



23rd October 2021

Final Report

Human Decision Making when boarding
public transportation



Meher Singh 27983633
FINAL YEAR ECSE PROJECT 2021

i. Significant Contributions

Created and implemented a SUMO-based simulation of a bus route from Chadstone Shopping Centre to Monash University, including:

- Importing a network file from open street map and developing a bus route including bus stops
- A randomisation of passengers per hour boarding the bus from several bus stops
- Creating a server that can run and gather data from the simulation

Added human-in-the-loop capacity to the SUMO-based simulation. Developed a python-based server that acts as a gateway for communication between the SUMO-based simulation of the bus route, and a real person with an Android application on their phone

Designed and developed an android application that can receive details from the simulation relating to bus positions and numbers of simulated people on board and saves data relating to human preferences on which bus to thus take.

ii. Poster



MONASH University
Engineering

Department of Electrical and
Computer Systems Engineering

ECE4095 Final Year Project 2021

Meher Singh

Human Decision Making on Boarding Public Transport

Supervisor: Dr Wynita Griggs



python™

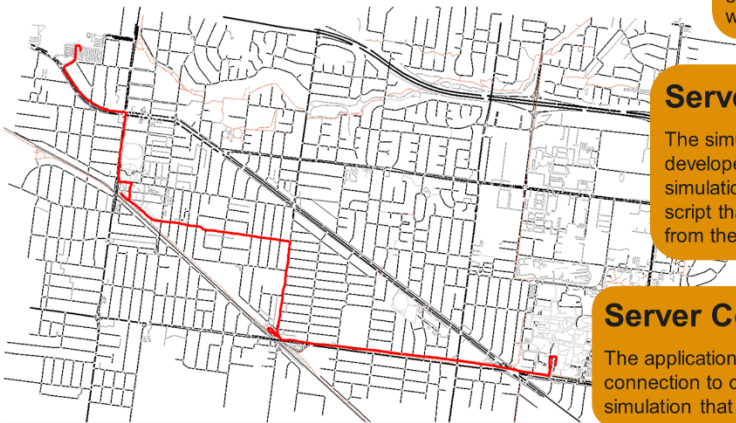


JavaScript



Project Aim

Design and develop a human-in-the-loop simulation platform, based on SUMO, for gathering data on human decision making when boarding public transport.

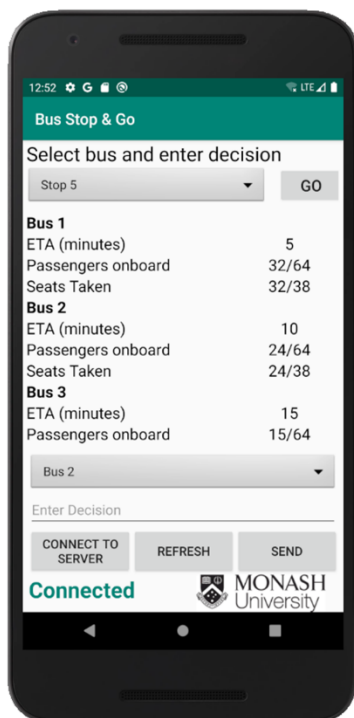


Server/Simulation

The simulation developed in this project is developed on the open source, continuous traffic simulation software SUMO. It is run from a python script that acts as the server, receiving information from the simulation and sending it to the application.

Server Connectivity

The application developed in this project uses a TCP socket connection to connect to the server running the bus simulation that is also developed in this project.



The app allows the user to select a bus stop in the route and view the buses arriving to that stop

The app displays key details for each bus arriving to the stop selected by the user. These details can allow the user to make a decision on which bus to take

The app allows the user to select a bus, and provide a reasoning behind their decision

Buttons to interact with the app

Status bar to show whether the app is connected to the server

iii. Executive Summary

Imagine that you are standing at a bus stop, and a bus pulls up that is almost to capacity with people forced to stand in the aisle. You are unaware of another bus arriving 5 minutes later that is less than half capacity with many seats available, and in turn must get on the first bus and stand next to a crowd for your journey. Now suppose you had an application that can inform you of this information. Would you take the first bus and get to your destination immediately, or wait for the second bus and have a more comfortable ride but arrive slightly late?

The main outcomes from the project are an application installed on a phone that can communicate with a server, and a server that can run a bus simulation, connect to the application, and save results to an excel database.

This report outlines the design, development, and results of an application for gathering data on human decision-making regarding boarding public transportation. The report goes into detail of the objectives for the project, the overview of the solution, and a detailed discussion on the design choices, and testing results for the application and server.

1.0 Table of Contents

i. Significant Contributions	1
ii. Poster	2
iii. Executive Summary	3
1.0 Table of Contents	4
1.1 List of Figures	5
1.2 List of Tables	5
2.0 Introduction	6
3.0 Literature Review / Background	7
4.0 Overview	8
4.1 Android Application	8
4.2 Python-based server & SUMO simulation	12
5.0 Detailed Discussion	14
5.1 Android Application	14
5.2 Python-based Server & SUMO Simulation	17
5.2.1 SUMO Simulation	17
5.2.2 Python-based Server	20
6.0 Test/Experimental Results and Discussion	23
Test 1: Testing of the Sumo Simulation	24
Test 2: Testing of python script with the Sumo simulation	27
Test 3: Testing the communication of the app and the server	29
Test 4: Testing of saving data collected to the database	31
Test 5: Testing of the whole application	33
7.0 Conclusion and Future Work	40
7.1 Future Work / Improvements	40
7.2 Links	40
8.0 References	41
9.0 Appendices	42
9.1 Code snippets from the python-based server	42
9.2 Code snippets from the Android app	42

1.1 List of Figures

Figure 1: Flow chart for the Android Application	9
Figure 2: Final App Design	11
Figure 3: Flow chart for the python-based server	13
Figure 4: Blueprint for the app design	15
Figure 5: Design of the app	16
Figure 6: 601 bus route	18
Figure 7: 900-bus route	19
Figure 8: Manager Object for shared variables	20
Figure 9: Vehicle parameters from the SUMO simulation	21
Figure 10: Template of excel spreadsheet	22
Figure 11: Passenger walking on the road towards a bus stop	25
Figure 12: Bus stuck behind passenger	25
Figure 13: Passenger counted at a bus stop	26
Figure 14: Passenger not counted at a bus stop	26
Figure 15: Results from test 2 (0 passengers on board)	27
Figure 16: Results from test 2 (1 passenger on board)	27
Figure 17: Results from test 2 (2 passenger on board)	28
Figure 18: Results from test 2 (next bus stop change)	28
Figure 19: Server communication with the client	30
Figure 20: App display when requesting data from the server	30
Figure 21: Outcome from excel test	32
Figure 22: Test 5 Scenario 1 Application	34
Figure 23: Test 5 Scenario 1 Database	35
Figure 24: Test 5 Scenario 2 Application	36
Figure 25: Test 5 Scenario 2 Database	37
Figure 26: Test 5 Scenario 3 Application	38
Figure 27: Test 5 Scenario 3 Database	39
Figure 28: 2 options for generating passengers in the simulation	42
Figure 29: Creating a socket connection from the server	42
Figure 30: Client socket connection to the server	42

1.2 List of Tables

Table 1: Tests completed for the project	23
--	----

2.0 Introduction

The aim for this project is to design and develop a human-in-the-loop simulation platform, based on SUMO, for gathering data on human decision making when boarding public transport. The application has the capability to obtain bus data from a simulation such as passengers on board and time of arrival, display this data for a user to see, and allows the user to select the best option and provide a response as to why the user chose that option. A server running the simulation will be able to receive and store the data obtained from the application.

SUMO, also known as Simulation of Urban Mobility, is an open-source traffic simulation that can simulate a variety of traffic scenario. For the purposes of this project, SUMO is being used to simulate a portion of the 900-bus route from Caulfield to Stud Park Shopping Centre (Rowville) taken from Public Transport Victoria. The route being simulation is from the bus stop at Chadstone Shopping Centre in Chadstone towards the bus stop located at Monash University in Clayton, taking in all the 8 bus stops that are located within this section.

Currently there are projects being developed around the world in gathering data on human decision making when boarding public transportation. These projects involve creating methodologies to study commuter's preference, conducting research to understand the reasons behind avoidance of using public transportation, and following the travel experiences of regular public transport users to understand the positive user experiences and travel needs of the users. While these projects are contributing to the research into the topic, this project is more of a design and build project. The purpose is to build an application that has the potential to improve the uptake of public transportation by improving customer experience. The application has the ability for users to voice their thought process when boarding public transportation and these results can be analysed to improve public transportation in the future.

The objectives of the project have been split into 3 different sections. At a high level, the application with the user installed on their phone will be able to communicate with a SUMO simulation of a bus route, the application will show the user details of buses including time of arrival, number of passengers already on board and number of seats taken, and the application will allow the user to record their reasoning behind their decision on which bus to take: the next one to arrive or a bus further away along the route. These high-level requirements have been broken down into functional, non-functional, and communication requirements that the application will require to have.

3.0 Literature Review / Background

Work in the field has been occurring around world about user decision making while using public transportation. Currently there is a project being conducted by a team from the Massachusetts Institute of Technology and Singapore-MIT Alliance for Research and Technology. This project's goal is to create a methodology to study commuters' preferences when travelling. This is done using GPS enabled devices such as mobile devices, followed by a survey to gather information [1].

