

**GPS SOLUTION FOR ACTIVE QUEUE MANAGEMENT USING ANDROID
PLATFORM
BY
LIU YU YAO**

A REPORT
SUBMITTED TO
Universiti Tunku Abdul Rahman
in partial fulfillment of the requirements
for the degree of
BACHELOR OF COMPUTER SCIENCE (HONOURS)
Faculty of Information and Communication Technology
(Kampar Campus)

MAY 2021

REPORT STATUS DECLARATION FORM

Title: GPS Solution for Active Queue Management Using Android Platform

Academic Session: May 2021

I, LIU YU YAO, declare that I allow this Final Year Project Report to be kept in Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.

Verified by,



(Author's signature)



(Supervisor's signature)

Address:

108-B, Kampung Dato Seri
Kamaruddin, 32040 Seri Manjung,
Perak

Ts Yeck Yin Ping

Supervisor's name

Date: 2/9/2021

Date: 2/9/2021

Universiti Tunku Abdul Rahman			
Form Title : Submission Sheet for FYP/Dissertation/Thesis			
Form Number: FM-IAD-004	Rev No.: 0	Effective Date: 21 JUNE 2011	Page No.: 1 of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
UNIVERSITI TUNKU ABDUL RAHMAN

Date: 2/9/2021

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that Liu Yu Yao (ID No: 17ACB00765) has completed this final year project entitled “GPS Solution for Active Queue Management Using Android Platform” under the supervision of Ts Yeck Yin Ping (Supervisor) from the Department of Computer Science, Faculty of Information and Communication Technology.

I understand that University will upload softcopy of my final year project in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

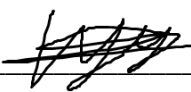
Yours truly,



Liu Yu Yao

DECLARATION OF ORIGINALITY

I declare that this report entitled “**GPS Solution for Active Queue Management Using Android Platform**” is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____ 

Name : _____ Liu Yu Yao _____

Date : _____ 2/9/2021 _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Ts. Yeck Yin Ping and my moderator, Dr. Ooi Boon Yaik for giving me this opportunity to engage in this project. This is my first step of establishing a career in IT field. A million thanks to my supervisor and moderator.

I am also like to thank my parents for supporting me throughout the course as well as their love and continuous encouragement.

Abstract

This proposed project proposes an active queue management system in multiple location integrated with GPS (Global Positioning System) Android mobile application. This project introduces a model for effective management of queue in different locations using GPS as a medium. Since most of the existing software on the market does not manage tickets in an efficient way and does not provide route planning function, this proposed project aims to solve the aforementioned problems by develop an Android mobile application which check travel time for user before taking ticket, allows user to delay ticket in case of not enough queue time and help user to plan route on multiple places as well as navigate user to the location. The application developed will help public to save their precious time by utilising the waiting time into doing something more worthy such as part of queue time. Since queueing in different locations is considered as a optimisation problem, several algorithms are reviewed to tackle the problem, such as brute force method, nearest neighbour algorithm, and branch and bound algorithm. Four tickets management applications, namely QueueBee, powerQ, WaveTec and QLess are reviewed to understand both strengths and weaknesses of existing application on current market. A comparison table is constructed to compare their functions to visualise differences between them. From the table, the proposed application can identify which function to focus on. Tools used to develop the project is Android Studio with Google Firebase as database of this project. This project uses prototype developing methodology which the application will improve on every deployment. If user has at least one ticket taken, system can generate an optimised route using branch and bound method. User can also send swap ticket request to next user. If the user accepts the request, both tickets will be swapped. The final deliverable of this project would be a multi-location queue management Android application with GPS supported.

TABLE OF CONTENTS

TITLE PAGE	i
REPORT STATUS DECLARATION FORM	ii
FYP THESIS SUBMISSION FORM	iii
DECLARATION OF ORIGINALITY	iv
ACKNOWLEDGEMENTS	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF FIGURES	xi
LIST OF TABLES	xiii
LIST OF ABBREVIATIONS	xiv
CHAPTER 1: INTRODUCTION	1
1.1 Background Information	1
1.2 Problem Statement	3
1.3 Project Objectives	4
1.4 Project Scope	5
1.5 Highlights of What Have Been Achieved	6
1.6 Outline	7
CHAPTER 2: LITERATURE REVIEW	8
2.1 Algorithm to Solve Combinatorial Optimization Problem	8
2.1.1 Brute Force	8
2.1.2 Nearest Neighbour	8
2.1.3 Branch and Bound Algorithm	9
2.1.4 Algorithm Comparison Table	10
2.1.5 Discussion	10
2.2 Existing Queue Management System	11
2.2.1 QueueBee	11
2.2.1.1 Brief	11
2.2.1.2 Strength	12
2.2.1.3 Weakness	12
2.2.1.4 Recommendation	12

2.2.2 powerQ	12
2.2.2.1 Brief	12
2.2.2.2 Strength	13
2.2.2.3 Weakness	13
2.2.2.4 Recommendation	13
2.2.3 WaveTec	14
2.2.3.1 Brief	14
2.2.3.2 Strength	14
2.2.3.3 Weakness	14
2.2.3.4 Recommendation	15
2.2.4 QLess	15
2.2.4.1 Brief	15
2.2.4.2 Strength	15
2.2.4.3 Weakness	16
2.2.4.4 Recommendation	16
2.2.5 Application Comparison	16
2.2.6 Discussion	16
CHAPTER 3: SYSTEM DESIGN	18
3.1 System Framework	18
3.1.1 Data Storing Flow	18
3.1.2 Queue Management Flow	18
3.1.3 Ticket Swapping Flow	18
3.1.4 Routing Flow	19
3.2 Project Algorithm and Method	20
3.2.1 Branch and Bound for Optimal Route	20
3.2.1.1 General Idea	20
3.2.1.2 Flow Chart	24
3.2.2 Method for Checking Delay Viability	28
3.3 Use Case Diagram	31
3.4 Use Case Description	31

3.4.1 Register Account	31
3.4.2 Login Account	32
3.4.3 Logout Account	32
3.4.4 Take Ticket	32
3.4.5 Show Route	32
3.4.6 Check Tickets Taken	32
3.4.7 View Ticket Details	32
3.4.8 Delay Ticket	32
3.4.9 Cancel Ticket	33
3.4.10 Check Inbox	33
3.4.11 Response to Swapping Request	33
3.5 Activity Diagram	34
3.5.1 Login	34
3.5.2 Register Account	35
3.5.3 Take Ticket	36
3.5.4 Show Route	37
3.5.5 Delay Ticket	38
3.5.6 Respond to Swap Request	39
3.6 Sequence Diagram	40
3.6.1 Take Ticket	40
3.6.2 Show Route	40
3.6.3 Delay Ticket	41
3.7 Firebase Structure	42
CHAPTER 4: SYSTEM METHODOLOGY AND TOOLS	45
4.1 System Methodology	45
4.2 Project Workflow	45
4.2.1 Planning Phase	45
4.2.2 Analysis Phase	45
4.2.3 Design Phase	46
4.2.4 Implementation Phase	46
4.2.5 System Prototype Testing Phase	46
4.2.6 Deploy Phase	46
4.3 Project Timeline	47
4.4 Technologies and Tools Involved	49

4.4.1 Computer	49
4.4.2 Emulator	49
4.4.3 Android Studio	50
4.4.4 Google Firebase	50
4.4.5 Google Directions API	51
4.4.6 Google Distance Matrix API	51
4.5 System Requirements	52
4.5.1 Functional Requirements	52
4.5.2 Non-Functional Requirements	52
CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING	53
5.1 Scenario	53
5.2 System Implementations	55
5.2.1 Login/Register	55
5.2.2 Map Interface	57
5.2.3 Ticket List Interface	62
5.2.4 Ticket Details Interface	63
5.2.5 Inbox Interface	64
5.3 System Testing	65
5.4 Algorithm Comparison	69
5.4.1 Performance Test	69
5.4.2 Accuracy Test	71
CHAPTER 6: CONCLUSION	73
6.1 Project Review	73
6.2 Novelties and Contribution	73
6.3 Future Work	74
REFERENCES	76
Weekly Report	A-1
Final Year Project Poster	B-1
Plagiarism Check Result	C-1
FYP2 CHECKLIST	D-1

LIST OF FIGURES

Figure Number	Title	Page
Figure 2.2.1.1.1	Example of QueueBee queue information	11
Figure 2.2.2.1.1	Screenshots of powerQ	13
Figure 2.2.3.1.1	Steps of joining a queue using WaveTec	14
Figure 2.2.4.1.1	Steps of using QLess	15
Figure 3.1.1	System Framework	18
Figure 3.2.1.2.1	Initialisation Flow Chart	24
Figure 3.2.1.2.2	Update Solution Part 1 Flow Chart	25
Figure 3.2.1.2.3	Update Solution Part 2 Flow Chart	26
Figure 3.2.1.2.4	Searching Branch Flow Chart	27
Figure 3.2.1.2.5	Pruning Branch Flow Chart	28
Figure 3.2.2.1	Queue Timeline	29
Figure 3.2.2.2	Requester Queue Timeline	29
Figure 3.2.2.3	Receiver Queue Timeline	30
Figure 3.3.1	Use Case Diagram	31
Figure 3.5.1.1	Login Activity Diagram	34
Figure 3.5.2.1	Register Account Activity Diagram	35
Figure 3.5.3.1	Take Ticket Activity Diagram	36
Figure 3.5.4.1	Show Route Activity Diagram	37
Figure 3.5.5.1	Delay Ticket Activity Diagram	38
Figure 3.5.6.1	Respond to Swap Request Activity Diagram	39
Figure 3.6.1.1	Take Ticket Sequence Diagram	40
Figure 3.6.2.1	Show Route Sequence Diagram	40
Figure 3.6.3.1	Delay Ticket Sequence Diagram	41
Figure 3.7.1	Top Layer of Firebase	42
Figure 3.7.2	Places Child Layer	43
Figure 3.7.3	Users Child Layer	44
Figure 4.1.1	System Methodology	45
Figure 4.3.1	Gantt Chart for Project I	47
Figure 4.3.2	Gantt Chart for Project II	48
Figure 4.4.3.1	Android Studio	50

Figure 4.4.4.1	Firestore	50
Figure 5.2.1.1	Login/Register Page	55
Figure 5.2.1.2	Forgot Password Page	56
Figure 5.2.2.1	Map Interface	57
Figure 5.2.2.2	Drawer Navigation	57
Figure 5.2.2.3	Select Queue Location and Service	58
Figure 5.2.2.4	Empty Pending List	59
Figure 5.2.2.5	Pending List with Items	59
Figure 5.2.2.6	Item On Click	60
Figure 5.2.2.7	Route on Click	61
Figure 5.2.2.8	Navigation Interface	61
Figure 5.2.3.1	Ticket List Interface	62
Figure 5.2.4.1	Ticket Details Interface	63
Figure 5.2.5.1	Inbox Interface	64
Figure 5.2.5.2	Inbox Item Interface	64
Figure 5.3.1	Visualisation of Optimal Route	67
Figure 5.4.1.1	Performance Test Graph Shown up to 1000ms	70
Figure 5.4.1.2	Performance Test Graph Shown up to 16000ms	70

LIST OF TABLES

Table Number	Title	Page
Table 2.1.4.1	Table of Comparison	10
Table 2.2.5.1	Table of Application Comparison	16
Table 4.4.1.1	Computer Used	49
Table 4.4.2.1	Emulator Used	49
Table 5.3.1	Table of Different Input Order Test	65
Table 5.3.2	Table of Algorithm Testing	66
Table 5.3.3	Table of Data of Queue Locations	67
Table 5.4.1.1	Table of Performance Test	69
Table 5.4.2.1	Table of Minimum Cost Found	71
Table 5.4.2.2	Table of Minimum Cost Found for 5 Queue Places	71

LIST OF ABBREVIATIONS

<i>QR Code</i>	Quick Response Code
<i>GPS</i>	Global Positioning System
<i>TSP</i>	Travelling Salesman Problem
<i>API</i>	Application Programming Interface
<i>CPU</i>	Central Processing Unit
<i>RAM</i>	Random Access Memory
<i>ABI</i>	Application Binary Interface

CHAPTER 1: INTRODUCTION

CHAPTER 1: INTRODUCTION

1.1 Background Information

A queue is defined as a line of people or things waiting for something (QUEUE, n.d.). According to Professor Haslam (n.d.), queue is described as a social norm, governed by unspoken rules promoting efficiency and equality. He also pointed out that queuing exists because there is an imbalance between the supply and demand of services (Haslam, n.d.). Therefore, nowadays society perceive queue as a normal phenomenon when comes to getting a service in First Come First Serve method, which everyone in society treated this as the fairest approach to get a service. Queueing in real life is an inevitable action in real life such that almost every day we will have the need of queueing, either to be queueing in a line, or to wait in waiting area and have your ticket number called, for example queue in front of a restaurant to wait for empty seats, waiting in bank to wait for your turn through ticketing queueing system, waiting for doctor available in hospital, waiting for your ticket number to be called in Tealive to get your milk tea and etc. However, people waiting in physical queue has to waste time standing in queue or staying in that particular place to wait for their numbers to be called in order to get the services. They cannot leave the queue to do other stuff since there is no estimated service time and estimated remaining time until their turn, and if they leave the physical queue, there is a risk of missing their turn which will waste more time to queue again.

Virtual queue is then introduced to try to solve problems occurred in physical queue. The main difference between a physical queue and a virtual queue is that whether you need to personally line up in the place you get your service (Tšernov, 2020). By using virtual queue, people can freely do their other works without having to stay at the places waiting for their turn when there is still a long duration before reaching their turn. This not only allow people to utilise their waiting time for other stuffs, but also able to estimate waiting time so that people can decide when to come back to the place to get the service within several minutes, meaning to say that people can get the service they want without having to wait for a long time.

In order to manage a queue properly and efficiently whether be it physical queue or virtual queue, a good queue management system is required not only to reduce queue waiting time and queue length, but also helps to reduce the chances of people leaving

CHAPTER 1: INTRODUCTION

a queue. However, do we even know what is a queue management system, or rather what is the element of a queue management system? According to Swati Rai (n.d.), a queue management system is an automated system to help managing queue of people. Using a queue management system for virtual queue, a customer can take a ticket with details of the queue such as estimated waiting time and number of customers ahead using his or her mobile phone. The customer will also be notified when it is his or her turn through mobile notification, and the customer is ready to leave the queue for the service.

Nowadays, more and more businesses actually turn their attention to virtual queueing systems (Tšernov, 2020). By implementing virtual queueing technology, it provides several benefits to both company and customer. For customer, they can get better services, have a more enjoyable waiting experience, and have their time back to do their own things. For company wise, customer may potentially add more items to basket or shopping cart if the company is retail-based or provide materials about other product provided if it is bank. Data can also be collected by using virtual queue (What are virtual queues and how are they used?, 2019). Using virtual queue will also eliminate cutting-line problem, where people may cut in line if they are physically queueing. To further improve the performance of virtual queue, most existing mobile application integrated Google Maps and Global Positioning System (GPS) into queue management system so that users can select service provider from Google Maps itself. The GPS is an earth-orbiting-satellite based navigation system which provides precise position and precise time, and this service is usually used to show user locations on map (Dana, 1997).

However, to queue in multiple different locations, there is no existing queue management system able to do that. Most of the current existing queue management systems are focusing on queueing in one location, and they are performing well in managing queues in single location. Queue in multiple locations has brought the problem to be solved to another level, which is solving combinatorial optimisation problem. The most popular combinatorial optimisation problem is said to be Travelling Salesman Problem according to Colorni, et al.(1996), which is quite similar to this project. Therefore, the algorithm chose to use is based on the efficiency in solving Travelling Salesman Problem.

CHAPTER 1: INTRODUCTION

1.2 Problem Statement

- i. Current queue management systems only manage queue on their own company or a single location.**

Current queue management applications can only manage the queue on a company but not manage the queue for multiple locations. There is no existing queue management system in the current market that supports more than one company. If user wants to queue in multiple places, he or she needs to use every company's own queue management system to take ticket.

- ii. Current queue management system does not provide an efficient approach for both company and customer if customer cannot reach on time.**

If customer after taking a ticket realises that he or she cannot reach the destination on time, current applications on market such as QLess does not provide an efficient way for user to elongate their arriving time. Customer will have no other choice but to discard their current ticket and take a new one (Digital Queuing System to Eliminate Lines, n.d.). Although some applications such as WaveTec application allow users to postpone their ticket, this approach will cause the company staff to have to wait for the postponed customer and thus will have nothing to do at the counter (Mobile Queuing and Virtual SMS App, n.d.), causing a waste of staff's time.

- iii. Current system does not provide any recommendation for user on which place to go first and does not plan the travelling route.**

While some of the current systems on the market does not support multiple tickets to be taken, some systems do indeed have the function for user to take multiple tickets. However, these systems do not recommend user to go to which place first so that the user will not miss any ticket. For instance, a user has taken 5 tickets and has to plan which place to go first by comparing all tickets remaining time. This will be a tedious work and have a chance that the user miscalculates the path, causing the user to miss one or more tickets. For example, QLess application supports multiple tickets to be taken, but it does not help user to plan route (Digital Queuing System to Eliminate Lines, n.d.).

1.3 Project Objectives

- i. **To propose a queue management system that can queue in multiple location with minimal queue time in each location.**

The system will first require user to select the places he or she wants to queue. Using Branch and Bound algorithm, system will queue the user in a way such that user has the least queue time in every location. The algorithm will consider travel time between every place and queue time required when queueing in every place to decide queue place order that has minimum queue time in every place by trying to utilise travel time as part of queueing time. In best case scenario, user can receive service shortly after user has reached a destination, without needing to queue for a long time.

- ii. **To propose an effective way to manage ticket whose user does not have enough travel time by implementing the function of delay tickets.**

If user wants to change swap ticket to have more time before reaching the destination, or system detects that user will not be able to reach the destination on time, user can request swap with next ticket holder. If a user receives a swapping request, user can choose to accept or reject the request. If system calculates that user will not reach the destination on time after swapping, system will auto reject the request.

- iii. **To develop route suggestion function based on every ticket a user has taken.**

System will be able to calculate the best possible route based on all tickets taken. The route generated is based on the remaining time of every ticket, meaning that system will suggest user to go to the place which has shortest remaining time on ticket. The route can be generated with the help of Google Direction API which will return shortest path as result in the form of JSON data. The route is only a suggestion for user which user can choose to follow the route or go his or her own way.

CHAPTER 1: INTRODUCTION

1.4 Project Scope

The scope of the proposed a model prototype for queue management system which supports queueing in multiple places in shortest time possible with optimal route generated according to tickets taken. In this project, an Android application is to develop to help user to calculate the best path to follow in terms of time and taking the ticket for user in every place to queue. The tickets decided by system should be the smallest ticket number that a user can take to reach every queue location on time. System will calculate the travel time between user's current location and all destinations where user will be taking ticket at. Using Branch and Bound algorithm, system will find the ticket numbers of every destination that will consume minimum time. After system has taken all the tickets, user will be able to see the suggested route from current location to every destination. Navigation function will be available through Google Maps app. User can opt to delay a ticket by either swap tickets with other user if next ticket was issued to the user, or simply increase ticket number if there is no next ticket holder. A notification will be sent to receiver regarding swapping request. User can response to a swap request by either accepting the request or rejecting the request. A calculation will be performed to make sure both users of swapping tickets will not miss any queue after swapping.

This application will implement the following modules:

i. Map and Routing module

This module will show user Google Maps that user can search for desire place to queue. This module will also detect user's current location and all tickets taken by user to calculate a fastest route to follow and show the path on Google Maps. System uses Google Direction API to get the path with shortest time from user's current location to destination.

ii. User Authentication module

User can login to an account if user has one already or register an account if user does not have one. User can login using email address. Any ticket taken will be added to ticket list of current user account, which will be used as the input of route planning.

CHAPTER 1: INTRODUCTION

iii. Queue Management module

This is the main module of the system. This module helps user to decide the ticket number to be taken in every queue location. User first selects all the destinations he or she wants to queue, system will then find the fastest route to queue in every place using Branch and Bound algorithm. Minimum ticket of every queue location will be calculated for ticket taking decision and for checking if the ticket can be exchanged with other user.

iv. Ticket module

This module allows user to choose available services provided by service provider selected by user. Data will be retrieved from database and displayed to user. After user has taken a ticket, user will see the details including current number and ticket number of that ticket. User can request swap ticket with another person on queue if there is person queueing behind user. After user has taken a ticket, the ticket will be saved on user's current ticket list. User will be able to see all tickets taken and can select a ticket to see the details such as current number and people waiting in this queue. A summary will also show to user without having user to click on a ticket to see which ticket is this. A remaining time countdown function will be implemented as well. When the remaining time of a ticket is less than 5 minutes, the summary will appear red to alert user.

1.5 Highlights of What Have Been Achieved

The main objective, which is to construct a queue management system that supports multiple queue locations, is achieved using branch and bound method to find the fastest route to travel through every location as well as calculated the tickets should be taken in every location. The project also implemented swap ticket function to allow user to swap ticket with others to have more queue time. Last but not least, a recommended route can be generated to determine which place to go first and see the travelling time and queue time.

CHAPTER 1: INTRODUCTION

1.6 Outline

This proposal consists of 6 chapters, which are Chapter 1: Introduction, Chapter 2: Literature Review, Chapter 3: System Design, Chapter 4: System Methodology and Tools, Chapter 5: System Implementations and Testing, and Chapter 6: Conclusion.

Chapter 1 describes the introduction of this project including background information, problem statements, project objectives, project scope, highlights of the project and outline of the report.

In Chapter 2, literature review has been done on describing Travelling Salesman Problem, analysis on existing algorithm to solve TSP, and 4 existing software related to queue management. A table of comparison is created between the said queue management software to show their differences visually.

Chapter 3 discussed about the algorithm and logic used in this project, and system design including system framework, use case diagram, activity diagram, sequence diagram and Firebase structure.

Chapter 4 shows the methodology used by this project, project timeline, tools and technologies used to complete this project, and system requirements of this project.

