Text

Description automatically generated

Table of Contents

[1. Introduction 6](#_Toc118839735)

[1.1 Data structures and Classes 6](#_Toc118839736)

[Data structures 6](#_Toc118839737)

[Push 6](#_Toc118839738)

[Append 7](#_Toc118839739)

[Left\_to\_right 8](#_Toc118839740)

[Right\_to\_left 9](#_Toc118839741)

[Admin classes 10](#_Toc118839742)

[class Admin 10](#_Toc118839743)

[tokenize 10](#_Toc118839744)

[Edit\_station\_everything 11](#_Toc118839745)

[Transaction\* swap 11](#_Toc118839746)

[bubbleSort\_name 12](#_Toc118839747)

[bubbleSort\_id 13](#_Toc118839748)

[Print\_list 14](#_Toc118839749)

[delete\_transaction 15](#_Toc118839750)

[Customer classes 15](#_Toc118839751)

[Class Customer 15](#_Toc118839752)

[append\_transaction\_customer 16](#_Toc118839753)

[append\_transaction\_list 16](#_Toc118839754)

[display\_transactions 17](#_Toc118839755)

[display\_receipt 18](#_Toc118839756)

[General classes 18](#_Toc118839757)

[get\_transaction\_id 18](#_Toc118839758)

[Struct Transaction 19](#_Toc118839759)

[class Station 20](#_Toc118839760)

[2. Implementation 20](#_Toc118839761)

[2.1 Algorithms 20](#_Toc118839762)

[Admin algorithms 20](#_Toc118839763)

[Create\_customers 20](#_Toc118839764)

[Delete\_stations 21](#_Toc118839765)

[Edit\_station\_everything\_page 22](#_Toc118839766)

[Edit\_station\_specific 23](#_Toc118839767)

[Add\_station 24](#_Toc118839768)

[Admin\_home\_page 26](#_Toc118839769)

[Admin\_login\_page 28](#_Toc118839770)

[Customer algorithms 29](#_Toc118839771)

[customer\_home\_page 29](#_Toc118839772)

[Customer\_login\_page 31](#_Toc118839773)

[General algorithm 32](#_Toc118839774)

[Get\_time 32](#_Toc118839775)

[Check\_service\_hours 33](#_Toc118839776)

[Check\_ticket\_time 34](#_Toc118839777)

[Print\_list\_of\_station 35](#_Toc118839778)

[confirmAndCreate\_transaction 36](#_Toc118839779)

[Auto list\_of\_station 36](#_Toc118839780)

[3. Result 37](#_Toc118839781)

[3.1 System Input / Output Screenshot 37](#_Toc118839782)

[Main page 37](#_Toc118839783)

[Customer login page 37](#_Toc118839784)

[Customer home page 38](#_Toc118839785)

[Purchase ticket for now(left to right) 38](#_Toc118839786)

[Receipt for now ticket (left to right) 39](#_Toc118839787)

[Purchase ticket for now (right to left) 40](#_Toc118839788)

[Receipt for now ticket (right to left) 41](#_Toc118839789)

[Purchase for later ticket 41](#_Toc118839790)

[Receipt for later ticket 42](#_Toc118839791)

[View transaction history 42](#_Toc118839792)

[Login customer 2 43](#_Toc118839793)

[Station selection safety feature 43](#_Toc118839794)

[Station time safety feature 44](#_Toc118839795)

[Admin login page 44](#_Toc118839796)

[Admin home page 45](#_Toc118839797)

[Admin view all customer transaction history 45](#_Toc118839798)

[Admin view all customer transaction (ID sorted) 46](#_Toc118839799)

[Admin view all customer transaction (name sorted) 46](#_Toc118839800)

[Admin edit/add station page 47](#_Toc118839801)

[Admin edit a whole station 47](#_Toc118839802)

[Admin edit specific information 48](#_Toc118839803)

[Add station 48](#_Toc118839804)

[Edit station input check 49](#_Toc118839805)