Another project being conducted by the Urban Mobility Lab at the Massachusetts Institute of Technology is about the impacts of Covid-19 on the industry. This project tries to understand the long-term effects that have occurred in behaviour and preference changes of public transit users. Key factors that are being assessed are the residual fears and new habits that have formed once the public health threat is gone. This project has exclusive access to transit cards and trip data across 3 major cities around the world, and surveys provided to passengers to reach their goals [2]. This project is very interesting as it is related to the current pandemic happening around the world.

A paper titled 'Factors affecting modal choice in urban mobility' by Yannis Tyrinopoulos and Constantinos Antoniou aims to understand the key factors that discourage users to take public transportation. The cast study for this project took place in Kalamaria, a densely populated suburb of Greece. The method of testing involved a detailed survey to understand the needs and requirements of the residents of the town as well as their travel choices. A second survey outlined the demographic and socioeconomic questions. The reasoning behind the two surveys taken in this study was to decrease the response bias by making participants not aware of their socioeconomic status when answering the first survey. The outcome from this paper summarises the results drawn from the participants. Key results found that high fare levels do not discourage passengers then previously thought, while other issues such as unreliability and overcrowding play a higher role in discouraging passengers. The report also concluded that parking availabilities in the city influences residents to drive and suggested decreasing public parking spaces to increase intake of public transportation [3].

An article published in 2018 investigated the preferences and requirement public transport users have with an application that trip planning data. The application created, Travel Companion, is a front-end user interface providing door-to-door travel experience. The article tested for the performance expectancy, usability, reliability, habit, and compatibility of the application. The research conducted involved analysis of interaction points, interviews, and workshops to improve the design and functionality of the travel companion application [4].

A journal paper from the 22nd International Academic Mindtrek Conference held in October 2018 focuses on creating a more pleasurable travel experience of users' public transport especially bus transportation. Investigation into passenger needs and bus travel experiences was conducted with a three-week field study. The study conducted involved 10 participants, who are regular users of bus transportation, to understand their travel needs and focused on passenger experience and needs [5].

4.0 Overview

This project had 2 main sections that were produced. The first being the application and the second being the python-based server with the SUMO simulation.

4.1 Android Application

The Android application is a phone app that the user has on their phone. The purpose is to provide the user with information gathered from the SUMO simulation and present this in a manner that is easiest for the user. It will allow the user to view this information, select a bus that best suits the user's needs and allows the user to provide a small description behind their thought process when choosing the best bus option. The information presented to the user is being sent from the server which is running the SUMO simulation and all inputs made by the user to the application will be sent to the server.

Figure 1 below illustrates how the Android application will run and communicate with the server.

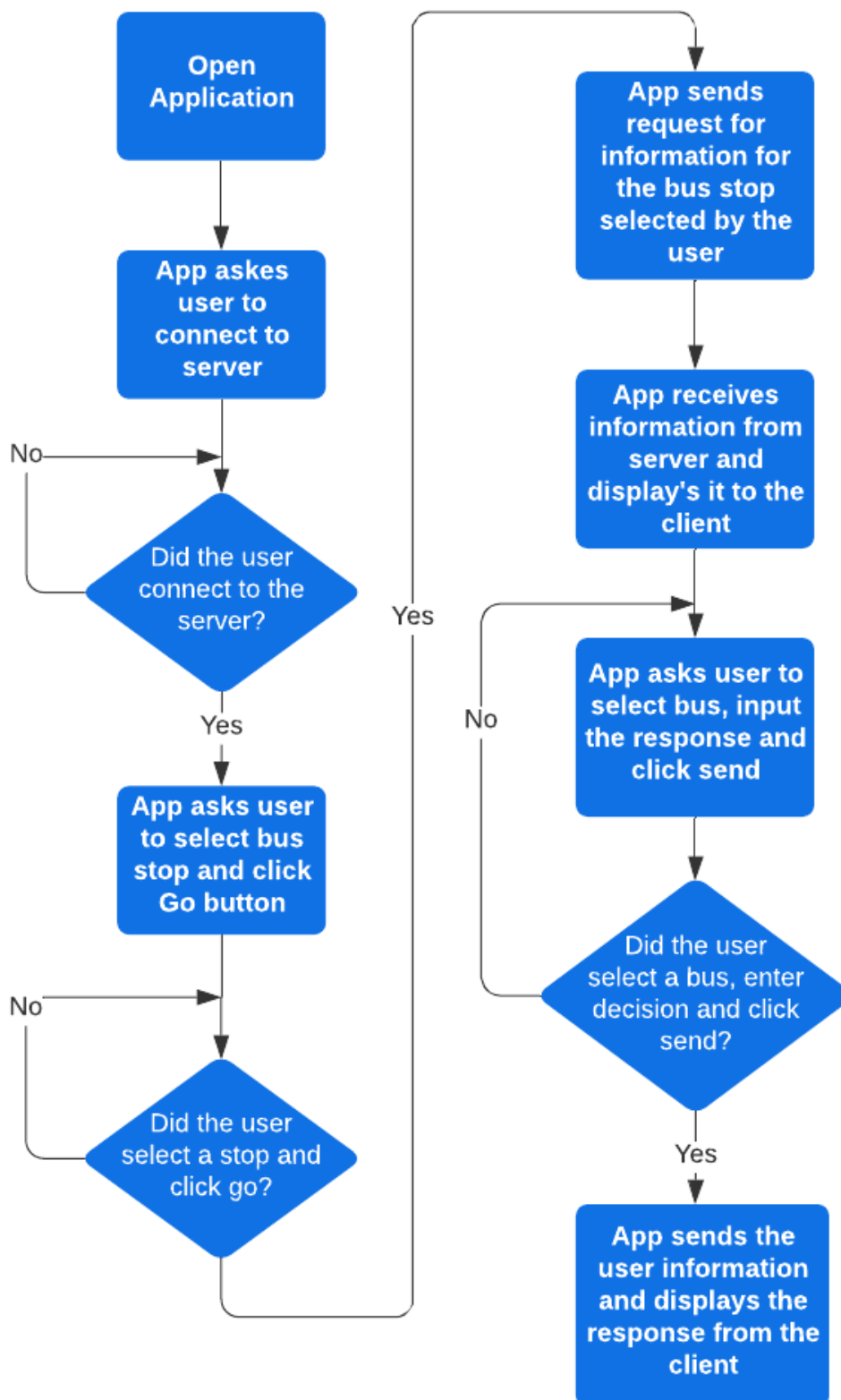


Figure 1: Flow chart for the Android Application

The app is designed to be easy to use with instructions written informing the user how to use the app. Figure 2 below shows a snapshot of the application in use. The app is broken down into rows of information. The first being the application name 'Bus Stop & Go'.

Following this is the title text. This text shows the instructions on how to use the app and what stage of the app the user is at. Once the user has performed a task, the text will update and let the user know the next task that needs to be completed

Following this is the bus stop selection row. Here the user will be able to select a bus stop and click 'go' to view bus information for that selected stop.

The next information on the app is the bus data. This information takes up much of the screen and is the focus point for the app. It shows information regarding buses that are arriving to the bus stop including the estimated time of arrival for the bus, the number of passengers on board and the number of seats taken.

Below the bus information is where the user can provide their input. The user can select the bus that best suits their needs at that time and provide a small description of their thought process when making their decision.

The next row provides buttons that the user can interact with. The first being the connect/disconnect button. Here the user can either connect to the server or disconnect once the session is finished. The next button is the 'refresh' button. Once there is bus data being displayed on the app, the user can click the 'refresh' button to update the information. The last button is the 'send' button. Once the user has made their decision on the bus to take and provides a small description, they can click the 'send' button to send their response to the server.