In Chapter 5, the scenario when using this application is briefly described and the user interface of the application is shown. Besides, algorithm testing and comparison are also conducted.

Chapter 6 discussed about the project review, contributions, novelties, and future work of this project.

CHAPTER 2: LITERATURE REVIEW

CHAPTER 2: LITERATURE REVIEW

2.1 Algorithm to Solve Combinatorial Optimization Problem

2.1.1 Brute Force

Brute force, also called as naive approach, is a method that considered all the possible routes and pick the minimum cost path. In this project context, we will have a fix starting point which is user's current location, therefore we can consider user's location as starting point. Then we will need to generate all permutations of n locations, which is $(n-1)!$ permutations. Next, the algorithm requires us to calculate every permutation cost and keep track of permutation with minimum cost. After all the permutation costs have been calculated, we will take the permutation with minimum cost as our result. Since all possible permutations will be generated and the one with minimum cost is selected, the result is guaranteed to be the globally optimal result, which means with respect of this project, the selected path is considered to be shortest path. For this approach, since we will need to generate $(n-1)!$ permutations which the time required will increase greatly as the number of points increase, the time complexity of this approach is $O(n!)$ (GeeksforGeeks 2018).

2.1.2 Nearest Neighbour

Nearest neighbour algorithm is a type of greedy algorithm which will take only local optimum points as consideration. According to Raju(2020), greedy algorithm has the characteristics of choosing best choice in every step to guarantee optimised solution and will not reverse any decision made. This indicates that the solution found might not be globally optimised due to the nature of the algorithm but will consume a considerable short time to execute since it considers only one location to proceed. The algorithm starts with a starting point and finds an unvisited location with lowest cost, which will then be selected as next point. The process repeats until all location has been explored. The solution route and travel cost will then be produced. Using this algorithm will need to set a starting point, generate nearest neighbour of every point, and calculate the cost between one point and every other points, therefore the time complexity of nearest neighbour algorithm is $O(n^2)$ (Weru 2021).

CHAPTER 2: LITERATURE REVIEW

2.1.3 Branch and Bound Algorithm

The concept of branch and bound solution is to first compute a bound on best possible solution we can get on a node. If the bound is worse than the best computed so far, we will ignore the subtree of the node. There are 2 costs we need to consider: cost from root to a node, and cost from node to a leaf. The first cost determines the current cost when we reach a node, while the second cost to compare with the bound to decide whether to ignore this subtree or not. There are 2 optimization problems, namely maximization and minimization problems can use branch and bound algorithm to find a solution. In this project we are interested in minimization problem, which has a lower bound to inform us the minimum cost if we follow the node (GeeksforGeeks 2020).

According to Morrison, et al.(2016), there are three important components that is required for Branch and Bound algorithm to function properly, namely search strategy, branching strategy and pruning rules. A search strategy is required to decide which subproblem is selected to explore first. Some examples of search strategy is depth first search and breath first search. For branching strategy, it can be divided into two groups, that are binary and non-binary branching. This strategy decides the way that a tree is branched into a subtree, either binary or non-binary. Lastly, pruning rules decide when a subtree can be pruned or ignored. The pruning rules need to be updated for every subtree to ensure accuracy.

For searching strategy, according to Tarjan(1972), depth first search or backtracking, starts from one vertex of a graph and select an edge. The selected edge is traversed to reach new unexplored vertex. The process is repeated until all vertices in a path are explored, which will then backtrack or move backwards on the same path to find unexplored vertices to traverse. The idea of depth first search is to traverse as deep as possible before backtracking to search for next available vertex. Breadth first search, on the other hand, traverse through vertex of a graph level by level. Starting from root vertex, it finds all the possible edges connecting root vertex and all other vertices on current level. The search will then continue on next depth level and find all possible edges of all vertices in this level.

CHAPTER 2: LITERATURE REVIEW

The branching strategy of branch and bound can be divided into 2 different section, namely binary branching and wide branching according to Morrison, et al.(2016). Binary branching split a subproblem into two mutually exclusive and smaller subproblems. This strategy can often be seen in solving knapsack problem, where one branch indicates an item is selected while the other indicates the item is not included in knapsack. In contrast, wide branching branches into every unvisited vertex in every subproblems.

Pruning rule, or the bound, used in branch and bound search is responsible for excluding any infeasible search region. For example, if a subproblem S_1 contains a solution that is better than any solution in subproblem S_2 , then it is enough to just search for subproblem S_1 , meaning that it is safe to prune subproblem S_2 since S_2 contains a solution which is at most same optimised solution.

When using branch and bound algorithm, the worst case will be the same as brute force algorithm that is $O(n!)$ because there might not have any node to prune. However, according to Rai(2020), the algorithm is performing well in practice and the complexity is really depending on the pruning rule of the algorithm.

2.1.4 Algorithm Comparison Table

Algorithm	Time Complexity	Optimal Solution
Brute Force	$O(n!)$	Yes
Nearest Neighbour	$O(n^2)$	Only locally optimal
Branch and Bound	$O(n!)$, but faster than brute force most of the time	Yes

Table 2.1.4.1 Algorithm Comparison Table

2.1.5 Discussion

From Table 2.1.4.1, we can see that both brute force and branch and bound algorithm provide optimal solution, while nearest neighbour algorithm will only produce locally optimal solution, which is not what we want to achieve in this project. Among the other two algorithm, branch and bound algorithm is chosen to be the algorithm used in this project since it performs better than brute force in terms of time.

CHAPTER 2: LITERATURE REVIEW

2.2 Existing Queue Management System

2.2.1 QueueBee

2.2.1.1 Brief

QueueBee provides mobile queue solution for customer to download their application and let them has real time queue status remotely (QueueBee Mobile Queue, n.d.).

User can either login with an account before using this application, or simply continue as guest, making it more user friendly. User can then scan QR Code or select service provider from a list, or even use Maps function of the application to find service provider from Google Map. Next, user will choose a service from a list and a virtual queue number will be given to user. The queue information such as current number, waiting position and estimated call time will be shown to user, as shown in Figure 2.2.1.1.1. QueueBee application will alert the user when the turn comes (QueueBee Mobile Queue, n.d.).

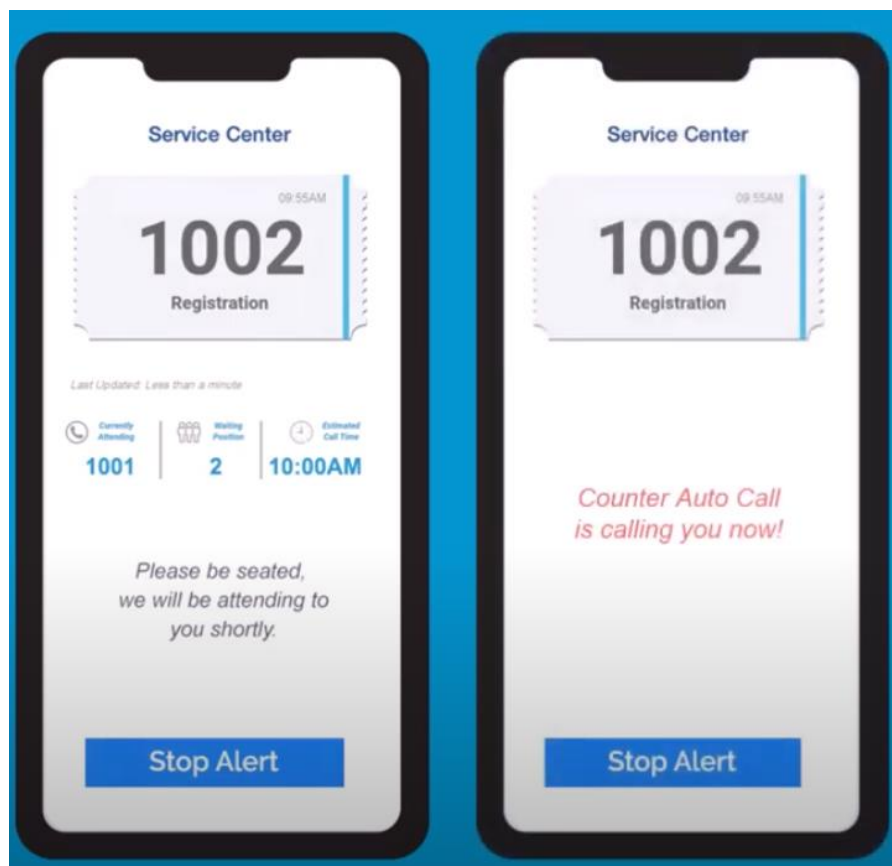


Figure 2.2.1.1.1 Example of QueueBee queue information

CHAPTER 2: LITERATURE REVIEW

2.2.1.2 Strength

- Provide map function that allows user to choose service provider from map
- Show distance between current location and target destination
- Provide filter categories
- Simple user interface which promotes user-friendliness
- Show estimated waiting time

2.2.1.3 Weakness

- Unable to cancel ticket
- Unable to exchange ticket with others
- System does not calculate route if user wants to go to multiple places
- System does not calculate travel time

2.2.1.4 Recommendation

It is recommended to implement a method in case of user is not able to reach on time. It can either allows user to cancel ticket, postpone ticket or exchange ticket with others. Since this application provides map function, it can implement route planning function to save user's time.

2.2.2 powerQ

2.2.2.1 Brief

powerQ is a mobile application which supports both Android and IOS, meaning that user can download this application at Google Play Store and Apple App Store. Currently powerQ application only provide services at Visa, Pass & Permit Division, Immigration Putrajaya. Services provided are for Social Visit Pass (Extension) and Long-Term Social Visit Pass (Extension - Spouse programme) at the time being. By using this application, user can book a time slot, view booked time slot, view current queue, and manage user account (powerQ, n.d.). After user booked a slot, a QR Code will be generated for user to scan it at kiosk. Screenshots of PowerQ application is shown in Figure 2.2.2.1.1.

CHAPTER 2: LITERATURE REVIEW

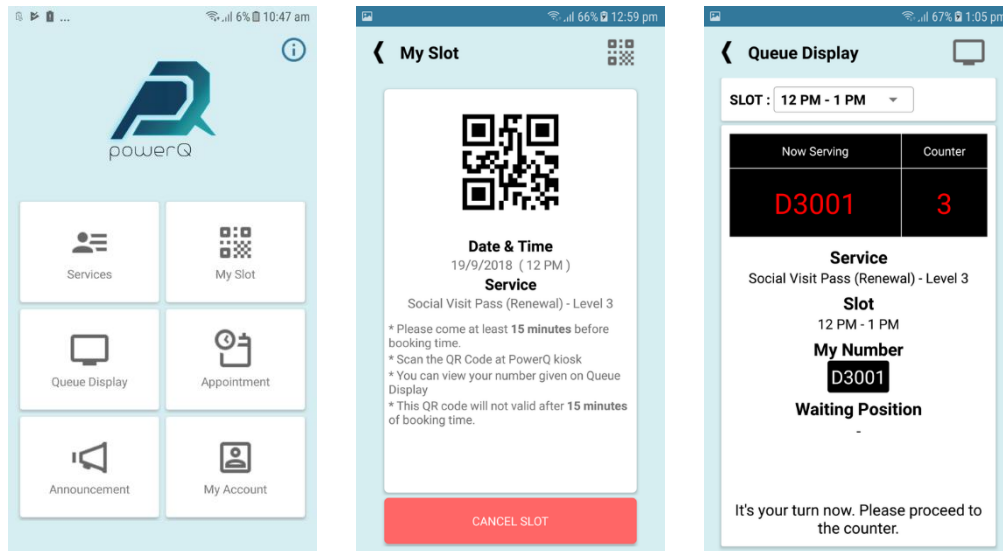


Figure 2.2.2.1.1 Screenshots of powerQ

2.2.2.2 Strength

- Easy-to-understand UI
- Provide tutorial for first time user
- Able to cancel time slot booked

2.2.2.3 Weakness

- Does not provide map function
- Cannot exchange ticket with other once book a time slot
- System does not calculate route if user wants to go to multiple places
- Does not show estimated remaining waiting time
- Does not provide filter categories
- Can only book one time slot at a time
- Does not calculate travel time

2.2.2.4 Recommendation

This application only supports one ticket to be taken at a time. This application can try to increase number of places user can take ticket from and provide map function as well as plan route for user.

CHAPTER 2: LITERATURE REVIEW

2.2.3 WaveTec

2.2.3.1 Brief

WaveTec provides a queue management mobile application for both Iphone and android user. This application allows user to join a virtual queue easily with a few clicks before physically arriving at a branch. This mobile application, named Mobile-Q, aims to eliminate physical queue and create seamless and efficient customer experiences (Mobile Queuing and Virtual SMS App, n.d.). Figure 2.2.3.1.1 shows the steps of how this application works.

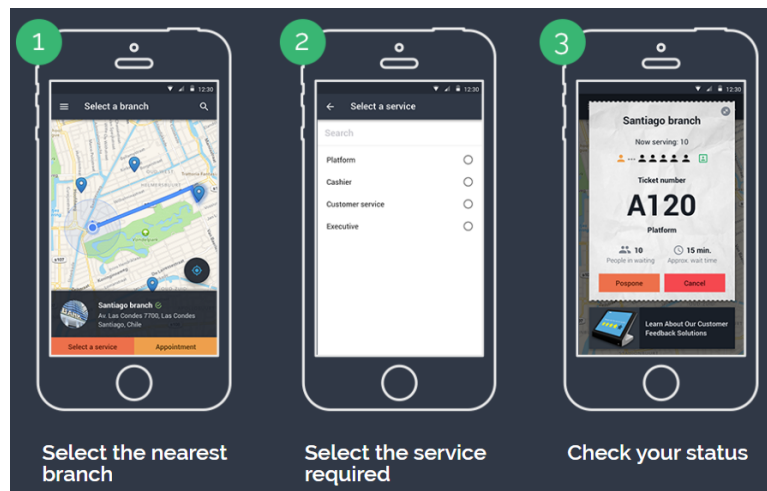


Figure 2.2.3.1.1 Steps of joining a queue using WaveTec

2.2.3.2 Strength

- Provide map function
- Able to postpone if customer unable to reach in time
- Able to cancel ticket
- Show estimated waiting time

2.2.3.3 Weakness

- System does not calculate route for multiple places
- Does not calculate travel time
- Does not provide filter categories
- Cannot swap ticket

CHAPTER 2: LITERATURE REVIEW

2.2.3.4 Recommendation

Since this application provides map function, the application can add function which allows user to take tickets from multiple places and route planning function can be implemented to save user's time.

2.2.4 QLess

2.2.4.1 Brief

QLess aims to eliminate long queueing lines and speed up transactions by using virtual queue. This application allows customers to utilise their smartphone with alerts about waiting time and scheduled appointments, allowing them to queue virtually which leads to high customer satisfaction and increase business revenue (Digital Queuing System to Eliminate Lines, n.d.). Figure 2.2.4.1.1 shows the steps on how to use this application.



Figure 2.2.4.1.1 Steps of using QLess

2.2.4.2 Strength

- Able to reschedule an appointment
- Show distance between current location and destination
- Able to cancel ticket
- Show estimated waiting time
- Able to virtually queue in multiple places

CHAPTER 2: LITERATURE REVIEW

2.2.4.3 Weakness

- Does not calculate route if user wants to go to multiple places
- Does not provide map function
- Does not provide filter categories
- Does not calculate travel time

2.2.4.4 Recommendation

This application can add map function into its system. Since this application supports multiple tickets to be taken, route planning function can be implemented.

2.2.5 Application Comparison

Application	QueueBee	powerQ	WaveTec	QLess	Proposed Application
Map Function	✓	✗	✓	✗	✓
Delay Tickets	✗	✗	✓	✗	✓
Calculate Route	✗	✗	✗	✗	✓
Cancel Tickets	✗	✓	✓	✓	✓
Show estimated waiting time	✓	✗	✓	✓	✓
Support Multiple Queue Location	✗	✗	✗	✓	✓
Calculate Travel Time	✗	✗	✗	✗	✓

Table 2.2.5.1 Table of Application Comparison

2.2.6 Discussion

From Table 2.2.5.1, although most applications do utilise Google Map, they do not provide any generate route or route-planning function. Also, no existing application implement the function of swap tickets, but most application do have cancel tickets function, which is a not-so-efficient method. Besides, all existing application does not

CHAPTER 2: LITERATURE REVIEW

calculate travel time between user's current location and destination. This could lead to user takes a ticket which he or she may not reach destination on time. There is only one application support multiple queues. On the good side, most applications show estimated waiting time which is good to show remaining time until user's turn.

After comparing the functions of existing applications, the proposed mobile application aims to improve the weaknesses. The proposed application supports map function which integrate Google Maps into the system. User can choose service provider on the Google Maps. The proposed application also implements the function of showing estimated waiting time. Unlike other application which only show static time, the proposed application implements a timer function to show real-time estimated remaining time. It also supports multiple queues in different locations which allows user to queue in multiple places.

There are some new functions being implemented on the proposed application. First, the proposed application supports ticket swapping. This is to avoid people who cannot reach on time has to force to discard ticket. Next, the proposed application will calculate both shortest route and smallest ticket number a user can have in every destination. Then, system will calculate travel time and estimated arrival time between user's current location and destination. A shortest route will be automatically drawn after user has taken any ticket.

CHAPTER 3: SYSTEM DESIGN

CHAPTER 3: SYSTEM DESIGN

3.1 System Framework

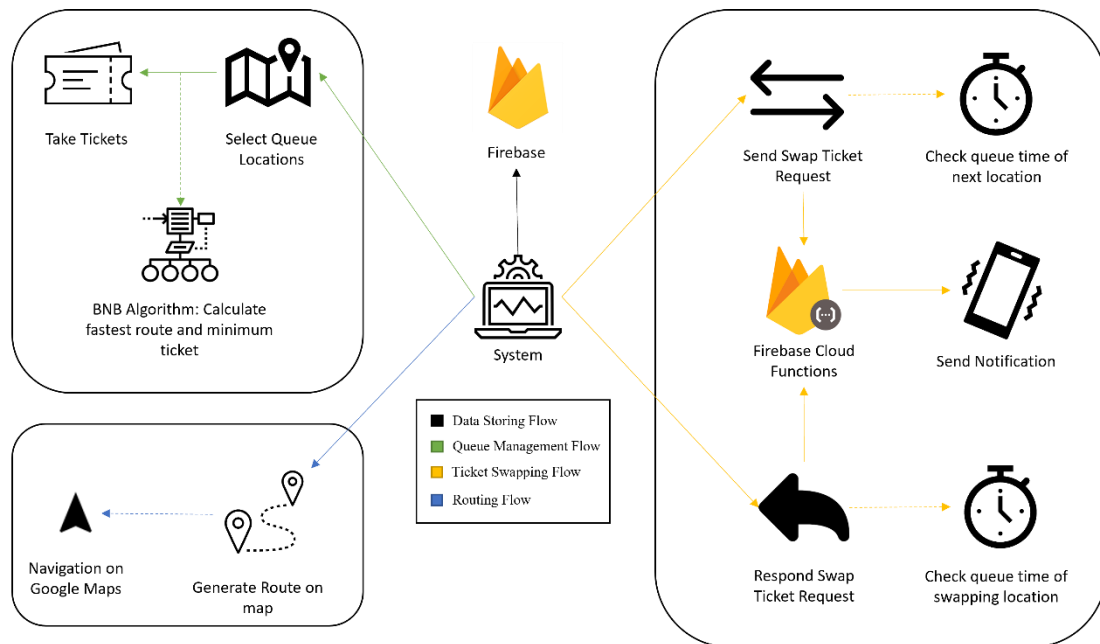


Figure 3.1.1 System Framework

3.1.1 Data Storing Flow

Using an Android mobile phone as a medium, the system can store data to database. The database of this project is Google Firebase.

3.1.2 Queue Management Flow

Through the system, a user can select the locations where he or she wants to queue at. The selected locations will be added to a pending list. After user has finished choosing the locations, the system will use branch and bound algorithm to find the optimum route to travel through all the places and all the tickets that should be taken in every places.

3.1.3 Ticket Swapping Flow

User can perform 2 actions on this flow. If user wants to swap ticket with others, system will first check with the next location queue time of swap ticket location to ensure even if user has swapped the ticket, he or she can still reach next destination on time. For example, user 1 sends swap request to user 2 at location A. Before sending the request, system will check queue time at next location of location A, say location B. System will check if the queue time of location B is less than service time of

CHAPTER 3: SYSTEM DESIGN

location A, system will block the swap request. If swap is allowed, system will send swap request to user 2, then access to function stored in Firebase Cloud Function to send notification to the other user who will be respond to the swap request. On the other hand, if user 2 responds to a swap request, system will check the queue time and service time of location B to make sure user 2 can reach the destination on time after swapping. Similar to send request, a response on swap request will also trigger a function in Firebase Cloud Function to send notification to user who send the request to inform him or her the request has been responded.

3.1.4 Routing Flow

After user has taken any ticket, the system will generate a shortest and optimised route for user to follow. User can click on the route to see when to arrive, travel distance and travel time. User can also click navigate button to open Google Maps application to navigate user to destination.

CHAPTER 3: SYSTEM DESIGN

3.2 Project Algorithm and Method

3.2.1 Branch and Bound for Optimal Route

3.2.1.1 General Idea

3.2.1.1.1 Creating Cost Matrix

Before the algorithm is able to calculate optimal path, it is important to first create a cost matrix for system to refer so that system will not make unnecessary and repeated action of having to calculate the path between two places. For this project, the cost is different from typical optimisation problem which only take travel distance or travel time. The cost of this project needs to consider travel time, service time, post buffer and pre buffer time. Service time is time required to take a service in a location. Post buffer time is time reserved for user after the user has finished the service and before he or she is ready to depart to next location, such as walk time from building to car. Pre buffer time is time reserved for user to enter the building after the user has reached the place, such as parking time.

