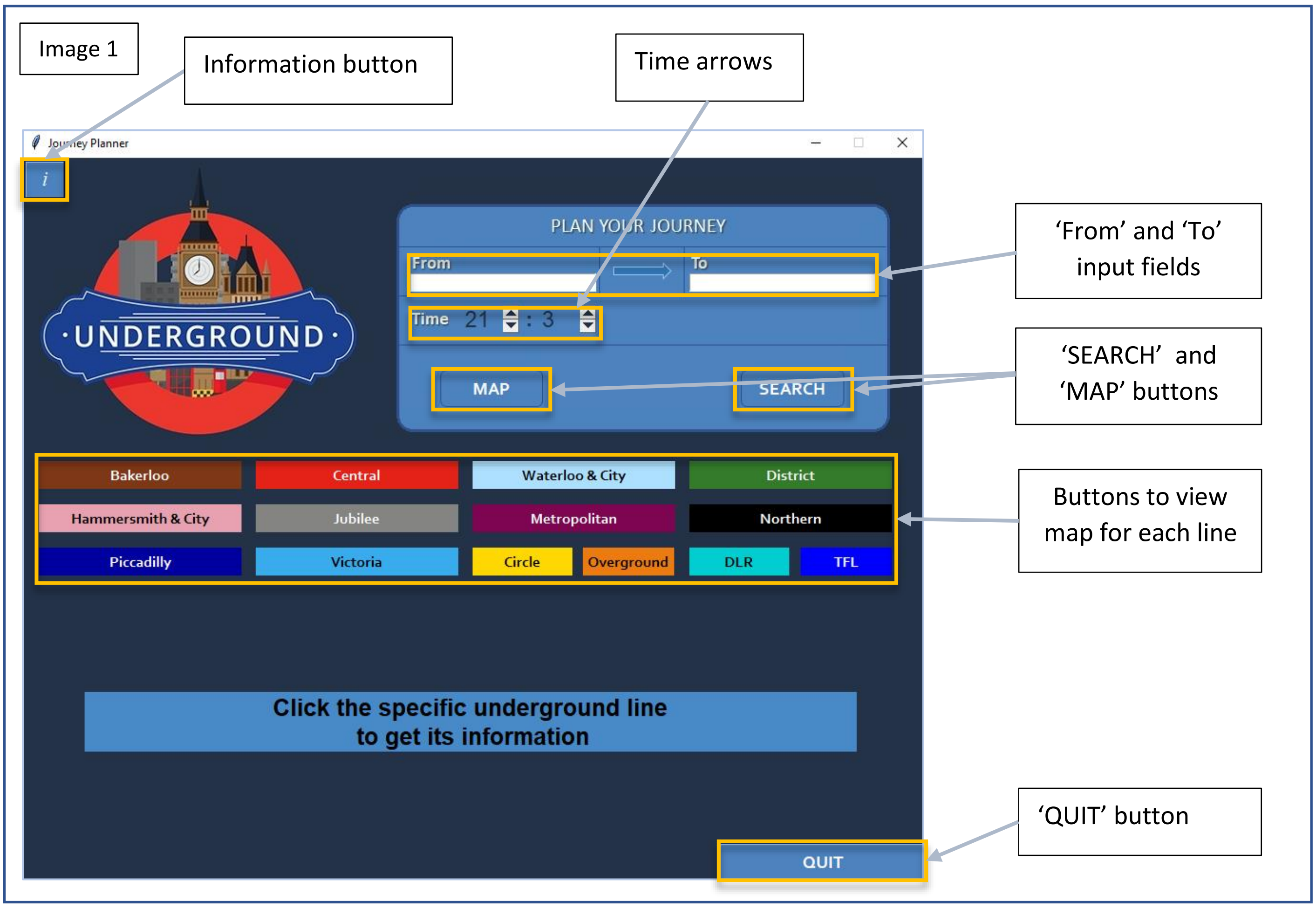
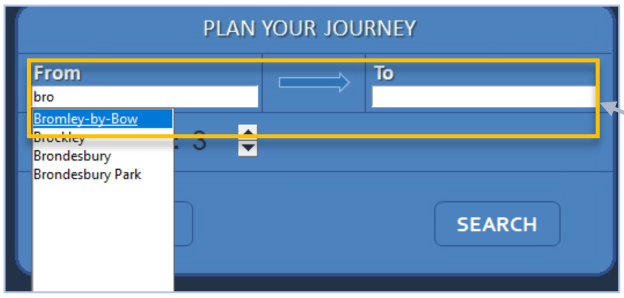
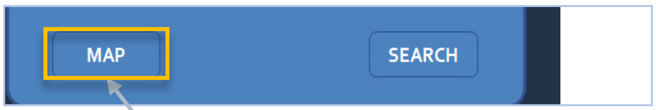


