



INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT108-3-1-PYP

PYTHON PROGRAMMING

APU/APD1F2206-

PE/TE/EEE/MMT/SE/CGD/CE/IT/CS/CS(DF)/CS(IS)/CS(CYB)

HAND OUT DATE : 18th July 2022
HAND IN DATE : 20th September 2022
WEIGHTAGE : 100%
NAME : LEE ZHI XUAN
TP_NUMBER : TP065525
INTAKE : APD1F2206ME

INSTRUCTIONS TO CANDIDATES:

1. Submit your assignment online in MS Teams unless advised otherwise
2. Late submission will be awarded zero (0) unless Extenuating Circumstances (EC) are upheld
3. Cases of plagiarism will be penalized
4. You must obtain at least 50% in each component to pass this module

1.0 Table of Contents

1.0 Table of Contents	2
2.0 Introduction.....	5
3.0 Assumptions.....	5
4.0 Design of the Program	7
4.1 Pseudocode	7
Title Page	7
Role Selection Page	7
Admin Login Page	8
Customer Role Page.....	9
Customer Journey KL-SG Page.....	10
KL-SG Timetable.....	11
KL-Thai Timetable Route A.....	14
KL-Thai Timetable Route B	17
Admin Actions Page	19
Admin Edit Timings Page.....	22
Edit Timings Function	23
Admin Add Station Page	25
Add Station Function Page	26
Remove Station Page	28
Admin Remove Station Function Page.....	29
Admin Edit Trip Starting Time Page	31
4.2 Flow Charts	34
Title Page	34

Admin Login Page	35
Customer Role Page.....	36
Customer KL-SG Page	37
KL-SG Timetable Page.....	38
KL-Thai Timetable Route A Page	39
KL-Thai Timetable Route B Page	40
Customer KL-Thai Route A Page.....	41
Customer KL-Thai Route B Page	42
Admin Actions Page	43
Admin Edit Timings Page.....	44
Edit Timings Function Page.....	45
Admin Add Stations Page.....	46
Add Station Function Page	47
Admin Remove Station Page	48
Remove Station Function Page.....	49
Edit Trip Starting Time Page	50
5.0 Program Source Code and Explanation	51
File Handling	51
If-Else Statement.....	53
While Loop	54
For Loop.....	55
Functions.....	55
Lists.....	57
6.0 Sample of Input/Output and Explanation	59
Starting of the Program.....	59

Customer Role	59
Timetables.....	60
Admin Login.....	62
Admin Action.....	63
Edit Timings.....	64
Add Station	67
Remove Station.....	69
Edit Trip Starting Time.....	71
7.0 Conclusion	73
8.0 References.....	74

2.0 Introduction

The popularity of public transport has been steadily increasing globally over the past couple of decades. There are many factors that contribute to this change such as the increase in awareness of climate change and our individual impacts can have on the environment. In addition, the Malaysian public transport system has also improved greatly, thus making it an attractive transport option for its citizens. The primary form of public transport is trains which come in many forms such as light rail transit (LRT), medium rail transit (MRT), and standard rail. With the increasing popularity of these transport, modern systems are also required to handle and manage the time schedules of these systems.

The objective of this assignment is to create a train scheduling system using the programming language Python. The company that this simulated project is for is the Malayan Railway Limited, more commonly known as KTM. The company offers standard rail train services between Kuala Lumpur, Singapore, Thailand, as well as many other places via multiple different routes. This program allows the customers to view timetables of trains for each of the routes, including their departure, arrival, and travel time.

In addition, this program also includes admin functionality such as editing the stopover and turnaround time of trains. This function may come in useful to reflect changes such as the introduction of new locomotives allowing higher speeds, thus reducing the travel duration. Moreover, the admin can also add or remove stations from the routes as they wish to reflect changes in the real world such as the opening of a new station or the temporary closing of a station. A reset functionality is also included to for the system to be able to restore its default timetables in the event that the admin made a mistake in the settings of the timetables.

3.0 Assumptions

From the program requirements, there are only 2 routes, both of which have Thailand and Singapore as their terminus stations. However, it was noticed that Route A and B both share the same route from Kuala Lumpur to Singapore. As a result, it is decided that this route shall be split

from the other two, creating route C and the previous 2 routes will be shortened to terminus at Kuala Lumpur station.

In this program, it is assumed that KTM offers train journeys in three different routes. Route A, which is from Kuala Lumpur to Thailand via Butterworth, Route B, which is from Kuala Lumpur to Thailand via Kelantan, and Route C, which is from Kuala Lumpur to Singapore. All these train routes are bidirectional, meaning that the trains travel to and then back from the terminus station after a turnaround.

As per the requirements, the company offers 2 daily bidirectional trains on Route A from KL to Thailand via Butterworth, and 1 on Route B from KL to Thailand via Kelantan. The requirements allow the freedom to choose however many trips between Kuala Lumpur and Singapore, thus it is decided that there will be 5 daily round trips on Route C from KL to Singapore. The number of trains are determined based on the higher demand of the KL-SG route due to the strong economic and cultural ties between the two countries.

Moreover, it is assumed that Kuala Lumpur station is the hub station and is where all trains initially depart from. This station was chosen as KTM is a Malaysian Company and Kuala Lumpur is the capital of the country. With this configuration, passengers can travel from Singapore to Thailand with a connection at Kuala Lumpur station.

The default password for the admin login menu is '1111'.

4.0 Design of the Program

4.1 Pseudocode

Start

Import compiler OS system

Define colour class

 Define 10 colour codes into classes

Open all station and timing files for read

Write file contents into lists

Close all files

End

Title Page

Start

Define title page:

 Print header bar

 Print company name and title

 Print footer bar

End

Role Selection Page

Start

Define select role page:

 Title page function

 Print header

 Print instructions

```
role_selected = 0
Do while role_selected is 0, then
    Print legend
    If user role is 1, then
        Print selection confirmation
        role_selected = 1
        customer role page function
    Else if user role is 2, then
        Print selection confirmation
        role_selected = 1
        Admin login page function
    Else if user role is e, then
        Print exit text
    Else, then
        Print error text
    Endif
Enddo
End
```

Admin Login Page

Start

Define admin login page:

```
passed = 0
fail_count = 0
Print header
Do while passed is 0, then
    Print legend
    Input password
    If fail count more >= 4, then
```



```
        Print max login message
        passed = 1
        Select role page function
    If password is correct, then
        passed = 1
        Print login confirmation
        Admin actions page function
    Else if password is b, then
        passed = 1
        Print back to page text
        Select role page function
    Else, then
        fail_count = fail_count + 1
        attempts_left = 5 - fail_count
        Print error message with remaining attempts
    Endif
Enddo
End
```

Customer Role Page

Start

Define customer role page:

Print header

Print instruction text

customer_journey_selection_complete = 0

Do while customer_journey_selection_complete is 0, then

Print legend

Input customer selection

If customer selection is b, then

```
        customer_journey_selection_complete = 1
        Print selection confirmation
        Select role page function
    Else if customer selection is e, then
        customer_journey_selection_complete = 1
        Print exit text
    Else if customer selection is 1, then
        customer_journey_selection_complete = 1
        Print selection confirmation
        KL-SG page function
    Else if customer selection is 2, then
        customer_journey_selection_complete = 1
        Print selection confirmation text
        KL-Thai Route A page function
    Else if customer selection is 3, then
        customer_journey_selection_complete = 1
        Print selection confirmation text
        KL-Thai Route B page function
    Else, then
        Print error text
    Endif
Enddo
End
```

Customer Journey KL-SG Page

Start

Define customer journey KL-SG page:

```
Title page function
Print header
```

```
Print instruction text
KL-SG timetable 1 function
KL-SG timetable 2 function
KL-SG timetable 3 function
KL-SG timetable 4 function
KL-SG timetable 5 function
customer_journey_kl_sg_complete = 0
Do while customer selection complete is 0, then
    Print legend
    If customer selection is b, then
        Print selection confirmation
        customer_journey_kl_sg_complete = 1
        Customer role page function
    Else if customer selection is e, then
        customer_journey_kl_sg_complete = 1
        Print exit text
    Else if customer selection is 1, then
        customer_journey_kl_sg_complete = 1
        Print selection confirmation
        KL-Thai page function
    Else, then
        Print error text
    Endif
Enddo
End
```

KL-SG Timetable

Define KL-SG timetable page:

```
Print header
```

```
c_hour = start_time_list
c_minute = start_time_list
Create c_departure list
Create c_arrival list
count = 0
For loop for c_timings_content_list - 1, then
    c_minute = c_minute + c_timings_content_list[i]
    count = count + 1
Enddo
Do while c_minute >= 60, then
    c_minute = c_minute - 60
    c_hour = c_hour + 1
Enddo
Do while c_hour >= 24, then
    c_hour = c_hour - 24
    c_time = c_hour + c_minute fill to 2 integers each side
Enddo
If count is odd number, then
    Append c_time in c_departure
Else, then
    Append c_time in c_arrival
Endif
Reverse timings content list
For loop for c_timings_content_list - 1, then
    c_minute = c_minute + c_timings_content_list[i]
    count = count + 1
    Do while c_minute >= 60, then
        c_minute = c_minute - 60
        c_hour = c_hour + 1
    Enddo
```

```
Do while c_hour >= 24, then
    c_hour = c_hour - 24
    c_time = c_hour + c_minute fill to 2 integers each side
Enddo
If count is odd number, then
    Append c_time in c_departure
Else, then
    Append c_time in c_arrival
Endif
Reverse timings content list
Enddo
count = 1
Create c_duration list
For loop for c_timings_content_list – 1, then
    If count is odd number, then
        Append timings content list in c_duration
        count = count + 1
    Endif
Enddo
Reverse timings content list
For loop for c_timings_content_list – 1, then
    If count is odd number, then
        Append timings content list in c_duration
        count = count + 1
    Endif
Enddo
Reverse timings content list
count = 1
Print timetable header
For loop for c_station_content_list – 1, then
```

```
        Print station content list to station content list, c_departure, c_arrival, and
        c_duration
        count = count + 1
        Reverse station content list
        count = 1
    Enddo
    For loop for c_station_content_list - 1
        Print station content list to station content list, c_departure, c_arrival, and
        c_duration
        count = count + 1
        Reverse station content list
    Enddo
End
```

KL-Thai Timetable Route A

Define KL-Thai timetable A page:

```
    Print header
    a_hour = start_time_list
    a_minute = start_time_list
    Create a_departure list
    Create a_arrival list
    count = 0
    For loop for a_timings_content_list - 1, then
        a_minute = a_minute + a_timings_content_list[i]
        count = count + 1
    Enddo
    Do while a_minute >= 60, then
        a_minute = a_minute - 60
        a_hour = a_hour + 1
    Enddo
```

```
Do while a_hour >= 24, then
    a_hour = a_hour - 24
Enddo
a_time = a_hour + a_minute fill to 2 integers each side
If count is odd number, then
    Append a_time in a_departure
Else, then
    Append a_time in a_arrival
Endif
Reverse timings content list
For loop for a_timings_content_list - 1
    a_minute = a_minute + a_timings_content_list[i]
    count = count + 1
    Do while a_minute >= 60, then
        a_minute = a_minute - 60
        a_hour = a_hour + 1
    Enddo
    Do while a_hour >= 24, then
        a_hour = a_hour - 24
    Enddo
    a_time = a_hour + a_minute fill to 2 integers each side
    If count is odd number
        Append a_time in a_departure
    Else
        Append a_time in a_arrival
    Endif
Reverse timings content list
count = 1
Create a_duration list
For loop for a_timings_content_list - 1
```

```
        If count is odd number
            Append timings content list in a_duration
            count = count + 1
        Endif
    Enddo
    Reverse timings content list
    For loop for a_timings_content_list - 1
        If count is odd number
            Append timings content list in a_duration
            count = count + 1
        Endif
    Enddo
    Reverse timings content list
    count = 1
    Print timetable header
    For loop for a_station_content_list - 1
        Print station content list to station content list, a_departure, a_arrival, and
        a_duration
        count = count + 1
    Enddo
    Reverse station content list
    count = 1
    For loop for a_station_content_list - 1
        Print station content list to station content list, a_departure, a_arrival, and
a_duration
        count = count + 1
    Enddo
    Reverse station content list
End
```


KL-Thai Timetable Route B

Define KL-Thai timetable A page:

Print header

b_hour = start_time_list

b_minute = start_time_list

Create b_departure list

Create b_arrival list

count = 0

For loop for b_timings_content_list - 1

 b_minute = b_minute + b_timings_content_list[i]

 count = count + 1

 Do while b_minute >= 60, then

 b_minute = b_minute - 60

 b_hour = b_hour + 1

 Enddo

 Do while b_hour >= 24, then

 b_hour = b_hour - 24

 Enddo

 b_time = b_hour + b_minute fill to 2 integers each side

If count is odd number, then

 Append b_time in b_departure

Else, then

 Append b_time in b_arrival

Endif

Reverse timings content list

For loop for b_timings_content_list - 1

 b_minute = b_minute + b_timings_content_list[i]

 count = count + 1

 Do while b_minute >= 60, then

```
        b_minute = b_minute - 60
        b_hour = b_hour + 1
    Enddo
    Do while b_hour >= 24, then
        b_hour = b_hour - 24
    Enddo
    b_time = b_hour + b_minute fill to 2 integers each side
    If count is odd number, then
        Append b_time in b_departure
    Else, then
        Append b_time in b_arrival
    Endif
Reverse timings content list
count = 1
Create b_duration list
For loop for b_timings_content_list - 1
    If count is odd number, then
        Append timings content list in b_duration
    Endif
    count = count + 1
Enddo
Reverse timings content list
For loop for b_timings_content_list - 1
    If count is odd number, then
        Append timings content list in b_duration
    Endif
    count = count + 1
Enddo
Reverse timings content list
count = 1
```

```
Print timetable header

For loop for b_station_content_list - 1
    Print station content list to station content list, b_departure, b_arrival, and
    b_duration
    count = count + 1
Enddo

Reverse station content list
count = 1
For loop for b_station_content_list - 1
    Print station content list to station content list, b_departure, b_arrival, and
    b_duration
    count = count + 1
Enddo

Reverse station content list

End
```

Admin Actions Page

Start

Define admin actions page:

```
admin_action_selected = 0
Print header
Print instruction
Do while admin_action_selected is 0
    Print legend
    Input admin_action
    If admin_action is b, then
        Print selection confirmation
        admin_action_selected = 1
        Select role page function
    Else if admin_action is 1, then
```

```
Print selection confirmation
admin_action_selected = 1
Admin edit timings page function
Else if admin_action is 2, then
    Print selection confirmation
    admin_action_selected = 1
    Admin add stations page function
Else if admin_action is 3, then
    Print selection confirmation
    admin_action_selected = 1
    Admin remove stations page function:
Else if admin_action is 4, then
    Print selection confirmation
    admin_action_selected = 1
    Edit trip start time page function
Else if admin_action is r, then
    admin_action_selected = 1
    Print reset confirmation
    Input reset_confirmation
    If reset_confirmation is YES
        Print selection confirmation
        default_a_stations = ['Kuala Lumpur', 'Butterworth', 'Kedah', 'Perlis',
                              'Thailand']
        Open file a_stations.txt as restore_a_stations
        For station in default_a_stations
            Write list into file
        Enddo
    Close file
    default_b_stations = ['Kuala Lumpur', 'Terengganu', 'Kelantan',
                          'Thailand']
```

```
Open b_stations.txt as restore_b_stations
For station in default_b_stations
    Write list into file
Enddo
Close file
default_c_stations = ['Kuala Lumpur', 'Johore', 'Singapore']
Open c_stations.txt as restore_c_stations
For station in default_c_stations
    Write list into file
Enddo
Close file
default_a_timings = [45, 240, 10, 60, 10, 45, 30, 15, 20]
Open a_timings.txt as restore_a_timings
For timings in default_a_timings
    Write list into file
Enddo
Close file
default_b_timings = [45, 285, 10, 90, 30, 45, 20]
Open b_timings.txt as restore_b_timings
For timings in default_b_timings
    Write list into file
Enddo
Close file
default_c_timings = [45, 270, 30, 30, 20]
Open c_timings.txt as restore_c_timings
For timings in default_c_timings
    Write list into file
Enddo
Close file
default_start_timings = [5, 15, 12, 15, 7, 15, 0, 15, 4, 15, 6, 15, 8,
```

```
15, 12, 15]
Open start_timings.txt as restore_start_timings
For timings in default_start_timings
    Write list into file
Enddo
Close file
Print reset confirmation
Print restart request
Else
    Print reset cancellation
    Admin actions page function
Endif
Else if admin_action is e
    Print exit text
    admin_action_selected = 1
Endif
Else
    Print error text
Endif
End
```

Admin Edit Timings Page

Define admin edit timings page:

```
Print header
admin_edit_timings_complete = 0
Print instruction
Do while admin_edit_timings_complete is 0
    Print legend
    Input admin_edit_timings_selection
```

```
If admin_edit_timings_selection is b, then
    Print selection confirmation
    admin_edit_timings_complete = 1
    Admin actions page function
Else if admin_edit_timings_selection is e, then
    Print exit text
    admin_edit_timings_complete = 1
Else if admin_edit_timings_selection is 1, then
    Print selection confirmation
    admin_edit_timings_complete = 1
    Edit timings A page function
Else if admin_edit_timings_selection is 2, then
    Print selection confirmation
    admin_edit_timings_complete = 1
    Edit timings B page function
Else if admin_edit_timings_selection is 3, then
    Print selection confirmation
    admin_edit_timings_complete = 1
    Edit timings C page function
Else, then
    Print error text
Endif
Enddo
End
```

Edit Timings Function

Define edit timings page:

```
Print header
Create edit_timings_a_intervals list
```

```
count = 0
For loop for a_station_content_list - 1
    Append edit_timings_intervals with station_content_list[i]
    count = count + 1
    station_append = station_content_list[i] + station_content_list[count]
Enddo
Append edit_timings_intervals with station_append
Append edit_timings_intervals with station_content_list[-1]
count = 0
Print timings header
For loop for edit_timings_a_intervals
    count = count + 1
    Print edit_timings_intervals[i], timings_content_list[i]
Enddo
Print instruction
edit_timings_completed = 0
Do while edit_timings_completed is 0
    Print legend
    Input edit_timings_select
    Check if edit_timings is numeric
    If edit_timings_select is b, then
        Print selection confirmation
        edit_timings_completed = 1
        Admin edit timings page function
    Else if edit_timings_select is e, then
        Print exit text
        edit_timings_completed = 1
    Else if edit_timings is numeric, then
        If count >= edit_timings_a_select > 0, then
            Input new_timings
```



```
        timings_content_list[edit_timings_select - 1] = new_timings
        Open file timings.txt as edit_timings
        For timings in timings_content_list
            Write list into file
        Enddo
        Close file
        Print change confirmation text
    Else, then
        Print error text
    Endif
Else, then
    Print error text
Endif
Enddo
End
```

Admin Add Station Page

Start

Define admin add station:

```
    Print header
    admin_add_station_complete = 0
    Do while admin_add_station_complete is 0
        Print legend
        Input admin_edit_station_selection
        If admin_edit_station_selection is b, then
            Print selection confirmation
            admin_add_station_complete = 1
            Admin action page function
        Else if admin_edit_station_selection is e, then
```

```
        Print exit text
        admin_add_station_complete = 1
    Else if admin_edit_station_selection is 1, then
        Print selection confirmation
        admin_add_station_complete = 1
        Add station page A function
    Else if admin_edit_station_selection is 2, then
        Print selection confirmation
        admin_add_station_complete = 1
        Add station page B function
    Else if admin_edit_station_selection is 3, then
        Print selection confirmation
        admin_add_station_complete = 1
        Add station page C function
    Else, then
        Print error text
    Endif
Enddo
End
```

Add Station Function Page

Define add station page:

```
    Print header
    count = 1
    Print table header
    For loop for station_content_list - 1
        Print station_content_list[i]
        count = count + 1
    Enddo
```

```

Print instructions
admin_add_station_complete = 0
While admin_add_station_complete is 0
    Print legend
    Input add_station_select
    If add_station_select is b, then
        Print selection confirmation
        admin_add_station_complete = 1
        Admin add station page function
    Else if add_station_select is e, then
        Print exit page
        admin_add_station_complete = 1
    Else if add_station is numeric, then
        If 0 < add_station_select <= count - 1, then
            Input new_station
            Print instruction
            Input new_station_travel_duration
            If new_station_travel_duration is numeric and
new_station_travel_duration <= a_timings_content_list[add_station_select], then
                Print instruction
                Input new_station_stopover
                If new_station_stopover is numeric, then
                    Insert station_content_list with add_station_select,
new_station
                    Insert a_timings_content_list with
add_station_select * 2 - 1, new_station_travel_duration
                    Insert a_timings_content_list with
add_station_select * 2, new_station_stopover
                    timings_content_list[add_station_a_select] * 2 + 1] -
timings_content_list[(add_station_select * 2 + 1] - new_station_travel_duration
                    Open file a_stations.txt as edit_a_stations

```

```

        For stations in station_content_list
            Write list into file
        Enddo
        Close file
        Open file timings.txt as edit_timings
        For timings in timings_content_list
            Write list into file
        Enddo
        Close file
        Print changes confirmation
        admin_add_station_complete = 1
        Admin add station page function
    Else
        Print error text
    Endif
Else
    Print error text
Endif
Else
    Print error text
Endif
Else
    Print error text
Endif
End
End
```

Remove Station Page

Define admin remove station page:

Print header

```
admin_remove_station_complete = 0
Do while admin_remove_station_complete is 0
    Print legend
    Input admin_edit_station_selection
    If admin_edit_station_selection is b, then
        Print selection confirmation
        admin_remove_station_complete = 1
        Admin action page function
    Else if admin_edit_station_selection is e, then
        Print exit text
        admin_remove_station_complete = 1
    Else if admin_edit_station_selection is 1, then
        Print selection confirmation
        admin_remove_station_complete = 1
        Remove station A page function
    Else if admin_edit_station_selection is 2, then
        Print selection confirmation
        admin_remove_station_complete = 1
        Remove station B page function
    Else if admin_edit_station_selection is 3, then
        Print selection confirmation
        admin_remove_station_complete = 1
        Remove station C page function
    Else, then
        Print error text
    Endif
Enddo
End
```

Define remove station page:

Print header

count = 1

Print table header

For loop for a_station_content_list

Print count, station_content_list[i]

count = count + 1

Enddo

Print instruction

admin_remove_station_complete = 0

Do while admin_remove_station_complete is 0

Print legend

Input remove_station_select

If remove_station_select is b, then

Print selection confirmation

admin_remove_station_complete = 1

Admin remove station page function

Else if remove_station_select is e, then

Print exit text

admin_remove_station_complete = 1

Else if remove_station is numeric, then

If $1 < \text{remove_station_select} \leq \text{count} - 2$, then

Pop station_content_list remove_station_select - 1

Pop timings_content_list remove_station_select * 2 - 2

timings_content_list[remove_station_select * 2 - 3] = \

timings_content_list[remove_station_select * 2 - 2] \

+ timings_content_list[remove_station_select * 2 - 3]

Pop timings_content_list remove_station_select * 2 - 2

Open file stations.txt as edit_stations

For stations in station_content_list

```
        Write list into file
    Enddo
    Close file
    Open file timings.txt as edit_timings
    For timings in timings_content_list
        Write list into file
    Enddo
    Close file
    Print changes saved confirmation
    admin_remove_station_complete = 1
    Admin remove station page function
Else if remove_station_select is 1, then
    Print error text
Else if remove_station_select is count – 1, then
    Print error text
Else, then
    Print error text
Else, then
    Print error text
Enddo
End
```

Admin Edit Trip Starting Time Page

Define edit trip starting time page

```
Print header
start_time_1 = start_time_list[0] + start_time_list[1]
start_time_2 = start_time_list[2] + start_time_list[3]
start_time_3 = start_time_list[4] + start_time_list[5]
start_time_4 = start_time_list[6] + start_time_list[7]
```

```
start_time_5 = start_time_list[8] + start_time_list[9]
start_time_6 = start_time_list[10] + start_time_list[11]
start_time_7 = start_time_list[12] + start_time_list[13]
start_time_8 = start_time_list[14] + start_time_list[15]
Print table header
print 1, KL-Thai Route A Train 1, start_time_1
print 2, KL-Thai Route A Train 2, start_time_2
print 3, KL-Thai Route B Train 1, start_time_3
print 4, KL-SG Route C Train 1, start_time_4
print 5, KL-SG Route C Train 2, start_time_5
print 6, KL-SG Route C Train 3, start_time_6
print 7, KL-SG Route C Train 4, start_time_7
print 8, KL-SG Route C Train 5, start_time_8 + colour.end
Print instructions
trip_start_time_complete = 0
Do while trip_start_time_complete is 0
    Print legend
    Input trip_start_time_selection
    If trip_start_time_selection is b, then
        Print selection confirmation
        trip_start_time_complete = 1
        page_admin_action
    Else If trip_start_time_selection is e, then
        Print exit text
        trip_start_time_complete = 1
    Else If trip_start_time_selection is numeric and 0 < trip_start_time_selection <= 8,
then
        Print instructions
        new_start_hour = input Start hour
        If new_start_hour is numeric and 0 <= new_start_hour < 24
```