The cost between 2 places is shown as such:

$$C_{ij} = T_{ij} + S_i + P_i + p_j \quad (1)$$

where

C_{ij} is cost between i and j, in seconds

T_{ij} is travel time between i and j, in seconds

S_i is service time at i, in seconds

P_i is post buffer time at i, in seconds

p_j is pre buffer time at j, in seconds

Using Google distance matrix API will get the shortest travel time from one place to another place. A travel time matrix can be generated using this API. The matrix is then updated using equation (1). All the data required such as service time can be retrieved from Firebase, which is the database used for this system. After the cost matrix is generated, the matrix is passed to the algorithm as one of the parameters to determine optimal path.

CHAPTER 3: SYSTEM DESIGN

3.2.1.1.2 Deciding Target Nodes

To make sure the algorithm is able to traverse through every place while ignoring visited place, a list of visited boolean variable is created to keep track of the visited places. When selecting next place to explore, system will first check the visited list and skip the place if the place is explored. The algorithm is following depth first search method, therefore after reaching last available place, system will traverse back one level up from the current path to continue explore other combination.

3.2.1.1.3 Calculating current weight

When a place is selected to be next place to traverse, a calculation on current weight and ticket number of that place will be performed. Ticket number is decided using the following equation:

$$T_j = \text{Ceil}((Q_i + C_{ij}) / S_j) + t_j \quad (2)$$

where

T_j is minimum ticket number can be taken at j

Q_i is queue duration or weight from origin to i , in seconds

C_{ij} is cost between i and j , in seconds

S_j is service time at j , in seconds

t_j is current ticket number that is being processed

Using equation (2) will first calculate the smallest ticket number that can be taken by user in current route order. However, this ticket number still need to be checked with other tickets taken by other users to prevent taking the same number. Before that, system will first check if user has already taken a ticket in this location. If this is the case, system will set the ticket number to taken ticket number as long as the taken ticket is larger than or equal to the smallest ticket number. Otherwise, system will invalidate this ticket since current path order is impossible. For example, user has taken a ticket with number 5 at location B and wants to take ticket at location A. If the system is currently calculating the route from A to B, and the minimum ticket for location B in this route is 8, then system will invalidate this route because user is holding ticket number 5 to continue to search for other possible routes. If user does

CHAPTER 3: SYSTEM DESIGN

not hold a ticket in this location, system will check with all tickets taken by the other users and select the smallest possible ticket number that is larger than or equal to the calculated minimum ticket number.

After the ticket number of this location is decided, the queue duration or the weight of the algorithm will be calculated. The equation is as such:

$$Q_i = (T - t_i) * S_i \quad (3)$$

where

Q_i is queue duration or weight from origin to i , in seconds

T is the ticket number decided by the system

t_i is current ticket number that is being processed

S_i is service time at i , in seconds

The queue duration can be calculated by using equation (3). Next, to make sure user can reach the destination and process the service before closing hour, the queue time will be compared with the closing hour of the location. The location will be skipped if user cannot reach and process service before closing hour, otherwise system will proceed to next step, which is pruning section.

3.2.1.1.4 Pruning Condition

After system has calculated the queue duration, system will check the value with current best value. The value more than current best value indicates that any route branches from current route will be not optimal, thus will be pruned. The system will then find other routes to proceed. If the weight or the queue duration calculated is less than current best value, this location will be added as part of the route and the weight as well as ticket number will be recorded.

3.2.1.1.5 Updating Pruning Rule

If a branch or a route is reaching the end, in other words all locations are explored, system will make sure the route contains tickets that has been taken by user. Next, system will check number of places in the route that are marked as exceeded working

CHAPTER 3: SYSTEM DESIGN

hour. If the number is less than or equal to current recorded number of places that are exceeded working hour, and the final weight of the route is less than current best weight, the prune value which is the current best weight will be updated and all the details of this route including ticket numbers of all location and travel order will be recorded. The system will then continue to explore other routes if possible.

3.2.1.1.6 Get Optimal Route

After system has explored every possible route, an optimum route will be found. System will then take tickets based on the calculated tickets of the optimum route.

CHAPTER 3: SYSTEM DESIGN

3.2.1.2 Flow Chart

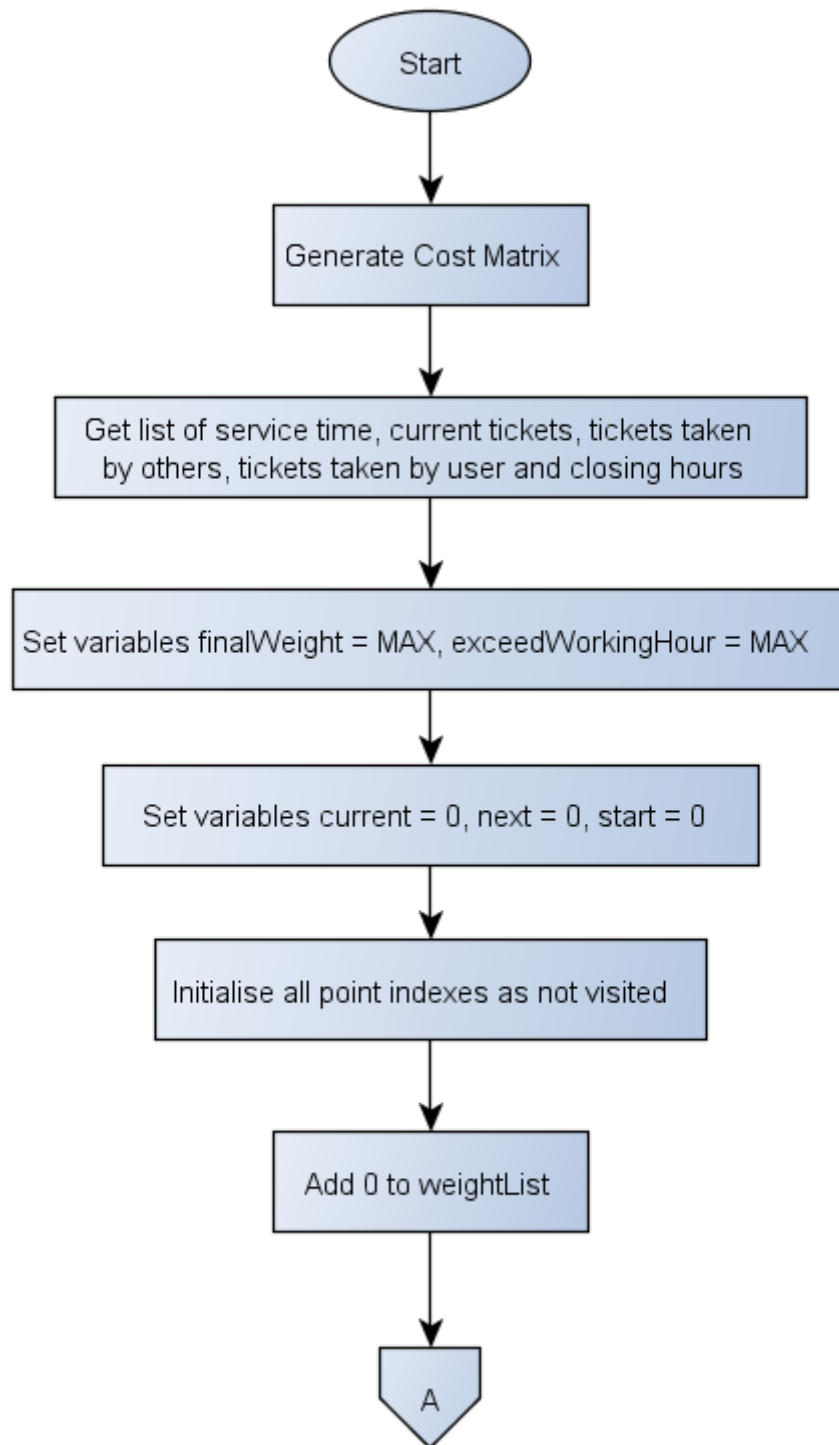


Figure 3.2.1.2.1 Initialisation Flow Chart

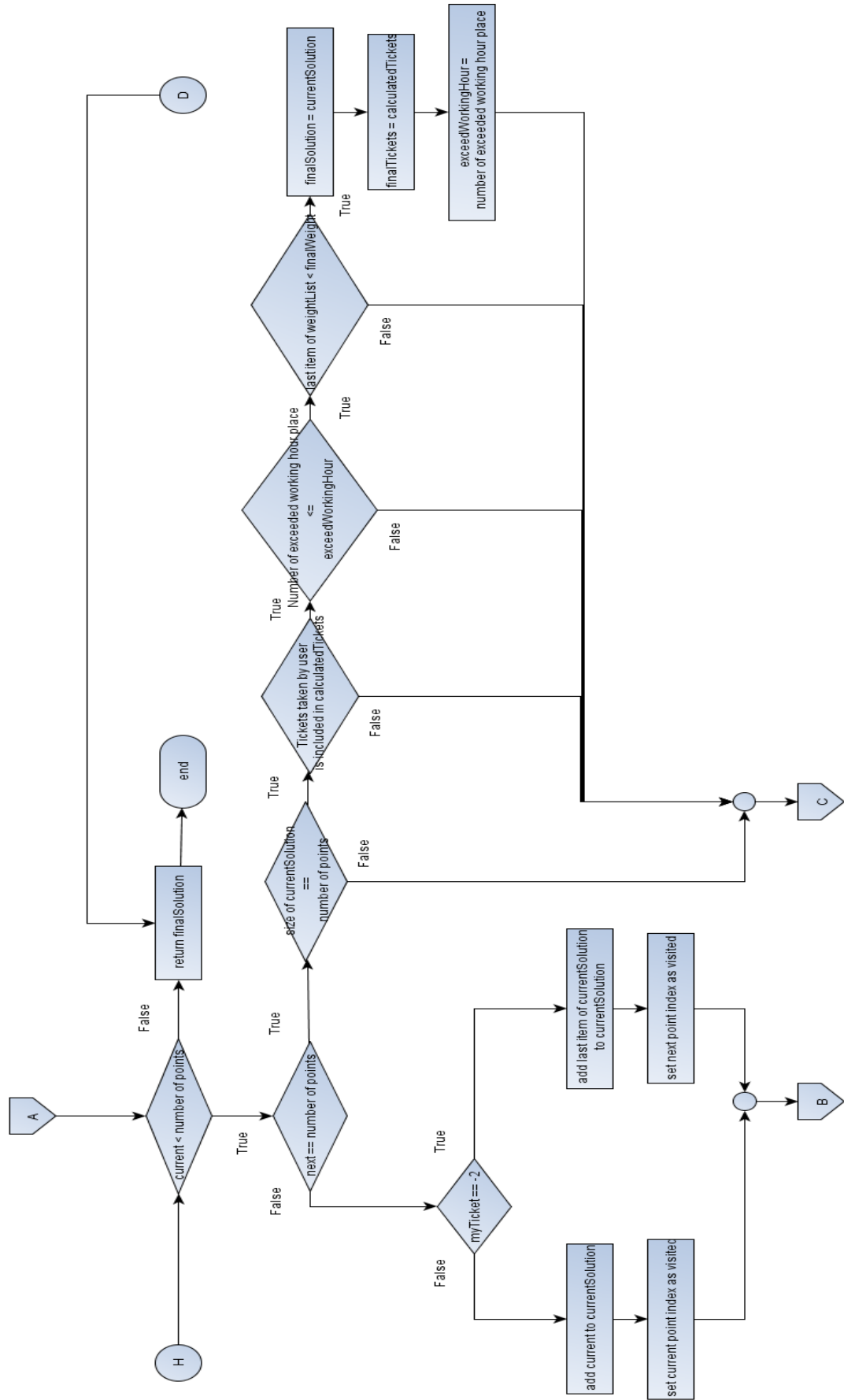


Figure 3.2.1.2.2 Update Solution Part 1 Flow Chart

CHAPTER 3: SYSTEM DESIGN

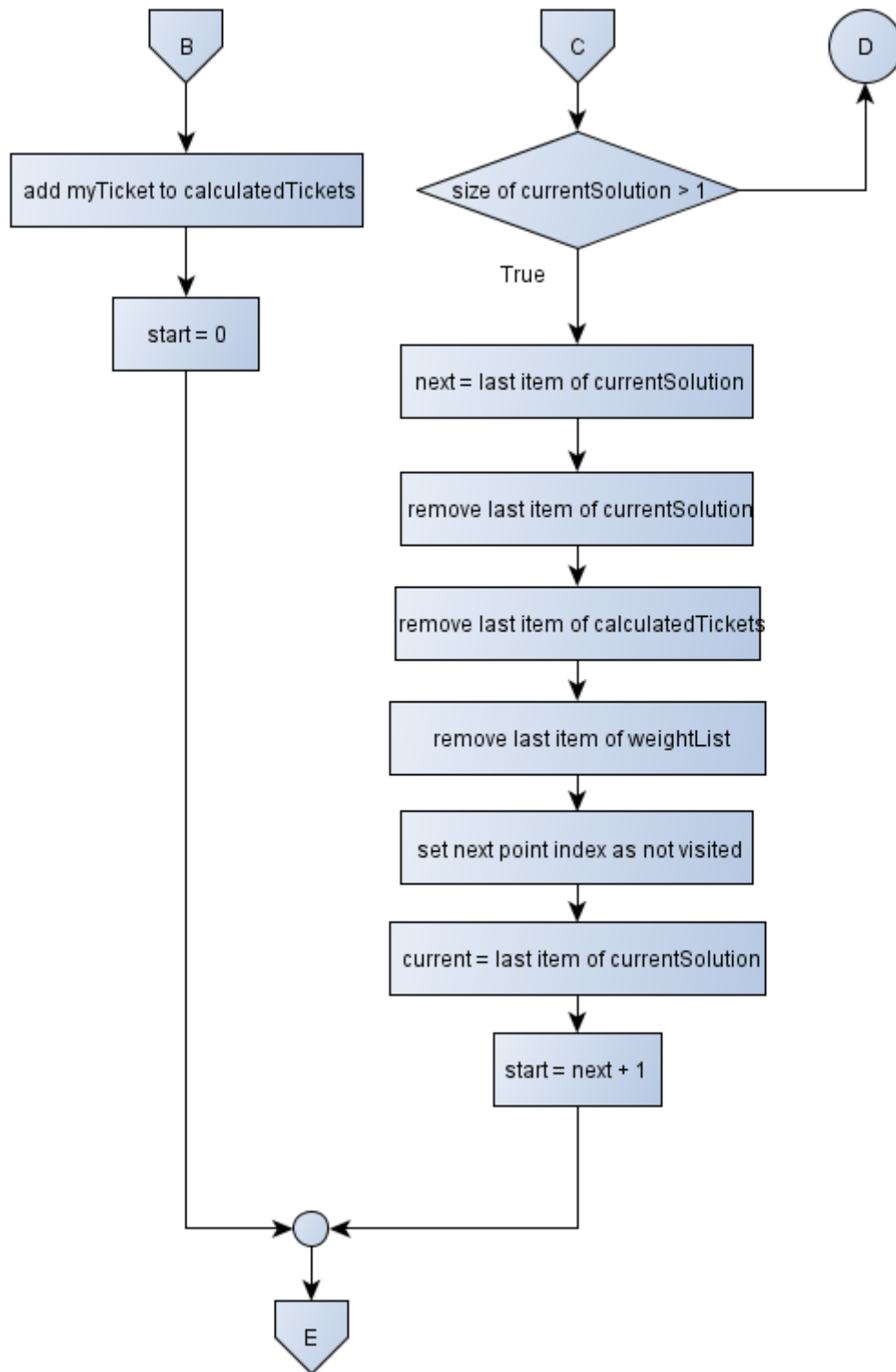


Figure 3.2.1.2.3 Update Solution Part 2 Flow Chart

CHAPTER 3: SYSTEM DESIGN

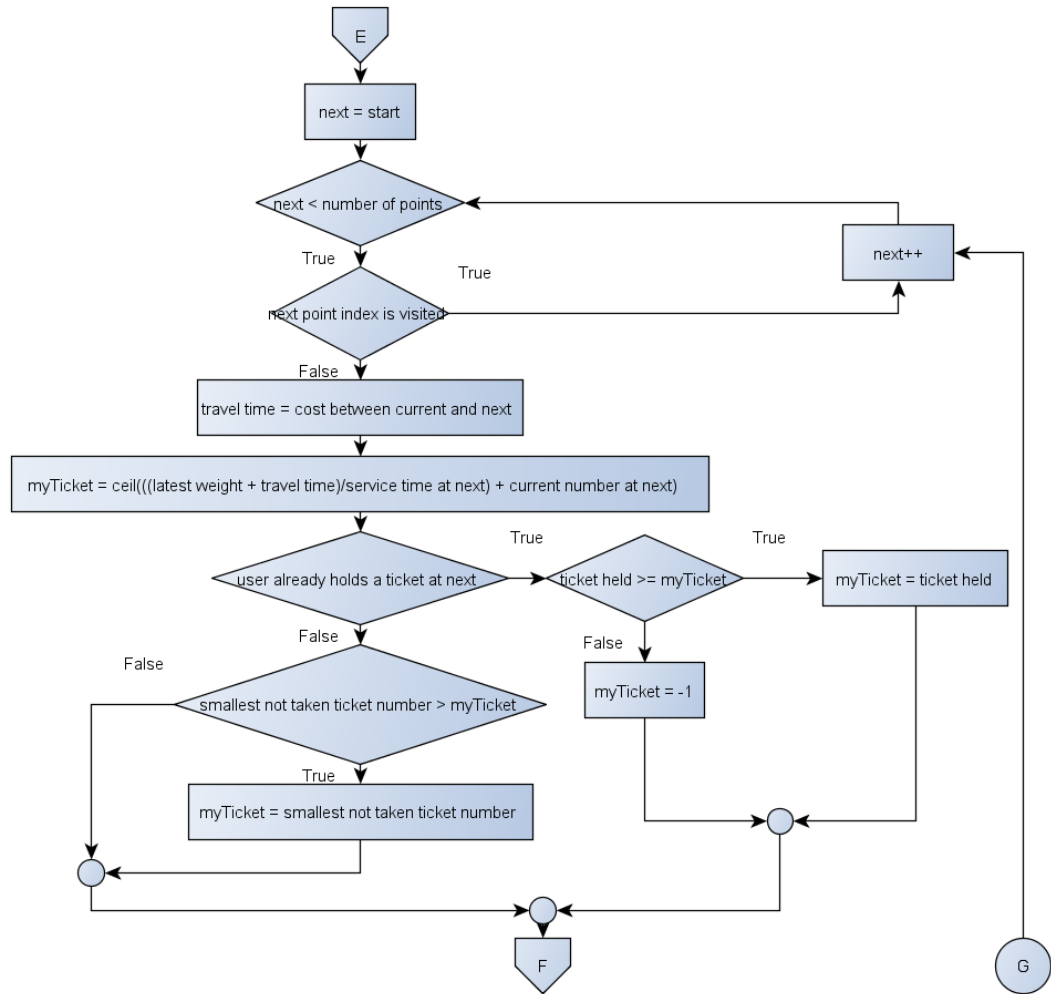


Figure 3.2.1.2.4 Searching Branch Flow Chart

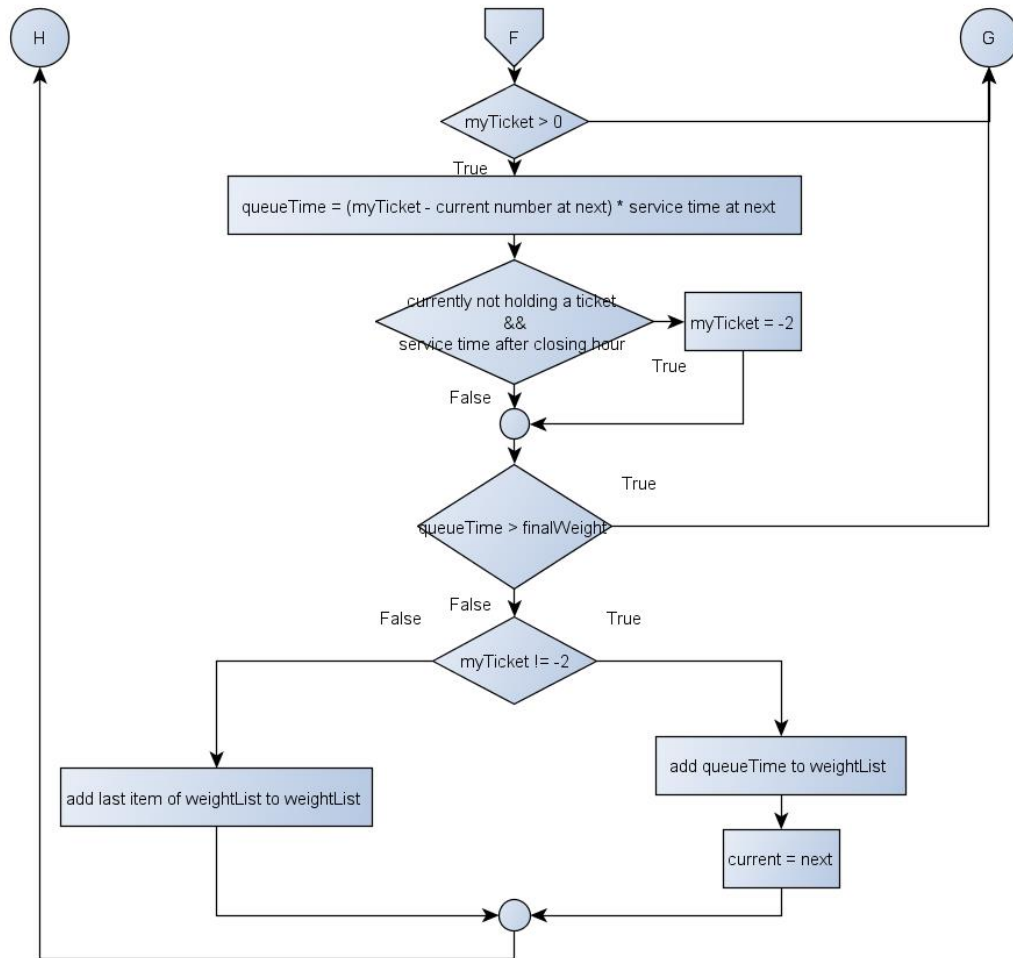


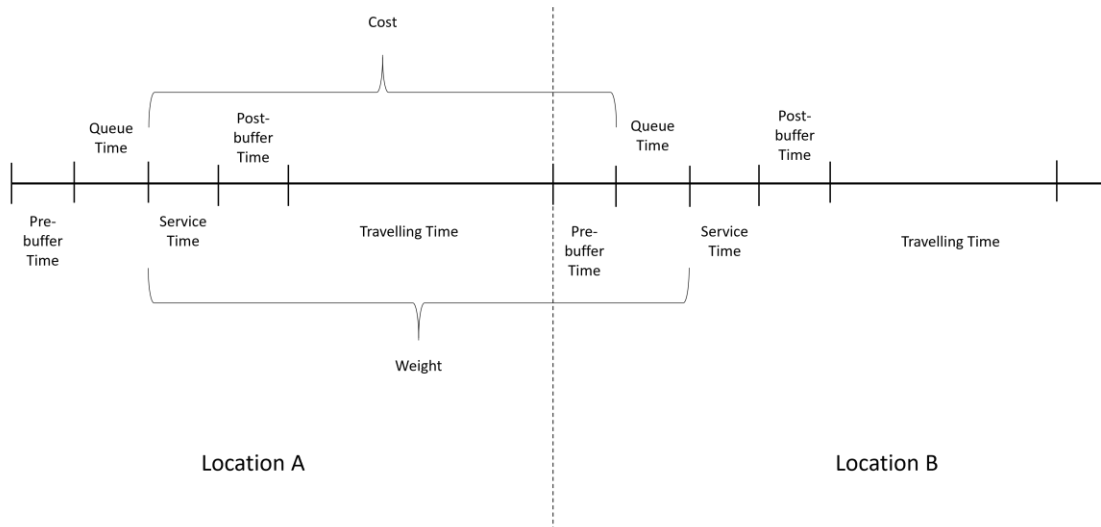
Figure 3.2.1.2.5 Pruning Branch Flow Chart

3.2.2 Method for Checking Delay Viability

To make sure user can still reach every queue location after swapping tickets, a verification process needs to be carried out. Both requester and receiver need to check for swapping viability.