Image 2



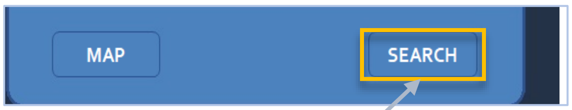
In the "Plan Your Journey" field, the user can enter the departure station and the arrival station. In order to make the choice of the route more effective, the program shows as suggestions the names of the stations that begin with the same letters entered by the user. To select the desired station, the user must click twice on it, and it will automatically register on the input box.

Image 3



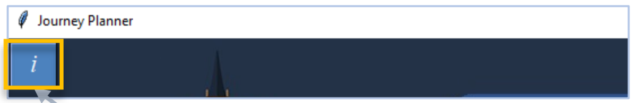
The user has the option of viewing the complete tube map by clicking the “Map” button.

Image 4



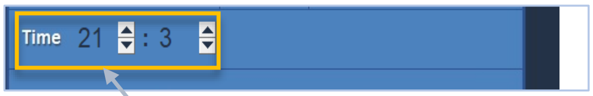
Once the desired journey and time have been selected, the user must click the “Search” button and will be redirected to a second window.

Image 5



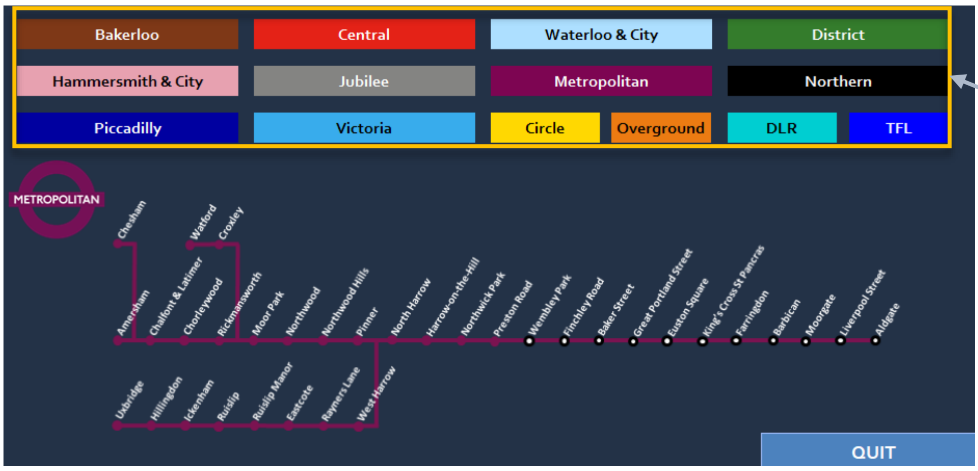
The user can click on the info button to view more information on travel times. Info includes the timeframe of trains and much more.

Image 6



These are the time arrows which allow the user to change the time they would like to start the journey. When the program is opened the time will be automatically set to the current time.