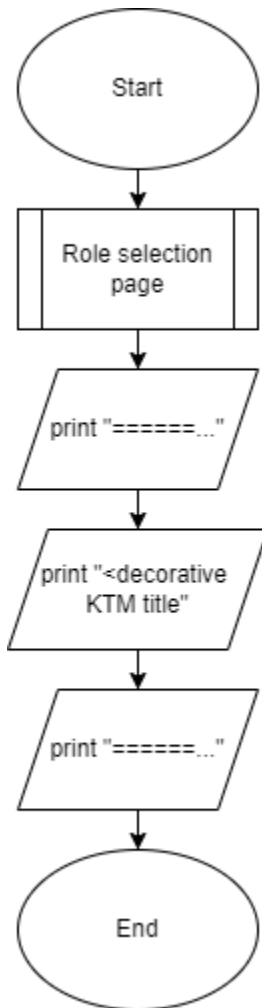


```
new_start_min = input Start minute
If new_start_min is numeric and 0 <= new_start_min < 60
    start_time_list[trip_start_time_selection*2-1] =
new_start_min
    start_time_list[trip_start_time_selection*2-2] =
new_start_hour

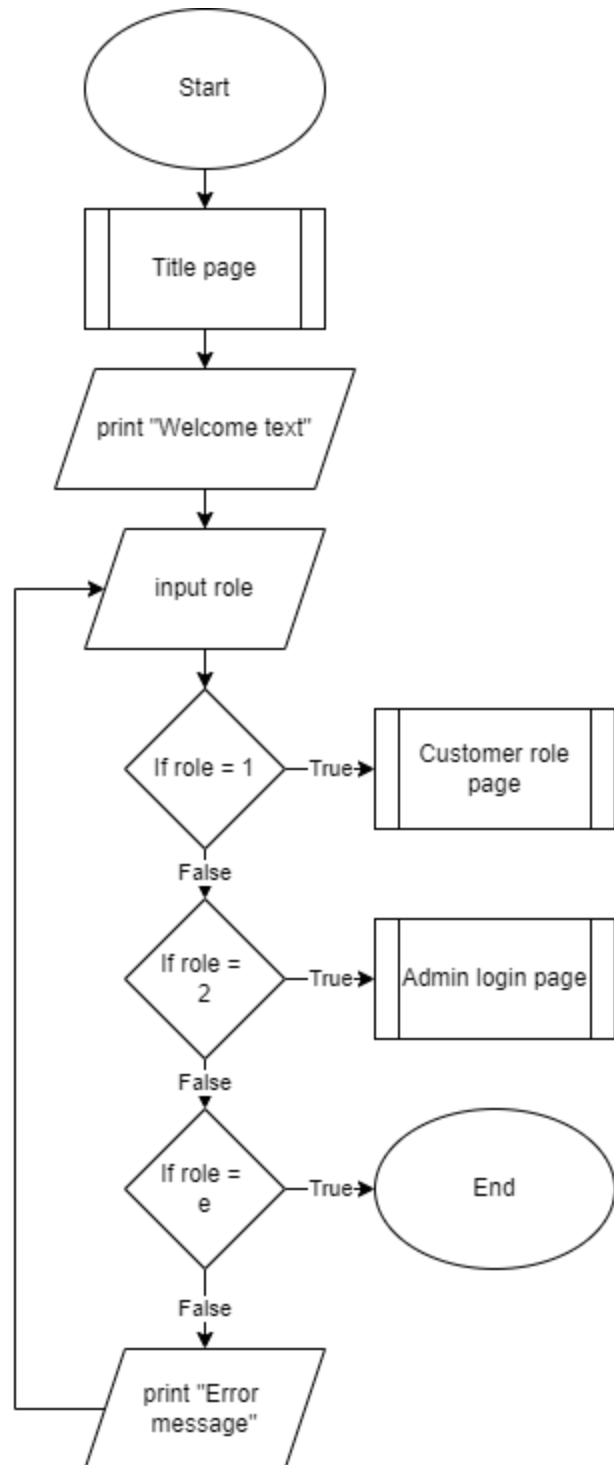
    Open file start_timings.txt as edit_start_timings
    For timings in start_time_list
        Write list into file
    Close file
    trip_start_time_complete = 1
    Print changes saved text
    Edit trip starting time page function
Else, then
    Print error text
Else, then
    Print error text
Else, then
    Print error text
Enddo
End
```