[Push new station 49](#_Toc118839806)

[Append new station 50](#_Toc118839807)

[4. Conclusion, future works, and reflection 51](#_Toc118839808)

[5. Appendix: Workload Matrix Table with signature 52](#_Toc118839809)

# Introduction

We as a group are required to make a system that is to be used for a train station system where people are allowed to purchase tickets to travel from one station to another. We are going to be using a data structure to store the information regarding the train station so that customers can see the details of the desired station and make the necessary purchases. The system that is going to be made will be using C++ language with structures such as a linked-list. We will also be showing and demonstrating source codes, algorithms, and the output of the code.

We as junior software developers from FIRST IT Ltd has been assigned to help the city council of Subang Jaya to make this system. We are going to include features for roles such as passengers and the Admin. Passengers are allowed to choose the train stations from Titiwangsa to Chan Sow Lin and vice versa, searching the selected station details, purchasing a subway ticket, viewing their transaction history, and deleting their purchase transaction. As for admins, they have special privileges compared to passengers such as being able to add subway station details. This allows the admin to update the station details by editing it. They also can view and sort the purchase transactions by searching customer ticket purchase information and being able to edit the information for the customer like the travel information that is printed on the ticket. Finally, admins are also allowed to delete the customers ticket purchase information.

# Data structures and Classes

## Data structures

### Push

Text

Description automatically generated

Figure 1 Push

This part of the code will push a new node in front of the linked list. The new\_node is declared as a pointer by station which contains station\_id, station\_name, travel\_time, travel\_distance, and fee. The new\_node->next will be the \*head\_ref while the new\_node->prev will be null. So if \*head\_ref is not null, then once the new\_node is in front, it will become head.

### Append

Text

Description automatically generated

Figure 2 Append

The figure above shows the code for appending a new station. First a new node is created and the last node is created to loop through the station linked list to get to the end. Once last reaches the end of the linked list last->next will become the new node. If the station linked list is empty, head\_ref is null, set head\_ref equal to new node.

### Left\_to\_right

Text

Description automatically generated

Figure 3 Left\_to\_right

This code is the first part of the double linked list. The station will be defined as the head node. If node is not empty, then the process continues to the next if statement. Else it will go to the time\_check function.In the if statement, if the station\_id node is equal to departure\_station, the departure\_info will be set to the node->station\_name. It will also output the station name in the interface of the app. In the do function, the process will repeat itself until it is broken with a break function. The total fee, distance, and time will also be set to their respective nodes like fee, travel\_distance, and travel\_time. In the next if statement, if the next station\_id node is equals to the arrival\_station, the arrival\_info will be set to the next station\_name node along with displaying the next station\_name in the app interface. If not, then it will display the next station\_name with the travel\_time. If the station\_id is not the same with the arrival\_station, it stops the loop. The time\_check uses the check\_ticket\_time function which will include the total\_time, now, and time\_for\_later. If time\_check is true, then it will output the total\_fee, total\_distance, and total\_time. After displaying the details, it will use the confirmAndCreate\_transaction function that will include the customer, total\_fee, total\_distance, total\_time, departure\_info, and arrival\_info before returning.

### Right\_to\_left

Text

Description automatically generated

Figure 4 Right\_to\_left

This part of the code is identical to the left\_to\_right as it is meant for the opposite direction of the double linked list. Instead of next stations, it will be previous stations.

## Admin classes

### class Admin

Text

Description automatically generated

Figure 5 Admin class

This is the admin class that only contains the username and password. There would only be 1 admin on this system so we just need a way to prove that the user is the admin.

### tokenize

Graphical user interface, text

Description automatically generated

Figure 6 Tokenize

This function allows for multiple inputs used for editing the station details.

### Edit\_station\_everything

Text

Description automatically generated

Figure 7 Edit\_station\_everything

The figure above shows the code for editing everything in the station. After the admin has input the format of the new station detail, it will print out the information before and after for the station. Then if the admin input y then the system will replace the old station detail to the new station detail that the admin just input. If the admin input anything else it would end the function.