Image 7

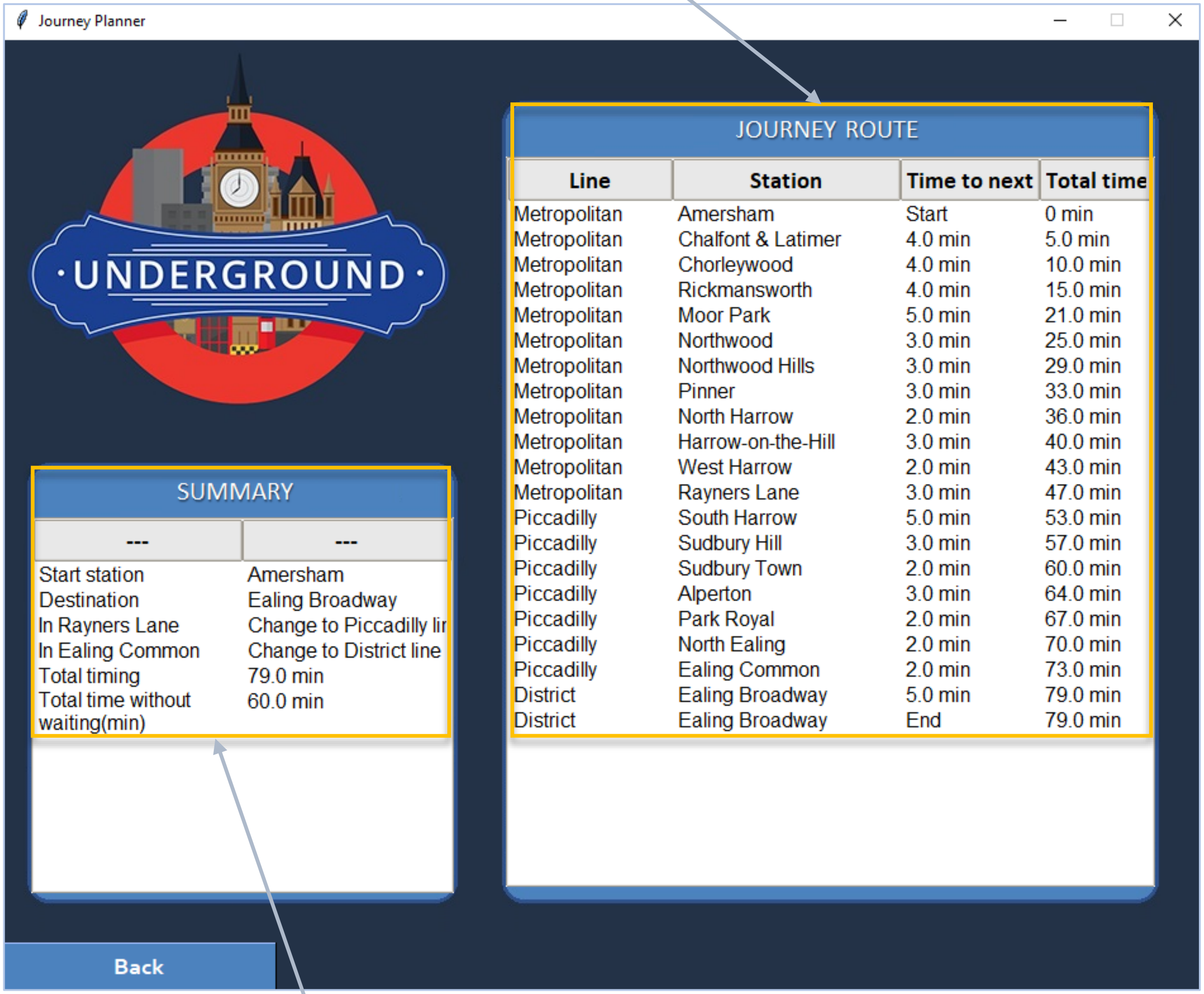


The user can select any of these buttons to view the map for the selected line. For example, in this case the ‘Metropolitan’ button is clicked, and the Metropolitan line map is showed.

Image 8

The second page contains two tables. The main table has four columns which contain information regarding the lines, stations, time required to next stop and total time; they provide different information about the journey:

- In the 'Line' column, all the lines for each station traveled is showed.
- In the 'Station' column, every traveled station is showed.
- In the 'Time to next' column, the time taken from one stop to the next one is showed.
- In the 'Total time' column, the time is added to the previous one also including 1-minute waiting for each station.