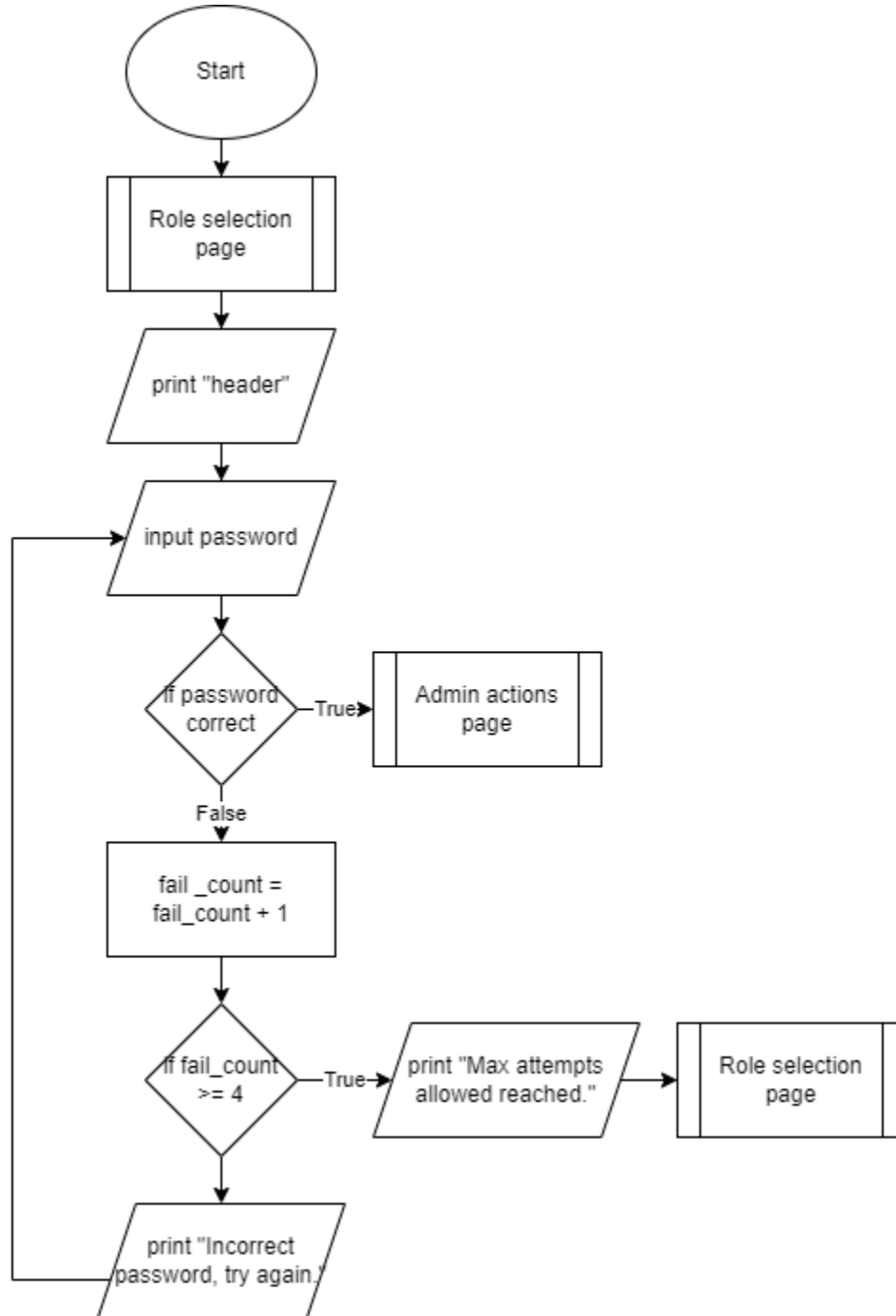
4.2 Flow Charts

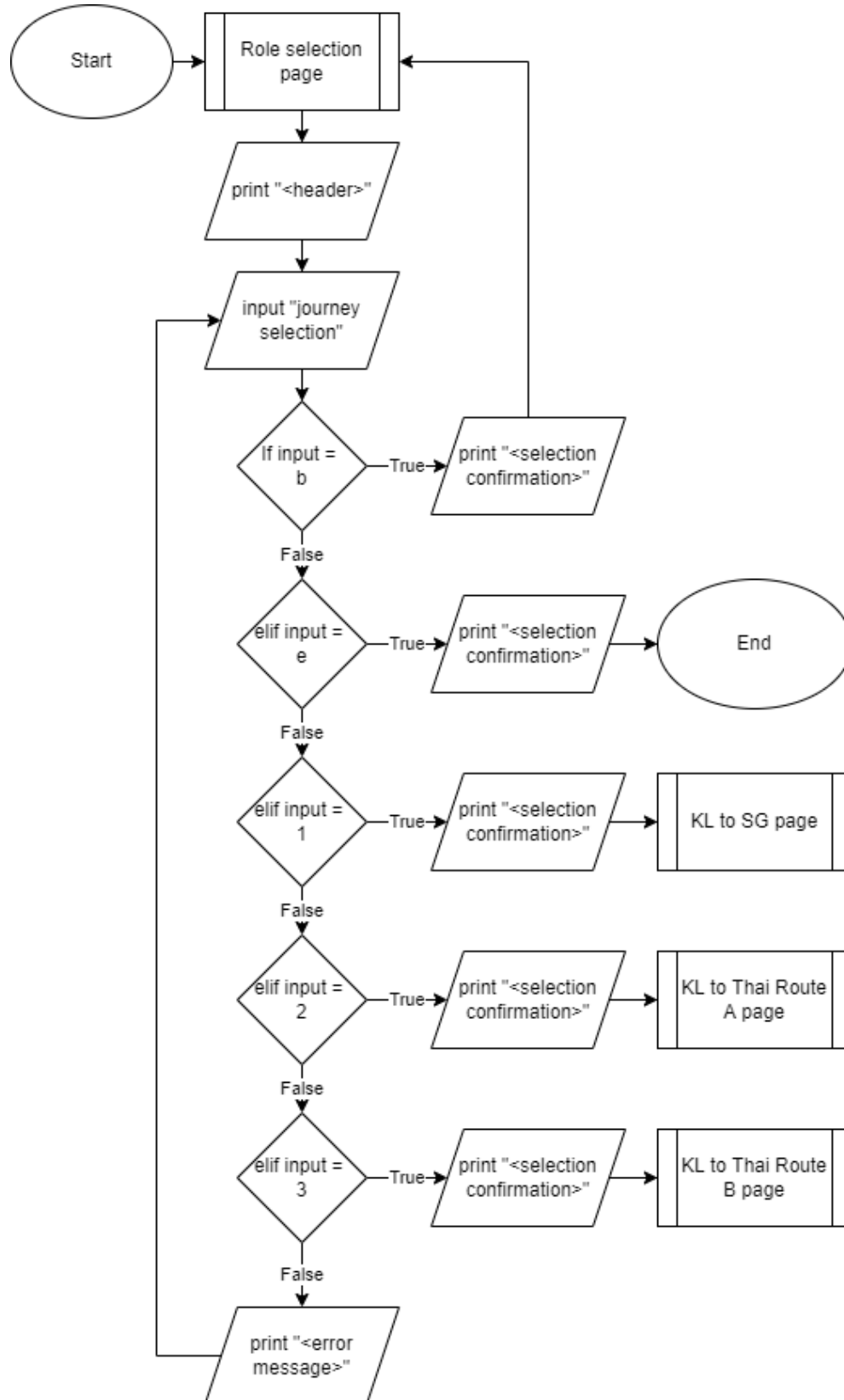
Title Page

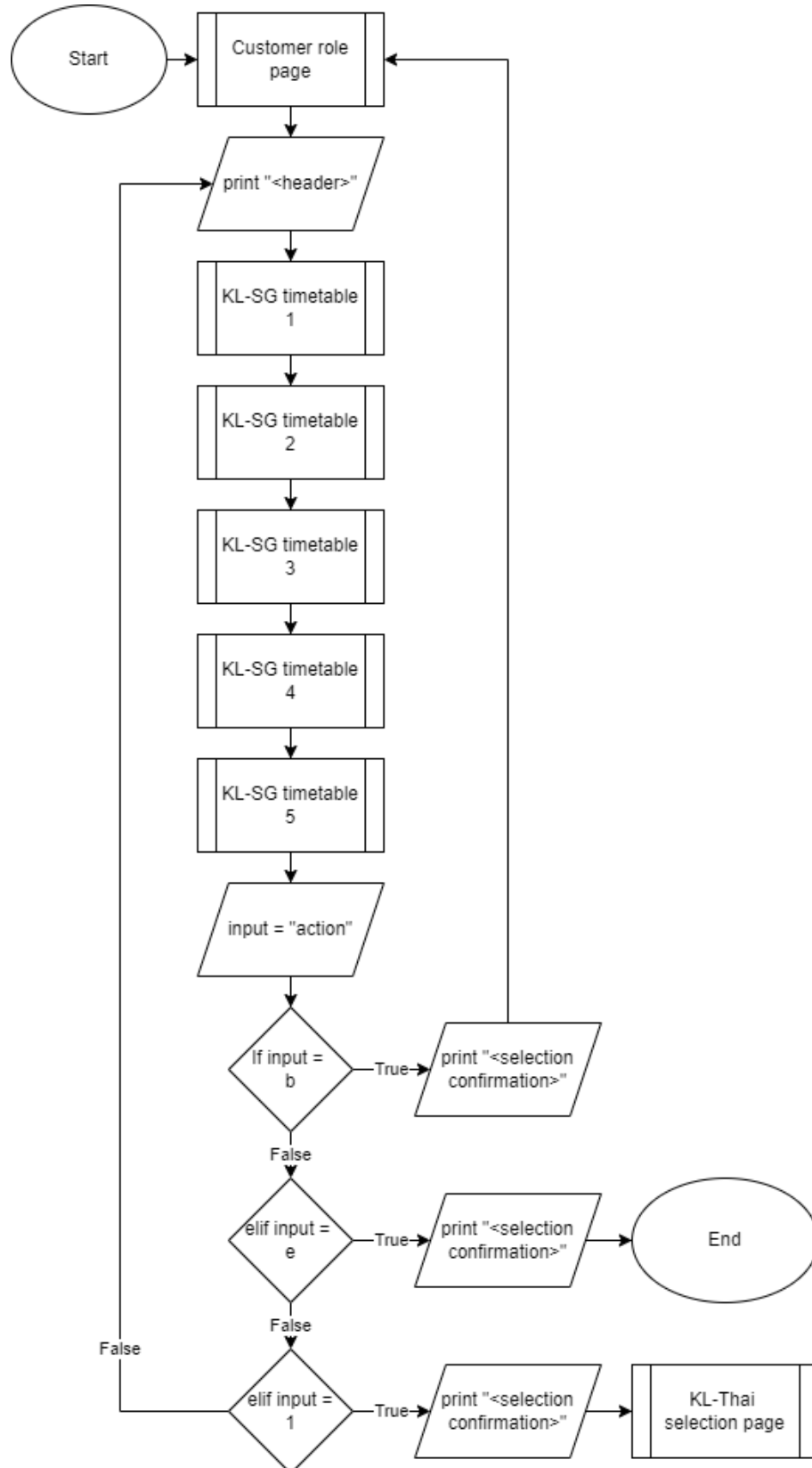


Role Selection Page



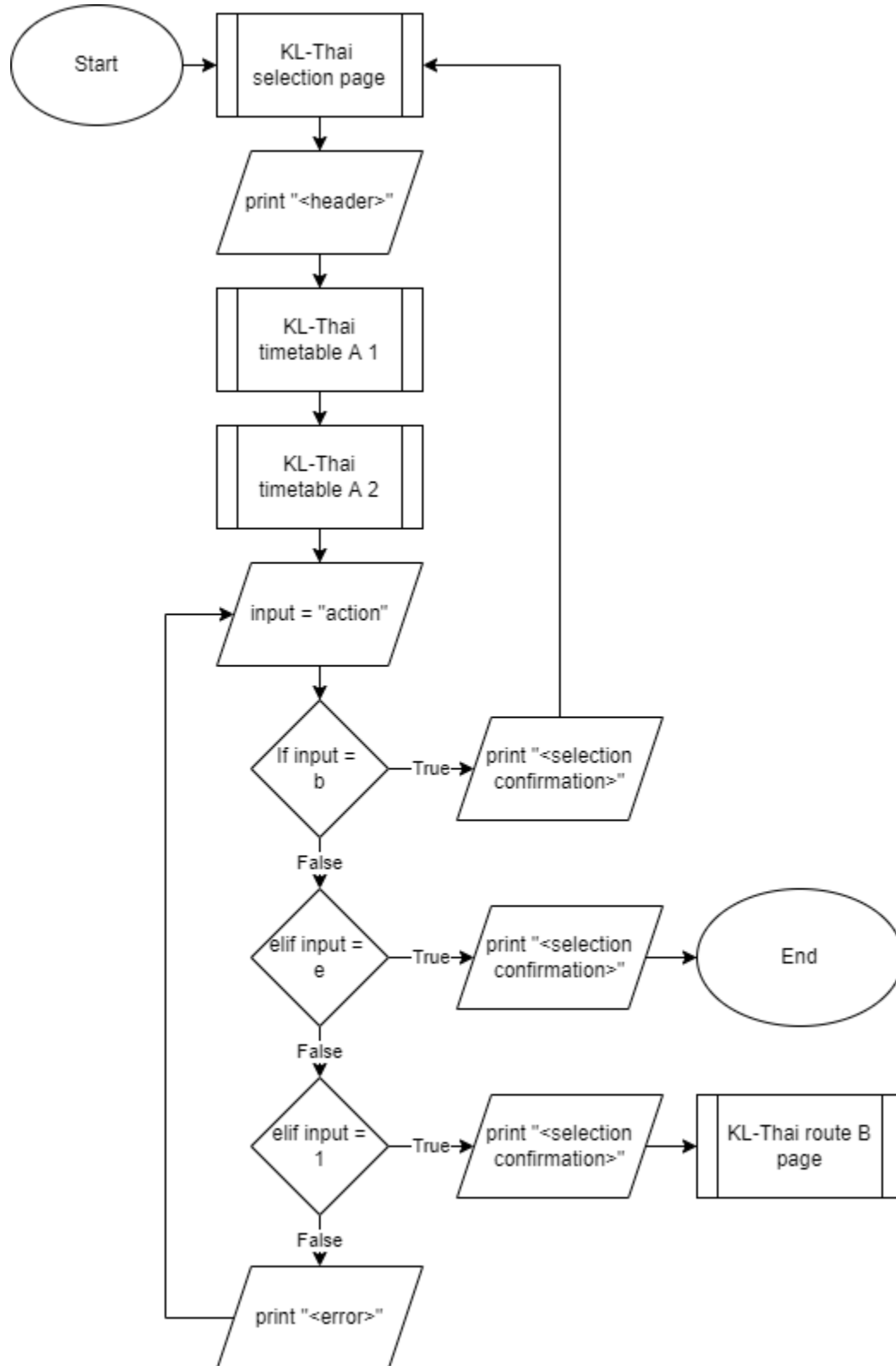
Admin Login Page

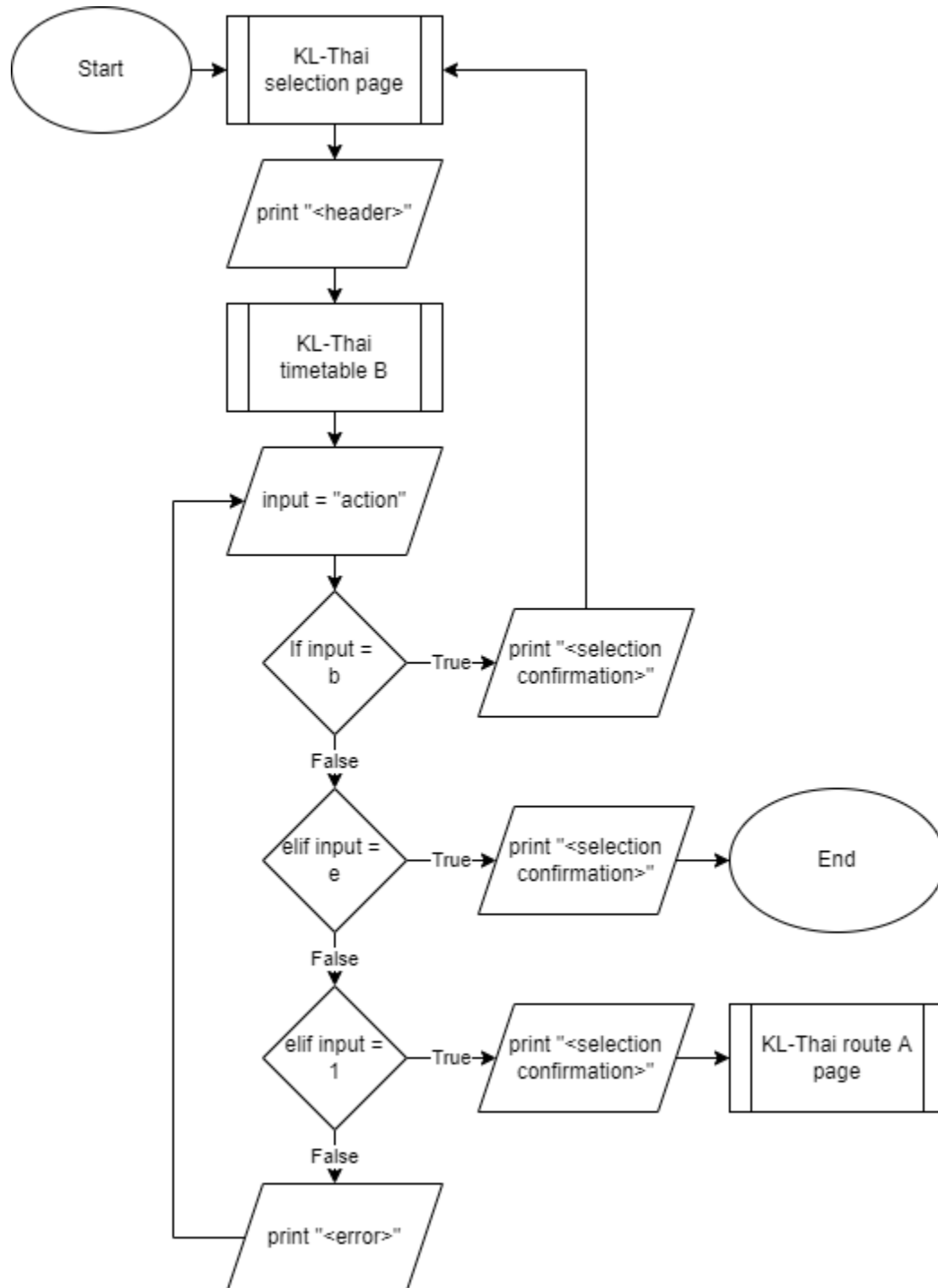
Customer Role Page

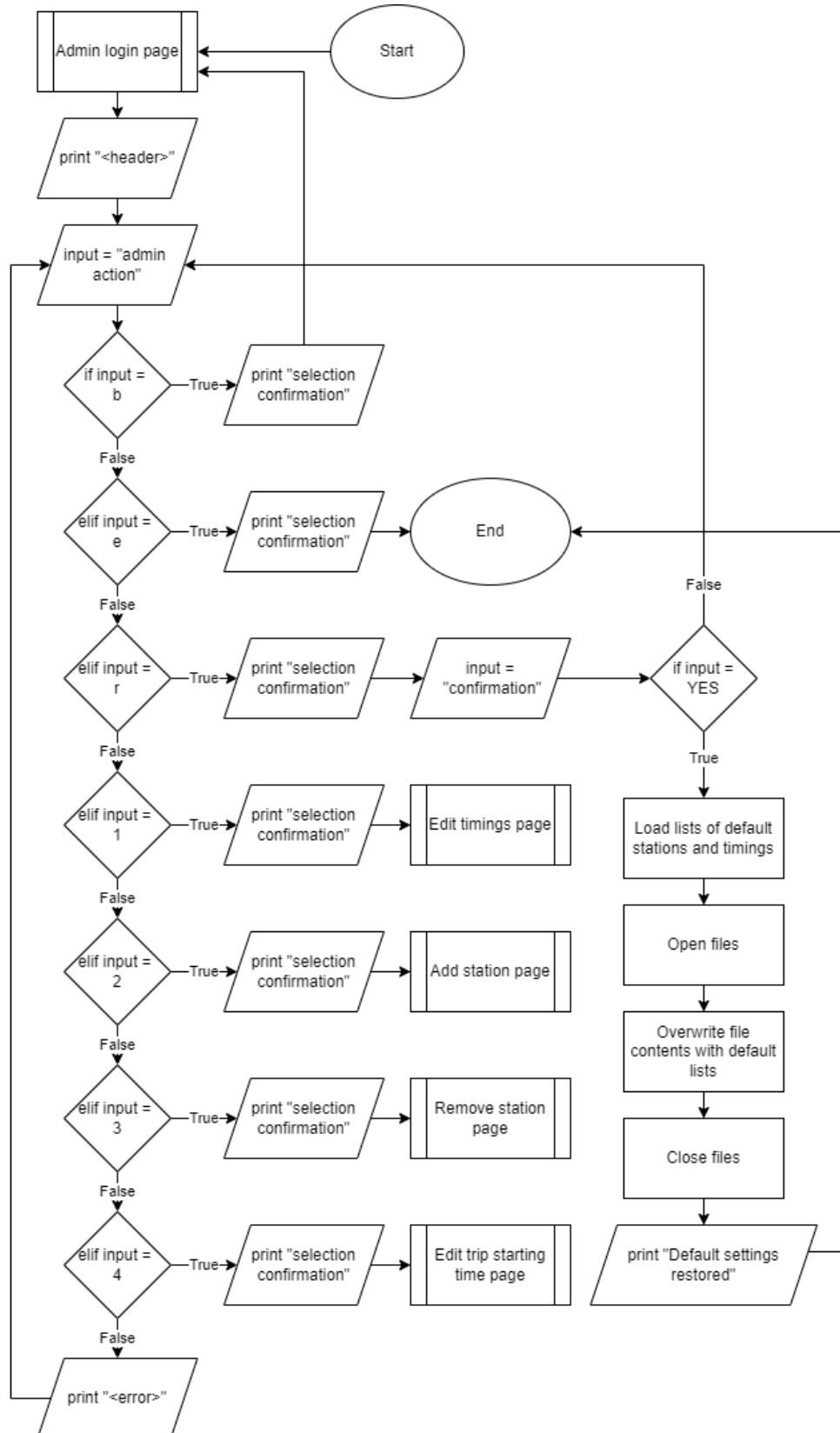
Customer KL-SG Page

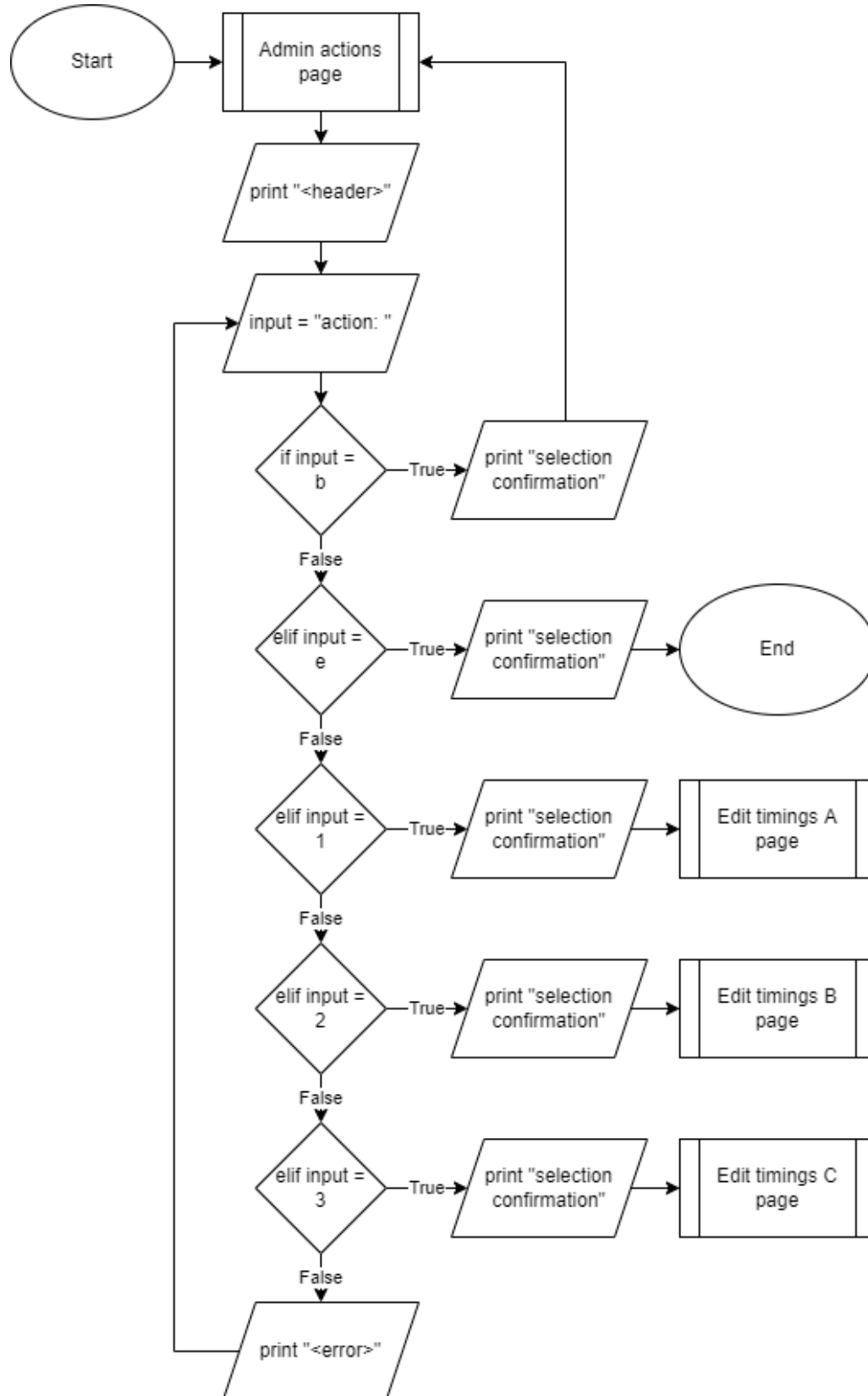
KL-Thai Timetable Route A Page

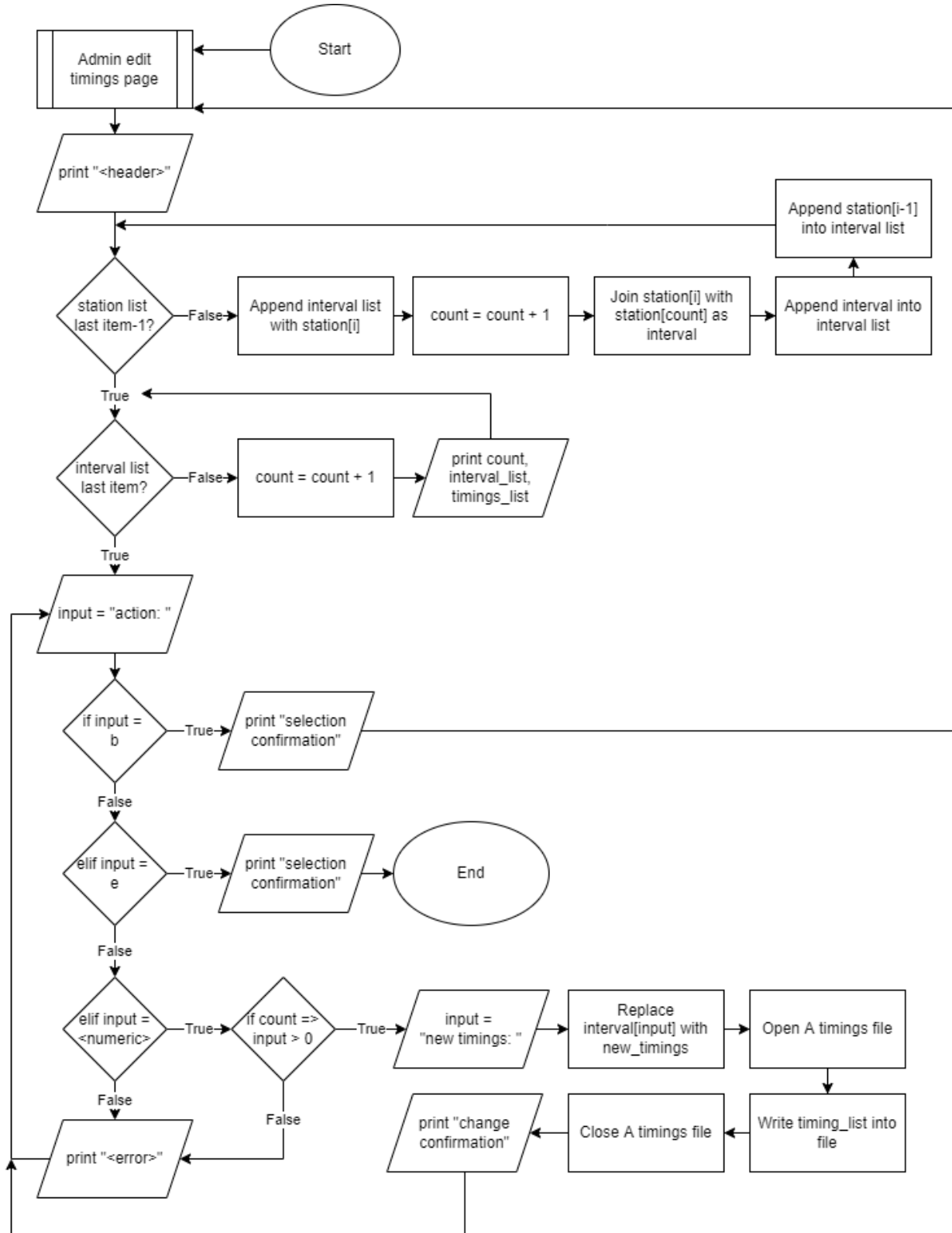
KL-Thai Timetable Route B Page

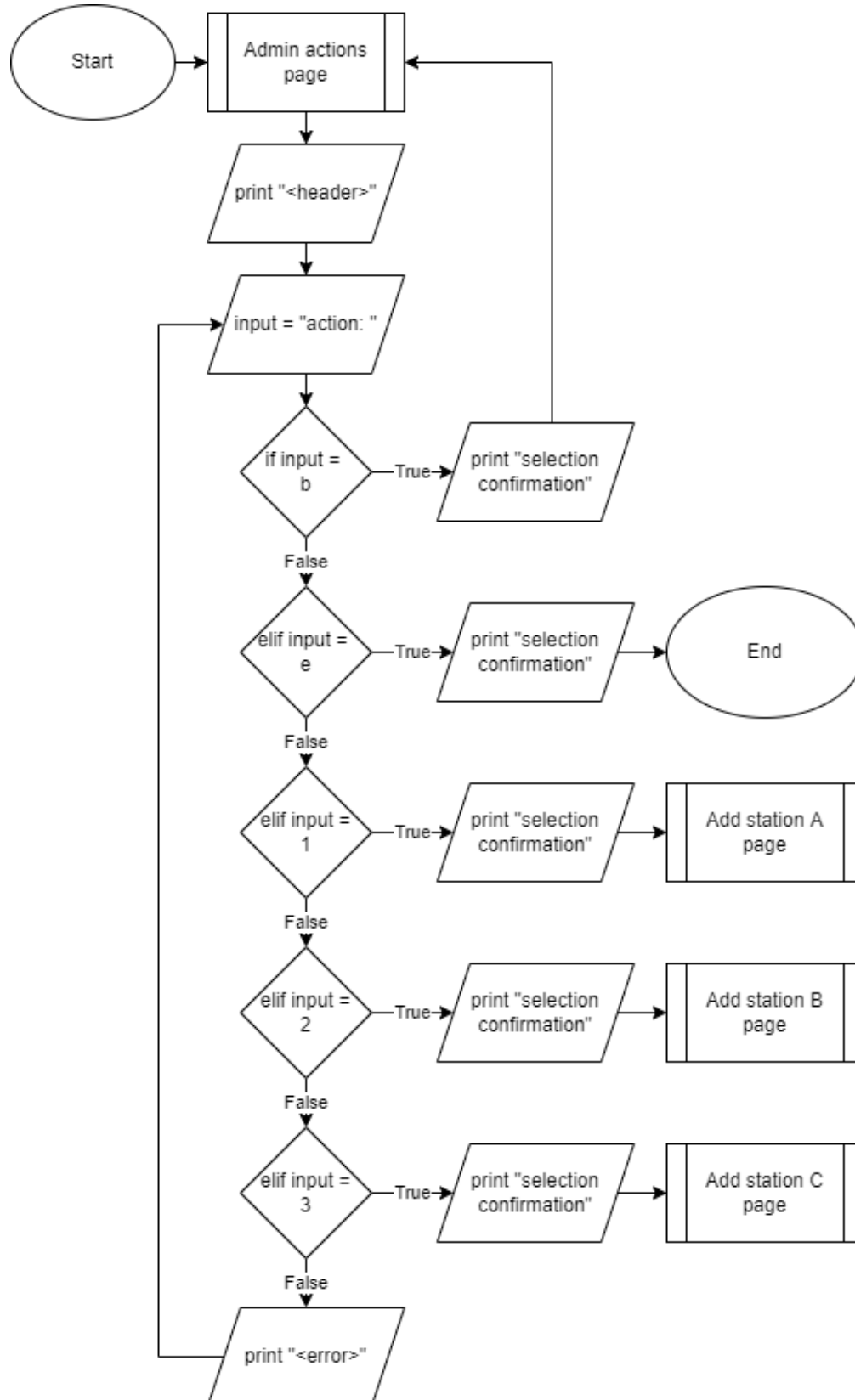
Customer KL-Thai Route A Page

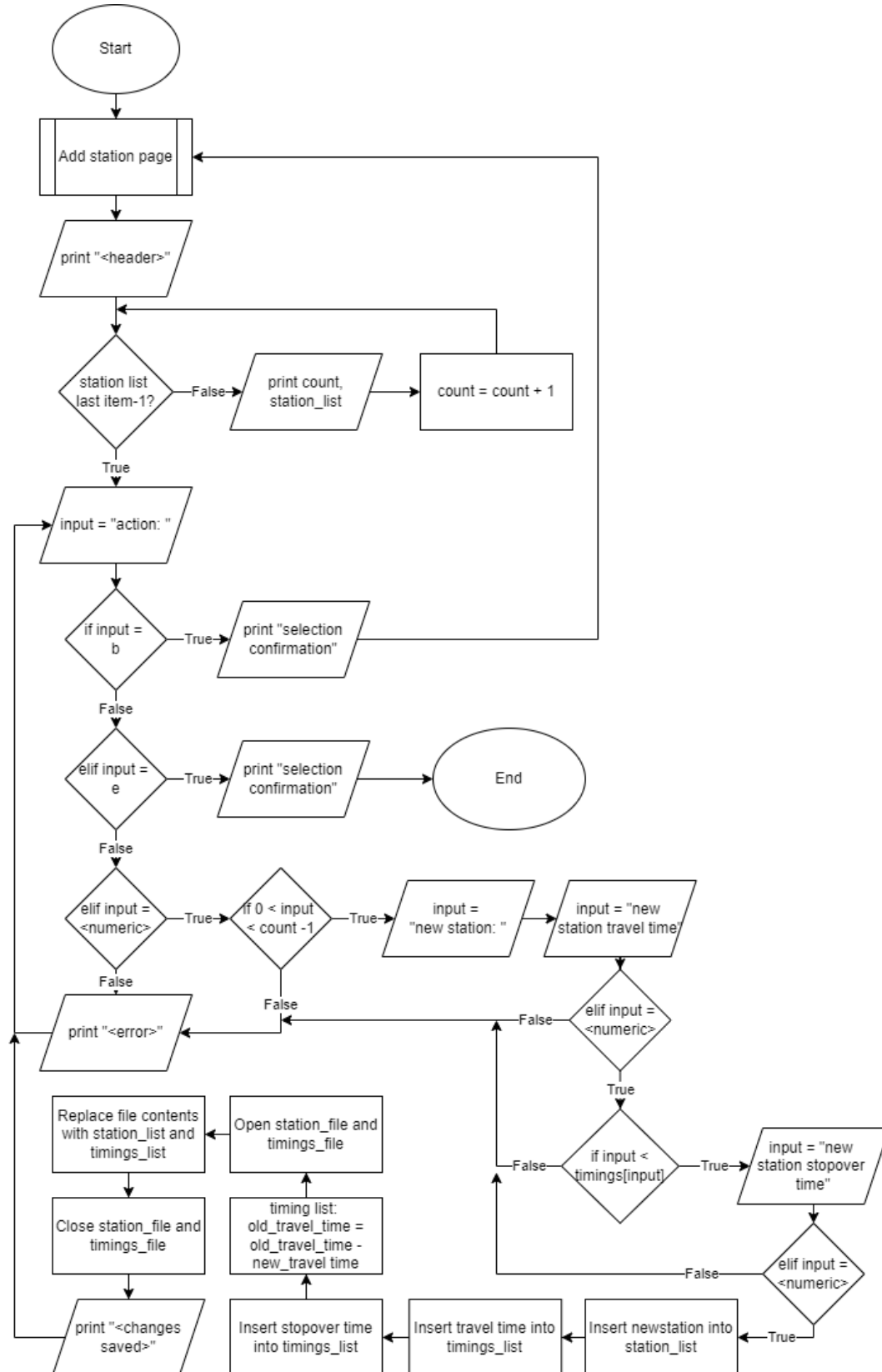
Customer KL-Thai Route B Page

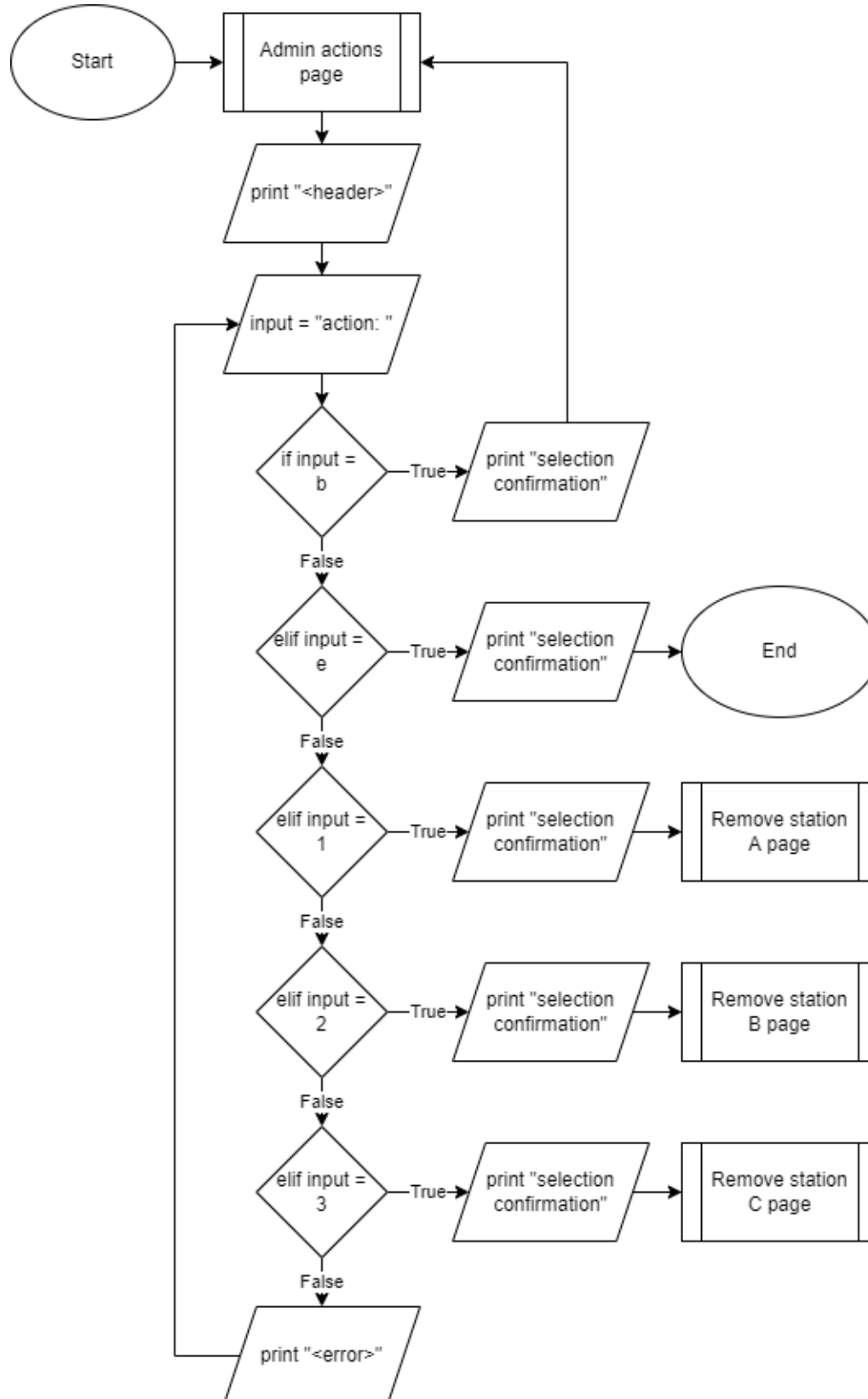
Admin Actions Page

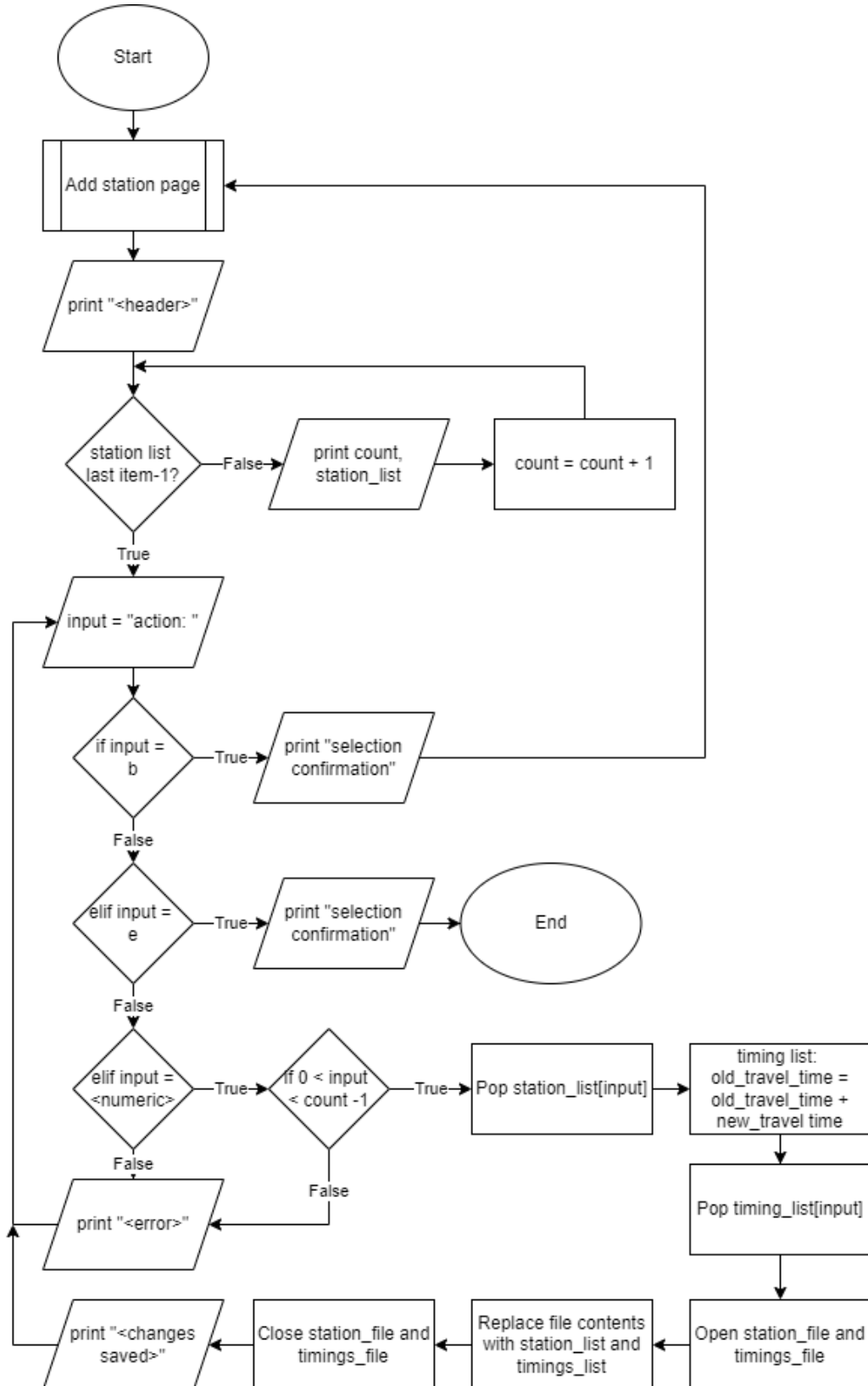
Admin Edit Timings Page

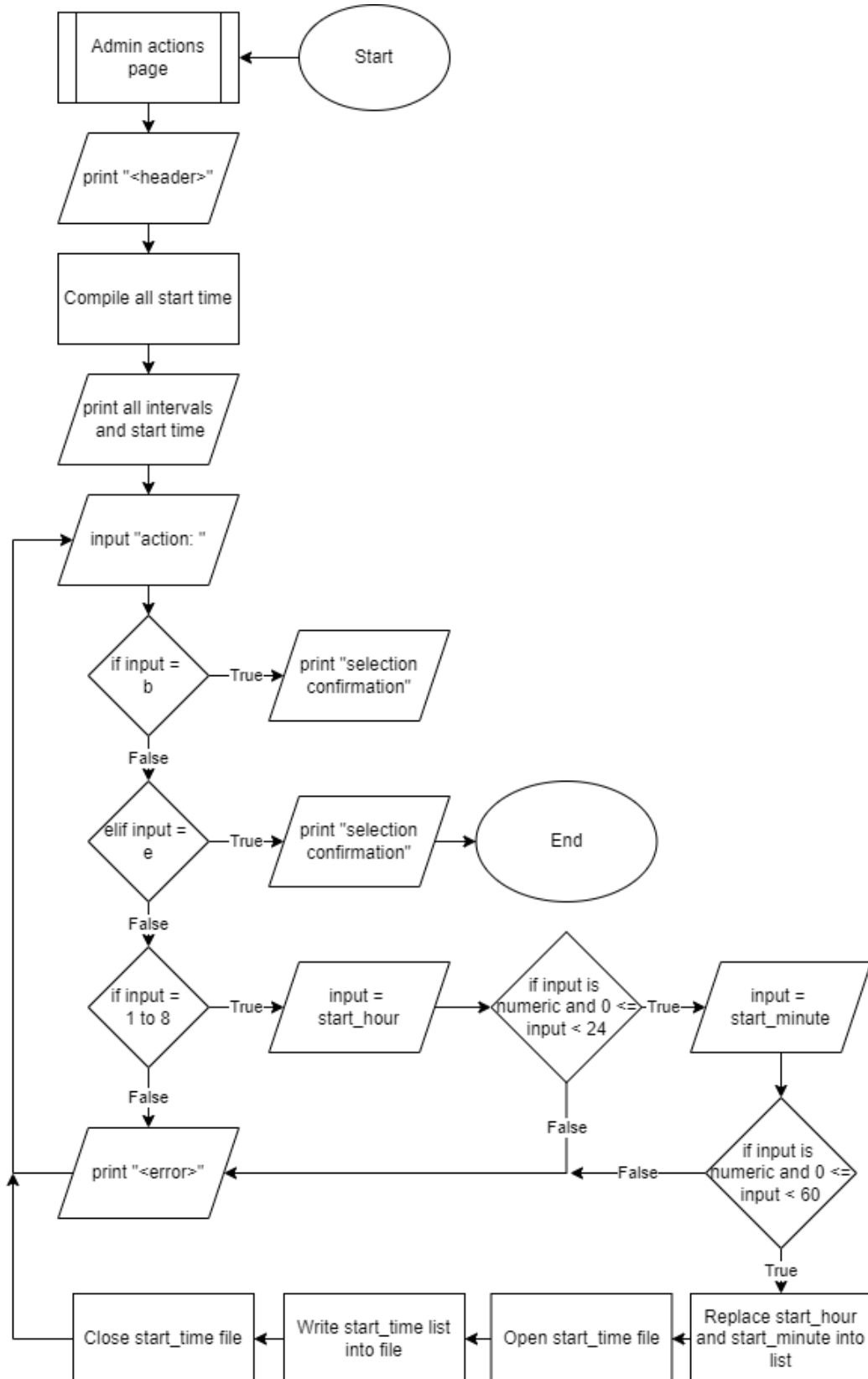
Edit Timings Function Page

Admin Add Stations Page

Add Station Function Page

Admin Remove Station Page

Remove Station Function Page

Edit Trip Starting Time Page

5.0 Program Source Code and Explanation

The following section are programming concepts used in the program with attached examples from the program.

File Handling

The data inputted in a while running a python program is volatile, meaning that all the values that are not in the source code such as values of variables and lists are lost when the program is closed and is not recoverable on the following session. To solve this issue, external files are used to store the data such that it remains persistent across different sessions.

Since the files are not stored locally on the program itself, the data from the files must first be imported and opened. Then, the data in the file is converted into a format that is easier to use inside the program such as a list with separators spacing out each of the data. To avoid potential conflicts, it is best to close the file after importing.

After a use session on the program is completed, any changes made to the data in the lists must be stored back in the files to preserve persistence into the next session. To do this, the file is first opened, then the data of the list is overwritten into the file with a separator spacing out each of the data. The file is closed prior to closing the program.

The following are examples of file handling used in the program:

```
a_stations_file = open("a_stations.txt")
a_stations_file_contents = a_stations_file.read()
a_station_content_list = a_stations_file_contents.splitlines()
a_stations_file.close()
```

Figure 5.1: Program file import.

This section of code first opens and reads the text file. Then, the data of the file is imported into a list with each line as a new separator for new data using the splitlines function. Finally, the file is closed.

```
with open("a_stations.txt", "w") as restore_a_stations:
    for station in default_a_stations:
        restore_a_stations.write("%s\n" % station)