### Transaction\* swap

Text

Description automatically generated

Figure 8 Edit\_station\_everything

This is the bubble sort swap function where the bubble sort function happens.

### bubbleSort\_name

Text

Description automatically generated

Figure 9 bubbleSort\_name

The code above is the bubble sort for admin class where it would sort out the transaction by the transaction name.

### bubbleSort\_id

Text

Description automatically generated

Figure 10 bubbleSort\_id

This is the function under admin class for the transaction bubble sort. Where this function will sort the transaction according to the transaction ID.

### Print\_list

Text

Description automatically generated

Figure 11 print\_list

This is the print\_list function under admin class.

### delete\_transaction

A screenshot of a computer

Description automatically generated

Figure 12 delete\_transaction

This function is to delete transactions. Firstly it asks the user for the transaction ID to be deleted. If temp->next is not null, the statement continues with a if statement. If temp->transaction\_id is the same as the id inputted by the user and temp->next->next is not null. This outputs the transaction id and customer name that was deleted.

## Customer classes

### Class Customer

Text

Description automatically generated

Figure 13 Class Customer

Customer class contain all detail of all customer.​ Customer class include the username, password, full name, IC number of the customer and head node for station and transaction.​

The head node for station and transaction is used for the link list so it can save the address of the station head node inside the customer object, then later we can call it within the customer object.

### append\_transaction\_customer

Text

Description automatically generated

Figure 14 append\_transaction\_customer

This is the transaction append for customer’s own transaction list head.

### append\_transaction\_list

Text

Description automatically generated

Figure 15 append\_transaction\_list

This function works similarly with the append function where it adds the new transaction at the back of the list. This list will be the global list where transactions will be in.

### display\_transactions

Text

Description automatically generated

Figure 16 display\_transactions

This figure displays the individual transaction for the customer. If there is data in the node it would bring out the title along with the data of the transaction. If there is no data in the node it would tell the user that there is no transaction.

### display\_receipt

Text

Description automatically generated

Figure 17 display\_receipt

The figure above displays the details of the ticket after the customer purchase.

## General classes

### get\_transaction\_id

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Figure 18 get\_transaction\_id

The figure above shows the global variable of the station head pointer, customer array pointer and the transaction head pointer.

### Struct Transaction

Chart

Description automatically generated with low confidence

Figure 19 Struct Transaction

For the transaction struct contain all detail for the payment when customer pays for the ticket.​

The transaction struct includes Total Fee, Total distance, Total Time, Customer Name, Bought Time for the ticket, Departure Station, Arrival Station and Transaction ID. ​We made these classes by following the variable of a receipt where the departure and arrival station was the station that was selected by the customer.

### class Station

Text

Description automatically generated

Figure 20 class Station

Stations class contain all details for all stations.​ In this struct it includes station ID, Station Name, Travel Time, Travel Distance and Fee.​ Travel Time, Travel Distance and Fee which are used to calculate the payment for each station.

# 2. Implementation

# 2.1 Algorithms

## Admin algorithms

### Create\_customers



Graphical user interface, text

Description automatically generated

Figure 21 Create\_customers

The figure above shows the data of all the customers. The customers is the pointer for the string Customer. The pointer would have 3 variables and in each variable it will create a new customer containing their details.

### Delete\_stations

Text

Description automatically generated

Figure 22 Delete\_stations

The figure above shows the code for the admin to delete the station. The function will first get the pointer of the current station and set a pointer to null which is now not pointing to any data.

### Edit\_station\_everything\_page

Text

Description automatically generated

Text

Description automatically generated

Figure 23 Edit\_station\_everything\_page

### Edit\_station\_specific

Text

Description automatically generated

Figure 24 Edit\_station\_specific

In this code, it will allow the station detail to be edited. If 1 was pressed, it will go into the change station id where the input will be set as the node->station\_id. 2 changes the node->station\_name with the string\_input entered by the user. The case 3, 4, and 5 has a similar process.

user is prompted to enter "y" to indicate whether or not they want the function to continue to re-enter the username and password.