For requester, when user decides to delay a ticket, system will need to make sure user can reach next destination on time. Figure 3.2.2.1 shows the timeline when user is queueing in multiple locations.

CHAPTER 3: SYSTEM DESIGN

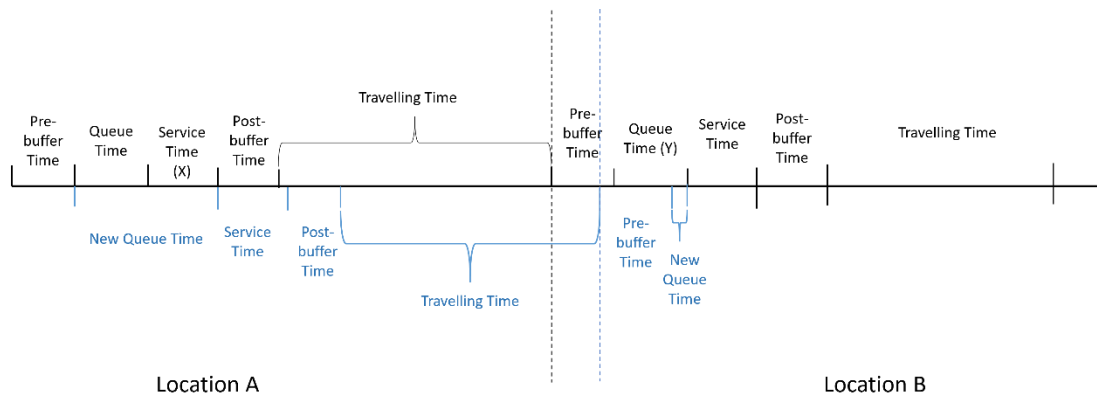


Cost = Time required to reach next destination

Weight = Time required to reach user's ticket number

Figure 3.2.2.1 Queue Timeline

To make a ticket delay in Location A possible, the queue time in Location B must be more than service time in Location A so that user will not miss his or her turn in Location B. Figure 3.2.2.2 shows the queue timeline of a user who can delay a ticket.



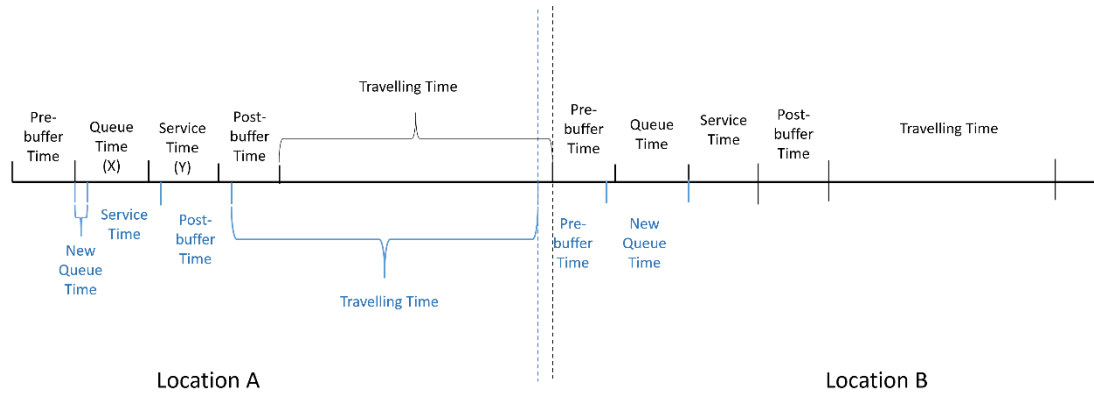
Service Time (X) < Queue Time (Y)

Figure 3.2.2.2 Requester Queue Timeline

In Figure 3.2.2.2, we can clearly see that the new queue time in Location B will be shorter than Queue Time (Y). This reduction is contributed by the service time at Location A, therefore we can conclude that service time of Location A must be less than queue time in Location B to make a delay request viable.

CHAPTER 3: SYSTEM DESIGN

For receiver, when user receives a swap request, system will need to make sure user can reach the swap location on time before user can accept a request. Figure 3.2.2.3 shows the timeline of user who can accept a swap request.



Service Time (X) < Queue Time (Y)

Figure 3.2.2.3 Receiver Queue Timeline

In Figure 3.2.2.3, the new queue time is shorter than Queue Time (X) in Location A. Note that for user who accepts a swap request, there will be no critical affection on Location B although the new queue time will be elongated. User can still reach on time no matter what. The problem lies in queue time and service time of Location A. To make accepting of swap request viable, the service time must be less than the queue time in Location A.

CHAPTER 3: SYSTEM DESIGN

3.3 Use Case Diagram

Figure 3.2.1 shows the use case diagram of what users can do using the system. User will first need to login or register an account to keep track of tickets taken. User will then be navigated to map interface which is the location where taking ticket process is happening. User can select the queue locations to add them into pending list. After user has added all the desired location to pending list, user can click a button to queue in every selected place after system has finished calculating the optimal route. User can also check all the taken tickets in ticket list interface. In ticket list interface, user can click on a particular ticket to see the ticket details. From the ticket detail, user can choose to delay ticket or cancel ticket. In inbox interface, user can see all the swap requests and respond to the requests.

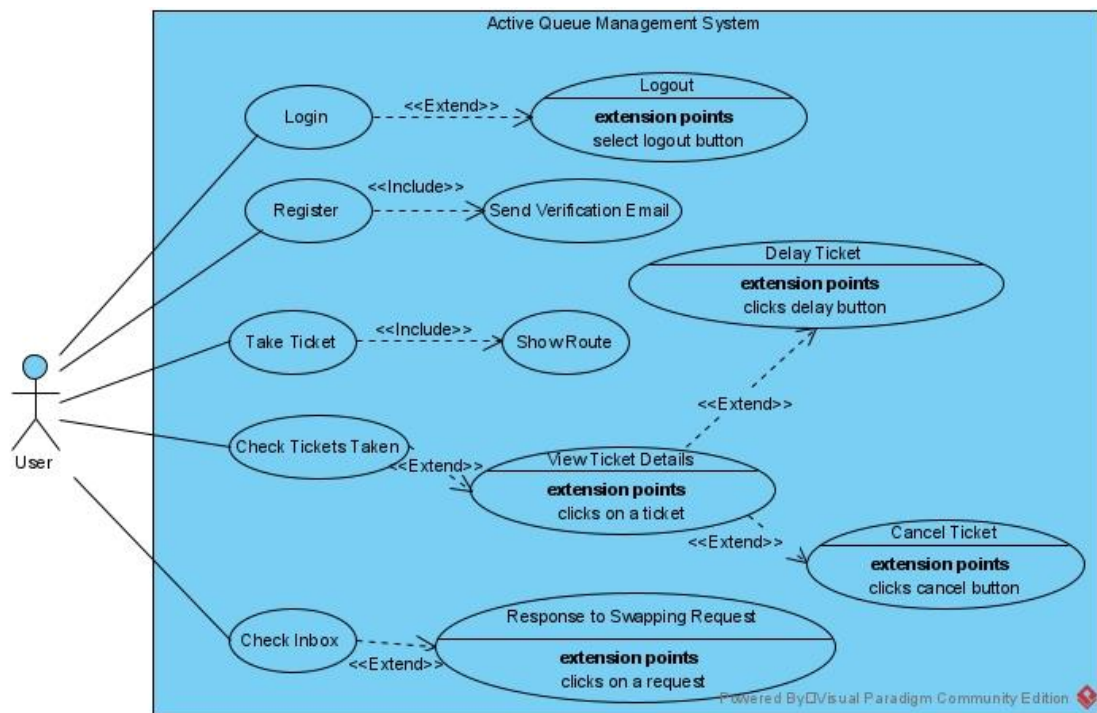


Figure 3.3.1 Use Case Diagram

3.4 Use Case Description

3.4.1 Register Account

This function allows user to register a new account using email. This step is required for new user as the system relies on account to keep track of list of taken tickets. A verification email will be sent to registered email. User needs to verify through the email to use the application.

CHAPTER 3: SYSTEM DESIGN

3.4.2 Login Account

This function allows user to login to the system using registered personal account. User is required to login to the system before he or she can start using the system.

3.4.3 Logout Account

This function allows logged in user to log out from personal account. User is required to log in again if he or she wants to use the application again.

3.4.4 Take Ticket

This function is the main function of the application. User can select multiple locations to queue up. The selected location will be added into pending list which is where the user can click a button to allow system to calculate optimal path and ticket number.

3.4.5 Show Route

This function shows the recommended route based on the tickets taken by user. The route is shown on map interface and is allowed to be clicked by user to see the details of the route such as travelling time and distance as well as user's current ticket number.

3.4.6 Check Tickets Taken

This function allows user to check a list of his or her taken tickets.

3.4.7 View Ticket Details

This function allows user to select a ticket from the list of taken tickets. The details such as current number, ticket number, estimated queue duration and person in queue will be shown in this interface. Delay and cancel ticket button can be found in this interface.

3.4.8 Delay Ticket

This function allows user to delay slightly his or her ticket to next ticket number. If there is no person holding next ticket number, system will straight take the number, otherwise a swap request will be sent to the person holding the number. System will

CHAPTER 3: SYSTEM DESIGN

first ensure user can reach next destination if the swap is successful before allowing user to send swap request. This is to make sure user stays on the route planned.

3.4.9 Cancel Ticket

This function allows user to cancel the ticket.

3.4.10 Check Inbox

This function allows user to see all the swap request sent to the user. User can select one of the items to make respond.

3.4.11 Response to Swapping Request

This function allows user to either accept or reject a request. If user chooses to reject, no ticket swapping will occur. If user chooses to accept, system will first check if user can reach the destination before response. If system calculates that user will not be able to reach destination on time after accepting the swap request, system will not swap both tickets. Otherwise, both tickets are swapped.

CHAPTER 3: SYSTEM DESIGN

3.5 Activity Diagram

3.5.1 Login

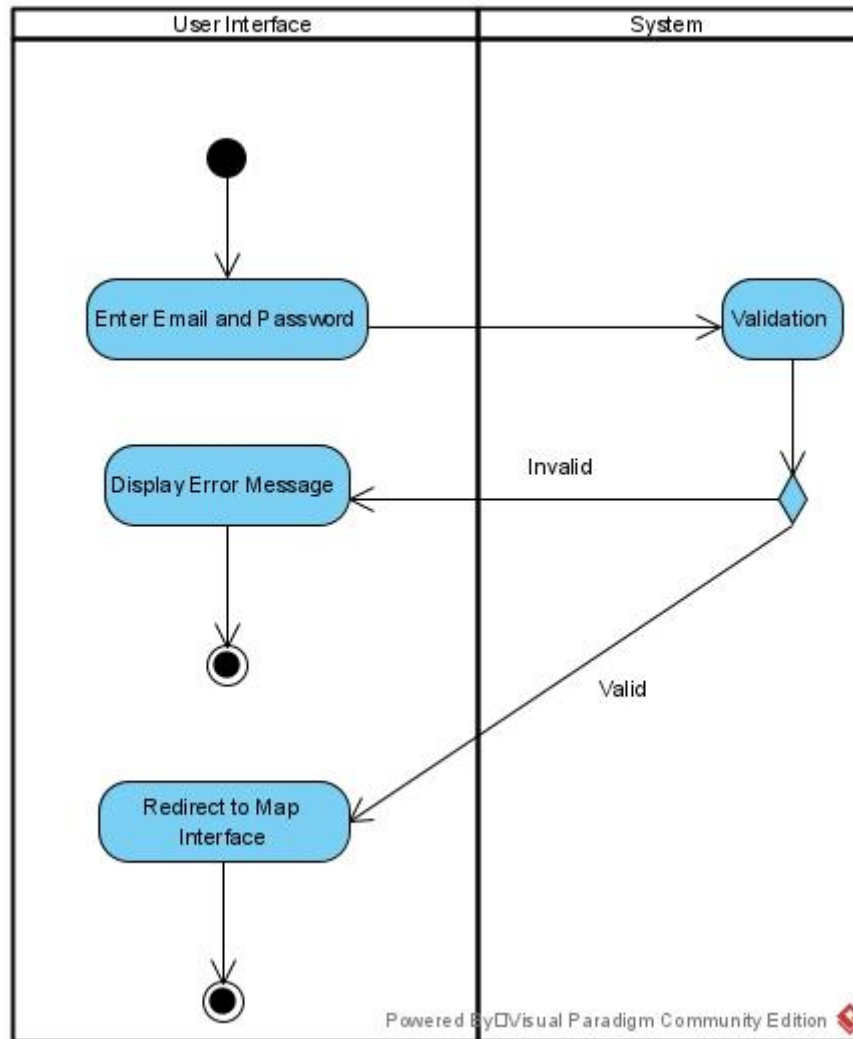


Figure 3.5.1.1 Login Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.5.2 Register Account

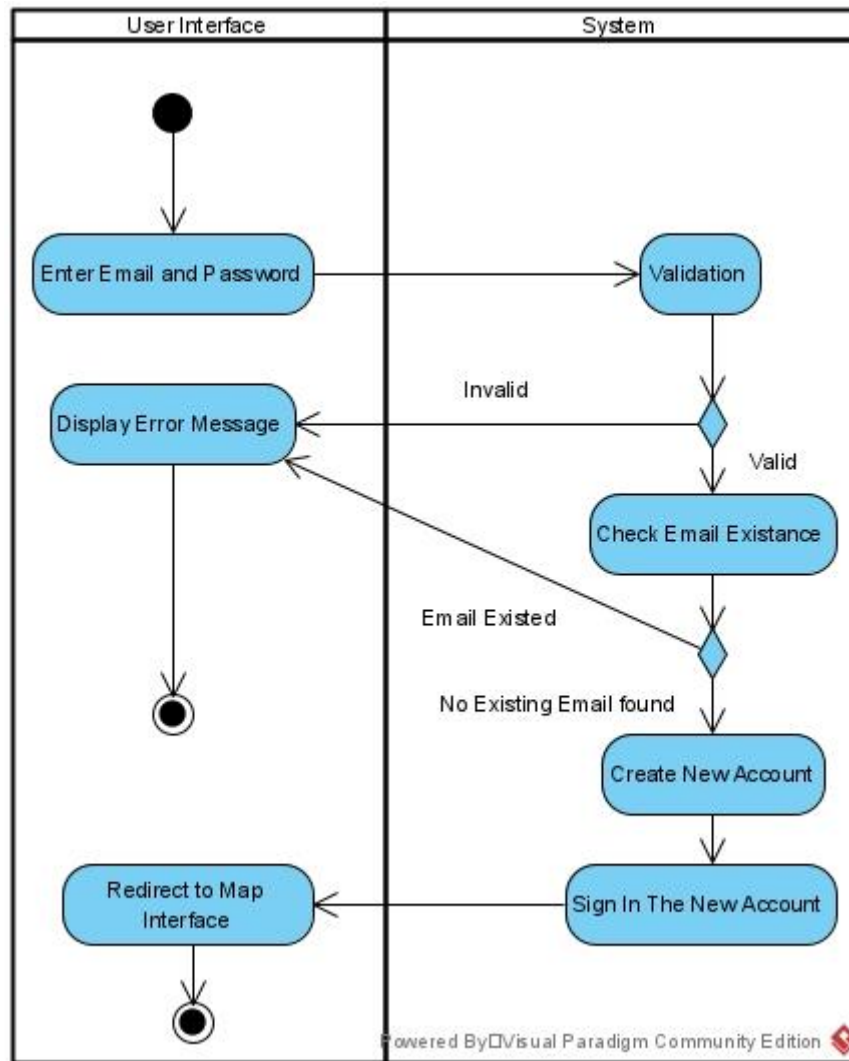


Figure 3.5.2.1 Register Account Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.5.3 Take Ticket

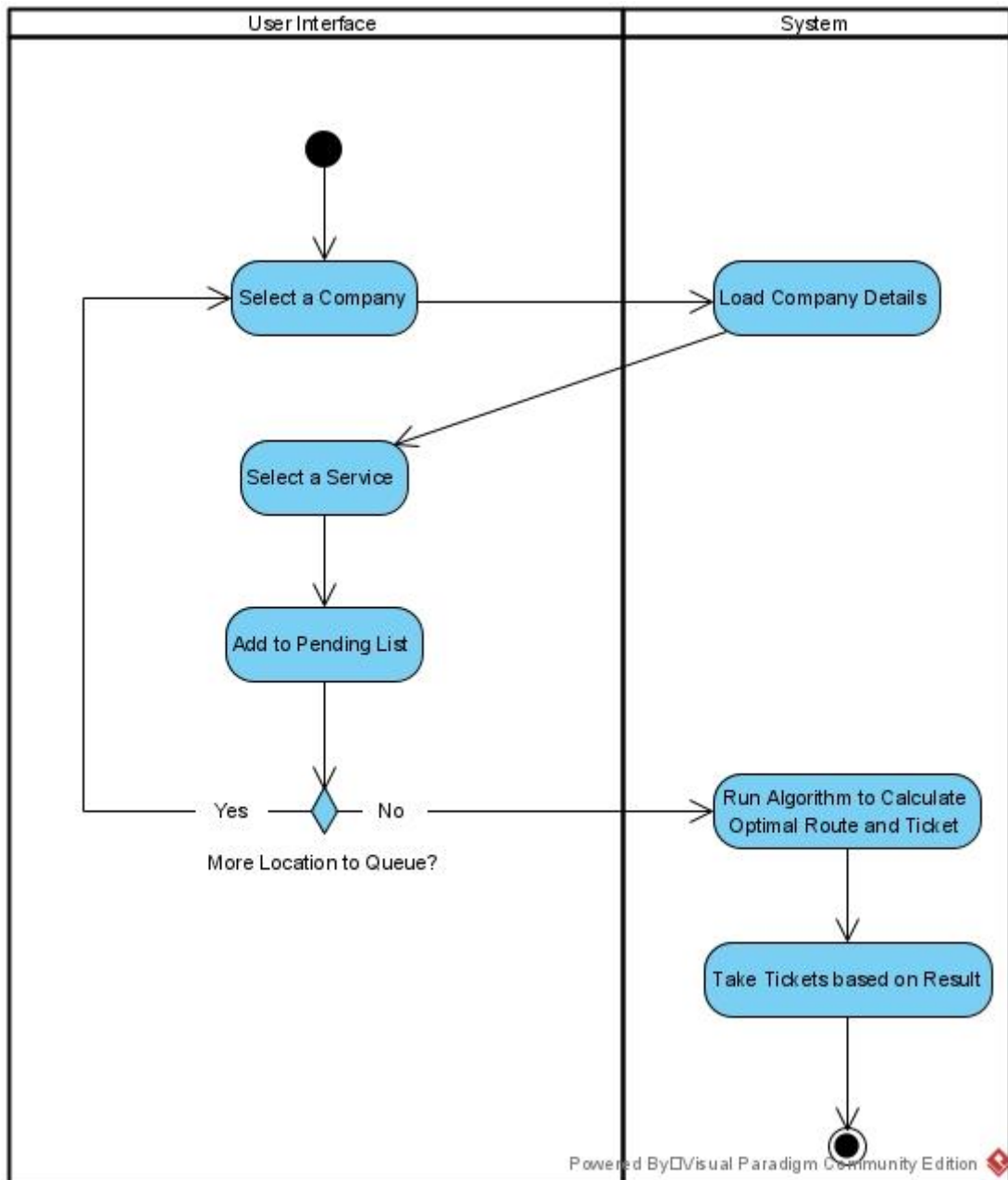


Figure 3.5.3.1 Take Ticket Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.5.4 Show Route