restore_a_stations.close()
```

Figure 5.2: Write list into file.

The code in figure 5.2 first opens the text file. Next, the data from a list is overwritten into the text file with new lines as separator of the data using “\n” nested in a for loop. Finally, the file is closed.

```
1 Kuala Lumpur
2 Butterworth
3 Kedah
4 Perlis
5 Thailand
6 |
```

Figure 5.3: Station file.

Figure 5.3 is one of three text files that is used to store the station names. The data is stored in such a way that every new line is a new data, and the stations represent unidirectional travel. In the example above, it represents travel from Kuala Lumpur to Thailand. To create the bidirectional schedules displayed in the timetables, the list created from this text file is reversed for the return trip and reversed again once the processed is complete to avoid future conflict when using the list.

```
1 45
2 240
3 10
4 60
5 10
6 45
7 30
8 15
9 20
10 |
```

Figure 5.4: Timings file.

Figure 5.4 is the timings text file for the list of stations shown above. For however many stations there are in the stations text file, there are twice minus one timing in the timings file. This is because the timings file store both the stopover time/turnaround time, and the travel duration between each of the stations. All the timings are stored in minutes for easier computations.

In the example above, the first line represents the stopover time of the first station, which is 45 minutes. The second line represents the travel time between the first station and the second station, which is 240 minutes, or 4 hours. The third line again represents the stopover time of the second

station, and so on. The way the program later differentiates between the stopover times and the travel times is via the iteration count being odd or even. For example, off numbered counts represent stopover times whereas even numbered counts represent travel duration.

If-Else Statement

An if statement first checks for whether a statement matches the requirements. If the result is positive, it will execute the command it is attached to. If not, it will skip the command and move on to the next line.

And else-if statement, written in programming as “elif”, usually follows an if statement if there’s more than possibilities. It functions the same way as an if statement. It proceeds the first if statement and makes up the middle portion of an if-else statement chain. If the statement did not find a match, it moves on to the next line.

An else statement is usually the last of an if-else chain. If none of the previous checks returned a positive result, the else statement will execute its attached command.

The following is an example of an if-else chain used in the program:

```
if user_role == str(1):
    print(colour.bold + colour.pink + "Role selected: Customer\n" + colour.end)