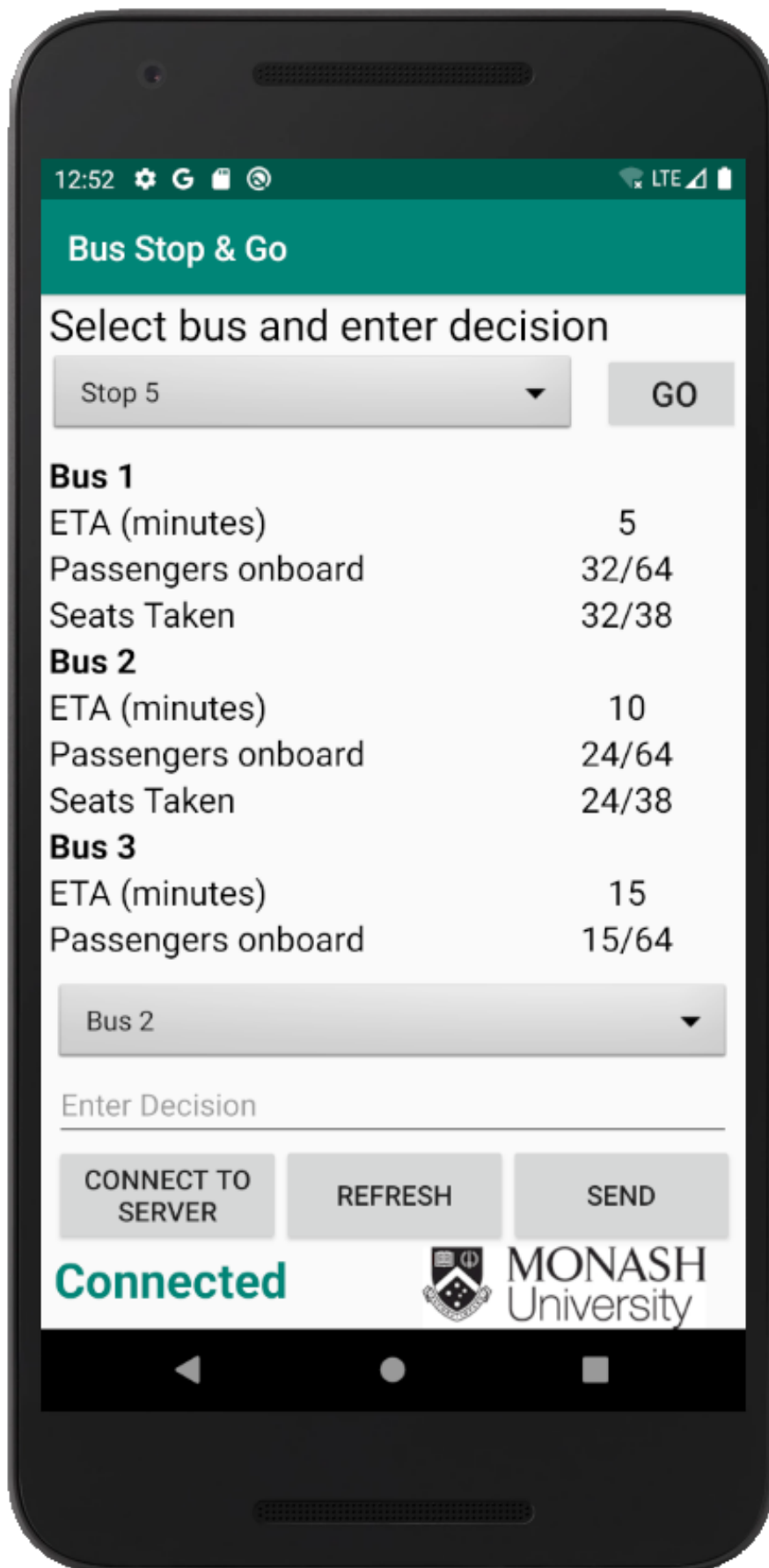


Figure 2: Final App Design

4.2 Python-based server & SUMO simulation

The bus route modelled in the SUMO simulation was modelled from the 900-bus route taken from Public Transport Victoria. This simulation views a small section of the 900 bus from Chadstone Shopping Centre to Monash University Clayton Campus and all 8 bus stops within.

The simulation is being run through the python-based server. The server is able listen for incoming connections from the Android application, open and run the SUMO simulation and open an excel workbook to save output results into. Throughout this, the server is also actively listening for information requests from the client and responding with data obtained from the running simulation. Once the client has finished and is sending its final user input, the server saves this data in the opened excel workbook in a new sheet.

Figure 3 below illustrates how the python-based server and the SUMO simulation will run and communicate with the Android application.

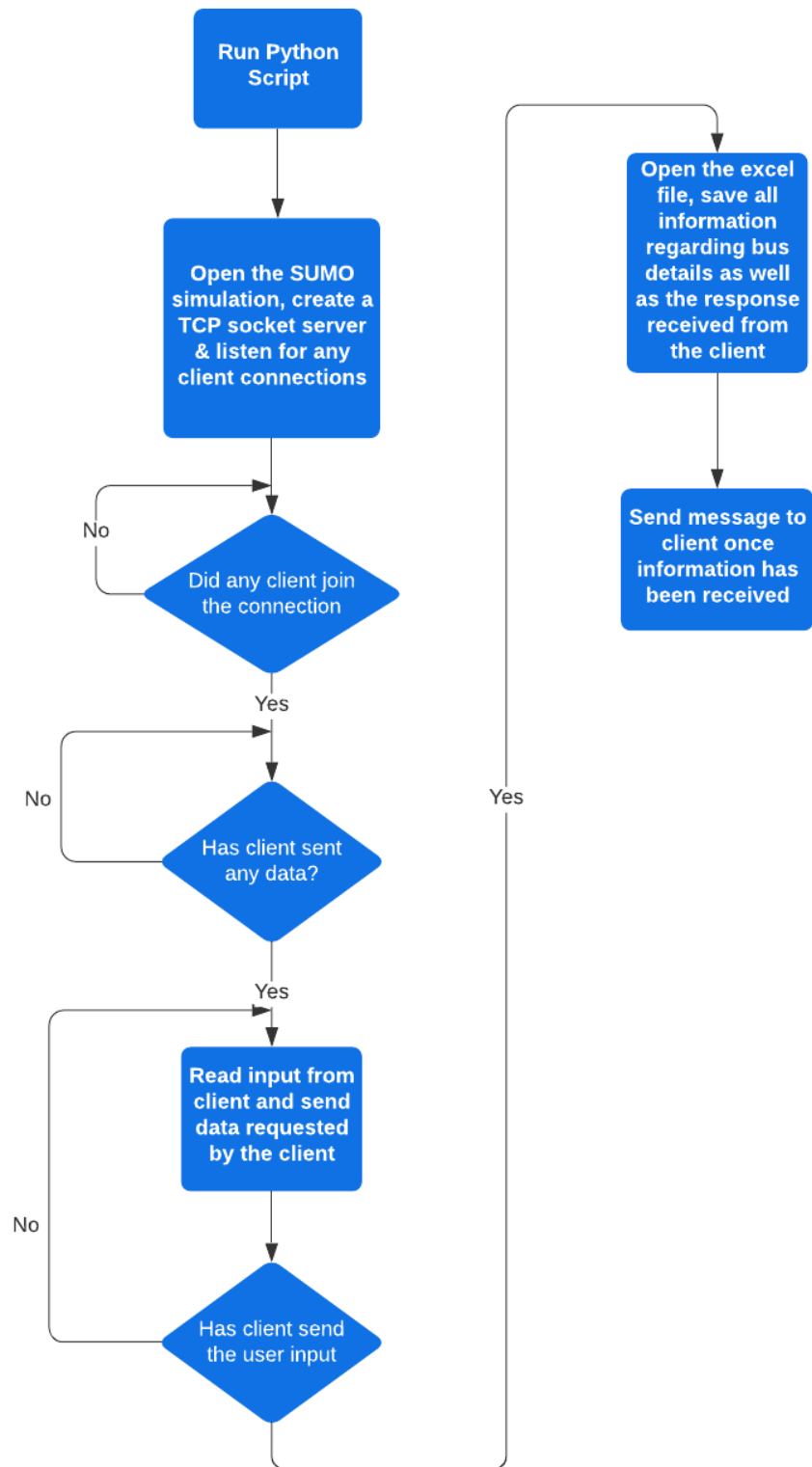


Figure 3: Flow chart for the python-based server

5.0 Detailed Discussion

5.1 Android Application

The application was designed and built using Android Studio. During the project there were 2 main operating systems that could have been chosen to build the app on, Apple's iOS and Android OS. Upon further investigation, Android OS with Android Studios was chosen as the operating system for the app. This was mainly due to the high number of users in the community and the coding language used. Android OS apps are written in Java which is a well-known language that is widely used and has many online resources for. Compared to this, iOS uses the swift coding language which is new compared to Java and only used for apple devices. This means there will be less resources online to provide support for when completing the project.

The application was designed using extensible markup language on android studio. Figure 4 and Figure 5 below provide a blueprint and layout design of the app. In Figure 4 it is shows via the blueprint that the app is designed in a table view with all its text boxes clearly visible. This layout format was chosen as the data being displayed was repetitive with each bus having the same data attributes. This made it easy to implement a table as each bus required only 4 rows of information. Due to the amount of data being displayed here, and the screen size available, a scrollable design was implemented to allow the user to experience all information via swiping up in the middle section.

The rest of the display consisted of buttons and spinners (drop down lists). Each button serves a purpose with the first button 'go' being used in conjunction with the spinner next to it. The combination of the button and spinner allows the user to select a stop and request information regarding that stop. The design of the app also included multi-use buttons such as the 'Connect to Server' button. Once this button is clicked and the app successfully connects to the server, the button can then be used again to disconnect from the server.

In the send section of the design, 3 different user interface features are being used together. A spinner, a user input text box and a button are all used to provide the user with the ability to provide feedback on their decision-making abilities and send that information to the server.

The status bar at the bottom of the design was implemented for user convenience so the user is aware of the connection status of the application. Once the app connects to the server, the status bar will change to reflect this.

While the original goal for the application was to use the GPS sensors on the mobile phone to automatically select a bus stop when the user arrives within a radius of the bus stop, issues came up with build of this functionality. This feature required users testing the application to be physically at the bus stop location for the app to work. Due to the current coronavirus restrictions in place testing of this functionality to ensure proper usage was not possible resulting in this feature being phased out of the final design.