### Add\_station

Text

Description automatically generated

Text

Description automatically generated

Figure 25 Add\_station

This code allows the admin to add stations in front or at the back of the train station list. It will ask the user to input the station\_id, station\_name, travel\_time, travel\_distance, and fee. Once all the details has been entered, it will display the changes and will confirm the changes with the user with y or n. If y is entered, it will ask to append or push the new station. If not it will return to the main page.

### Admin\_home\_page

Text

Description automatically generated

Text

Description automatically generated

Figure 26 Admin\_home\_page

The figure above shows the home page for admin. First it would print out some tile for the admin to pick. Then a switch function is created to let the admin the function that wants to be executed. There would be functions for example displaying all translations for customers, sorting with ID, sorting with name, editing the station and editing and deleting the translation.

### Admin\_login\_page

Text

Description automatically generated

Figure 27 Admin\_login\_page

The figure above shows the function of checking the password for the admin. To begin storing information from the user, a string consisting of a username, password, and input is constructed. The loop function is then used to check and compare the admin username and password. If the username is true it will bring the user to the admin home page. When a mismatch occurs, the n it will output login failed. The user can then decide to continue by inputting y, whereas n will return the function.

## Customer algorithms

### customer\_home\_page

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Figure 28 customer\_home\_page

The figure above shows the code for the customer home page. The code will first print the tile and detail of the function to let the customer select, then the customer can pick the function that they want to execute, for example purchasing a ticket, viewing their transaction history and exit to the main menu.

### Customer\_login\_page

Text

Description automatically generated

Figure 29 Customer\_login\_page

The figure above shows the code for the customer login page. Firstly, a string of username,password and input is created for saving user input data.Then a string match and num\_customers is created to check and give validation for the loop function. Then a for loop function is created to check the password if it is true it would go to the customer user page. If the match is false it would ask the user to input “y” to continue or end the function.

## General algorithm

### Get\_time

Graphical user interface, text, website

Description automatically generated

Figure 30 Get\_time

The figure above is a function to get the current time which will be used in other functions for checking purposes. This uses the chrono library which deals with time and date. The system\_clock gets time of the computer. Ctime will convert the time to the calendar local time and return the time.

### Check\_service\_hours

Text

Description automatically generated

Figure 31 Check\_service\_hours

This is the code for checking where the current time is on working hours. The time uses a 24 hour format and if the time passes 6 it won’t allow the user to book the ticket. If it is in the working hour then the book ticket function will work.

### Check\_ticket\_time

Text

Description automatically generated with medium confidence

Text

Description automatically generated

The code above it to check customer ticket booking time. When the customer buys a ticket it would first check the current time which is get\_time(). Then it would turn the time into a character array. Then it would add the current time with the estimated time from the station that customer picks. If the total is exit the work time the function will end and if it is during the work time the ticket would be purchased.

Text

Description automatically generated

Figure 32 Check\_ticket\_time

### Print\_list\_of\_station

A screenshot of a computer

Description automatically generated

Figure 33 Print\_list\_of\_station

Figure above shows the list of all stations. First an int named “count” is created and its variable is set to 1. Then it would print the count number with the station name. If the node has a next station it would print a double arrow and add the count by 1 and repeat the function. While if the node has nothing to go next it would print a long line to end the code.

### confirmAndCreate\_transaction

Text

Description automatically generated

Figure 34 confirmAndCreate\_transaction

In this code, it will ask the user if the user wants to purchase the ticket if the input equals to y, then customer object access both append\_transaction and display\_receipt functions.

### Auto list\_of\_station

Text

Description automatically generated

Figure 35 Auto list\_of\_station

Station\* is defined as the head which will be set as null. Then the push and append is for the stations to be listed in the head with details such as the id, name, distance, time and price. Once all the stations have been pushed and appended, it will return.

