Project 2

M00861483(Thomas),
M00874096(Ben),
M00664455 (Ahmed)

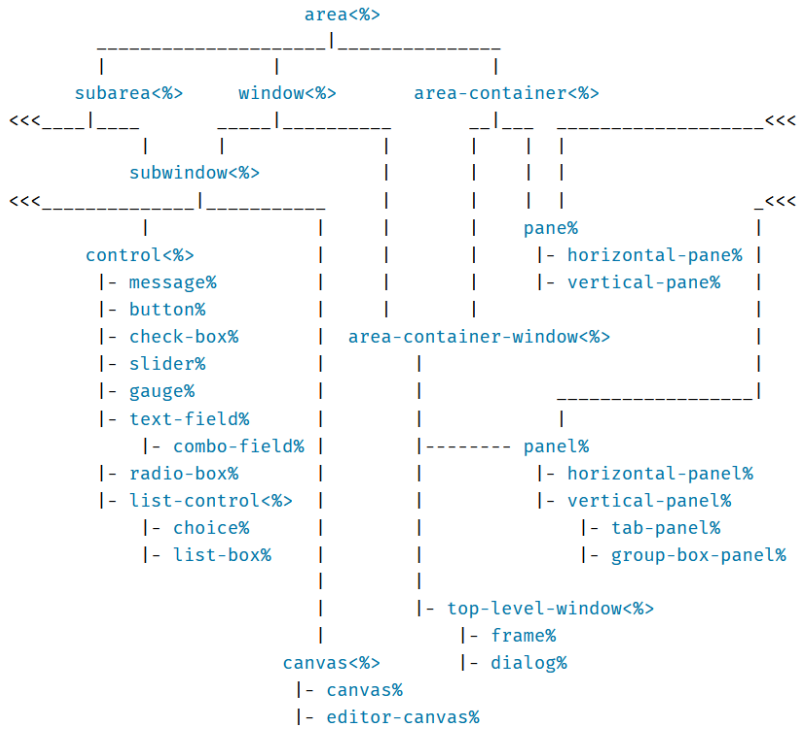
Summary

Our system, “Easy Access TFL London” is an application for your phone that allows users to input a current location and a target destination. The system will compute the most effective way of navigating the city to arrive at the target destination. The system determines what is the most “effective” way of travelling based on a series of inputs from the user that list requirements or desires such as wheelchair accessibility or that the user wants transport that offers a higher level of comfort. These filters will impact how the system computes what modes of transport to take such as fewer transfers, taking trains over buses or working around closures and flooding.

Here we are designing how the system will look like considering the potential users and using the design requirements/key features established in the first project. We will Refine the original design concept and make it more realised and practical, using the Racket Graphical Interface Tool to design a prototype based on the design and make improved design sketches.

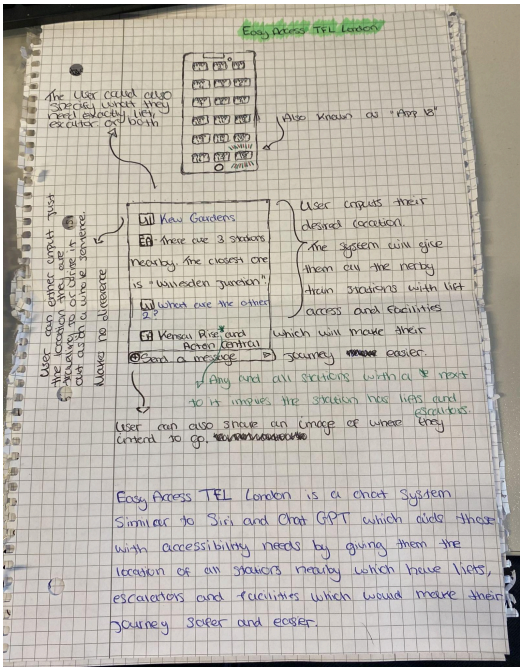
Racket Graphical interface tool(Racket GUI)