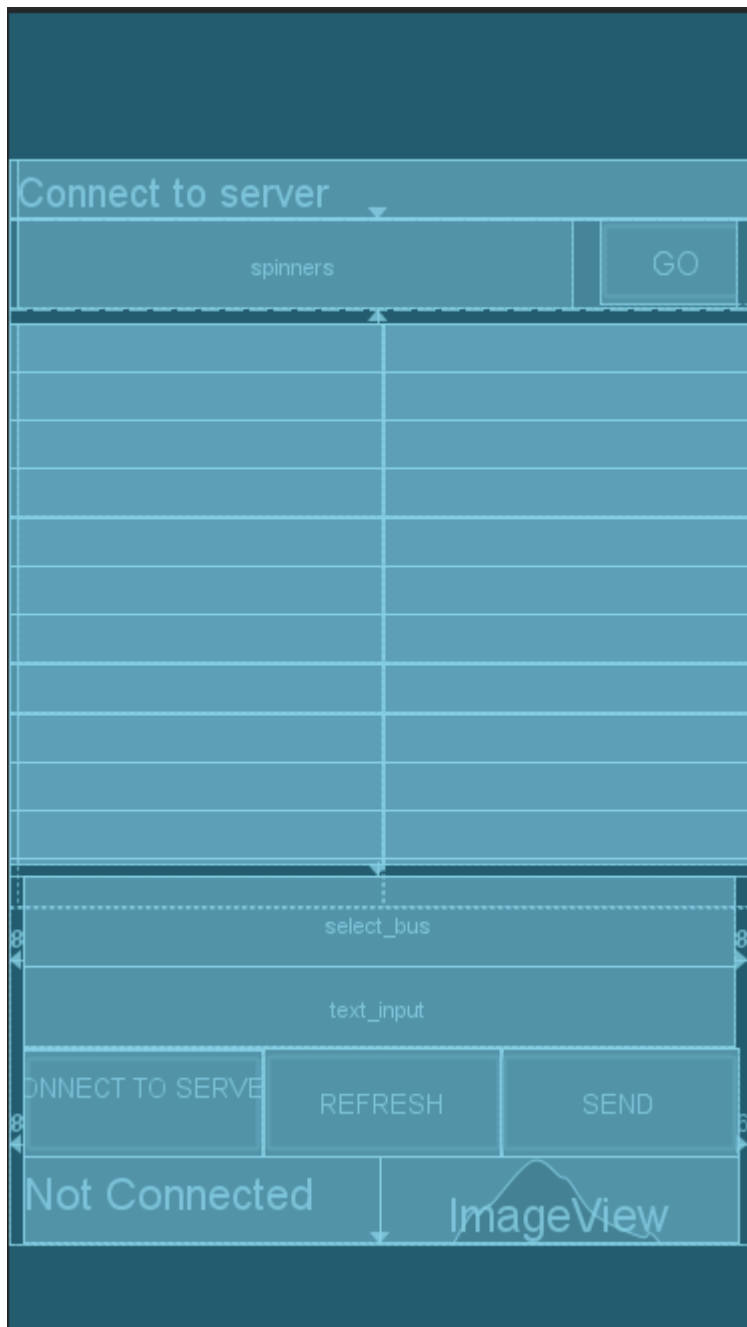


Figure 4: Blueprint for the app design

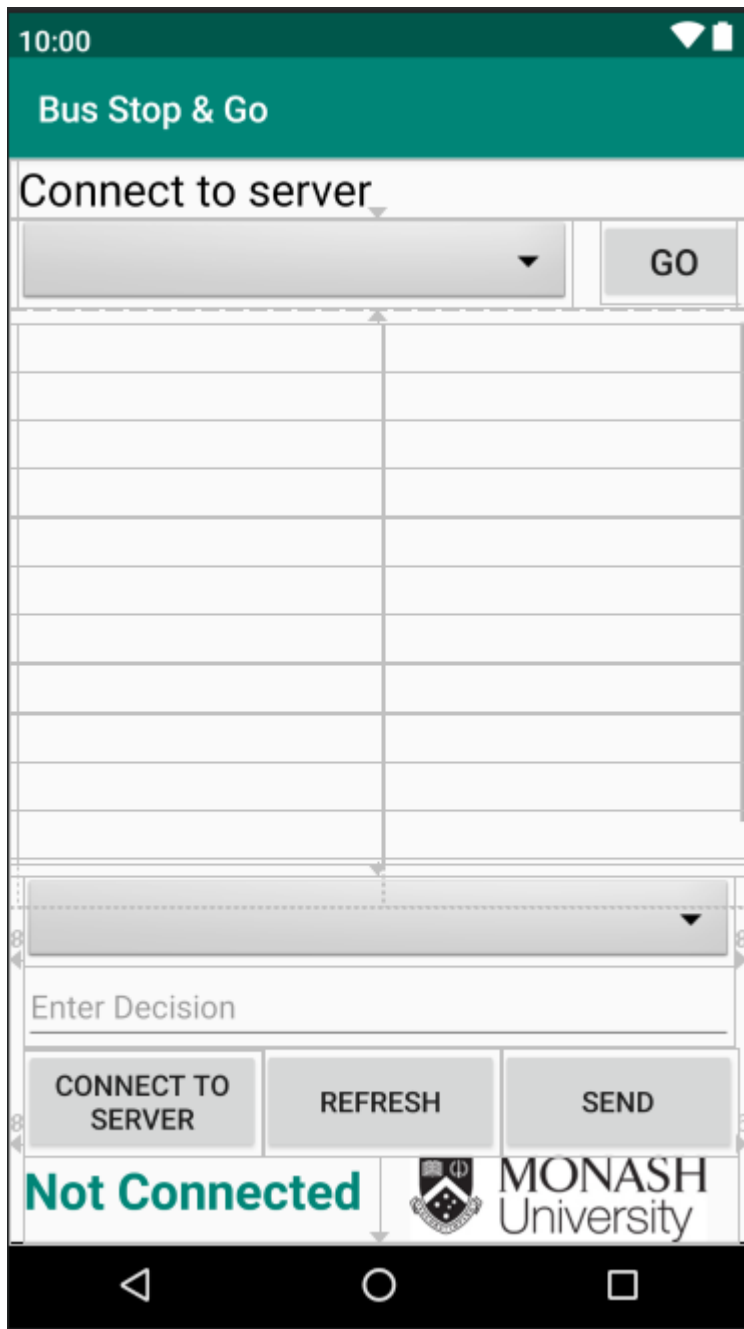


Figure 5: Design of the app

5.2 Python-based Server & SUMO Simulation

5.2.1 SUMO Simulation

The traffic simulation was simulated with the Simulation of Urban Mobility Software. The SUMO software was suggested by the project brief and was ultimately chosen to be the best solution for simulating real work traffic.

This project had the opportunity to be completed without a simulation, by having randomised numbers coded for the estimated time of arrival, passenger numbers and seats taken written directly into the python code. However, this method for the project wouldn't be able to provide a more real-world situation with real world problems that may arise. This includes delays due to traffic, stopping at each bus stop and traffic light cycles that may influence data obtained from the simulation. Due to this, the simulation was the better choice as it can simulate more real-world scenarios that can be tested and can influence a humans decision making abilities. SUMO also provided the freedom to easily change parameters of the simulation to see the effect of different scenarios such as a change in passenger numbers, adding/removing bus stops and adding more buses per hour to see how each bus manages the passenger numbers.

A bus route needed to be chosen for the project and one that started at or ended at Monash University was ideal as it could provide a better public transport experience for the hundreds to thousands of students that take public transport to university and could help encourage more students to take public transportation instead of driving. This made the 601 bus from Huntingdale Station to Monash University the best candidate. It was originally chosen as many students from Monash University take this bus with students seen lining up every day at the bus stop to take this bus. Due to its high demand and overcapacity of students to buses, it seems like a fine choice. However, this route did have one flaw, that it was an end-to-end bus route with no stops from the first and last bus stop. As the aim for the project was to have users viewing buses arriving to a stop to take them to another stop, this bus route was not suitable for the project. Another bus route needed to be used, and from research conducted on the Public Transport Victoria website a section of the 900-bus route was more suitable for the project. This route started at Chadstone Shopping Centre and ended up at Monash University with 8 total bus stops providing. Figure 6 & Figure 7 below shows the two different route options. The benefit of using the 900-bus is that the route overlaps with the 601-bus route.