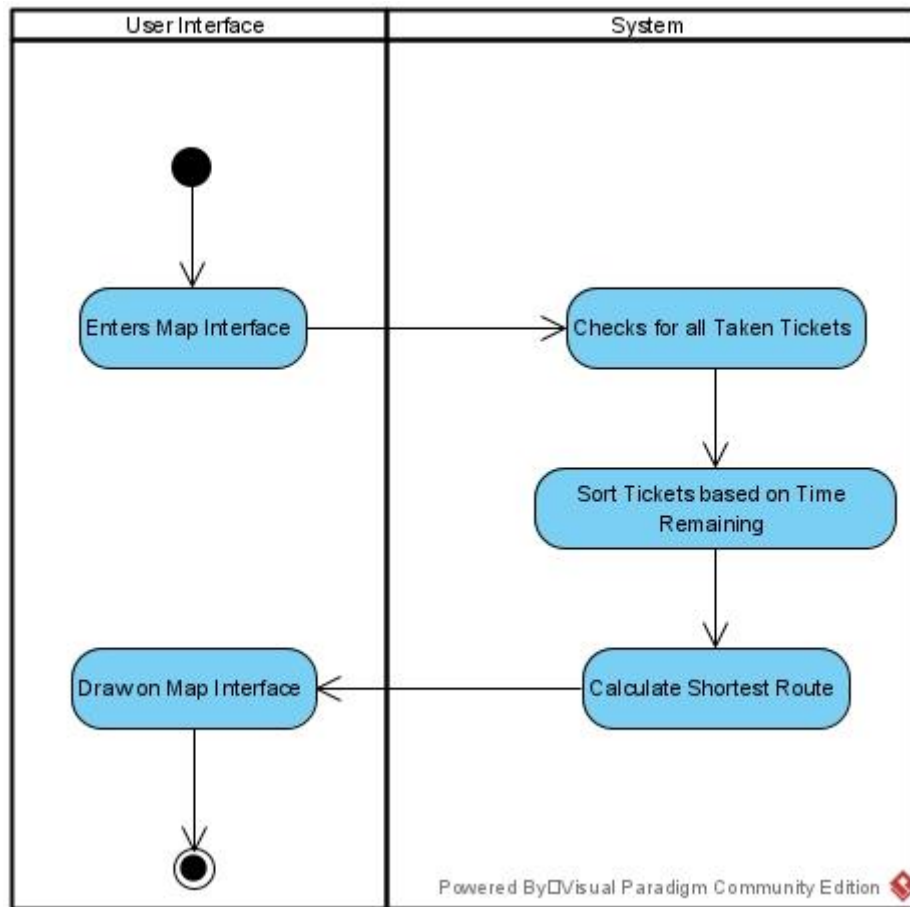


Figure 3.5.4.1 Show Route Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.5.5 Delay Ticket

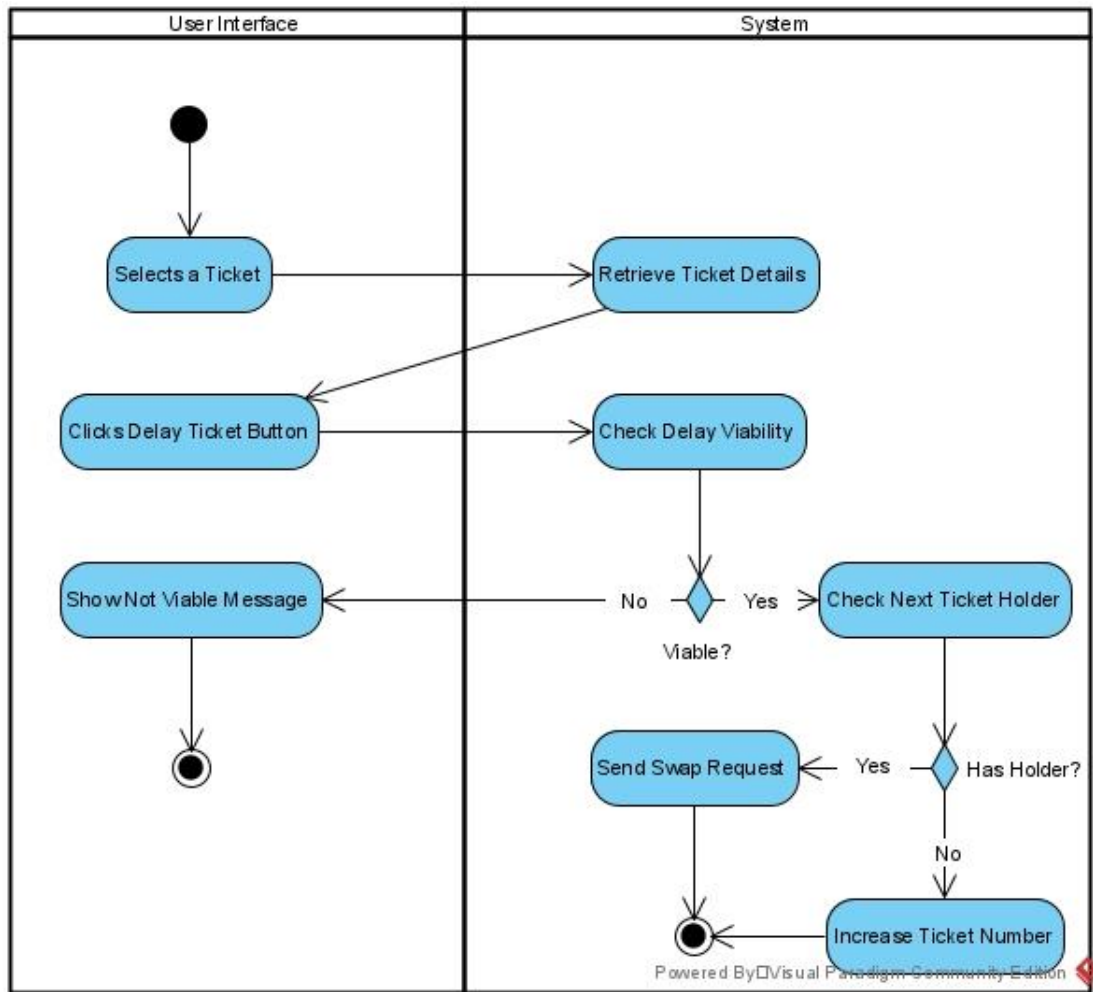


Figure 3.5.5.1 Delay Ticket Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.5.6 Respond to Swap Request

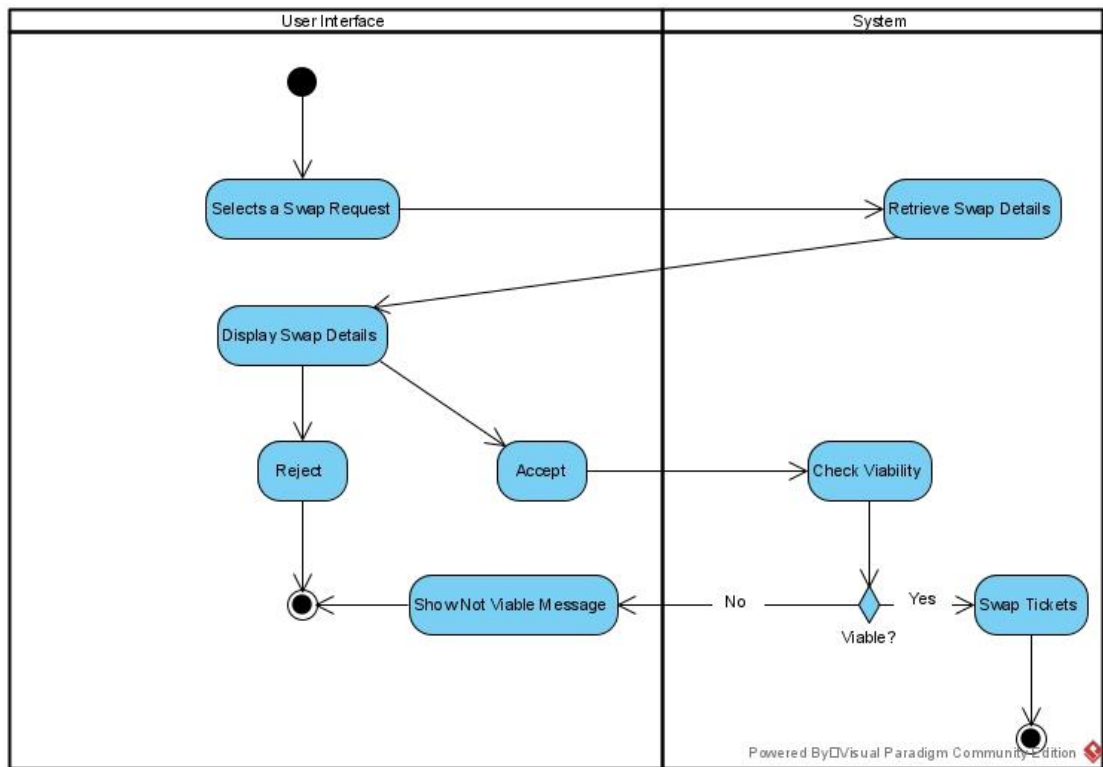


Figure 3.5.6.1 Respond to Swap Request Activity Diagram

CHAPTER 3: SYSTEM DESIGN

3.6 Sequence Diagram

3.6.1 Take Ticket

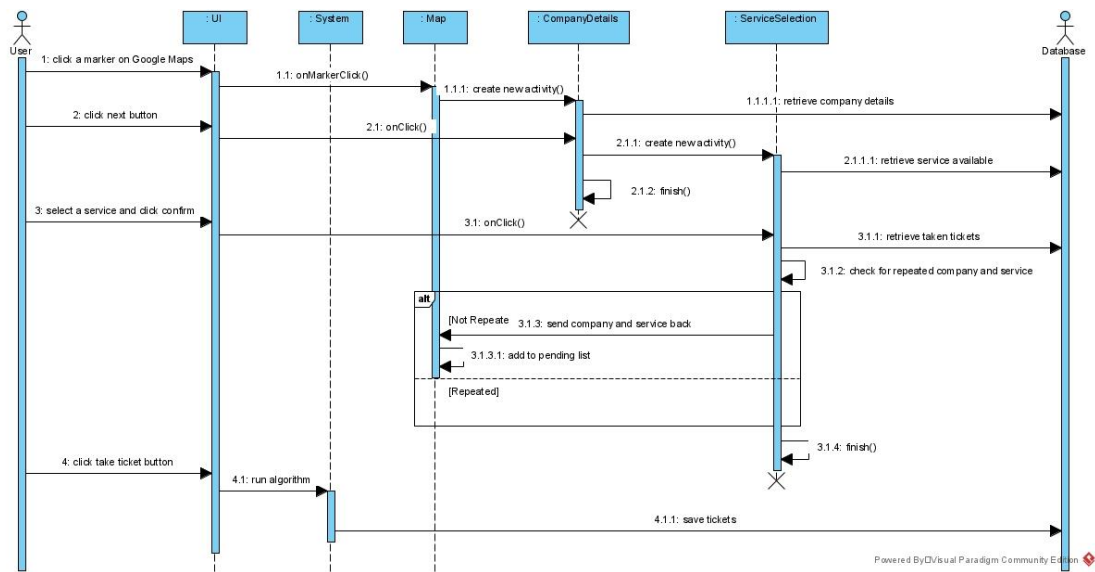


Figure 3.6.1.1 Take Ticket Sequence Diagram

3.6.2 Show Route

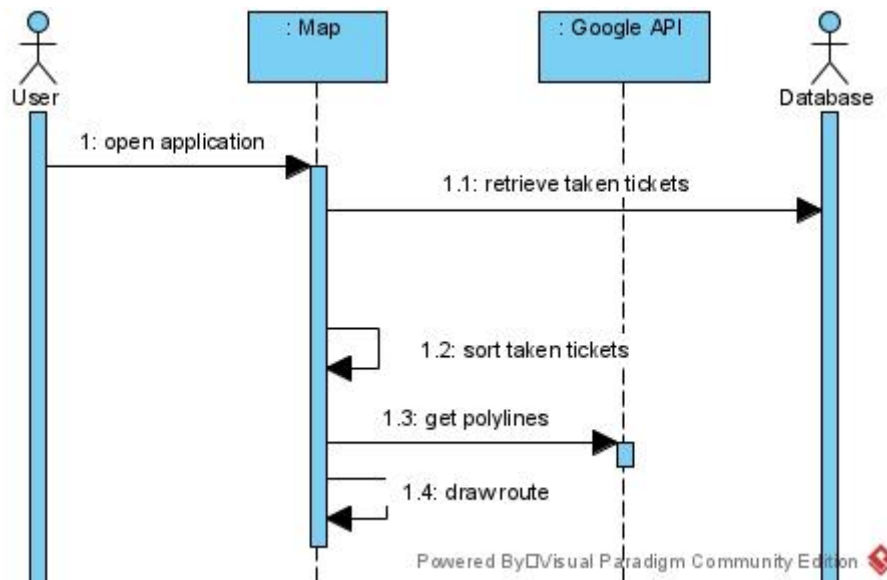


Figure 3.6.2.1 Show Route Sequence Diagram

3.6.3 Delay Ticket

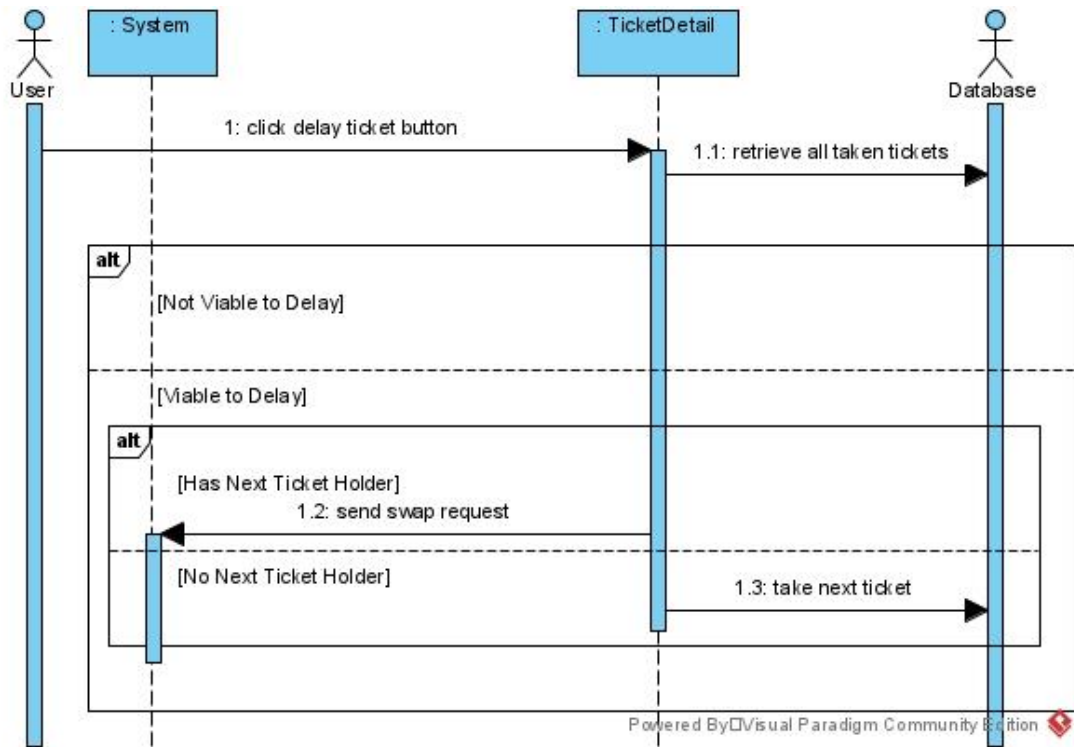


Figure 3.6.3.1 Delay Ticket Sequence Diagram

CHAPTER 3: SYSTEM DESIGN

3.7 Firebase Structure



Figure 3.7.1 Top Layer of Firebase

CHAPTER 3: SYSTEM DESIGN

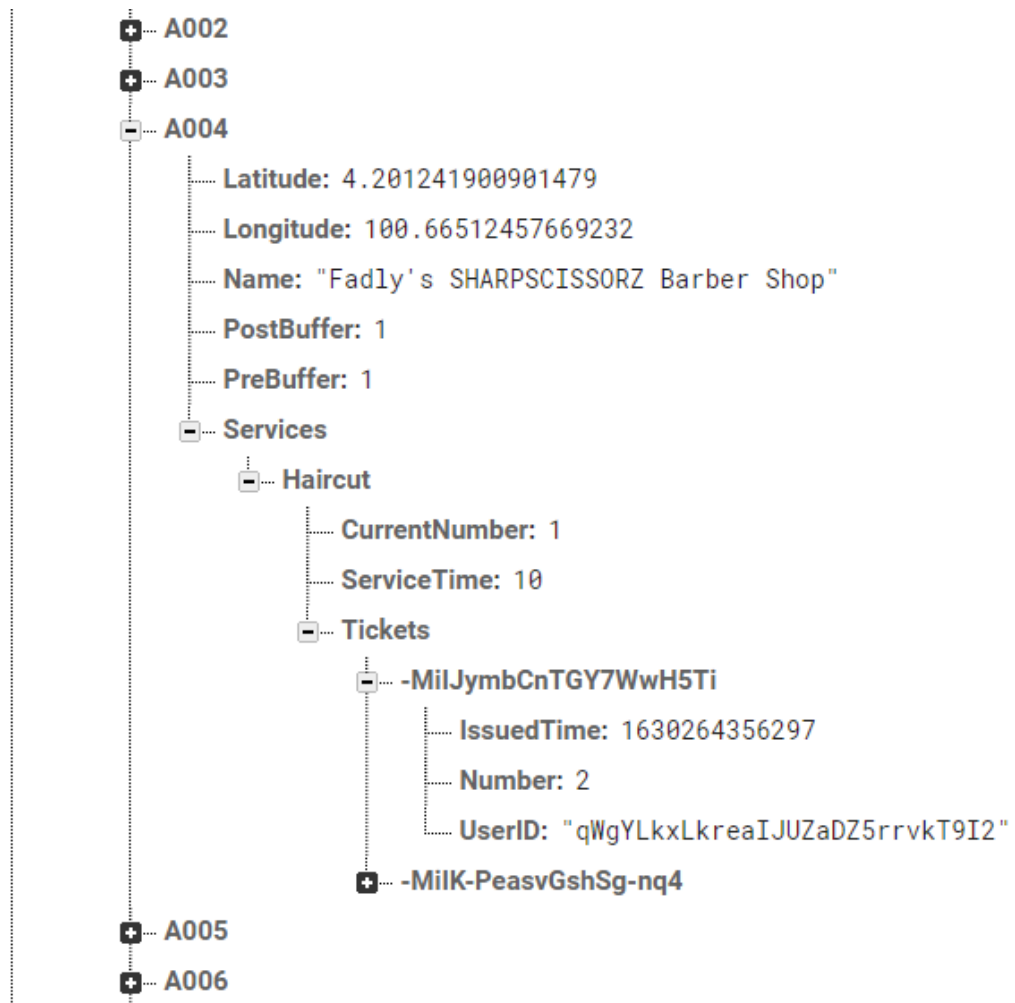


Figure 3.7.2 Places Child Layer

CHAPTER 3: SYSTEM DESIGN



Figure 3.7.3 Users Child Layer

CHAPTER 4: SYSTEM METHODOLOGY AND TOOLS

4.1 System Methodology

The methodology for developing this system is prototype developing methodology.

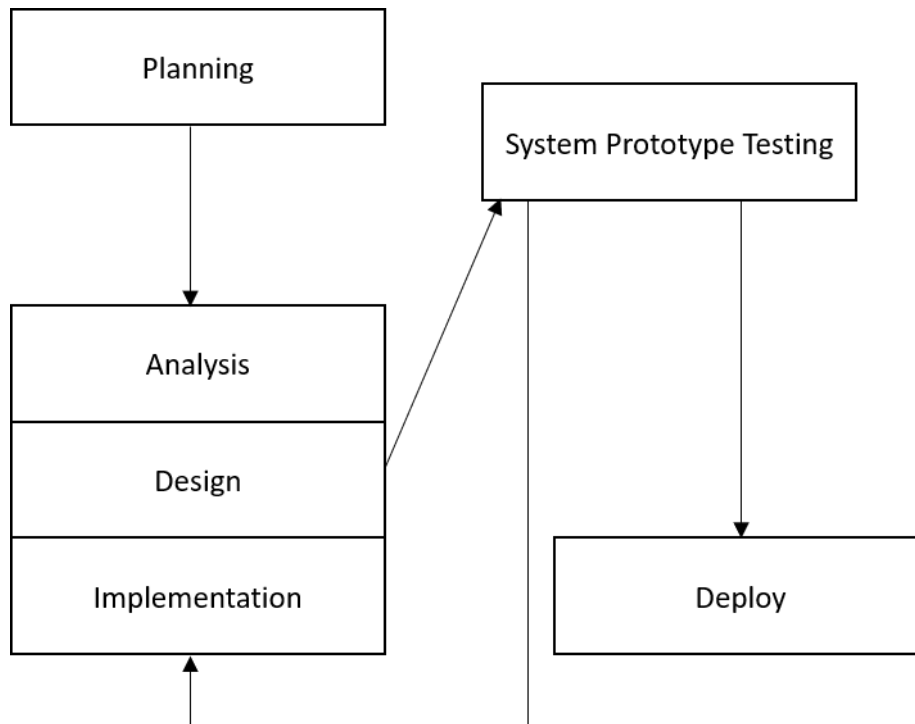


Figure 4.1.1 System Methodology

4.2 Project Workflow

4.2.1 Planning Phase

During planning phase, a set of plans is created to manage time, cost, quality, and issues. A timeline is created to plan the development time. Besides, the preliminary report is evaluated to define problem statements, project objectives, project scopes, and contribution. The proposed application is developed based on the timeline created and project objectives stated. The plans created act as guides to make sure development is according to plan and within expectation.

4.2.2 Analysis Phase

During analysis phase, analysing of similar projects is done to gather information needed. The applications on current market are observed and evaluated to find out their strengths and weaknesses. A table of comparison is created to visualise the differences. The strengths of the analysed applications are recorded which will be implemented to proposed application while trying to solve the weaknesses. Besides,

CHAPTER 4: SYSTEM METHODOLOGY AND TOOLS

some algorithms are also being reviewed to choose a suitable algorithm for this project. The proposed solution is also refined according to the reviews.