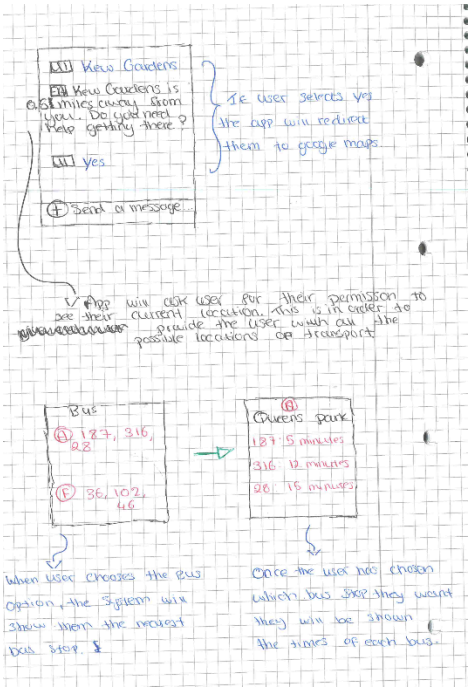
The Racket Graphical interface is an object oriented library, providing all the tools to create a GUI with the Racket language.

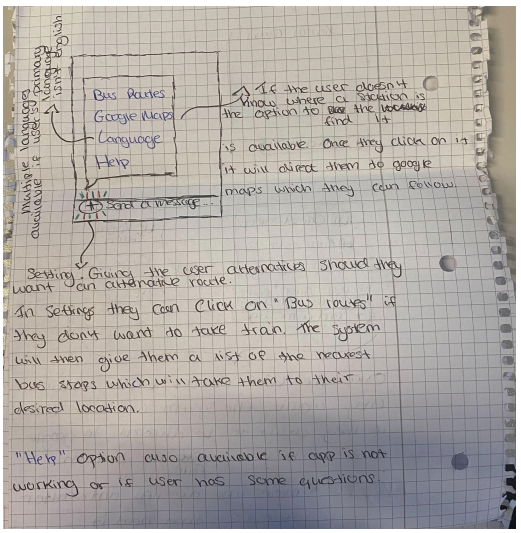


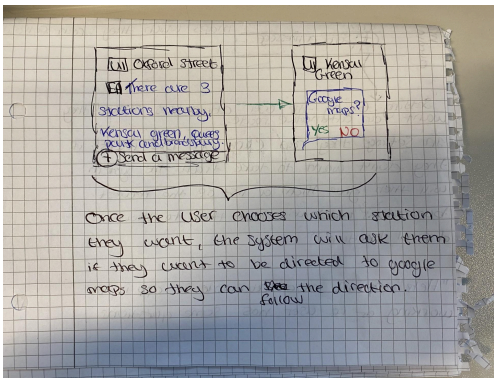
Design

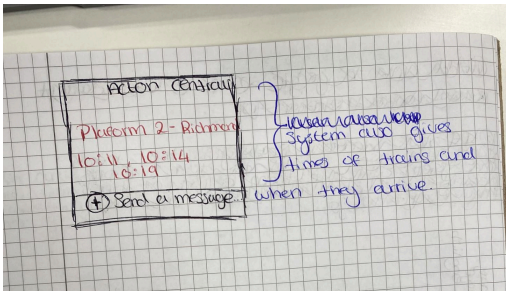
Initial design sketches







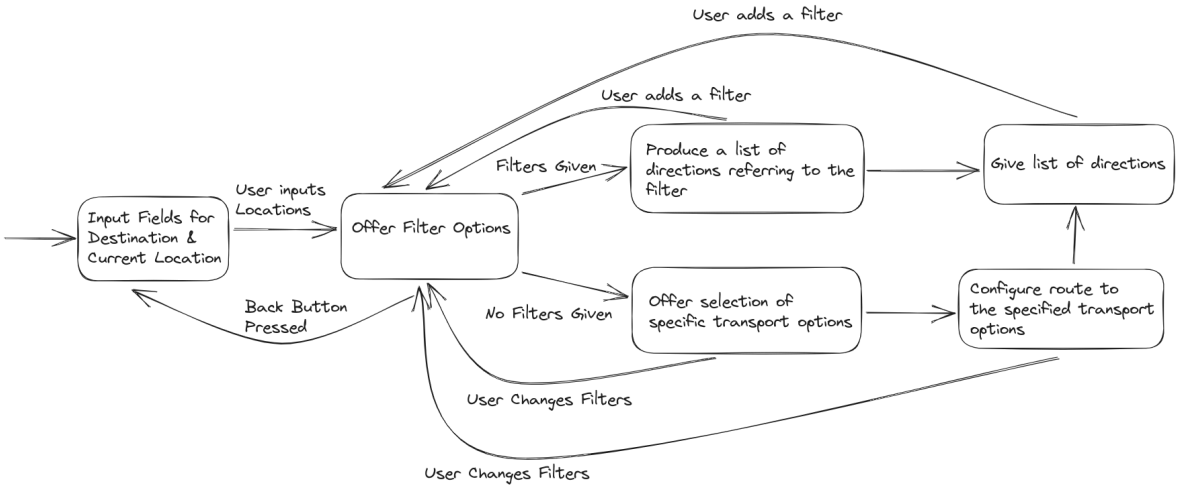




Revised Design

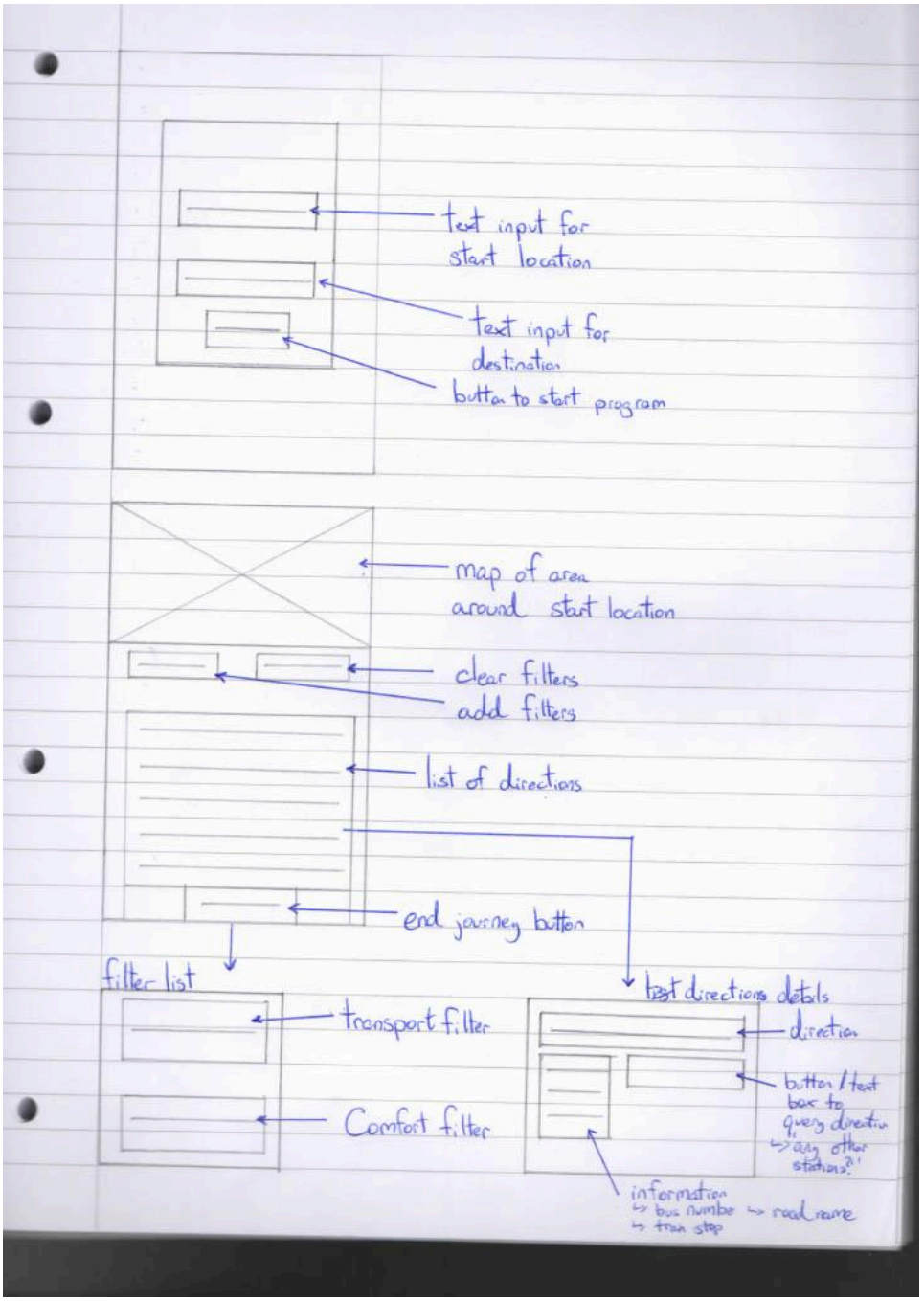
To represent the London Underground and bus routes we will be using a branching network of nodes which will represent bus stops and train stations, the user will input a starting and ending station and the code will list out the ways to travel between these two nodes/bus stops/train stations. Using data structs with this we can represent the stations/stops , as well as data about those stations./stops, e.g. what line or part of that line it's on or accessibility information.