Figure 6: 601 bus route

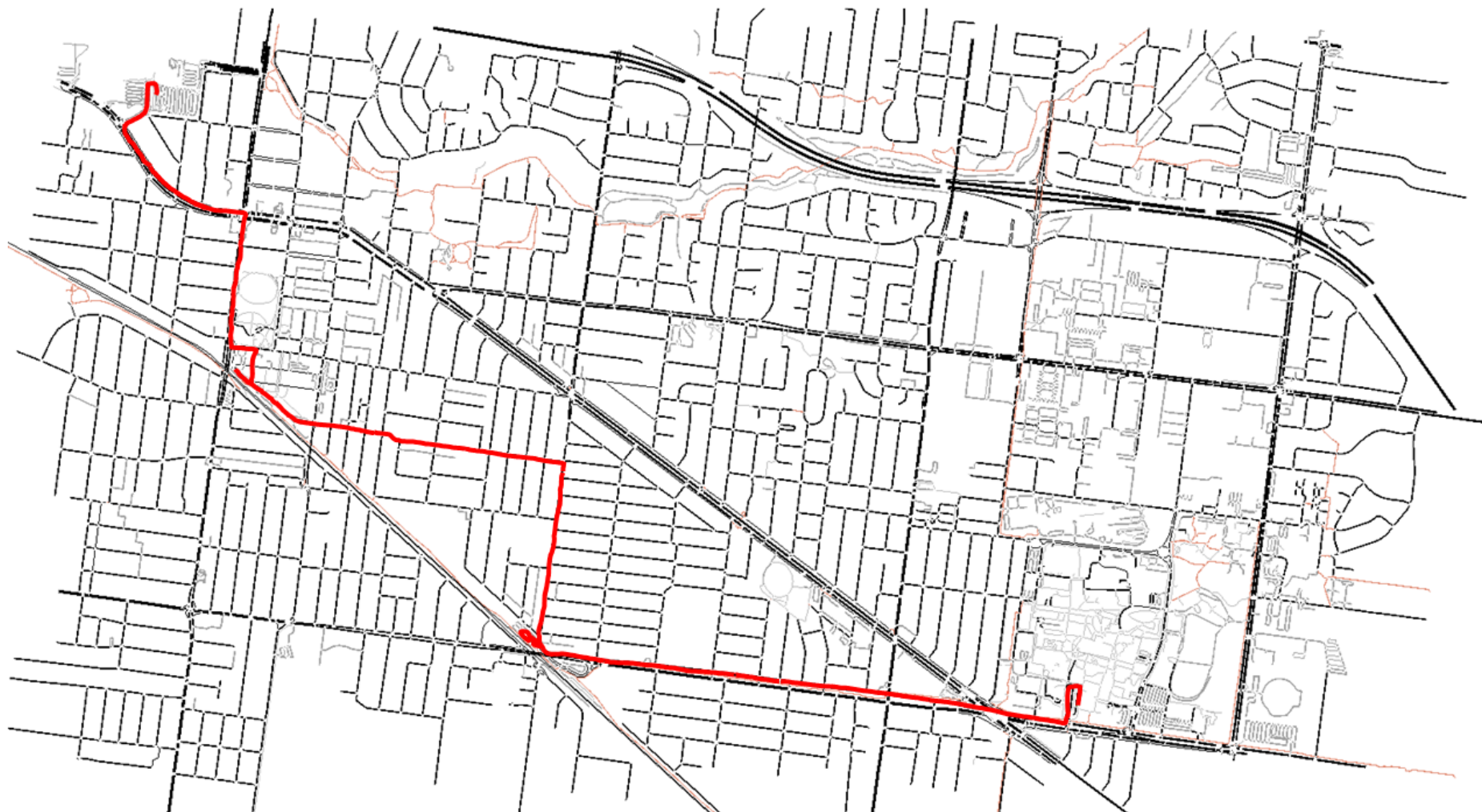


Figure 7: 900-bus route

5.2.2 Python-based Server

The aim for the server is to be able to handle communication from the Android application, the SUMO simulation and to the excel workbook. This means it needs to have integration with all 3 applications. SUMO has a traffic control interface (TraCI) that primarily has support with the python coding language using a TCP based architecture to handle the communication [7]. This allows the python code to act as a server for the simulation, and this method is also used with the connection from the python script and the Android application. The python script acts as a server by creating a socket connection and actively listening for clients to connect.

As a socket connection requires an actively listening server that can respond to requests from a client, a thread was used to create the server so it can actively run concurrently with the simulation. Within the server thread is a function that will consistently run when the server receives data from the client. This way all data being sent is received.

A key issue with the server being run in a separate thread to the main code is the sharing of information. As regular variables can't be passed like they are passed in functions and global variables are not suitable for this case. Manager objects are used to carry crucial information obtained from the simulation to the server thread to send to the client. The use of the objects allowed several pieces of information to be stored in each dictionary. This way all information regarding a bus can be stored in one dictionary as seen in Figure 8 below.

```
bus_info = manager.dict()
received = manager.dict()
solution = manager.Value(c_wchar_p, "")
```

Figure 8: Manager Object for shared variables

Obtaining data from the SUMO simulation via the python script requires the use of the TraCI library. This allows all functions to be used and all data regarding vehicles to be obtained. Figure 9 provides all the information that can be obtained from the simulation and sent to the application. To satisfy the aim of the project, only a select few items from this list are required. This includes the lane id, the remaining occupancy, stop information, bus line and passengers on board.

vehicle:bus_flow.1 Parameter		
Name	Value	Dynamic
lane [id]	791849034#5_0	✓
position [m]	11.80	
lateral offset [m]	0.00	
speed [m/s]	7.21	
lateral speed [m/s]	0.00	
acceleration [m/s^2]	1.79	
angle [degree]	277.57	
slope [degree]	0.00	
speed factor	1.00	✗
time gap on lane [s]	-1.00	
waiting time [s]	0.00	
waiting time (accumulated, 100.00s) [s]	15.00	
time loss [s]	184.90	
impatience	0.00	
last lane change [s]	-100.00	
desired depart [s]	154.00	✗
depart delay [s]	0.00	✗
odometer [m]	5115.54	
remaining [#]	62	✗
insertion period [s]	144.00	✗
stop info	next: busStop:601stop1	✓
line	bus	✗
CO2 [mg/s]	32694.48	
CO [mg/s]	48.31	
HC [mg/s]	8.40	
NOx [mg/s]	249.15	
PMx [mg/s]	5.13	
fuel [ml/s]	13.94	
electricity [Wh/s]	0.00	
noise (Harmonoise) [dB]	80.46	
devices	person	✗
persons	5	
containers	0	
lcState right	unknown	✓
lcState left	stay strategic	✓

Figure 9: Vehicle parameters from the SUMO simulation

To save all data obtained from the simulation and the application, an excel workbook is being used. With the integration of commands on python, it can be easily written to with all the information gathered. A simple layout was designed to make every session follow the same format. Figure 10 below illustrates the template being used for the storage of information. Firstly, each session will

have its own sheet for the information with a unique name. The title for each sheet reflects the date and time that the session was conducted. In the sample template below, this session was conducted on 16-Oct-2021 at 20:19:06. The template goes on to show each parameter for each bus as well as the values and finishes off with all the information obtained from the Android application at the bottom. As more sessions are run, new sheets will be created for each session following the same template illustrated in Figure 10

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Data gathered from session with app												
2													
3	Bus 1												
4	ETA	5											
5	Passengers Onboard	32/64											
6	Seats Taken	32/38											
7													
8	Bus 2												
9	ETA	10											
10	Passengers Onboard	24/64											
11	Seats Taken	24/38											
12													
13	Bus 3												
14	ETA	15											
15	Passengers Onboard	15/64											
16	Seats Taken	15/38											
17													
18													
19													
20	The following response was obtained from the app												
21	Bus stop that the user is at	Stop 5											
22	Bus selected by the user	Bus 2											
23	User reasoning behind the bus decision	Bus 2 is only 5 minutes after bus 1 and has a higher likelihood of having a seat available to sit on											
24													
25													
26													
27													
28													
29													
30													
31													
32													
33													
34													
35													
36													
37													
38													

Figure 10: Template of excel spreadsheet

6.0 Test/Experimental Results and Discussion

Tests

- Testing connection of client to server
- Testing of commands to gather information from sumo (TRACI commands)
- Viewing parameters to ensure matching with what was obtained (from sumo)
- Testing of bus carrying passengers (passengers waiting and then getting on the bus)
- Testing communication between server and app (sending and receiving data)
- Testing of excel sheet

Throughout the project several tests were conducted to ensure the application and the simulation was working correctly. These have been recorded in the following table and explained below:

Table 1: Tests completed for the project

Test Number	Description
1	Proper working off the sumo simulation with passengers waiting at the bus stop and on buses
2	Proper working of the python-script with the sumo simulation with data being sent from the simulation to the server
3	Testing of the connectivity of the app to the server
4	Testing of save storage of information obtained in the simulation into the excel database stored
5	Testing of the app as a whole

Test 1: Testing of the Sumo Simulation

Once the simulation was created, testing was done to ensure all features work correctly. This involved running the simulation on its own to ensure all entities that were coded into the simulation are showing up and can be viewed. The criteria for this test involved:

- Ensuring all bus stops are displayed
- Viewing the route each bus takes to ensure it is the correct route planned for the project,
- Ensuring buses stop at every bus stop within the route
- Ensuring all passenger entities are generated at the correct bus stop and are boarding the next arriving bus
- Ensuring all passenger entities are disembarking the bus at the correct stops.

The following procedure was followed for completing this test. To start the test, the simulation was simulated as a standalone without the python script. This was done as the test was to ensure correct working operation for the simulation and not the python script. To run the test all that was required was to run the simulation and view each bus stop for approximately 5 minutes to ensure a successful outcome because the simulation runs as a repeated loop. The test concluded with a positive results with only one issue.

When testing the simulation, issues arose with how the passengers behave before boarding a bus at a bus stop. Initially there was a plan to have passengers waiting at a bus stop counted as this could be a potential new added feature that can enhance the user experience on top of the already defined requirements for the project. However, when this feature was tested the results were not satisfactory as passengers were starting the simulation away from the bus stop and walking onto the road when walking to the bus stop. This affected buses as the bus would need to stay behind a passenger walking on the road and can not move until the passenger reaches the bus stop. Figure 11 & Figure 12 both show how the passengers are walking on the road to get to the bus stop. Figure 12 was the cause to have the passenger behaviour changed. It shows a bus driving behind a passenger unable to get ahead of the passenger that is walking on the road to the bus stop. As more passengers arrive to the stop, more buses were held up the same way.

To fix this issue, passengers would need to start their journey already at a bus stop waiting for a bus rather than walking to the bus stop to wait for a bus. This however resulted in the passenger not being counted at a bus stop even though they are waiting for a bus. Figure 13 shows how a passenger, who walked on the road getting to the bus stop, is counted to be at the stop, while Figure 14 shows how a passenger, who started their journey at the bus stop waiting for the bus, is not counted to be at the stop. While this fixes the issue of passengers affecting buses on the road, it results in the feature above to not be possible and resulted in it being removed. Code for this solve can be found in Figure 28 located in the 9.0 Appendices

Once the new solution was implemented, the test was repeated with all criteria mentioned above working correctly.