4.2.3 Design Phase

In design phase, a general design specification for whole system is performed based on the requirement gathered. User interface and system flow is designed to make sure application developed is well organised.

4.2.4 Implementation Phase

As soon as the design specification is finalised with no problems, implementation phase will be started which in this stage coding is being done and first version of prototype is created. The development is based on the design specification stated on design phase.

4.2.5 System Prototype Testing Phase

System testing is conducted once the prototype is created. Three phases of analysis, design and implementation are repeated to continue developing next version until the prototype is fully developed and ready to implement into system. If the prototype passes the testing, it will be implemented into system.

4.2.6 Deploy Phase

During this phase, the final system is released to end user. User feedback will be collected, and maintenance will be made based on user feedback.

4.3 Project Timeline

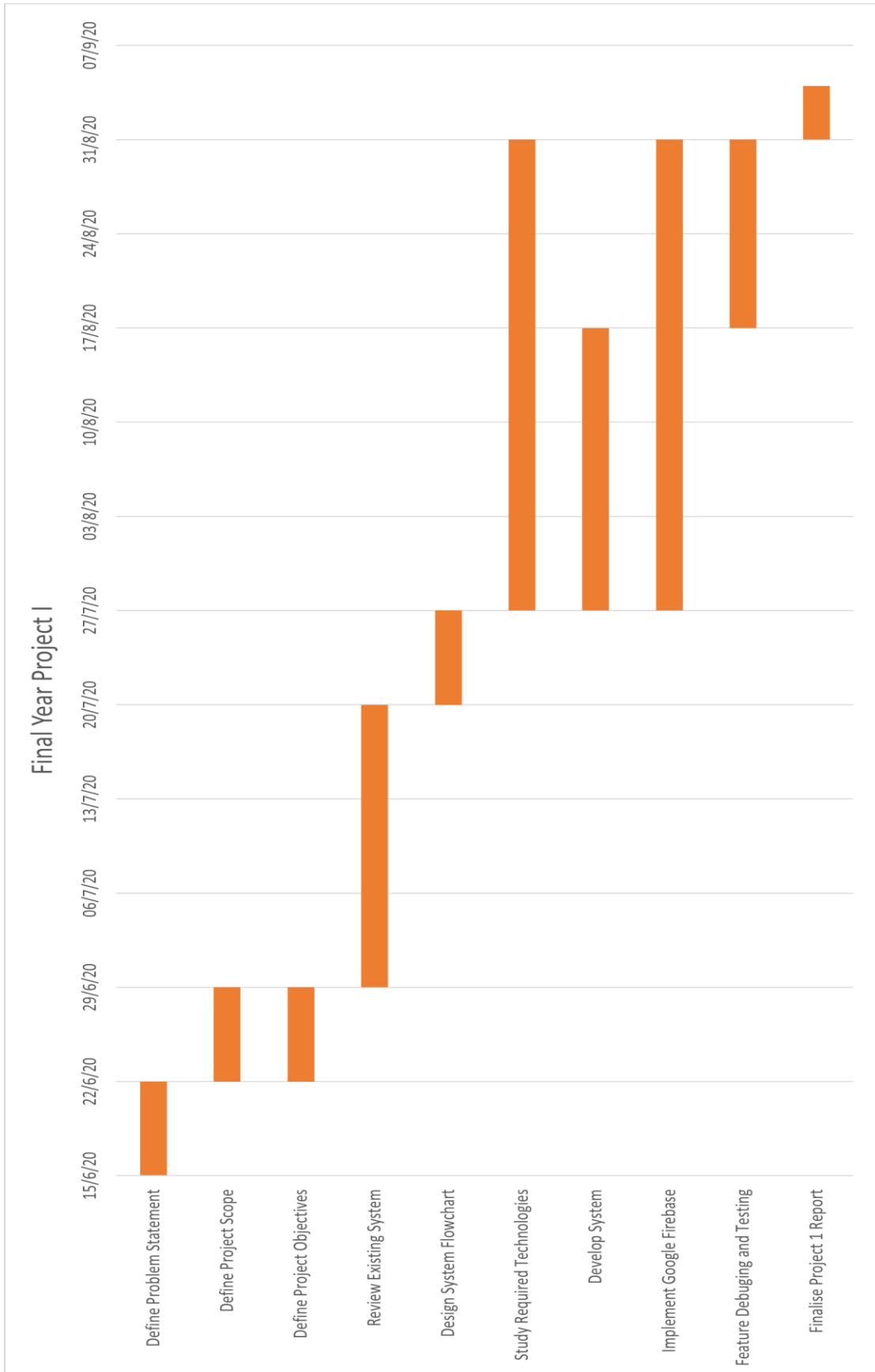


Figure 4.3.1 Gantt Chart for Project I

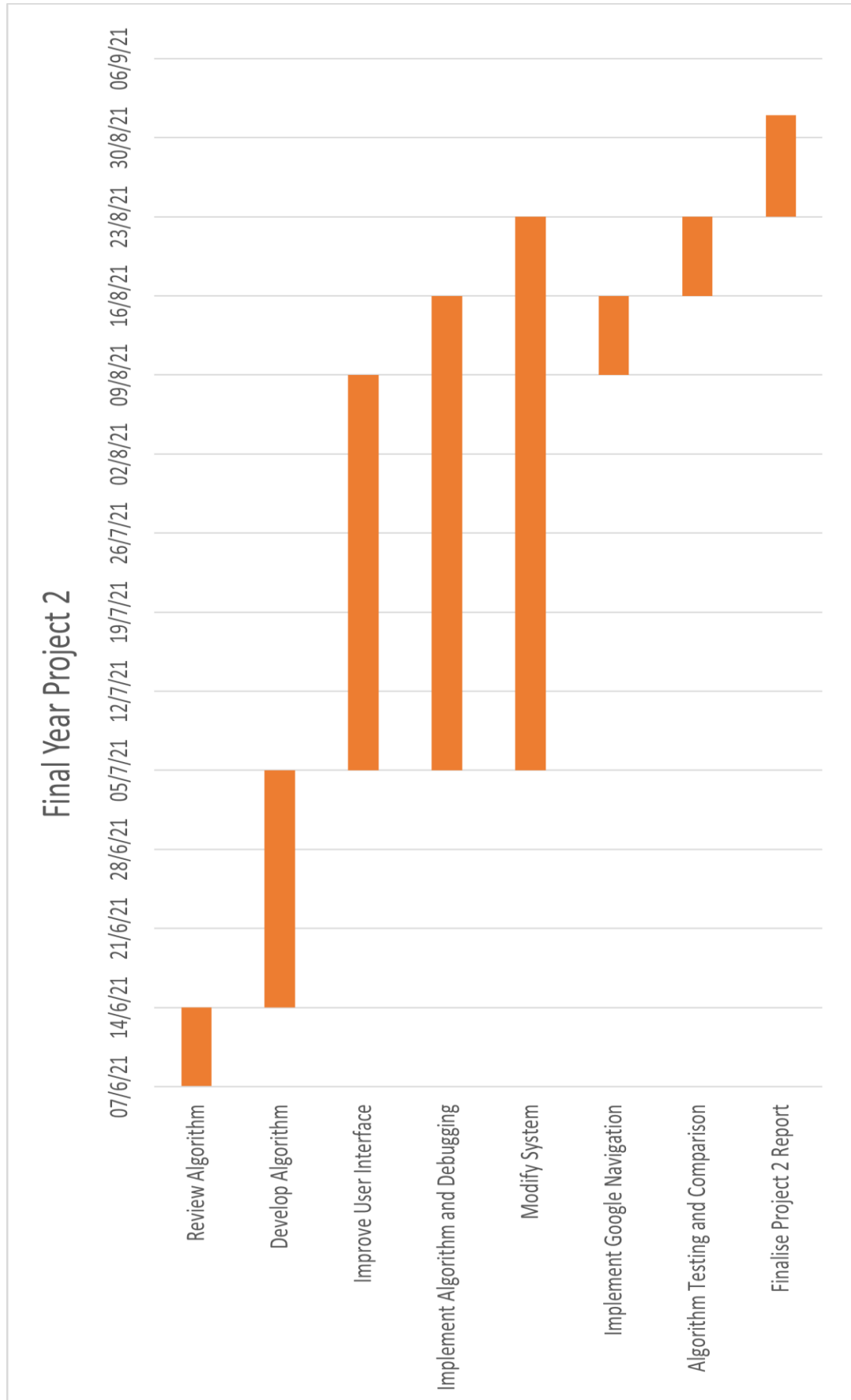


Figure 4.3.2 Gantt Chart for Project II

CHAPTER 4: SYSTEM METHODOLOGY AND TOOLS

4.4 Technologies and Tools Involved

4.4.1 Computer

Processor	Intel(R) Core(TM) i5-8400 CPU @ 2.80GHz
RAM	16GB
Hard Drive	1TB HDD
Operating System	64-bit Microsoft Windows 10
Graphic Card	GTX 1050 3GB
Monitor	AOC 2470W 24'' inch

Table 4.4.1.1 Computer Used

Table 4.4.1.1 shows the computer used to develop the system. The processor used is Intel i5-8400 with 2.8GHz clock speed. A 16GB RAM is used to make sure Android Studio can be run smoothly.

4.4.2 Emulator

Model	Galaxy Nexus
CPU/ABI	Google APIs Intel Atom (x86)
Android Version	Android 8.0
RAM	1GB
Internal Storage	800MB

Table 4.4.2.1 Emulator Used

Table 4.4.2.1 shows the emulator used to test the develop application. The Android Version is set to Android 8.0 to ensure the notification function works as expected. 1GB of RAM is used to make sure there is enough space to execute the application.

4.4.3 Android Studio



Figure 4.4.3.1 Android Studio

Android Studio is the official Integrated Development Environment (IDE) that is mainly used to create android mobile application. It is based on IntelliJ IDEA which provides powerful code editor and development tools. Besides, Android Studio also provide Gradle-based build system, emulator and built-in support for Google Cloud Platform (Android Developers 2020). The project of Android Studio is written in Java language and XML language.

4.4.4 Google Firebase



Figure 4.4.4.1 Firebase

CHAPTER 4: SYSTEM METHODOLOGY AND TOOLS

Google Firebase provides real-time database which is efficient for mobile application. It stores and synchronises data with NoSQL cloud database in real-time (Firebase Realtime Database 2020). Google Firebase also provides cloud functions which is useful to send real-time notification to specific users. When data is added into Firebase, this can trigger the cloud function in real-time manner. Google Firebase also provides authentication service which manages user accounts that are registered through mobile application.

4.4.5 Google Directions API

Google Directions API is a web service that uses HTTP request to get details such as travel distance, travel time and polylines between multiple locations. The polylines or routes calculated by this API are always optimised according to travel time. The response of the HTTP request is formatted in JSON format (Google Developers 2021).

4.4.6 Google Distance Matrix API

Google Distance Matrix API, in contrast to Direction API, only return travel distance and travel time as response (Google Developers 2021). This API is useful if polylines between locations are not needed when processing the data.

4.5 System Requirements

4.5.1 Functional Requirements

- User should be able to login to account using email.
- User should be able to register a new account using email.
- User should be able to queue in multiple locations.
- User should have the ability to view calculated route on map.
- User should be able to search a location by typing the location name.
- User should have the ability to view all tickets taken by user.
- User should be able to see ticket details of any ticket taken by user.
- User should be able to cancel a ticket.
- User should be able to delay a ticket.
- User should be able to respond to a swap ticket request.
- User should have the ability to modify personal account.
- User should be able to log out from the account.
- System should be able to connect to Firebase Authentication Service.
- System should be able to connect to Firebase Realtime Database.
- System should be able to get user's current location.
- System should be able to generate optimal route for user.
- System should be able to find ticket numbers according to optimal route.
- System should be able to ask permission from user.

4.5.2 Non-Functional Requirements

- System should be able to retrieve accurate data from Google Firebase.
- System should be able to get accurate current location data of user.
- System should be able to prompt correct error message.
- System should be able to send notification to user.

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

5.1 Scenario

When user wants to queue in several places, user needs to select all the places he or she wants to go before system can calculate the shortest route in terms of time. The input is important since adding a new destination to a calculated shortest route may affect the accuracy of the route. If user chooses to add new destination while the user has already queueing in other places, the system will need to take all taken ticket places into consideration while finding optimal path, which may lead to suboptimised route. Therefore, it is recommended to decide all queue places in one shot rather than selecting queue places one by one. Due to the limitation of Google Distance Matrix API, the number of maximum places per HTTP request is 10 locations. In other words, user can select up to 9 queue locations because the HTTP request will need to include user's current location.

When calculating the cost, system will consider travel time between places and time buffer reserved for user such as parking time, walking time from car to building and time required for user to ready to depart after finishing a service in a place. The buffer time used in this project is predefined value since it is very difficult to find all the buffer time very accurately. When finding the tickets number for every place, the system will first discover the smallest possible number, then compare the number with to be issued ticket number. If the next issue number is less than or equal to the calculated number, system will use the calculated number to calculate current cost.

If user has already taken some tickets in some places, and user wants to queue in a new place, system will record all taken tickets number taken by user as one of the parameters of the algorithm. The case here is that the taken tickets cannot be modified, therefore system will use these tickets number to decide if it is possible for user to queue in between two queueing spot or can only queue the location as last destination. System will check if the newly added destination can be fit in between any two points.

For user who wants to delay a ticket, the system allows user to either swap ticket with another user or increase the ticket number. A delay viability check will be conducted if user decides to delay a ticket. Delaying a ticket in in one location (Location A) will

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

affect the queue time in next location (Location B). This is because when the delay is successful, the user needs to stay in Location A for a longer time. This shortens the queue time in Location B which means that user has shorter queue time in Location B. To measure this, system will calculate the queue time in Location B, and check the queue time with the service time in Location A. If the queue time is more than the service time, then it is possible for user to reach Location A after delay, therefore user is allowed to delay the ticket. Similarly, for user received a swap request, system will need to make sure user can reach the location on time. Assume a user received a swap request for Location A. System will compare the queue time and service time of Location A: if service time is more than queue time, system will block user from accepting the ticket; otherwise, user is allowed to accept the swap request.

5.2 System Implementations

5.2.1 Login/Register

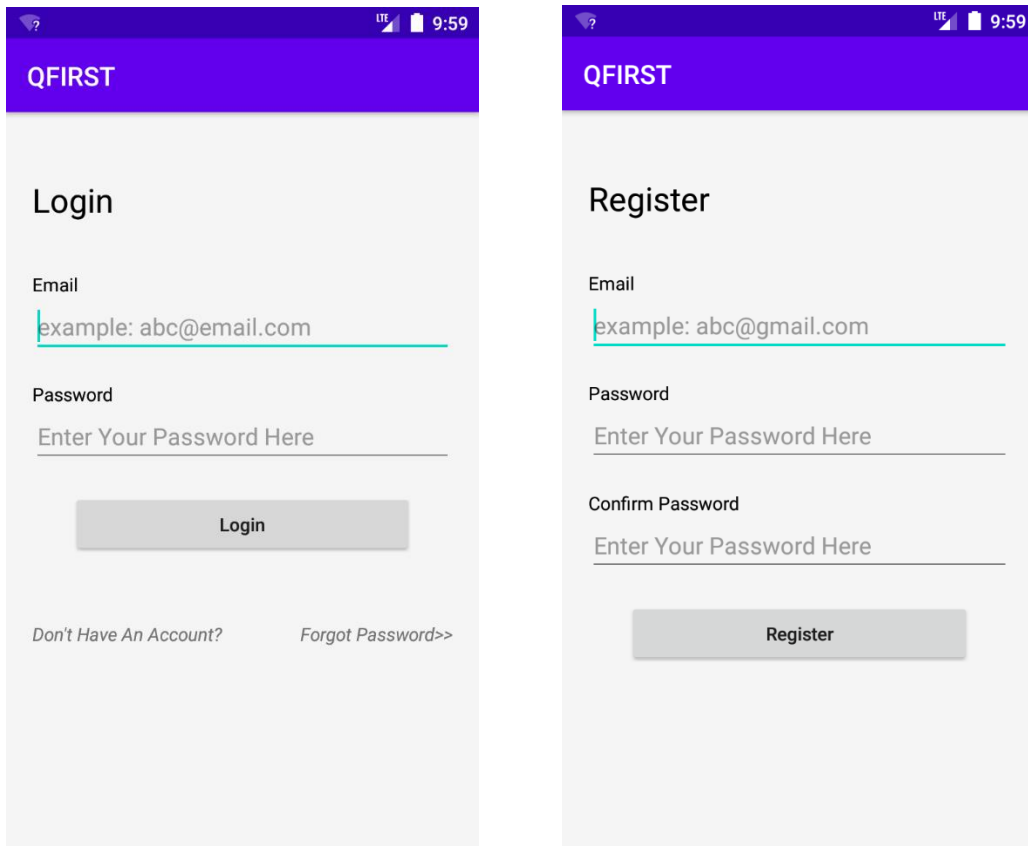


Figure 5.2.1.1 Login/Register Page

When user first open the application, user will be redirected to login page. If a user does not have an account, the user needs to register an account using email. If user has an account, the user can login using email and corresponding password. Figure 5.2.1.1 shows both login and register interfaces. The user will be redirected to map interface after logged in.

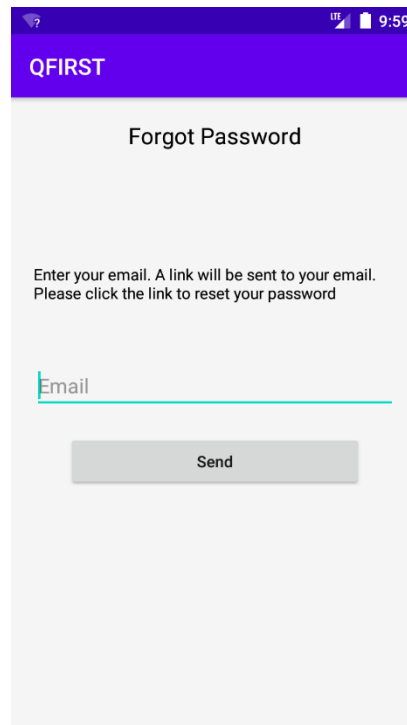


Figure 5.2.1.2 Forgot Password Page

Figure 5.2.1.2 shows forgot password interface. If user forgot the password, user could click the “Forgot Password” text shown on login page. User will be redirected to forgot password interface. An email containing a link will be sent to user’s email after user clicks send button.

5.2.2 Map Interface



Figure 5.2.2.1 Map Interface

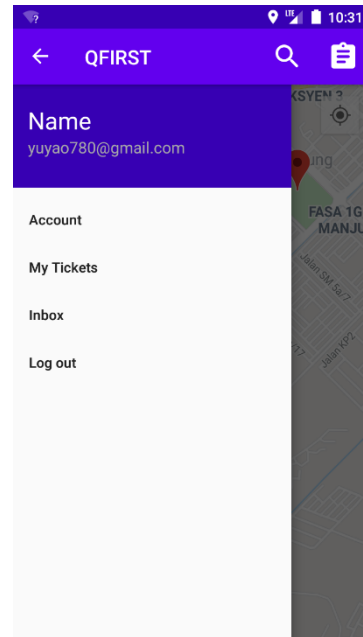


Figure 5.2.2.2 Drawer Navigation

Figure 5.2.2.1 shows the map interface. User can search for specific location by pressing search icon on the action bar. User can then type the name of a location and press the location shown on recommend list. The map camera will be relocated to the location and user is now able to see the queue location. Besides, user can click on the left of action bar to bring up the navigation drawer, as shown in Figure 5.2.2.2. In navigation drawer, the name and email of user are shown. Four selections are available for user to select, which are Account, My Tickets, Inbox and Log out.

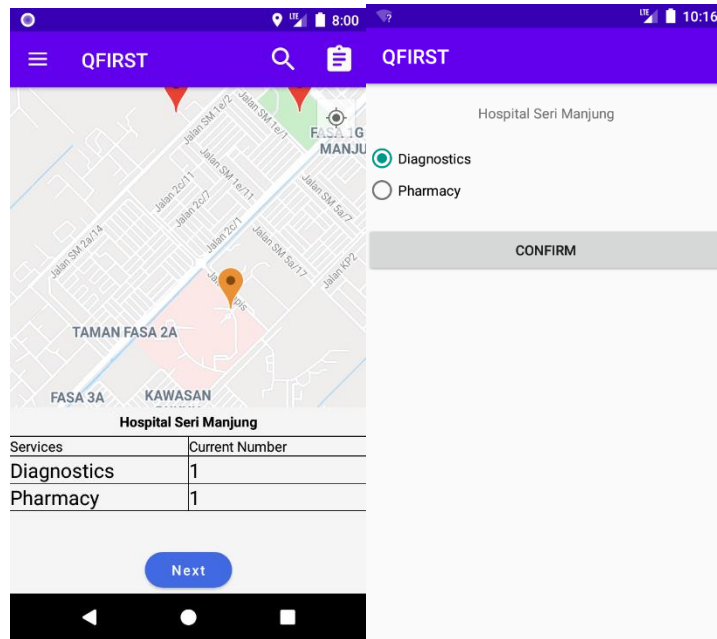


Figure 5.2.2.3 Select Queue Location and Service

To add a location to queue pending list, user can click on a marker on the map and select a service provided by the company, which is shown in Figure 5.2.2.3. When user clicks confirm button, the queue location will be added into pending list. The pending list icon is located on the right most side of action bar. User can simply click the icon to open the pending list.