This is a finite state machine to represent what we want to achieve with the user interface, from when the user inputs the where they're going from and where they're going too, and filters, with



The revised design changes the original design to make it more clear to the user where they should input and adding extra features to increase usability and functionality of the app.

- Making it so there's 2 text boxes one for starting location and a destination
- On the journey screen adding a map to show the user the journey and where to go.
- Adding filtering options so the user can filter by price, speed ect.

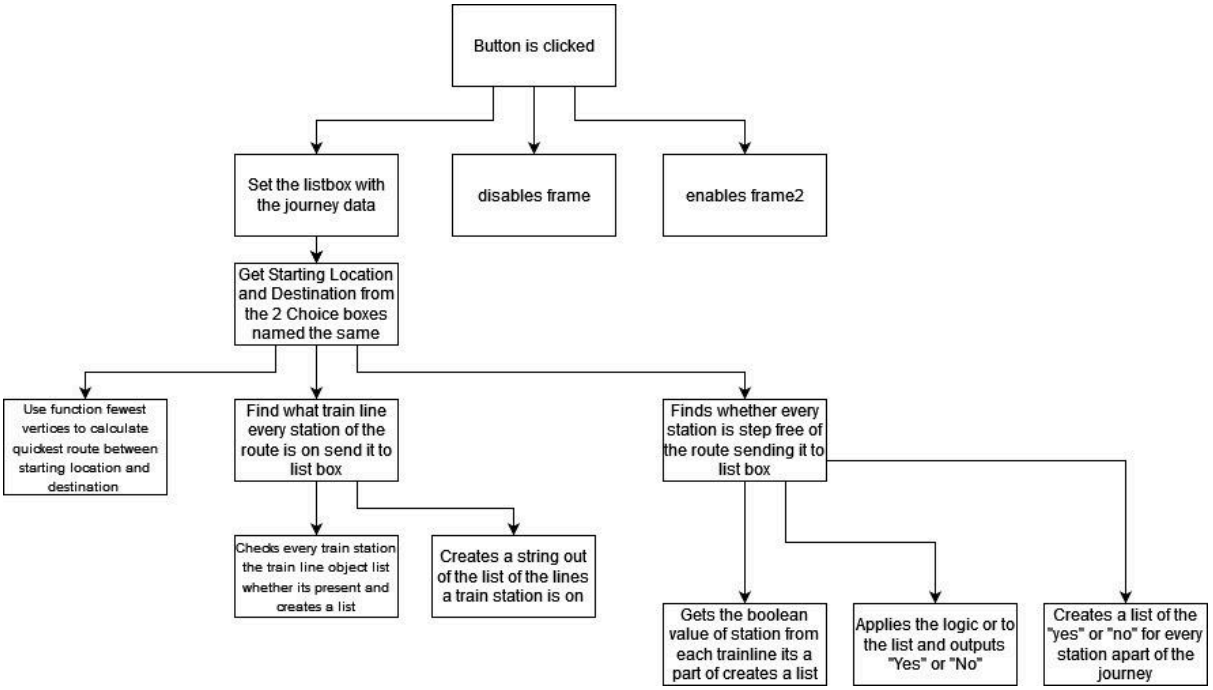


Prototype

GUI

The GUI side of the code has 2 primary windows, the window where the user inputs the information for the journey function and the window where the output of the journey function is displayed. In this explanation, we'll call them the input window and the output window.

The input window has the title of the software("Easy Access TFL") and a subtitle saying "Enter Journey info" for clarity. Then we have the 2 primary choices boxes(which we will now call "Starting location" and "destination" from now on). Additionally, there are an extra 2 choice boxes to filter the tube stations via line e.g. the Victoria Line or the Northern Line because of this the main choice boxes aren't one gigantic list making it easier and more manageable for the user to select which stations they need. As well as this the Go button gets the choices from the 2 main choice boxes, if the 2 choices are the same an error window appears to let the users know that it's an invalid input. Furthermore, it calls the get path function with the data from the 2 choice boxes to get the journey list and send it to the list box(the list box has 3 columns "Train Stations" "Train line" "Step-free"), with there being a for loop to determine which line it is on e.g. Kings Cross St Pancras would output "Northern Line Victoria Line" since its part of the class Northern Line and the class Victoria Line. Also, there is the for loop which determines whether the result of the get path is free outputting a list of "Yes" or "No". So if the journey was only Kings Cross St Pancras (which we'll entertain for this example), it'll output "Kings Cross St Pancras" "Northern Line Victoria Line" "Yes" to the first line of the list box under the three columns "Train Stations" "Train lines" "Step-free".