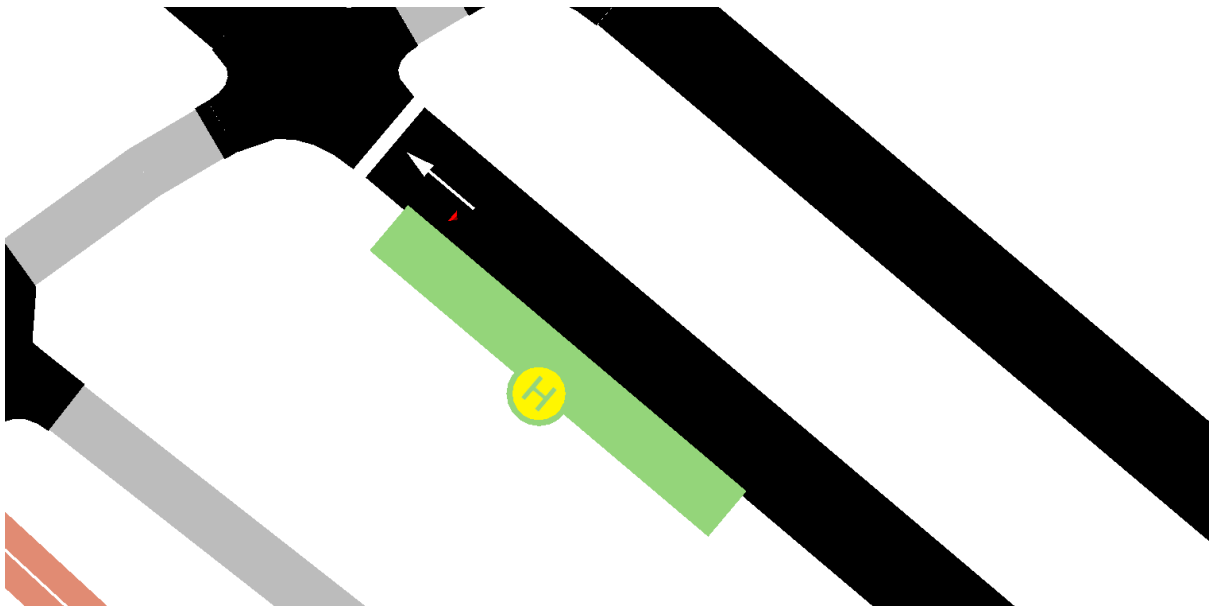


Figure 11: Passenger walking on the road towards a bus stop

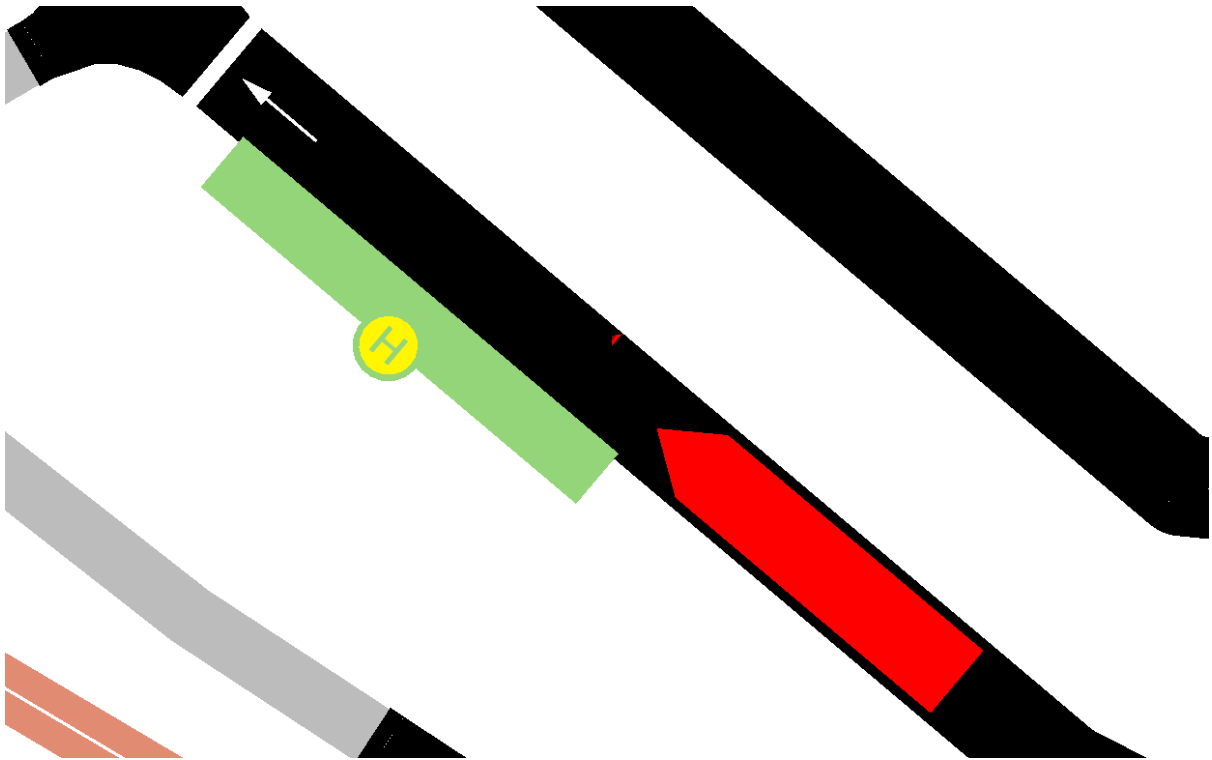


Figure 12: Bus stuck behind passenger

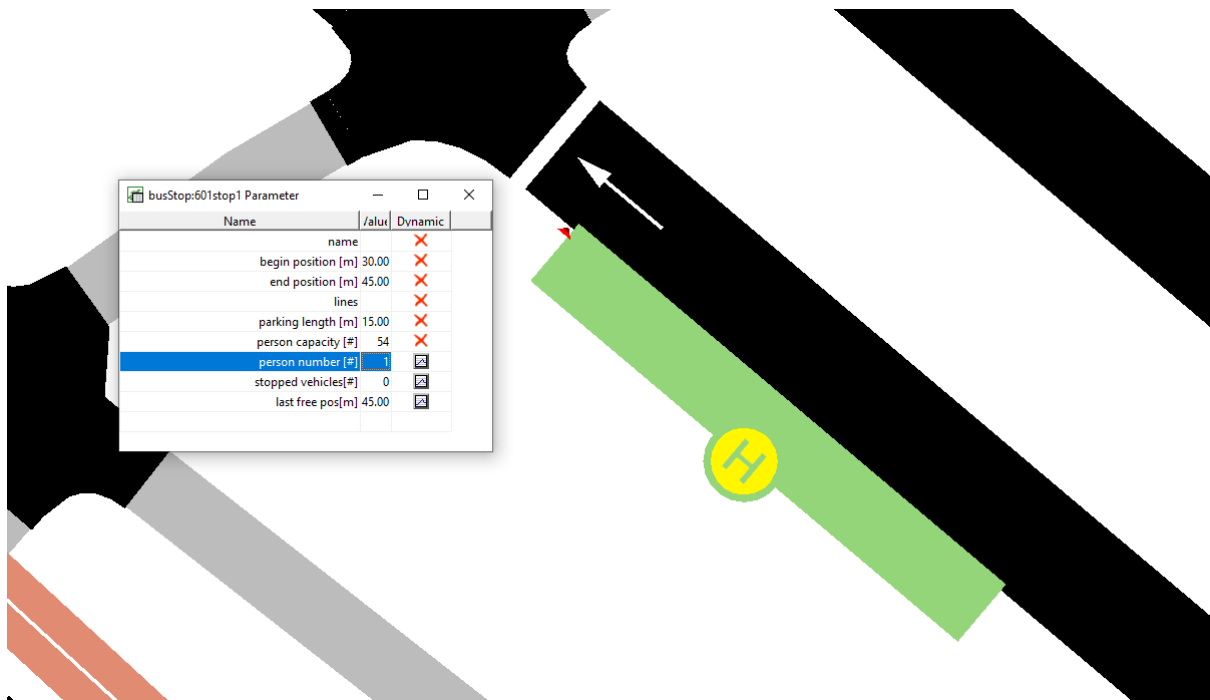


Figure 13: Passenger counted at a bus stop

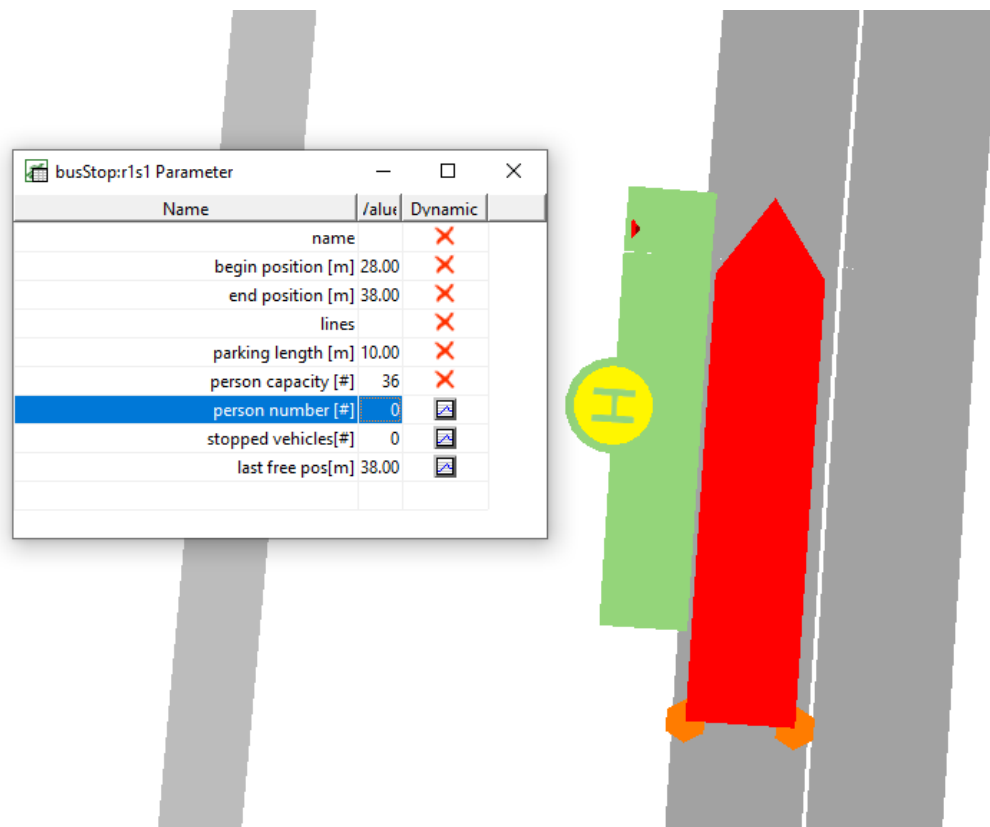


Figure 14: Passenger not counted at a bus stop

Test 2: Testing of python script with the Sumo simulation

The next test conducted for the project involved proper communication between the simulation and the python-based server script. This test was conducted to ensure the following criteria were met:

- The server was able to start the simulation
- The server was able to request data from the simulation
- The server was able to receive data from the simulation

The following procedure was followed for completing this test. To start the test, a command prompt window was opened to the directory where the python script and the simulation files are stored. The python script was run with the following command “sumodemo.py”.

These steps outlined above can run the python script which will in turn run the simulation and request information from it. For testing purposes, the following values were being obtained from the simulation:

- A list of names of all buses currently in the simulation
- The next bus stop for the buses in the simulation
- The current lane the bus is at
- The number of passengers on board the bus

The information obtained from the simulation was then printed out onto the command prompt window to validate the results with the simulation. Figure 15, Figure 16, Figure 17, and Figure 18 all provide the results from the test. These results show that the python script can successfully gather information from the SUMO simulation as it runs in real time. Figure 15 shows the original data obtained for the single bus on the simulation. Figure 16 provides an updated passenger number when a passenger boards the bus. Figure 17 also provides a passenger update when another passenger boards the bus. Figure 18 provides an update when the bus leaves the first stop and is heading for the next stop.

```
-----  
The following information was obtained from the simulation  
  
List of buses in the simulation: ('bus_flow.0',)  
Next stop for the bus: r1s1  
Current road lane the bus is at: -787851477  
Number of passengers on board: 0  
-----
```

Figure 15: Results from test 2 (0 passengers on board)

```
-----  
The following information was obtained from the simulation  
  
List of buses in the simulation: ('bus_flow.0',)  
Next stop for the bus: r1s1  
Current road lane the bus is at: -787851477  
Number of passengers on board: 1  
-----
```

Figure 16: Results from test 2 (1 passenger on board)

```
-----  
The following information was obtained from the simulation  
  
List of buses in the simulation: ('bus_flow.0',)  
Next stop for the bus: r1s1  
Current road lane the bus is at: -787851477  
Number of passengers on board: 2  
-----
```

Figure 17: Results from test 2 (2 passenger on board)

```
-----  
The following information was obtained from the simulation  
  
List of buses in the simulation: ('bus_flow.0',)  
Next stop for the bus: r1s2  
Current road lane the bus is at: -787851477  
Number of passengers on board: 2  
-----
```

Figure 18: Results from test 2 (next bus stop change)

Test 3: Testing the communication of the app and the server

The following test conducted for the project involved testing to ensure the TCP socket communication link between the Android app and the python-based server can be created and used to send and receive data across the two systems. This test was conducted to ensure the following criteria were met:

- The app can connect to the server via a push of a button
- The app can request information via a push of a button and receive information from the server
- The server can listen for incoming connections and connect to the first connection request
- The server can read incoming messages from a client and respond with the information requested

The following procedure was followed for completing this test. To start the test, a command prompt window was opened to the directory where the python script and the simulation files are stored. The python script was run with the following command "sumodemo.py". The Android application was opened on the emulator. For testing purposes, the following actions were performed by the user:

- The user pressed the "Connect to Server" Button
- The user selected "Stop 2" from the list of stops
- The user pressed the "Go" button

The output from this task is expected to provide the following results:

- The status bar on the bottom of the app changes from "Not Connected" to "Connected"
- The app should provide information regarding a single bus, with the estimated time of arrival to "Stop 2" as selected in the procedure, the number of passengers on board and the number of seats taken
- The server should provide feedback when the app connects to the server
- The server should provide feedback when the app requests data from the server

Figure 19 and Figure 20 below provides the results from the test conducted. Figure 19 shows the server communication with the application. The communication starts with the python script launching the SUMO simulation and the server. Once the user selected "Connect to Server", the server registers this connection records the date and time of the connection and outputs it onto the console. Once the user selects a stop and clicks "go" the server receives this request and responds with the data it obtains from the simulation. Figure 20 provides the application output from the test. Once the procedures for the test are followed, the output of the app shows the single bus in the simulation with the 2 passengers on board and the estimated time of arrival calculated. These results are what is expected with the test conducted.

```

=====
Beginning the main program.
=====

Connecting to SUMO via TraCI.

Launching the server.
The server has been launched.
Made a connection with ('127.0.0.1', 63070) on 20 Oct 2021 at 02:51:10.
Starting the server thread
Receiving data from client.
Receiving data from client.
Client requested bus information for stop 2.

```

Figure 19: Server communication with the client

Bus Stop & Go

Select bus and enter decision

Stop 2 ▼ GO

BUS 1

ETA (minutes)	12
Passengers onboard	2/64
Seats taken	2/38

Bus 1 ▼

Enter Decision

CLOSE SERVER REFRESH SEND

Connected MONASH University

Figure 20: App display when requesting data from the server

Test 4: Testing of saving data collected to the database

The following test conducted for the project involved testing to ensure data on the python-based server can be saved into an excel spreadsheet. This test was conducted to ensure the following criteria were met:

- The server can open the pre-defined excel spreadsheet
- The server can create a new page in the spreadsheet
- The server can store data in variables into the spreadsheet

The following procedure was followed for completing this test. To start the test, a command prompt window was opened to the directory where the python script and the simulation files are stored. The python script was run with the following command “sumodemo.py”. The Android application was opened on the emulator. For testing purposes, the following actions were performed by the user:

The following procedure was followed for completing this test. To start the test, a command prompt window was opened to the directory where the python script and the simulation files are stored. The python script was run with the following command “sumodemo.py”.

These steps outlined above can run the python script which will in turn run the simulation, request information from it and save it to the excel spreadsheet. For testing purposes, the following values were being obtained from the simulation:

- A list of names of all buses currently in the simulation
- The next bus stop for the single bus in the simulation
- The current lane the bus is at
- The number of passengers on board the bus

The expected outcome for this test is a new excel sheet containing the information requested above. Figure 21 provides the results from the test conducted. It shows an accurate list of values obtained from the simulation. These values are the same values obtained in Test 2 for testing the communication between the server and the SUMO simulation. The exact values can be seen in Figure 17.

	A	B	C	D	E	F
1	The following information was obtained from the simulation					
2						
3	List of buses in the simulation: ('bus_flow.0',)					
4						
5	Next stop for the bus: r1s1					
6						
7	Current road lane the bus is at: -787851477					
8						
9	Number of passengers on board: 2					
10						
11						
12						
13						

Figure 21: Outcome from excel test

Test 5: Testing of the whole application

The final test completed was the test of the entire project with all parts working together. This involved communications between the server and the simulation, the server and the client, and the server and the excel database. This test was conducted to ensure the following criteria were met:

- All communication with every component is working correctly
- All information is sent and received correctly
- Formatting of the excel sheet is working

The following procedure was followed for completing this test. To start the test, the python-based server needed to be started. A command prompt window was opened to the directory where the python script and the simulation files are stored. The python script was run with the following command "sumodemo.py".

The Android application was launched via the emulator built into Android Studio and was connected to the server. The following user inputs were added:

- User selects "Stop 5"
- User selects bus option 2
- User provides the following feedback: "Bus 2 was the best option as the estimated time of arrival fit well with my schedule"
- User sends the information to the server to save.

This test was run multiple times with different user inputs to ensure the system was working correctly. The 2nd scenario follows the same set up procedure as the 1st but with different user inputs:

- User selects "Stop 3"
- User selects bus option 1
- User provides the following feedback: "I picked bus 1 as it was the earliest to arrive at my stop"
- Users send the information to the server to save.

The final scenario tested was completed with the same set-up procedures as the 1st with the following user inputs:

- User selects "Stop 7"
- User selects bus option 3
- User provides the following feedback: "Bus 3 is my choice as I have a relaxed day and am not in a rush"
- Users send the information to the server to save.

The expected outcomes of these tests are a new excel sheet for every test with all key information shared between the server and the app, as well as any information received from the app all while ensuring the systems work without crashes.

The following results were achieved from the tests conducted. Figure 22, Figure 24, and Figure 26 provide a look into the application for each scenario with the bus data received and the user input. Figure 23, Figure 25, and Figure 27 provide information on the database storing all the information

from each test scenario. The results look promising as all information on the application matches all information obtained from the excel database. The Figures below show how each attribute for a bus is shown in the corresponding database and how the user input is accurately displayed on the dataset. Each new sheet in the data also matches the time of the test conducted.