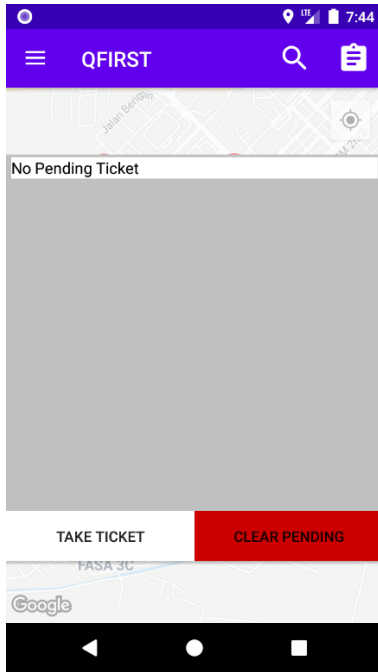


Figure 5.2.2.4 Empty Pending List

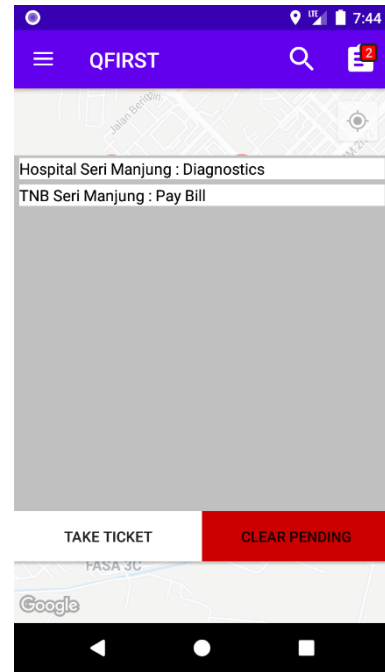


Figure 5.2.2.5 Pending List with Items

If there is no item inside pending list, an interface same as Figure 5.2.2.4 will be shown to user. Otherwise, the pending list icon on the action bar will show the number of items inside pending list on the upper right corner of the icon and queue location listed, as shown as Figure 5.2.2.5. To confirm to queue in every location inside queue pending list, user can click take ticket button.

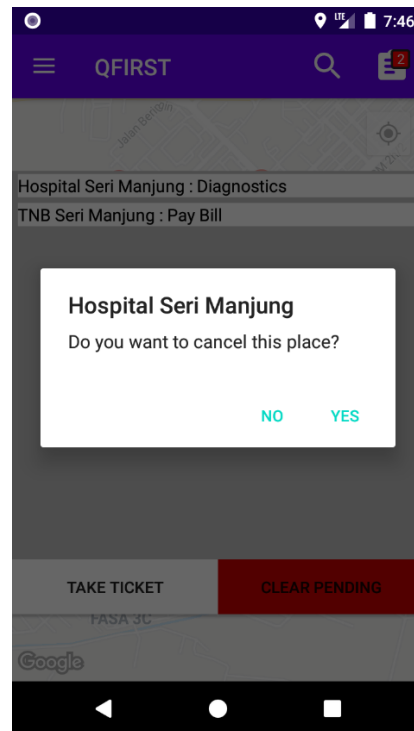


Figure 5.2.2.6 Item On Click

If user wants to delete items in pending list, user can either press an item in pending list, which will prompt user to confirm delete, or press the clear pending button in pending list. When user clicks on an item, a dialogue will be prompted to ask user to confirm delete, as shown in Figure 5.2.2.6.

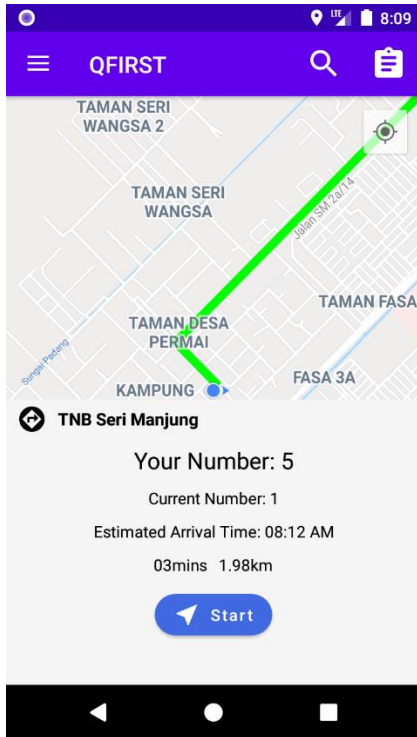


Figure 5.2.2.7 Route on Click

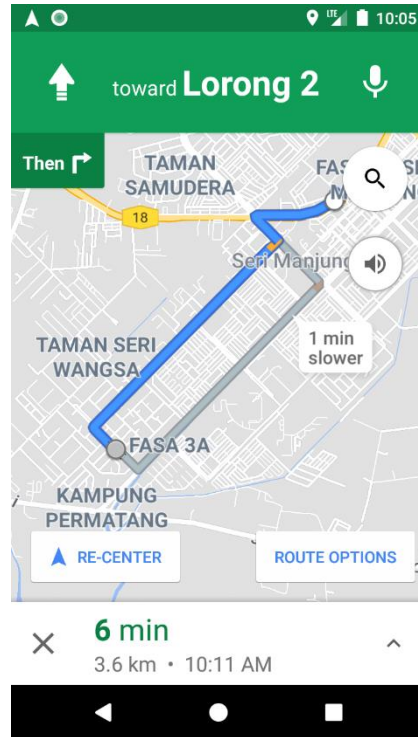


Figure 5.2.2.8 Navigation Interface

After user has successfully taken a ticket and the data is recorded into database, user will see a route automatically drawn on the map. As shown in Figure 5.2.2.7, the route can be clicked to show details such as ticket number of destination and travel time between two places. It is also possible for user to use navigation function of Google Maps by clicking the start button. Figure 5.2.2.8 shows an example of Google Maps Navigation.

5.2.3 Ticket List Interface

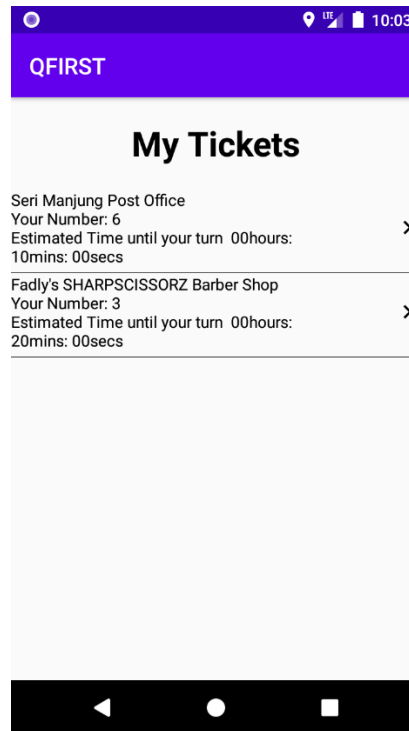


Figure 5.2.3.1 Ticket List Interface

When user selects “My Tickets” option in the navigation drawer, user can see a list of tickets taken by the user, which is shown in Figure 5.2.3.1. User can click an item to see more details of that selected ticket.

5.2.4 Ticket Details Interface

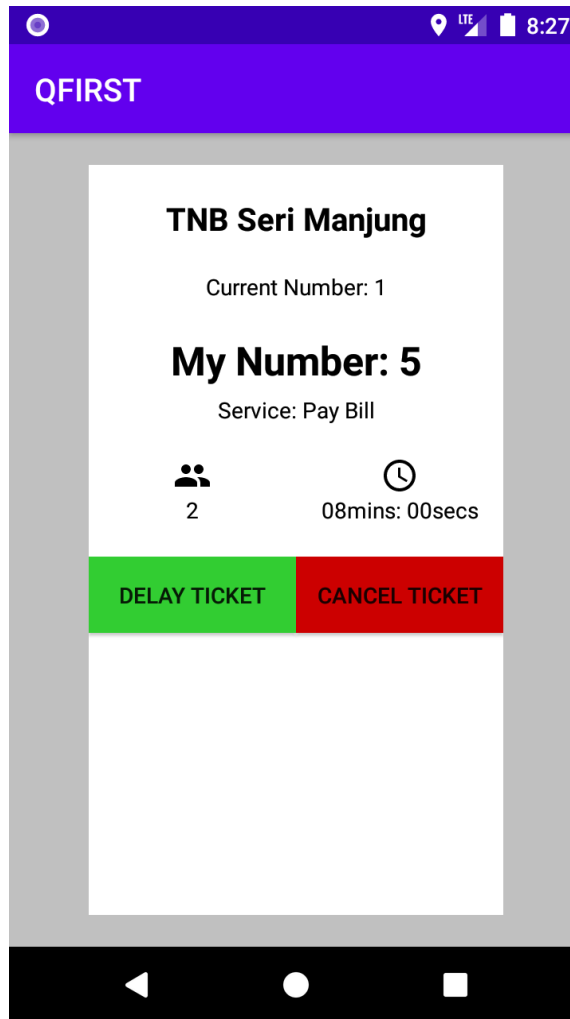


Figure 5.2.4.1 Ticket Details Interface

After user has selected a ticket from ticket list, the details will be shown to the user. Figure 5.2.4.1 shows the details user can see. Here user can choose to either delay the ticket or cancel the ticket if user wants to. Clicking delay button will trigger the system to check for viability to decide whether the user can delay the ticket. Clicking cancel button will prompt the user to confirm cancelling the ticket.

5.2.5 Inbox Interface

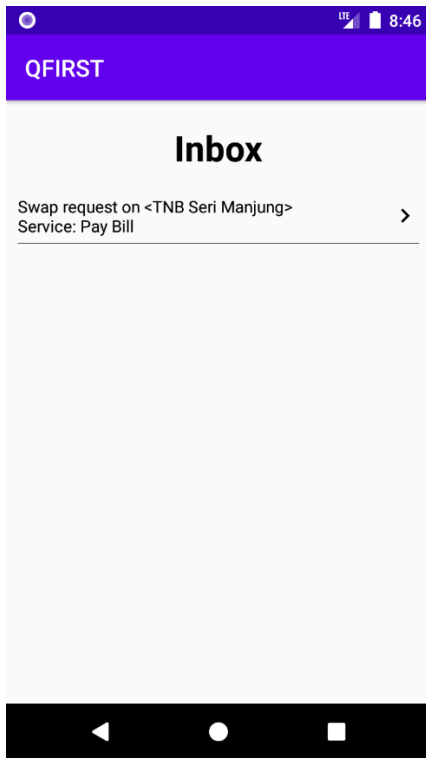


Figure 5.2.5.1 Inbox Interface

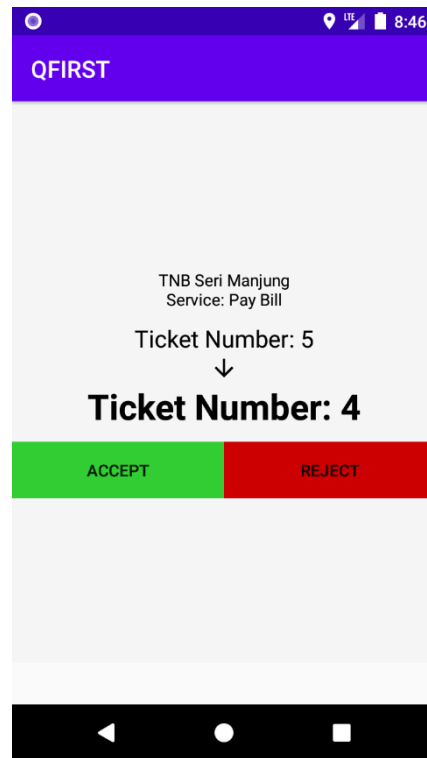


Figure 5.2.5.2 Inbox Item Interface

When user received a swap request, user can view the request inside the user's inbox. A list of available swap request will be shown here waiting for user to respond. Figure 5.2.5.1 shows a user has a swap request inside inbox User can simply click on an item to respond. Item details in Figure 5.2.5.2 will be shown to user, and user can choose to accept or reject the request.

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

5.3 System Testing

To make sure the route generated is optimal, brute force method is developed to test for the accuracy. Using brute force method will ensure all route combinations are explored, therefore the result is guaranteed to be the best. The places selected by every test is fixed for both algorithm and the location order selected for testing is random in every test. When selecting the locations to test, the locations are chosen in a random manner, meaning Location 1 for first test may be different from Location 1 for second test. This is to ensure the algorithm is still able to generate optimal route without having to fix the location number to certain point. To prove this point, a test is conducted using branch and bound by choosing three queue places, but the input order is different in each test.

Location 0: User current location

Location A: TNB Seri Manjung, Service: Pay Bill

Location B: Seri Manjung Post Office, Service: Bill Payment

Location C: Maybank, Service: Foreign Currency Exchange

Input Order	Result From Algorithm	Actual Path	Ticket at A	Ticket at B	Ticket at C
A, B, C	Path Taken: 0, 1, 2, 3 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7
A, C, B	Path Taken: 0, 1, 3, 2 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7
B, A, C	Path Taken: 0, 2, 1, 3 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7
B, C, A	Path Taken: 0, 3, 1, 2 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7
C, A, B	Path Taken: 0, 2, 3, 1 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7
C, B, A	Path Taken: 0, 3, 2, 1 Ticket Taken: 4, 9, 7	0, A, B, C	4	9	7

Table 5.3.1 Table of Different Input Order Test

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

Table 5.3.1 proves that no matter how is the input order, the final result will not be affected. Next, the testing on accuracy of branch and bound

Number of Places	Branch and Bound	Brute Force
2	Path: 0, 1, 2 Ticket: 10, 5	Path: 0, 1, 2 Ticket: 10, 5
4	Path: 0, 3, 1, 2, 4 Ticket: 3, 14, 9, 2	Path: 0, 3, 1, 2, 4 Ticket: 3, 14, 9, 2
6	Path: 0, 2, 6, 3, 4, 5, 1 Ticket: 4, 2, 35, 18, 29, 3	Path: 0, 2, 6, 3, 4, 5, 1 Ticket: 4, 2, 35, 18, 29, 3
8	Path: 0, 1, 2, 3, 5, 6, 7, 8, 4 Ticket: 3, 7, 11, 20, 4, 14, 151, 4	Path: 0, 1, 2, 3, 5, 6, 7, 8, 4 Ticket: 3, 7, 11, 20, 4, 14, 151, 4

Table 5.3.2 Table of Algorithm Testing

Table 5.3.1 shows the algorithm testing result. To evaluate the table, “Path” is the best or optimal path found by the algorithm, and “Ticket” is the ticket number of every queue location other than starting point. For example, for the test on two places, a “Path” of “0, 1, 2” shows the path starts from “0” which is user’s current location, then traverse to location “1” and finally location “2”. For “Ticket”, first ticket number “10” is the ticket number decided by system for location “1” and second ticket number “5” is the ticket number decided by system for location “2”.

Figure 5.3.1 is another example of how to evaluate data in Table 5.3.1.

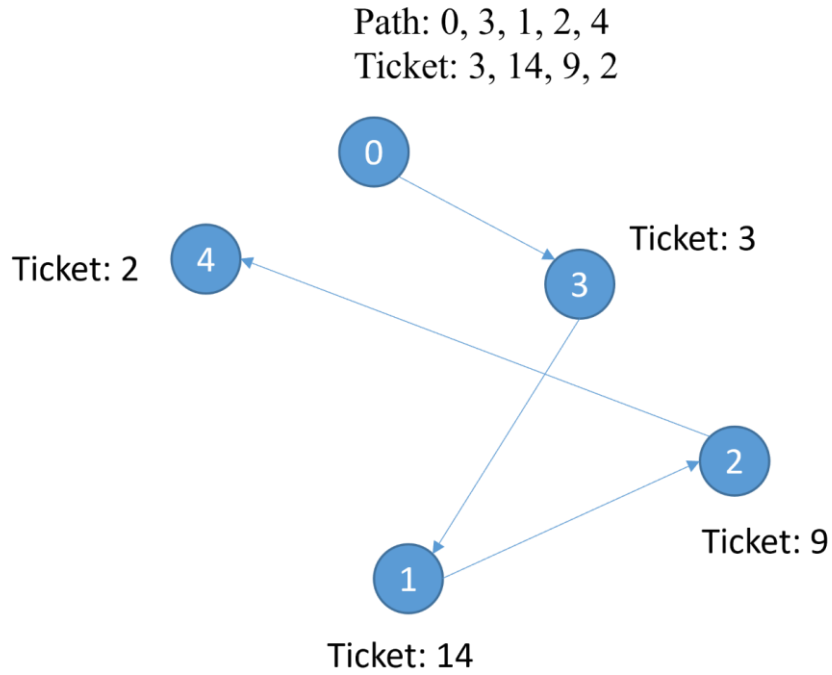


Figure 5.3.1 Visualisation of Optimal Route

From Table 5.3.1, both optimal path and ticket numbers of corresponding locations are the same, which indicates that the accuracy of branch and bound algorithm is reliable.

For ticket delay viability test, first we must find out a route which contains two or more queue locations. Using the system, we can find out that travelling from user’s current location to TNB Seri Manjung and Hospital Seri Manjung will generate an optimum route of user’s current location to TNB Seri Manjung to Hospital Seri Manjung. We simplify the route to 0, 1, 2 for easier understanding. The ticket taken at location 1 is 4 and location 2 is also 4. The cost of 0 to 1 is 348s and 1 to 2 is 502s. From database, we can get the service time, prebuffer and post buffer time. Travel time from one place to next place can be collected by calling Google Distance Matrix API. Table 5.3.3 shows the data collected in table form.

Location	Ticket	Current Ticket	Service Time(s)	Pre Buffer(s)	Post Buffer(s)	Travel time to next location(s)
0	-	-	0	0	30	198
1	4	1	120	120	60	202
2	4	1	300	120	120	-

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

Table 5.3.3 Table of Data of Queue Locations

From Table 5.3.3, queue time at location 2 can be calculated since it is one of the factors affecting the viability of delay ticket. The equation is as below:

$$t_i = (T_i - C_i) * S_i \quad (4)$$

Where

t_i is time requires to reach user's ticket at location i

T_i is user's ticket number at location i

C_i is current ticket number at location i

S_i is service time at location i

Using equation (4), we can get $t_1 = 360s$ and $t_2 = 900s$. To get real queue duration of location 2, the equation is as below:

$$Q_j = t_j - t_i - S_i - P_i - p_j - T_{ij} \quad (5)$$

Where

$$j = i + 1$$

Q_j is queue duration when arrived at location j

t_j is time requires to reach user's ticket at location j

S_j is service time at location j

P_j is post buffer time at location j

p_j is pre buffer time at location j

Using equation (5) we can get $Q_2 = 900 - 360 - 120 - 60 - 120 - 202 = 38s$. This indicates that user is expected to have 38seconds more when the user arrives at location 2 before it is user's turn. If user wants to delay ticket at location 1, the service time in location 1 must be less than queue time in location 2, which in this case is not. Therefore, the delay is not viable. When system is checking for viability, the logic is matched with the system function executed and the value calculated matches the testing value, therefore delay ticket function is working as expected.

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

On the other hand, for a user to accept a swap request in location 1, the service time in location 1 must be less than queue time in location 1 also, since accepting a swap request will decrease ticket number by 1 which requires enough queue time to do so. Refer again to Table 5.3.3 and equation (4), we can get $t_1 = 360s$ and $t_0 = 0$. To get real queue duration of location 1, equation (5) is used, and the result is $Q_1 = 360 - 0 - 0 - 30 - 120 - 198 = 12s$. Since the queue time is less than the service time at location 1, the swap request cannot be accepted. The value calculated by the system matches the value calculated in this section, therefore the function is also working as expected.

5.4 Algorithm Comparison

5.4.1 Performance Test

The performance test is conducted by comparing execution time of 3 different algorithms, which are branch and bound, brute force and nearest neighbour. The result is recorded in Table 5.4.1.1, as shown in below. A graph of execution time against number of places is generated based on the test result.