# 3. Result

## 3.1 System Input / Output Screenshot

### Main page

Text

Description automatically generated

Figure 36 Main Page

This page is the main menu and displays the account types to use.

### Customer login page

Text

Description automatically generated

Figure 37 Customer login page

This is the login page for customers. The customer will need to input their username and password in order to login.

Customer home pageText

Description automatically generated

Figure 38 Customer home page

This is the customer home page which shows the account logged in named carmen. It allows the user to purchase the ticket or view their transaction history.

### Purchase ticket for now(left to right)

Text

Description automatically generated

Figure 39 Purchase ticket for now(left to right)

When purchasing a ticket, it will give the user an option for now or later. This page for buying a ticket now which will show the stations, current time, total distance, and total time. The purchase ticket page uses the double linked list to determine if the station id is ascending or descending order. In this case, it is an ascending order which will be using the left\_to\_right function.

### Receipt for now ticket (left to right)

Text

Description automatically generated

Figure 40 Receipt for now ticket (left to right)

This is the receipt which will show the first station as the departure and the last station as the arrival.

### Purchase ticket for now (right to left)

Text

Description automatically generated

Figure 41 Purchase ticket for now (right to left)

Similar to the left to right ticket purchase but the stations are entered in descending order.

### Receipt for now ticket (right to left)

Text

Description automatically generated

Figure 42 Receipt for now ticket (right to left)

Similar to the left to right where departure is the last while the arrival is the first station.

### Purchase for later ticket

Text

Description automatically generated

Figure 43 Purchase for later ticket

This is the purchase ticket for the later page where it will ask for a time before selecting the current and destination station.

### Receipt for later ticket

Text

Description automatically generated

Figure 44 Receipt for later ticket

This is the receipt for the later ticket purchase. Similar to the other receipts but for the later ticket.

### View transaction history

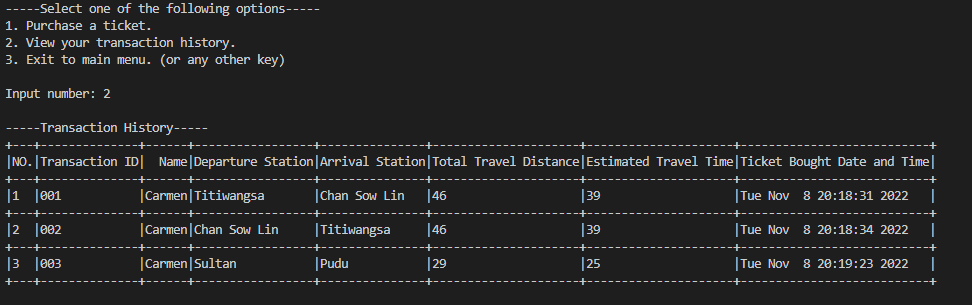


Figure 45 View transaction history

This is the transaction history for customers where they can view all their transaction details and ticket details in table form.

### Login customer 2

Text

Description automatically generated

Figure 46 Login customer 2

This an additional user that is in the list of customer accounts.

### Station selection safety feature

Text

Description automatically generated

Figure 47 Station selection safety feature

This is the station selection safety feature to make sure that the user does not set the same station for the current and destination station.

### Station time safety feature

Text

Description automatically generated

Figure 48 Station time safety feature

This is the time safety feature where it will make sure the user uses the correct time format.

### Admin login page

Text

Description automatically generated

Figure 49 Admin login page

This is the login page for the admin page.

### Admin home page

Text

Description automatically generated

Figure 50 Admin home page

This is the home page for the admin. It will display customer transactions, id sorted, name sorted, edit/add stations, and edit and delete transaction.

### Admin view all customer transaction history

Text

Description automatically generated

Figure 51 Admin view all customer transaction history

This page displays all the transactions of every customer. It includes the id, name, departure station, arrival station, total travel distance, estimated arrival time, and ticket bought date and time.