JOURNEY ROUTE

Line	Station	Time to next	Total time
Metropolitan	Amersham	Start	0 min
Metropolitan	Chalfont & Latimer	4.0 min	5.0 min
Metropolitan	Chorleywood	4.0 min	10.0 min
Metropolitan	Rickmansworth	4.0 min	15.0 min
Metropolitan	Moor Park	5.0 min	21.0 min
Metropolitan	Northwood	3.0 min	25.0 min
Metropolitan	Northwood Hills	3.0 min	29.0 min
Metropolitan	Pinner	3.0 min	33.0 min
Metropolitan	North Harrow	2.0 min	36.0 min
Metropolitan	Harrow-on-the-Hill	3.0 min	40.0 min
Metropolitan	West Harrow	2.0 min	43.0 min
Metropolitan	Rayners Lane	3.0 min	47.0 min
Piccadilly	South Harrow	5.0 min	53.0 min
Piccadilly	Sudbury Hill	3.0 min	57.0 min
Piccadilly	Sudbury Town	2.0 min	60.0 min
Piccadilly	Alperton	3.0 min	64.0 min
Piccadilly	Park Royal	2.0 min	67.0 min
Piccadilly	North Ealing	2.0 min	70.0 min
Piccadilly	Ealing Common	2.0 min	73.0 min
District	Ealing Broadway	5.0 min	79.0 min
District	Ealing Broadway	End	79.0 min

SUMMARY

---	---
Start station	Amersham
Destination	Ealing Broadway
In Rayners Lane	Change to Piccadilly lir
In Ealing Common	Change to District line
Total timing	79.0 min
Total time without waiting(min)	60.0 min

The second table shows the summary of the route. It contains the starting station, the final station, the changes that will be made, the total time and the total time without counting the wait at each stop.

2. Critical evaluation of the performance of the data structures and algorithms used (max 1 page) [15 marks]

The requirements of the program include the use of the doubly linked list to interconnect stations and the Dijkstra algorithm to calculate the route from one station to another.
We used the materials made available to us on the singly linked list and then fitted them to our project which necessitates the use of the doubly linked list, similarly we did with Dijkstra.

Our choice to use the *csv* format rather than the *xlsx* format of the file has been made by two factors, the first is that *csv* files are smaller than *xlsx* format in terms of size, the second is that the data are raw and are faster to be processed.

Dijkstra's algorithm works as follows: firstly, it chooses a starting node and marks all the other nodes as unvisited. Then it evaluates all the nodes connected to the current node and moves to the node with the lowest cost, and marks as visited the node from which it moves. So, carry on like this until it reaches the node it needs to get to.

Simplicity is the main feature of the algorithm as it is very intuitive and simple to be implemented, however, it has the drawback of slowing down the process in the event it is dealing with many nodes.
Unlike the singly linked list, the doubly linked list allows us to cross the list of stations in a bidirectional way, i.e., we can go from point A to point B and vice versa. However, this leads us to have to highlight one of the disadvantages of the doubly linked list which is the need to manage the node before the current, the head and tail as well as the data and node after the current.

In our project, each station is a node that contains information about the next and previous nodes, the journey time to reach them and data about the lines passing through the node.
As for the excel file containing the underground data, we found the following errors:

- Metropolitan line:
    - From Moore Park to Harrow-on-the-Hill 14 mins should be removed.
    - From Harrow-on-the-Hill to Finchley Road 16 mins should be removed as they are not directly connected.
    - From Harrow-on-the-Hill to Wembley Park 9 mins should be removed.
- Central Line:
    - Tottenham station should be named Tottenham Court Road.
- Piccadilly line:
    - Holborn Central Should be renamed to only Holborn.
    - From Hammersmith to Acton Town --> there is a station (Turnham Green) between them which was ignored.
    - Heathrow Terminal 1, 2, 3 should be only Heathrow Terminal 2,3 as there is no Terminal 1 station.

3. Discussion for the choice of test data you provide and a table detailing the tests performed (max 2 pages) [10 marks]

To test our program, we used a set of conditions, variables, steps, and expected result. The choice of the various types of testing was given by the need to show the functioning of the program in its correct and incorrect mode of use.
Both valid and invalid data were used to test the proper functioning of the program. In the first case, existing stations and an allowed time have been inserted, thus obtaining positive feedback from the research. In the second case, on the contrary, non-existent stations have been chosen, invalid characters entered, or correct data entered but with a time between midnight and 5 am.
Some tests were dedicated exclusively to the Bakerloo line to verify that at certain times of the day, the time taken to travel was halved compared to the normal journey.
A detailed test of the individual buttons was also performed to test the operation of the graphical interface.