Bus Stop & Go

Select bus and enter decision

Stop 5 ▼ GO

BUS 1

ETA (minutes)	0
Passengers onboard	7/64
Seats taken	7/38

BUS 2

ETA (minutes)	3
Passengers onboard	9/64
Seats taken	9/38

BUS 3

ETA (minutes)	5
Passengers onboard	8/64

Bus 2 ▼

estimated time of arrival fit well with my schedule

CLOSE SERVER REFRESH SEND


Connected  MONASH University

Figure 22: Test 5 Scenario 1 Application

	A	B	C	D	E	F	G	H	I	J
1	Data gathered from session with app									
2										
3	Bus 1									
4	ETA	0								
5	Passengers Onboard	7/64								
6	Seats Taken	7/38								
7										
8	Bus 2									
9	ETA	3								
10	Passengers Onboard	9/64								
11	Seats Taken	9/38								
12										
13	Bus 3									
14	ETA	5								
15	Passengers Onboard	8/64								
16	Seats Taken	8/38								
17										
18	The following response was obtained from the app									
19	Bus stop that the user is at	Stop 5								
20	Bus selected by the user	Bus 2								
21	User reasoning behind the bus decision	Bus 2 was the best option as the estimated time of arrival fit well with my schedule								
22										
23										
24										
25										
26										
27										
28										
29										
30										
31										
32										
33										
34										
35										
36										
37										
38										
<div> ◀ ▶ Introduction 16 Oct 2021 at 20.19.06 20 Oct 2021 at 03.34.35 20 Oct 2021 at 03.36.48 21 Oct 2021 at 13.59.05 ⊕ </div>										

Figure 23: Test 5 Scenario 1 Database

2:14

Bus Stop & Go

Select bus and enter decision

Stop 3

GO

BUS 1

ETA (minutes)	2
Passengers onboard	2/64
Seats taken	2/38

BUS 2

ETA (minutes)	5
Passengers onboard	7/64
Seats taken	7/38

Bus 1

d bus 1 as it was the earliest to arrive at my stop.

CLOSE SERVER

REFRESH

SEND

Connected






 MONASH University

Figure 24: Test 5 Scenario 2 Application

	A	B	C	D	E	F	G	H	I	J	K	L
1	Data gathered from session with app											
2												
3	Bus 1											
4	ETA	2										
5	Passengers Onboard	2/64										
6	Seats Taken	2/38										
7												
8	Bus 2											
9	ETA	5										
10	Passengers Onboard	7/64										
11	Seats Taken	7/38										
12												
13												
14												
15												
16												
17												
18	The following response was obtained from the app											
19	Bus stop that the user is at	Stop 3										
20	Bus selected by the user	Bus 1										
21	User reasoning behind the bus decision	I picked bus 1 as it was the earliest to arrive at my stop.										
22												
23												
24												
25												
26												
27												
28												
29												
30												
31												
32												
33												
34												
35												
36												
37												
38												
<div> <div>Introduction</div> <div>16 Oct 2021 at 20.19.06</div> <div>20 Oct 2021 at 03.34.35</div> <div>20 Oct 2021 at 03.36.48</div> <div>21 Oct 2021 at 13.59.05</div> <div>21 Oct 2021 at 14.11.03</div> </div>												

Figure 25: Test 5 Scenario 2 Database

2:40





Bus Stop & Go

Select bus and enter decision

Stop 7

GO

BUS 1

ETA (minutes)	8
Passengers onboard	5/64
Seats taken	5/38

BUS 2

ETA (minutes)	10
Passengers onboard	8/64
Seats taken	8/38

BUS 3

ETA (minutes)	12
Passengers onboard	7/64

Bus 3


oice as I have a relaxed day and am not in a rush

CLOSE SERVER

REFRESH

SEND

Connected



MONASH University

Figure 26: Test 5 Scenario 3 Application

	A	B	C	D	E	F	G	H	I	J
1	Data gathered from session with app									
2										
3	Bus 1									
4	ETA	8								
5	Passengers Onboard	5/64								
6	Seats Taken	5/38								
7										
8	Bus 2									
9	ETA	10								
10	Passengers Onboard	8/64								
11	Seats Taken	8/38								
12										
13	Bus 3									
14	ETA	12								
15	Passengers Onboard	7/64								
16	Seats Taken	7/38								
17										
18	The following response was obtained from the app									
19	Bus stop that the user is at	Stop 7								
20	Bus selected by the user	Bus 3								
21	User reasoning behind the bus decision	Bus 3 is my choice as I have a relaxed day and am not in a rush								
22										
23										
24										
25										
26										
27										
28										
29										
30										
31										
32										
33										
34										
35										
36										
37										
38										

Figure 27: Test 5 Scenario 3 Database

7.0 Conclusion and Future Work

The aim for this project is to design and develop a human-in-the-loop simulation platform, based on SUMO, for gathering data on human decision making when boarding public transport. During the start of the project clear objectives were outlined, and requirements were made to ensure a successful outcome. The high-level objectives outlined included:

- The application with the user installed on their phone will be able to communicate with a SUMO simulation of a bus route
- The application will show the user details of buses including time of arrival, number of passengers already on board and number of seats taken
- The application will allow the user to record their reasoning behind their decision on which bus to take: the next one to arrive or a bus further away along the route

Of these high-level objectives, all were completed with a successful outcome. The experimental testing results provide detailed steps of several key functionalities the high-level objectives were broken down into. The application created can connect to the server running the SUMO simulation and retrieve information from the simulation to display to the user, the user is able to select one of the bus options and provide a reasoning behind their decision, and the server is able to save these results in a database for later use.

7.1 Future Work / Improvements

The project's aim was to build an application that can be used to gather human decision-making data and save it into a database. While this aim was achieved further work can be done in the future to provide better user experience of people using public transportation. Adding GPS functionality to the app would allow a better experience for the user as the app would be able to provide details for a stop if the user is already at the stop. This would improve the user experience as the steps required to use the app would be less.

Future developers can use the app to model different real-world scenarios to see how public transport users adapt to changes in public transport. This includes changes such as frequency of buses on a route and routes becoming more popular with passenger numbers increase. Future developers can also use the application to test out new bus routes to view the uptake of the route to determine how many buses may be needed to provide the best user experience.

7.2 Links

Link to the GitHub code repository:

<https://github.com/msin0004/FYP-PROJECT>

Link to the FYP video:

https://youtu.be/KUVOF1FN_1Q

8.0 References

- [1] "Urban Mobility Behaviors & Preferences Test Bed | INTELLIGENT TRANSPORTATION SYSTEMS LAB", Its.mit.edu, 2021. <https://its.mit.edu/urban-mobility-behaviors-preferences-test-bed>.
- [2] "Medium-term impact of COVID-19 on urban mobility: Behavior, preference, and energy consumption | MIT Urban Mobility Lab", Mobility.mit.edu, 2021. <http://mobility.mit.edu/barr-project>.
- [3] Tyrinopoulos, Y., Antoniou, C. Factors affecting modal choice in urban mobility. *Eur. Transp. Res. Rev.* **5**, 27–39 (2013). <https://doi.org/10.1007/s12544-012-0088-3>.
- [4] A. Dolinayova et al., "Research of the Passenger's Preferences and Requirements for the Travel Companion Application", *Journal of Advanced Transportation*, vol. 2018, pp. 1-12, 2018. Available: <https://doi.org/10.1155/2018/8092147>.
- [5] E. Hildén, K. Väänänen and S. Syrman, "Modeling Bus Travel Experience to Guide the Design of Digital Services for the Bus Context", *Proceedings of the 22nd International Academic Mindtrek Conference*, pp. 143-152, 2018. Available: <https://doi.org/10.1145/3275116.3275120>.
- [6] C. Ho, C. Mulley and D. Hensher, "Public preferences for mobility as a service: Insights from stated preference surveys", *Transportation Research Part A: Policy and Practice*, vol. 131, pp. 70-90, 2020. <https://doi.org/10.1016/j.tra.2019.09.031>.
- [7] N. Klein, "More than just a bus ride: The role of perceptions in travel behaviour", *Urban Studies*, vol. 54, no. 11, pp. 2490-2503, 2016. Available: <https://doi.org/10.1177/0042098016649324>.
- [8] "Python: module traci._vehicle", *Sumo.dlr.de*, 2021. [Online]. https://sumo.dlr.de/pydoc/traci._vehicle.html.

9.0 Appendices

9.1 Code snippets from the python-based server

```
<!-- OPTION 2 -->
<personFlow id="BusPerson_0" begin="0" end="9000" personsPerHour="180" departPos="37">
  <ride from="-787851477" busStop="r1s8" lines="bus"/>
</personFlow>

<!-- OPTION 1 -->

<personFlow id="BusPerson_0" begin="0" end="9000" personsPerHour="180" departPos="37">
  <walk from="-787851477" busStop="r1s1"/>
  <ride busStop="r1s8" lines="bus"/>
</personFlow>
```

Figure 28: 2 options for generating passengers in the simulation

```
# create a socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# bind the socket to an address and port
host = '127.0.0.1' # localhost
port = 8080 # reserve a port for the service (i.e. a large number)
server_socket.bind((host, port)) # binds the socket to the hostname

# listen for incoming connections
server_socket.listen(backlog)
```

Figure 29: Creating a socket connection from the server

9.2 Code snippets from the Android app

```
clientSocket = new Socket( host: "10.0.2.2", port: 8080);
out = new BufferedWriter(new OutputStreamWriter(clientSocket.getOutputStream()));
in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
```

Figure 30: Client socket connection to the server