### Admin view all customer transaction (ID sorted)

Text

Description automatically generated with medium confidence

Figure 52 Admin view all customer transaction (ID sorted)

This page shows all the transactions sorted by the customer id.

### Admin view all customer transaction (name sorted)

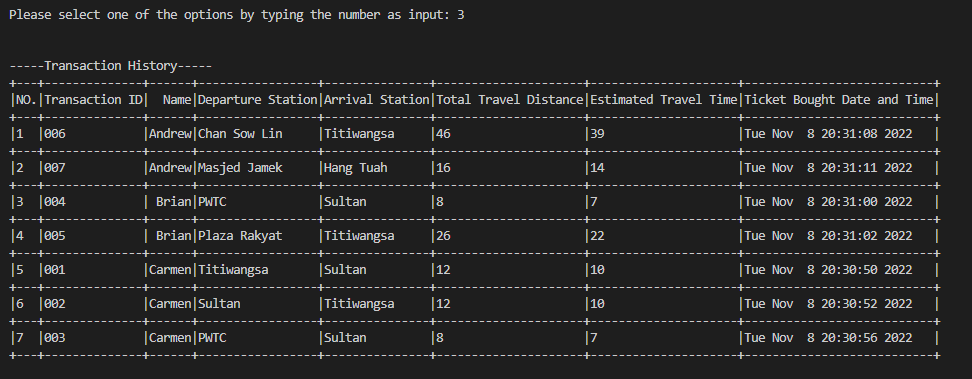


Figure 53 Admin view all customer transaction (name sorted)

This page shows all the transactions sorted by the customer name.

### Admin edit/add station page

Text

Description automatically generated

Figure 54 Admin edit/add station page

This page allows the admin to edit the station either by the layout of the stations or the details of the stations. It also allows the admin to add a new station to the station list.

### Admin edit a whole station

Text

Description automatically generated

Figure 55 Admin edit a whole station

For edit a whole station, it will require the admin to select a station id to edit.

Text

Description automatically generated

When editing station details it will require the admin to enter it in the correct format which is station id, name, travel time, distance, and fee.

### Admin edit specific information

Text

Description automatically generated

Figure 56 Admin edit specific information

Similar to the edit while station, it allows admin to edit but can select specifically which detail to edit. The admin will have to choose which detail to edit from id, name, time, distance, and fee.

Text

Description automatically generated

It will require the admin to replace the station name in this case. Once done, it will be shown in the station list where station 4 changes to mars.

### Add station

Text

Description automatically generated

Figure 57 Add station

In this function it will ask to push or append the new station. New stations can be placed behind or in front of the list.

### Edit station input check

Text

Description automatically generated

Figure 58 Edit station input check

This checks if the format is correct. If not, it will return to the admin main menu

### Push new station

Text

Description automatically generated

Figure 59 Push new station

This is the push function where it will require the admin to fill in the details one at a time. Once done, it will confirm with the admin before adding the station.

Text

Description automatically generated

New station will be added with an id of 0 and is named damansara when using the push function to add in front of the list.

### Append new station

Text

Description automatically generated

Figure 60 Append new station

Similar to the push function except for adding a new station at the end of the list.

# 4. Conclusion, future works, and reflection

This is the system that we have created as a team which will be presented to FIRST IT Ltd. This system allows the passenger to purchase tickets and make the necessary changes for their ticket and the admin is allowed to make changes to details of the customer ticket and the details of the train stations. With this we hope that we can make a more advanced system that has many more enhance features that could improve the overall quality of the system. Although this was out first major assignment to make this system, we managed to learn simple data structure and implementing linked list into our system that could help us with our projects in the future.

# 5. Appendix: Workload Matrix Table with signature

|  |  |  |  |
| --- | --- | --- | --- |
| Name | TP Number | Workload | Signature |
| Ng Hann Pinn | TP058002 | 25% |  |
| Wang Ju Lian | TP053175 | 25% |  |
| Nicklaus San Yew Joe | TP058303 | 25% |  |
| Lee Hong Yu | TP063138 | 25% |  |