More detailed information of all the data tests performed are shown in the table below.

Test Num.	Test Type	Test Data	Reason	Expected Outcome	Actual Outcome	Pass/Fail
1	Valid	From: Stanmore To: Kilburn	Enter valid information to check if the correct stations are displayed	Correct stations should be displayed from starting point to ending point	Correct stations are displayed from starting point to ending point	Pass
2	Valid	From: Stanmore To: Hammersmith	Enter valid information which pass through many junctions to see if the correct route is taken	The correct stations should be displayed where there are many junctions	The correct stations are displayed where there are many junctions	Pass
3	Valid	From: Epping To: Morden	Enter valid information to check if long distances affect the time taken for the search function	The time taken to run the search function should not vary	The time taken to run the search function did not vary with long distances	Pass
4	Valid	From: Euston To: Archway	Enter valid information and check if it is possible to travel between midnight and 5:00 am	It should not be possible to travel between midnight and 5:00 am and an error popup should be showed	It is not possible to travel between midnight and 5:00 am and an error popup is showed	Pass
5	Valid	From: Price Regent To: Hoxton	Enter valid information and check if changes work and are taken correctly	Changes between lines should be taken and should work correctly as intended	Changes between lines are taken and work correctly as intended	Pass
6	Valid	From: Heathrow Terminal 5 To: Shenfield	Check if we make a big journey the time is still calculated correctly	The travel time should be calculated, and a high travel time should be showed	The travel time between far stations very high which is what we were expecting	Pass

7	Valid	From: Maida Vale To: Lambeth North Time: 9:30 AM	Check if time is halved if starting time is between 9:00 AM to 16:00 PM and 19:00 PM to midnight for Bakerloo Line	The time taken for the travel should be halved at specific times for Bakerloo Line	Time is halved between the specific times for the Bakerloo Line	Pass
8	Valid	From: Maida Vale To: Lambeth North Time: 9:30 AM	Check if time is not halved if starting time is from 16:00 PM to 19:00 PM for Bakerloo Line	The time taken for the travel should not be halved at the specific time frame for Bakerloo Line	Time is not halved between the specific times for the Bakerloo Line	Pass
9	Valid	From: Amersham To: Hammersmith	Check if 'Journey Route' and 'Summary table' display information correctly after re-using the search button on Home Page	Journey Route' and 'Summary table' should reset and show only current route, should not display old search route	Journey routes are reset and every time we use the search button only the new information is displayed	Pass
10	Invalid	From: empty To: empty	Check if clicking on search button without giving input on 'From' and 'To' will give an error popup box	When we click the button 'Search' with empty inputs an error popup box should appear	An error popup appears if inputs are empty	Pass
11	Invalid	From: #@$! To: %&/!	Check if clicking on search button with random characters will allow to get on the journey summary page	When we click the button 'Search' with random character inputs an error popup box should appear	An error popup appears if inputs are random characters	Pass
12	Valid	From: Amersham To: Amersham	Check if 'Summary table' is displayed correctly with the same travel location input from both 'From' and 'To' fields	When we click the button 'Search' with the same valid inputs, information should appear only on 'Summary Table'	Information is shown only on the 'Summary Table'	Pass
13	Valid	Check 'Map' button	Check if map is showed correctly and interactive as intended when 'Map' button is clicked	Map should be showed correctly and be interactive as intended	Map is shown correctly, and we can interact with it as intended	Pass
14	Valid	Check tube information buttons	Check if tube line maps are displayed correctly when tube information buttons are clicked	On the click of each tube information button the correct tube line map should be displayed	When we click a tube information button the correct tube line map is displayed	Pass

4. Individual contribution by each team member and reflection (max 2 pages) [10 marks]

Name	**Allocation of marks agreed by team (0-100)**
Aisha Scacchi	100
Abdullah Albashed	100
Gabriele Qazolli	100
Nayem Miah	100