    role_selected = 1
    page_role_customer()
elif user_role == str(2):
    print(colour.bold + colour.pink + "Role selected: Admin\n" + colour.end)
    role_selected = 1
    page_role_admin_login()
elif user_role == str('e'):
    print(colour.bold + colour.green + " -- Thank you for using KTM Systems -- " + colour.end)
    role_selected = 1
    input()
else:
    print(colour.bold + colour.red + "[!] Invalid role, please try again." + colour.end)
```

Figure 5.5: Role selection menu.

This section of code is for the user role selection menu. Here, the user gets to choose whether to proceed in the program as a customer or an admin.

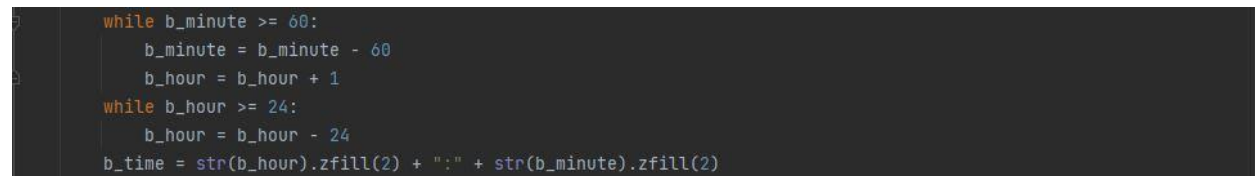
The if statement first checks whether the user_role variable is 1. If yes, it will execute the print command, the variable value change, and the function. If not, it will skip the commands and move onto the next line which is an else-if statement.

In the else-if statement, it checks for whether the variable is equal to 2. If yes, it will execute its commands. Otherwise, it will skip to the next line. The same goes for the next else-if statement.

If none of the previous checks returned positive, the else statement at the end will execute its print command, which is an error message.

While Loop

A while loop is a loop that will run its commands continuously as long as the included statement is true. The following is an example from the program:



```
while b_minute >= 60:
    b_minute = b_minute - 60
    b_hour = b_hour + 1
while b_hour >= 24:
    b_hour = b_hour - 24
b_time = str(b_hour).zfill(2) + ":" + str(b_minute).zfill(2)
```

Figure 5.6: Time overflow system.

This is the time processing in the timetable function of the program. As the timings in the program are stored in minute forms, the final timings need to be converted into hour and minute format for easier readability.

This section of code contains 2 while loops. The first while loop handles the minute to hour computations. It states that while the minute variable is more than or equal to 60, it will execute its commands which is to deduct 60 from the minute variable and add 1 in the hour variable. A while loop is used instead of an if statement because the minute variable could be multiple times greater than 60, thus if an if statement was used, it would only execute the command once. Whereas in a while loop, it would loop continuously until the minute variable is less than 60.

The second while loop handles the hour overflow. If the hour variable is more than or equal to 24, it will deduct 24 from the variable. As this program's time system does not handle days, it is not added.

The last section of the code just combines the hour and minute variables to form a single time variable to be displayed in the timetable.

For Loop