The output window deals with presenting the journey output and related information to the user; this is done with a list box with 3 columns as described above. Also having duplicates of the canvas' used on the input window with the second canvas text replaced with "Here's journey info:". Lastly, having a go-back button which takes the user back to the input window. The class train line works by having 2 fields Line Name and Train Stations. These are a string and a list, one to represent the train line name, the list to represent the train stations and the unweighted graph of that train line, the name as a string e.g. "Northern Line" and step-free access of that station represented as a boolean in nested lists e.g. '(("Tottenham Court Road" #t) '("Warren Street" #t)). Lastly, there are train connections which is an unweighted directed graph which represents the line. As well as this it has various methods, which sort the train stations list with just the name of the stations such as:

- line_name_get, as it says gets the field line name and outputs it.
- train_station gets the field train_stations which is a list.

- train_stations_gets takes the first part of ("Tottenham Court Road" #t) for every instance in the field Train Stations and outputs it as a list.
- step_get_free like train_station_get but get the second part of the ("Tottenham Court Road" #t) true output "Yes" if #f outputs "No".
- step_free is similar to step_free_get but only does it on one input rather than a list, check whether it is in the list if so outputs the step-free value for the station, e.g. given Warren it will get("Warren street" #t) it'll output true.
- train_connections_get gets the field train_connections for the graph function.

Sobs

Sob	Team member	What was done
111 Working individually or as part of a group, take a simple design task through from initial description to a working prototype, discussing …		
112 Demonstrate a clear awareness of the importance of user interface design issues …		
150 Create and manage a project in a repository such as GitHub.		
206 Take an open design task, refine the problem to a manageable scope, develop a robust working prototype …		
207 Demonstrate an understanding of the way a graphical user interface works with Racket …		
229 Write technical documentation that		

describes the specification, design, implementation and evaluation …		