Number of Places	Execution Time (ms)		
	Branch and Bound	Brute Force	Nearest Neighbour
1	2.6343	1.4832	0.0333
2	3.6422	3.5633	0.0598
3	8.3482	8.5274	0.0492
4	21.8526	17.8789	0.9666
5	23.2369	35.1155	0.0305
6	225.5302	109.4354	0.2263
7	108.5876	231.9107	0.0388
8	829.1857	1847.6455	0.0512
9	5719.1121	15912.6933	0.0557

Table 5.4.1.1 Table of Performance Test

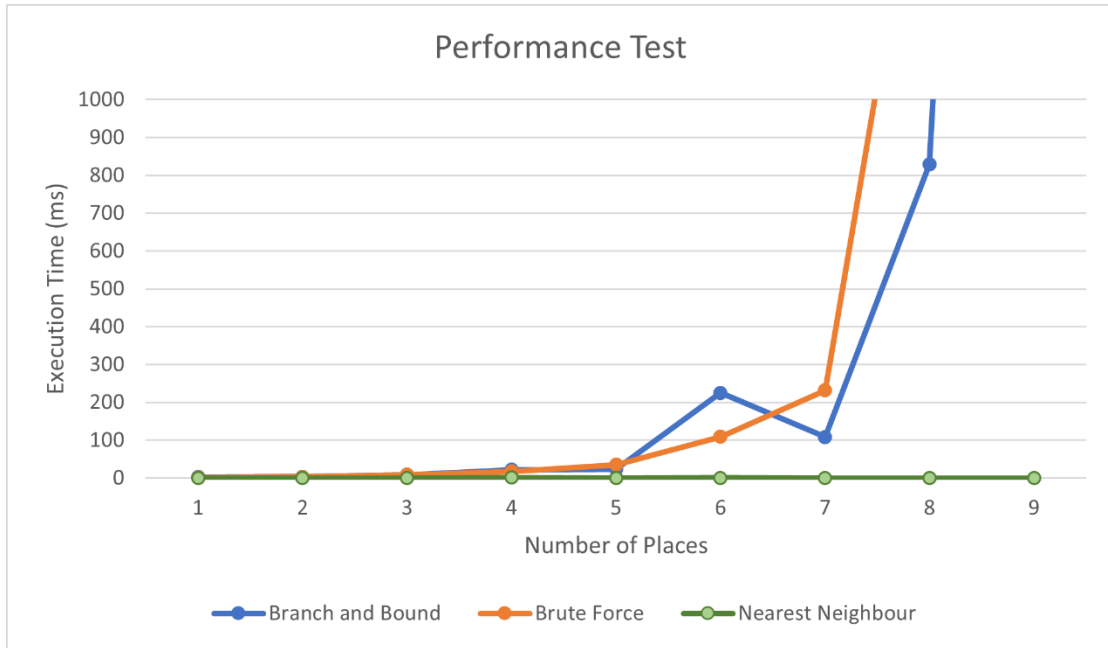


Figure 5.4.1.1 Performance Test Graph Shown up to 1000ms

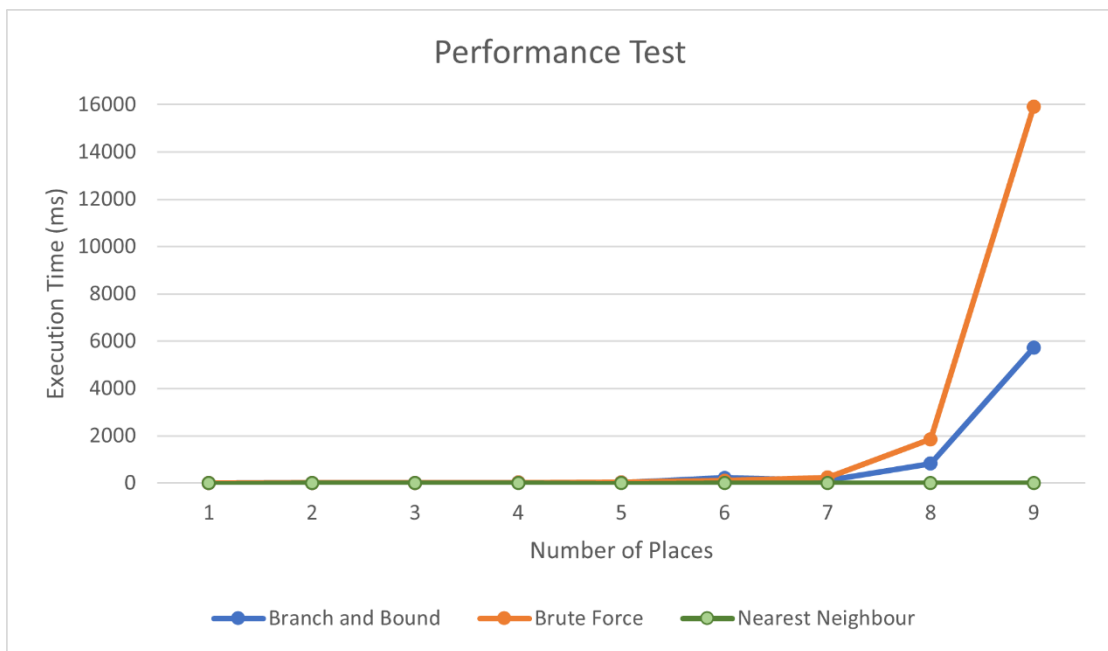


Figure 5.4.1.2 Performance Test Graph Shown up to 16000ms

Figure 5.4.1.1 shows the graph with y-axis shown up to 1000ms, while Figure 5.4.1.2 shows the graph with y-axis shown up to 16000ms. From the graph, we can see that nearest neighbour has fastest execution time. Brute force execution time increases drastically when the number of places increases. Branch and bound algorithm uses less execution time when compared to brute force.

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

5.4.2 Accuracy Test

To test for accuracy, the minimum cost is recorded, and brute force algorithm is used as a benchmark since the algorithm generates all the combination of routes and will return optimal route.

No. of Places	Minimum Cost Found(s)		
	Brute Force	Branch and Bound	Nearest Neighbour
1	480	480	480
2	1500	1500	1500
3	2160	2160	2160
4	2400	2400	2400
5	3600	3600	3600
6	7200	7200	10800
7	7200	7200	7200
8	10800	10800	10800
9	10800	10800	10800

Table 5.4.2.1 Table of Minimum Cost Found

In Table 5.4.2.1, three algorithms have most of the minimum cost similar, but when calculating optimal route for 6 places, nearest neighbour algorithm fails to get minimum cost. A further investigation is conducted to proof that nearest neighbour does not always find global minimum cost. Five random queue location is selected every trial to calculate for minimum cost.

Trials	Minimum Cost Found for 5 Queue Places(s)		
	Brute Force	Branch and Bound	Nearest Neighbour
1	7200	7200	7200
2	3000	3000	3600
3	5400	5400	6000
4	7200	7200	7200
5	3600	3600	3600

Table 5.4.2.2 Table of Minimum Cost Found for 5 Queue Places

CHAPTER 5: SYSTEM IMPLEMENTATION AND TESTING

As shown in Table 5.4.2.2, we can see that nearest neighbour fails to get global minimal cost at trials 2 and 3. Therefore we can conclude that while branch and bound algorithm is in fact finding for global minimum cost to generate an optimised route, nearest neighbour algorithm does not guarantee an optimised route.

CHAPTER 6: CONCLUSION

CHAPTER 6: CONCLUSION

6.1 Project Review

The project successfully realises the concept of utilising branch and bound algorithm to obtain optimised route with smallest ticket number in every queue location. This solves the problem of people need to manually plan route from places to places, and thus one of the objectives of this project that is proposing a queue management system integrating branch and bound algorithm has been achieved.

Besides, this project implemented a delay function which acts as a back up measure to not having enough travel time from one place to another. Instead of discarding a ticket, system allows user to swap ticket with next ticket holder providing that both users will not be disturbed in terms of queue route. This ensures a queue will not have empty space between two people queuing. Therefore, the second objective of this project, which is providing an effective method to deal with ticket whose holder does not have enough time, is met.

Next, after user has taken tickets, an optimised route will be generated on the map which also supports navigation function for user. User can simply follow the route or follow the navigation even without thinking which ticket is first to expire since the route is ordered in such a way that the expiring ticket is ordered first. Thus, we can conclude that the last objective of this project which is to generate an optimised route to be achieved.

6.2 Novelties and Contribution

It is always very time-consuming when it comes to planning a route to several location, not to mention that also having to queue in every destination after reaching a place. When planning a route, a person needs to gather all the travel time from all places to all places before making a real optimised route which is very time consuming and very error prone. One possible way to deal with this problem is to use a route planning app to generate an optimised route to every location, but the person might face an issue where there is a long queue in certain location and hence need to queue at the location before able to proceed to next location as planned using app.

One of the main novelties of this project is to combine route optimisation algorithm

CHAPTER 6: CONCLUSION

and queue management system together. With the implementation of the algorithm, user can now queue in many locations without needing to worry about route planning and issue of ticket expiring when arriving at a location. System will help user to manage queue such that user can reach destination one time while reducing queue time in every location. Therefore, this project will contribute to the idea of being able to combine both queueing process and route planning together so that user will not wasting time queueing in a place while still able to travel through every destination with minimum time consumed.

Besides, it is very important to maintain social distance due to the Covid pandemic. Using a mobile application of queue management system will help to reduce the number of people physically queueing in a place. For instance, people can queue in home using mobile application and depart to arrive to a location just in time to avoid excessive physical queue. Some applications of virtual queue which can utilise mobile queue management system would be queueing for Covid vaccination and queueing in barber shop. This project can contribute to help people practice social distancing by not queueing together under same physical space but rather queueing virtually. Many service providers such as public sector and hospitals can utilise this system to encourage virtual queueing which will lead to people practicing social distance.

Moreover, the swap ticket function may provide an insight on the implementation of exchanging ticket with 2 users while still ensuring the accuracy of the route, that is, both users will still be able to reach the destination on time without affecting other queue locations. Most of the time when a person has to arrive late on a location for whatever reason, the only choice the person can make is to discard current queue location and queue up again, which might cause the person minutes or hours of extra queueing time. The swap ticket function proposed in this project is important to avoid the mentioned case by exchanging the queue position of two users while also making sure they both are present when their turn is up.

6.3 Future Work

Future work can be done to further improve this system. First, the user interface of the application can be improved to enhance usability. Besides, the queue management

CHAPTER 6: CONCLUSION

model can be changed from taking ticket to taking a time slot that can monitor number of people taking a time slot. This improvement can extend the target companies which require queue but not limited to one service at a time. Some examples would be supermarket and grocery shop because such service providers need to restrict number of people remains inside the building due to Covid-19 pandemic. The algorithm used to calculate optimal route can also be improved to support multiple queue line in one place, which makes the system to become multi-queue-multi-location queue management system. The algorithm can also be improved by using travel distance as cost to calculate optimal route, which can allow user to change the preference cost as additional setting. Moreover, the delay ticket function can be improved to allow user to delay by a desire time rather than just delaying the ticket by one number. Furthermore, an iOS version of the system can also be developed to support iPhone users.

REFERENCES

- Android Developers. 2020. *Meet Android Studio*. [online] Available from:
<<https://developer.android.com/studio/intro>> [14 April 2020].
- Coloni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G. and Trubian, M.,
1996. Heuristics from nature for hard combinatorial optimization problems.
International Transactions in Operational Research, [ejournal] 3(1), pp.1–21.
<https://doi.org/10.1016/j.disopt.2016.01.005>
- Dana, P. H., 1997. Global Positioning System (GPS) Time Dissemination for Real-
Time Applications. *Real-Time Systems*, [ejournal] 12(1), pp.9–40.
<https://doi.org/10.1023/A:1007906014916>
- Dhayaalan Raju, 2020. *Travelling Salesman Problem via the Greedy Algorithm*.
[online] Available at:< <https://medium.com/ivymobility-developers/algorithm-a168afcd3611>> [Accessed 28 August 2021]
- Dictionary.cambridge.org. n.d. *QUEUE*. [online] Available from:
<<https://dictionary.cambridge.org/dictionary/english/queue>> [12 April 2020].
- Firebase. 2020. *Firebase Realtime Database*. [online] Available from:
<<https://firebase.google.com/docs/database>> [14 April 2020].
- GeeksforGeeks. 2021. *Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming)*. [online] Available at:
<<https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/>>
[Accessed 27 August 2021].
- Google Developers, 2021. *Distance Matrix API*. [online] Available at:
<<https://developers.google.com/maps/documentation/distance-matrix/overview>> [Accessed 29 August 2021].

Google Developers, 2021. *Getting directions through the Directions API*. [online]

Available at:

<<https://developers.google.com/maps/documentation/directions/get-directions>> [Accessed 29 August 2021].

Haslam, N., n.d. *Now We Know: Why We Stand In Queues*. [online] Available from:

<<https://pursuit.unimelb.edu.au/articles/now-we-know-why-we-stand-in-queues>> [13 April 2020].

JRNI, 2019. *What are virtual queues and how are they used?*. [online] Available

from: <<https://www.jrni.com/blog/what-are-virtual-queues-and-how-are-they-used>> [12 April 2020].

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C., 2016. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, [ejournal] 19, pp.79–102.

<https://doi.org/10.1016/j.disopt.2016.01.005>

Powerq.my. n.d. *Powerq*. [online] Available from: <<http://powerq.my/>> [13 April 2020].

QLess. n.d. *Digital Queuing System To Eliminate Lines*. [online] Available from:

<<https://www.qlless.com/>> [13 April 2020].

Queuebeesolution.com. n.d. *Queuebee Mobile Queue*. [online] Available from:

<<https://www.queuebeesolution.com/solution/mobileq.html>> [13 April 2020].

Rai, A. 2021. *Traveling Salesman Problem using Branch And Bound*. [online]

Available at: <<https://www.geeksforgeeks.org/traveling-salesman-problem-using-branch-and-bound-2/>> [Accessed 27 August 2021].

Rai, S., 2021. *What is Queue Management System*. [online]. Available at:

<<https://ace.aurionpro.com/what-is-queue-management-system/>> [Accessed 27 August 2021].

Tarjan, R., 1972. Depth-First Search and Linear Graph Algorithms. *SIAM Journal on Computing*, [ejournal] 1(2), pp146–160. <https://doi.org/10.1137/0201010>

Tšernov, K., 2020. *What Is Virtual Queuing? A Guide To Virtual Queues*. [online] Available from: <<https://www.qminder.com/virtual-queuing-systems/>> [12 April 2020].

Wavetec. n.d. *Mobile Queuing And Virtual SMS App*. [online] Available from: <<https://www.wavetec.com/solutions/queue-management/mobile-queuing/>> [13 April 2020].

Weru, L., 2021. *11 Animated Algorithms for the Traveling Salesman Problem*. [online]. Available at: <<https://stemlounge.com/animated-algorithms-for-the-traveling-salesman-problem/>> [Accessed 27 August 2021].

Weekly Report

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 2
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Reevaluated the whole program and made some coding changes in Branch and Bound algorithm to improve the algorithm.

Come out with an idea called “tolerance time”: a range of ticket number where a ticket can be without disturbing the order of the pre-planned route.

Added the idea of using QR code to inform system that the user has arrived the destination.

2. WORK TO BE DONE

Implement the idea of tolerance time to decide on the minimum and maximum number of ticket a person can have in a place, which is useful in designing the swap ticket function.

Show travel time of the planned route.

Display the destination order, i.e. which place user will go first.

Allow user to select fastest or shortest route.

So far the development is on the customer side, will develop the system for company side to stimulate the real queueing process.

3. PROBLEMS ENCOUNTERED

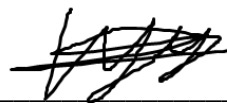
The development of the algorithm is slow because it needed a lot of time to test the accuracy. The main issue is the cost used to calculate the shortest path is not that straight forward since it consists of different dependencies. Although a seemingly working algorithm is already implemented, I still need some time to test the accuracy and further improve the algorithm to make it closer to “real-time” concept.

4. SELF EVALUATION OF THE PROGRESS

Expected to complete the system by week 5 of the semester. Will continue on software testing starting from week 6.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 4
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Implemented swap ticket function with proper check condition: swapping ticket will ensure the correctness of the route.

Recorded destination order.

Developed a prototype for company side to stimulate real queueing process.

2. WORK TO BE DONE

Show travel time of the planned route.

Display the destination order, i.e. which place user will go first.

Allow user to select fastest or shortest route.

Improve UI design.

3. PROBLEMS ENCOUNTERED

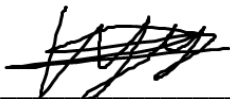
So far there is no serious problem encountered.

4. SELF EVALUATION OF THE PROGRESS

The progress of last fortnight is a bit slow which does not match my expected progress. Will try to finalize the development process in week 5 and start to do software testing.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 6
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Redesigned user interface to make the system more user friendly.
Display destination order.
Designed some test cases to test the accuracy of the algorithm.

2. WORK TO BE DONE

Show travel time of the planned route.
Continue on algorithm testing.
Write report.

3. PROBLEMS ENCOUNTERED

So far there is no serious problem encountered.

4. SELF EVALUATION OF THE PROGRESS

The progress is on track.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 8
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Updated Map UI: added navigation drawer.

Fixed some bugs.

Completed swap ticket function.

Updated route UI: highlights current route and grey out routes to other locations.

Demonstrated system done so far to supervisor to get feedback.

2. WORK TO BE DONE

Analysis on algorithms.

System testing.

Improve Google Maps interactions.

3. PROBLEMS ENCOUNTERED


After meeting with supervisor, I found that the user experience of my system is lacking. I will try my best to improve on this section.

4. SELF EVALUATION OF THE PROGRESS

The progress is slow. Need to put more efforts as the report submission date is around the corner.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 10
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed some bugs.

Tried to improve the Map UI.

Did some research on other algorithms which can solve TSP for literature review and analysis of algorithms, such as greedy algorithm and brute force method.

2. WORK TO BE DONE

Analysis on algorithms.

System testing.

Write report.

3. PROBLEMS ENCOUNTERED


Consumed too much time on improving user interface and user experience.

4. SELF EVALUATION OF THE PROGRESS

The progress is still slow. Will allocate more time to make sure the progress matches the scheduled progress.



Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y4S2	Study week no.: Week 12
Student Name & ID: LIU YU YAO 17ACB00765	
Supervisor: Ts Yeck Yin Ping	
Project Title: GPS Solution for Active Queue Management Using Android Platform	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

Fixed some bugs.

Added other algorithms into system such as greedy algorithm and brute force method for algorithm analysis, after that collected some data to check for correctness and performance.

Added navigation feature: Show routing navigation if user wants to.

Wrote chapter 1 and 2 of the report.

2. WORK TO BE DONE

Draw diagrams.

Explain Algorithm.

Put algorithm analysis data into report.

Improvement on UI and UX.

3. PROBLEMS ENCOUNTERED

Sometimes the coding of algorithm will produce an inaccurate result even the logic has been drafted. I need time to debug and fix the logic error which consume a lot of time.

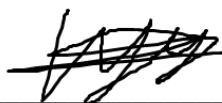
Also I found that I am weak in writing report, so need to spend more time on completing the report.

4. SELF EVALUATION OF THE PROGRESS

The progress is slow. Need to finish the report by next week Tuesday to make sure have enough time to submit the report. Improvement on UX and UI of application might need to take extra time to complete.



Supervisor's signature



Student's signature

Final Year Project Poster



Faculty of Information and
Communication Technology

GPS Solution for Active Queue Management Using Android Platform

INTRODUCTION

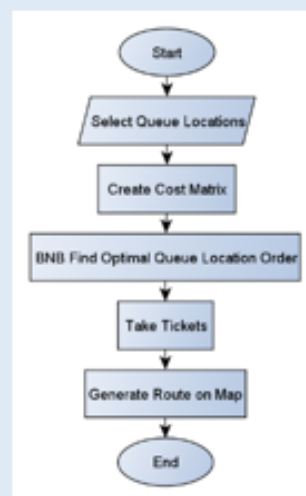
- Inefficient queue management system causes people to not able to queue in multiple locations at the same time.
- Even a person has taken multiple tickets in different locations, it is quite hard for a person to come out with an optimised route in a short time.
- Efficient delay method is lacking for those who cannot reach a location on time.

SYSTEM DESIGN



The system is developed into several flows. Using branch and bound algorithm, an optimal route together with calculated tickets can be generated. An optimal route will show on map based on tickets taken. Besides, the system supports the function of delaying a ticket through ticket swapping. Google Cloud Function will be triggered to send notification to users.

METHOD



RESULT & DISCUSSION

Select and Queue in Multiple Places

- System to find best ticket number in every queue location
- User to not worry about planning visit order

View Planned Route

- Generate optimal route based on tickets taken
- Navigate user to destination

Delay Function

- Extend a ticket number to have more queue time
- Swap with another user to not waste a ticket



By Liu Yu Yao

Project Supervisor: Ts Yeck Yin Ping

Plagiarism Check Result

GPS Solution for Active Queue Management Using Android Platform

ORIGINALITY REPORT

4%

SIMILARITY INDEX

2%

INTERNET SOURCES

1%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Informatics Education Limited Student Paper	<1 %
2	Submitted to Universiti Tunku Abdul Rahman Student Paper	<1 %
3	eprints.utar.edu.my Internet Source	<1 %
4	David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, Edward C. Sewell. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning", Discrete Optimization, 2016 Publication	<1 %
5	Submitted to Kuala Lumpur Infrastructure University College Student Paper	<1 %
6	Submitted to Ain Shams University Student Paper	<1 %
7	Submitted to Universidad Carlos III de Madrid Student Paper	<1 %

8	www.ukessays.com Internet Source	<1 %
9	Prashant K. Srivastava, Prachi Singh, Varsha Pandey, Manika Gupta. "Development of android application for visualisation of soil water demand", Elsevier BV, 2021 Publication	<1 %
10	Submitted to University of Greenwich Student Paper	<1 %
11	hdl.handle.net Internet Source	<1 %
12	Arsalan Khan, Farzana Bibi, Muhammad Dilshad, Salman Ahmed, Zia Ullah, Haider Ali. "Accident Detection and Smart Rescue System using Android Smartphone with Real-Time Location Tracking", International Journal of Advanced Computer Science and Applications, 2018 Publication	<1 %
13	Submitted to Asian Institute of Technology Student Paper	<1 %
14	Submitted to K12 Incorporated Student Paper	<1 %
15	chro.kungfuparma.it Internet Source	<1 %
www.qminder.com		

16	Internet Source	<1 %
17	Submitted to Emirates Aviation College, Aerospace & Academic Studies Student Paper	<1 %
18	Submitted to The British Schools Student Paper	<1 %
19	maxwellsci.com Internet Source	<1 %
20	www.lyncconf.com Internet Source	<1 %
21	archive.org Internet Source	<1 %
22	Chen, Chi-Tsong. "Analog and Digital Control System Design", Oxford University Press Publication	<1 %
23	analystcave.com Internet Source	<1 %
24	www.grin.com Internet Source	<1 %

Exclude quotes On
Exclude bibliography On

Exclude matches < 8 words

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1 of 1




FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Liu Yu Yao
ID Number(s)	17ACB00765
Programme / Course	BACHELOR OF COMPUTER SCIENCE (HONOURS)
Title of Final Year Project	GPS Solution for Active Queue Management Using Android Platform

Similarity	Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)
Overall similarity index: 4% Similarity by source Internet Sources: 2% Publications: 1% Student Papers: 2%	
Number of individual sources listed of more than 3% similarity: <u>0</u>	
Parameters of originality required and limits approved by UTAR are as follows: (i) Overall similarity index is 20% and below, and (ii) Matching of individual sources listed must be less than 3% each, and (iii) Matching texts in continuous block must not exceed 8 words <i>Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.</i>	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.



Signature of Supervisor
 Name: Yeck Yin Ping
 Date: 2/9/2021

Signature of Co-Supervisor
 Name: _____
 Date: _____



UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	17ACB00765
Student Name	Liu Yu Yao
Supervisor Name	Ts Yeck Yin Ping

TICK (√)	DOCUMENT ITEMS
	Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
N/A	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
N/A	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)

*Include this form (checklist) in the thesis (Bind together as the last page)

<p>I, the author, have checked and confirmed all the items listed in the table are included in my report.</p> <div style="text-align: center;"> <p>_____ (Signature of Student) Date: 2/9/2021</p> </div>	<p>Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction.</p> <div style="text-align: center;"> <p>_____ (Signature of Supervisor) Date: 2/9/2021</p> </div>
---	---