A for loop is a loop that will run its commands continuously over a sequence with a running iteration count. The sequence is typically the length of a list. The following is an example from the program:

```
for i in range(len(a)):
    print(count, "\t\t", a[i])
    count = count + 1
```

Figure 5.7: Print table loop example.

Figure 5.7 is a section of code from the remove station functionality of the program. This section of code prints the list of available stations to be removed.

Variable a is the station list. The for loop is configured to run the command for as many times as is the length of the station list. On each run, it executes its commands which prints a count variable, as well as the station list with iteration. Meaning that on the first run, it will print the first item in the station list, second item on the second run, and so on for the length of the list.

Functions

A function is a block of code that runs only when it is called. It is often used to create menus or to compact code that is used multiple times throughout a program. Functions can also include arguments to create general blocks of code that can be used with different configurations.

The following are examples of functions from the program:

```
def page_title():
    print(colour.bold + colour.blue + "\n===== " + colour.end)
    print(colour.bold + colour.yellow + "=====" + colour.white + " Malayan Railway Limited (KTM) "
          + colour.yellow + "=====" + colour.end)
    print(colour.bold + colour.blue + "===== " + colour.end)
```

Figure 5.8: Fancy title function.

Figure 5.8 is the title screen function from the program. This is a short function that only consist of 3 print commands. The function compacts these 3 lines of code to be used throughout the program when called like in the following:

```
def page_select_role():  
    page_title()  
    print(colour.bold + colour.green + "-=-" + colour.white + "Role Selection" + colour.green + "-=-" + colour.end)
```

Figure 5.9: Title function called.

This is the top portion of another function where on the second line, the title page function is called. Thus, it will first run the commands of the title page function before proceeding to the next line.

As previously mentioned, functions can also include arguments or parameters that can be used inside the function itself. This can be used to create general blocks of code to be used multiple times in a program each with unique outputs.

```
def page_kl_sg_timetable(a, b):  
    print(colour.bold + "===== " + colour.end)  
    c_hour = int(a)  
    c_minute = int(b)
```

Figure 5.10: Function with arguments in use example.

This is the top portion of the KL-SG timetable function. This function uses 2 arguments, a and b. The arguments act as variables that can be used inside the function. In this case, the a and b arguments are used to define the starting hour and minute variables of the timetable.