Nayem and Abdullah: Back-end

As Back-end programmers, we must ensure that the logical part of the project works efficiently. We have used the materials provided to study the functioning of the linked list and Dijkstra and then deepen them through research on the internet. First, we had to write each algorithm and test them separately, and then we had to think about how to group the various components to make them work cohesively and functionally. As for the Dijkstra algorithm, we had to make several changes in the code to make it as efficient as possible, trying to have a reasonable execution time even with the addition of DLR, Overground and TFL lines. The classes have been used in such a way that their functionalities can be reused in more parts of the project, making the code more efficient, organised, and easy to understand. The names of the variables, classes and functions have been assigned following the PEP 8 guidelines. As for the testing part of the backend we used the logging library to keep track of the reasons why the program crashes when the program is running, this helped us to detect some errors in the excel data.

The experience gained in the last university year has allowed us to divide up the Back-end efficiently and have good communication between us to keep up to date. We had to change parts of the Back-end to match part of one's code to the other person's code, which allowed us to learn that there are various solutions to solve the same problem.

Gabriele and Aisha: Front-end

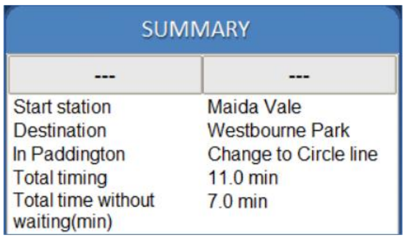
The main goal of every programmer is to make the final product user-friendly. As front-end managers we wanted to create an intuitive and simple graphic interface, understandable menus and functions, well-made and well-positioned icons, dynamics of use suitable for everyone.
Inspired by applications that we use daily and based on real experiences, we designed a layout that was simple to the user to look at, making the best use of the different graphic elements we had available. Before starting the actual programming work, like graphic designers, we produced concepts drafts and preliminary drawings sketched on paper. The designs for the first window were further developed digitally.

Regarding the programming phase we decided to use Tkinter as part of the Python library. Tkinter has a simple syntax and the text widget is remarkably powerful and very easy to work with. It uses native widgets on the mac and windows. Moreover, it is solid with few cross-platform idiosyncrasies.
Another fundamental point of the project was to optimize the spaces, and to be precise we decided to use “place” to position the widgets.

As a final comment, every decision made for the development of this program has been evaluated and approved by every single member of the group. A perfect synchronization work was done between Back-end and Front-end. After having illustrated in detail the work done in both sections, the next step was to combine the two parts into a single program.

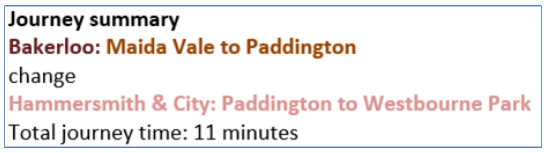
The stages of development of the Front-end and the Back-end were organized in view of the union of the two sections, therefore it was not a particularly complicated job.
In the phase of joining the two parts of the project (Back-end and Front-end), we found ourselves having to change a large part of the Back-end in order to adapt it to our GUI.
For satisfactory completion of the work, the program has been tested several times and the small bugs that have occurred have been effectively solved.

Additional Information



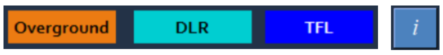
SUMMARY

---	---
Start station	Maida Vale
Destination	Westbourne Park
In Paddington	Change to Circle line
Total timing	11.0 min
Total time without waiting(min)	7.0 min



Journey summary
Bakerloo: Maida Vale to Paddington
change
Hammersmith & City: Paddington to Westbourne Park
Total journey time: 11 minutes

We decided to change the layout of the summary as we found our version more intuitive for understanding the changes to be made during the journey.



We decided to include Overground, DLR and TFL in our program. An Info button has also been added to illustrate some specifics on Underground rides to the user.

Reference (Front-end code)
[1] Basj. Tkinter canvas zoom + move/pan. Stack Overflow. Published January 14, 2017. https://stackoverflow.com/questions/41656176/tkinter-canvas-zoom-move-pan/48137257#48137257 [Accessed 10 Oct. 2020]
[2] uroshekic (2020). Tkinter - Autocomplete Entry Field. [online] Gist. Available at: https://gist.github.com/uroshekic/11078820 [Accessed 24 Oct. 2020].