When a function with arguments is called, values or variables are in place of the arguments and will be used inside the function. Example as follows:

```
page_kl_sg_timetable(start_time_list[6], start_time_list[7])  
page_kl_sg_timetable(start_time_list[8], start_time_list[9])  
page_kl_sg_timetable(start_time_list[10], start_time_list[11])  
page_kl_sg_timetable(start_time_list[12], start_time_list[13])  
page_kl_sg_timetable(start_time_list[14], start_time_list[15])
```

Figure 5.11: Function with arguments when called example.

This is part of the KL-SG Display page. These 5 function calls are the timetables for the 5 daily trains running between Kuala Lumpur and Singapore, each with their own start time. To use the same function for all five of the timetables, parameters for the start hour and minute is used in arguments. Thus, only one function is needed instead of 5 unique functions, thus saving space.

Lists

Python lists are a way of storing an array of data under a common name. Lists can be extended using the append function or shortened using the pop function. Items in a list can also be overwritten or changed.

Python lists are used throughout this program in two different forms, permanent lists, and temporary lists. Permanent lists are lists of data that is imported from files such as lists of station names, lists of timings, and the journey start timings. Temporary lists are lists created for a specific task only such as displaying items in the timetables.

The following are examples lists from the program:

```
for i in range(len(a) - 1):
    edit_timings_intervals.append(a[i])
    count = count + 1
    station_append = a[i] + " to " + a[count]
    edit_timings_intervals.append(station_append)
edit_timings_intervals.append(a[-1])
```

Figure 5.12: Append list example.

Figure 5.12 a portion of code extracted from the edit timings function of the program. The purpose of this code is to organize the station list into two temporary lists used to display the identification of the timing intervals. This example showcases the append function.

The 'a' variable is the station list, and the function is nested inside a for loop. For the first command, the program appends the edit timing intervals list with an item from the station list. This adds the item from the station list to the end of the edit timings intervals list. The next two lines of commands forms a constant which is later appended into the end of the edit timings intervals list on the next line of command. Finally at the last line, the list is appended again with another item from the station list. All of this creates a unique station name list used to identify all of the different timings in the timings list for easier editing.

```
elif remove_select.isnumeric() and 1 < int(remove_select) <= count - 2:
    a.pop(int(remove_select) - 1)
    b.pop((int(remove_select) * 2) - 2)
    b[(int(remove_select) * 2) - 3] = int(b[(int(remove_select)*2)-2]) + int(b[(int(remove_select)*2)-3])
    b.pop((int(remove_select) * 2) - 2)
```

Figure 5.13: Pop list example.

Figure 5.13 is from the remove station function of the program. This code's function is to carry out the removal of a station and its timings upon checks for the user selection. This example showcases the pop function of python lists.

The remove_select variable is the user input. The program first checks for whether the user input is numeric and is within the range of the station count or not. If it returns positive, the removal function is carried out on the selected station.

The pop function removes the specified item from the list. The 'a' and 'b' variables are the station list and timings list respectively. The first command removes the specified station from the station list and the second removes the stopover time of the specified station. The third and forth lines handle the removal of the travel time whilst combining the travel time of the removed station with the travel time of the previous station.

The following is an excerpt from the reset functionality of the program:

```
default_a_timings = [45, 240, 10, 60, 10, 45, 30, 15, 20]
with open("a_timings.txt", "w") as restore_a_timings:
    for timings in default_a_timings:
        restore_a_timings.write("%s\n" % timings)
restore_a_timings.close()
```

Figure 5.14: Example of a list (line 1)

The first line of code sets the lists. It is the default list of timings for Route A. The next four lines is responsible for writing the list into the file. First it opens the file for write. Then, using a for loop, the contents of the list is written into the file with each new line representing a new item in the list. This is done using the '\n' argument. Finally, the file is closed.

6.0 Sample of Input/Output and Explanation

Starting of the Program

```
=====
==== Malayan Railway Limited (KTM) ====
=====
-- Role Selection --
Welcome to the KTM train scheduling system. Which role do you wish to proceed as?
[Customer: '1', Admin: '2', Exit program: 'e']
Role:
```

Figure 6.1: Role selection menu.

This is the front title and role selection page of the program. The user is greeted with this menu as soon as the program is launched. The user has the option to choose between running the program as a customer, or an admin. The user may also choose to exit the program as well by inputting 'e'.

Upon selecting the customer role by inputting '1', the user will be redirected to the customer journey selection page where the user may choose which timetable they wish to view. In contrast, upon selecting the admin role by inputting '2', the user will be redirected to an admin login menu.

This menu is error proof. Meaning that if the user input is not within the possible inputs, an error message will be displayed, and the user is prompted to try again. The same is true for all user input menus in the program. Example of an invalid input:

```
[Customer: '1', Admin: '2', Exit program: 'e']
Role: test
[!] Invalid role, please try again.
```

Figure 6.2: Invalid input example.

Customer Role

```
-- Train Journey --
Which train journey schedule do you wish to view?
[Kuala Lumpur - Singapore: '1', Kuala Lumpur - Thailand Route A: '2', Kuala Lumpur - Thailand Route B: '3', Back: 'b', Exit program: 'e']
Route:
```

Figure 6.3: Customer journey selection menu.

This is the customer journey selection page where the user gets to choose which route they wish to view the timetable of. By inputting '1', the user will be redirected to the KL-SG route timetables

page. '2' will take the user to the KL-Thai Route A timetable page and '3' will lead to the KL-Thai Route B timetable page.

This menu is also equipped with the option for the user to choose to go back a menu by pressing 'b' which redirects the user back to the previous page they were on. As well as an exit program option by inputting 'e', which will end the program. These two functions will be consistent throughout the menus of this program.

Timetables

```

=====
==== Malaysian Railway Limited (KTM) =====
=====
-- Kuala Lumpur - Singapore --
KTM offers 5 daily round trips from Kuala Lumpur to Singapore.
=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Johore	01:00	05:30	270
Johore	to Singapore	06:00	06:30	30
Singapore	to Johore	06:50	07:20	30
Johore	to Kuala Lumpur	07:50	12:20	270

```

=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Johore	05:00	09:30	270
Johore	to Singapore	10:00	10:30	30
Singapore	to Johore	10:50	11:20	30
Johore	to Kuala Lumpur	11:50	16:20	270

```

=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Johore	07:00	11:30	270
Johore	to Singapore	12:00	12:30	30
Singapore	to Johore	12:50	13:20	30
Johore	to Kuala Lumpur	13:50	18:20	270

```

=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Johore	09:00	13:30	270
Johore	to Singapore	14:00	14:30	30
Singapore	to Johore	14:50	15:20	30
Johore	to Kuala Lumpur	15:50	20:20	270

```

=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Johore	13:00	17:30	270
Johore	to Singapore	18:00	18:30	30
Singapore	to Johore	18:50	19:20	30
Johore	to Kuala Lumpur	19:50	00:20	270

```

[Kuala Lumpur - Thailand Route A: '1', Kuala Lumpur - Thailand Route B: '2', Back: 'b', Exit program: 'e']
Action:

```

Figure 6.4: Schedules for KL-SG route.

This is the KL-SG timetable page where users get to view the train schedules. There are 5 daily bidirectional trains running on this route, thus, the timetable is divided up into 5 sections. The timetables clearly display the departure and destination station and times, as well as the travel duration between the stations. The 24-hour time format is used throughout this program.

At the bottom of the page, the user gets to choose to directly view the other timetables, go back a page, or to exit the program. By inputting '1', the user is redirected to the KL-Thailand Route A timetable page, and '2' routes the user to the KL-Thailand Route B timetable page.

The same 'b' and 'e' keys are for back and exit respectively as is on every page. User input error detector is also present on this page.

```

=====
----- Malayan Railway Limited (KTM) -----
=====
-- Kuala Lumpur - Thailand [Route A] --
=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Butterworth	06:00	10:00	240
Butterworth	to Kedah	10:10	11:10	60
Kedah	to Perlis	11:20	12:05	45
Perlis	to Thailand	12:35	12:50	15
Thailand	to Perlis	13:10	13:25	15
Perlis	to Kedah	13:55	14:40	45
Kedah	to Butterworth	14:50	15:50	60
Butterworth	to Kuala Lumpur	16:00	20:00	240

```

=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Butterworth	13:00	17:00	240
Butterworth	to Kedah	17:10	18:10	60
Kedah	to Perlis	18:20	19:05	45
Perlis	to Thailand	19:35	19:50	15
Thailand	to Perlis	20:10	20:25	15
Perlis	to Kedah	20:55	21:40	45
Kedah	to Butterworth	21:50	22:50	60
Butterworth	to Kuala Lumpur	23:00	03:00	240

```

[Kelantan route(B) : '1', Back: 'b', Exit program: 'e']
Action:

```

Figure 6.5: Schedules for KL-Thai Route A

This is the KL-Thailand Route A timetable which routes through Butterworth. KTM offers 2 daily round trips on this route, thus, 2 timetables are visible. Functionality wise, the coding behind this timetable works exactly the same as all other customer viewing timetables with the only difference being the lists used.

At the bottom of the menu the user gets the usual options such as back and exit program, as well as the ability to view the KL-Thailand Route B timetable by inputting '1'.

```

=====
----- Malayan Railway Limited (KTM) -----
=====
-- Kuala Lumpur - Thailand [Route B] --
=====

```

Departure	Destination	Departure Time	Arrival Time	Travelling Duration (Min)
Kuala Lumpur	to Terengganu	08:00	12:45	285
Terengganu	to Kelantan	12:55	14:25	90
Kelantan	to Thailand	14:55	15:40	45
Thailand	to Kelantan	16:00	16:45	45
Kelantan	to Terengganu	17:15	18:45	90
Terengganu	to Kuala Lumpur	18:55	23:40	285

```

[Butterworth route(A) : '1', Back: 'b', Exit program: 'e']
Action:

```

Figure 6.6: Schedule for KL-Thai Route B

Figure 6.6 is the KL-Thailand Route B timetable which routes through Kelantan. Since KTM only offers a single daily train on this route, there is only one timetable.

At the bottom of the menu the user gets the usual options such as back and exit program, as well as the ability to view the KL-Thailand Route A timetable by inputting '1'.

Admin Login

```
== Admin Login ==  
[Back: 'b']  
Please enter the password:
```

Figure 6.7: Admin login menu

Figure 6.7 is the admin login menu. The user is redirected to this menu upon selecting the admin role in the role selection menu. In here, the user will have to enter the correct password to proceed as the admin and access admin functionality. A back option is also available in case a customer accidentally chose the incorrect role.

In this menu, the user will get 5 attempts for inputting the correct password. If on the 5th attempt the user has not yet gotten the correct password, the program will force the user back to the role selection page. The following is the error message shown when an incorrect password is inputted:

```
== Admin Login ==  
[Back: 'b']  
Please enter the password: testing  
[!] Incorrect password, please try again. Attempts left: 4  
[Back: 'b']  
Please enter the password:
```

Figure 6.8: Incorrect password example.

For documentation purposes, the default password for this program is '1111'. Upon inputting the correct password, a login successful message will appear, and the user will be directed to the admin actions menu. A successful login is shown:

```
== Admin Login ==  
[Back: 'b']  
Please enter the password: 12345  
Login successful.
```

Figure 6.9: Successful login.

Admin Action

```
== Admin Panel ==  
Which action would you like to take?  
[Edit Timings: '1', Add Stations: '2', Remove stations: '3', Edit trip starting time: '4', Restore default settings: 'r'  
, Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.10: Admin actions menu.

Upon a successful login, the user will arrive on the admin actions menu. Here, the user can choose between a variety of functions such as edit timings, add, and remove stations, edit trip starting times, and restore default settings. The usual back and exit features are also present on this page.

By inputting '1', the user is redirected to the edit timings menu where the user can choose to edit the stopover time, turnaround time, as well as the travel time between stations. By inputting '2' or '3', the user is directed to the add station menu and remove station menu respectively. The same is the case where the user inputs '4', they will be directed to the edit trip starting time menu.

If the user wishes to reset the program's settings to its default configuration, they can input 'r', which will prompt the user for a confirmation as shown:

```
[Edit Timings: '1', Add Stations: '2', Remove stations: '3', Edit trip starting time: '4', Restore default settings: 'r'  
, Back: 'b', Exit program: 'e']  
Action: r  
Are you sure you want to restore to default settings?  
Type out 'YES' to confirm:
```

Figure 6.11: Reset confirmation request.

To acknowledge the confirmation, the user must type out 'YES' in the input section. This step was added to ensure that custom applied changes cannot be accidentally erased, and that the user is fully committed to what they are doing. Upon entering the confirmation, the program's settings will be restored to its default values and the program will prompt the user to restart the program for the changes to take full effect as shown:

```
== Admin Panel ==  
Which action would you like to take?  
[Edit Timings: '1', Add Stations: '2', Remove stations: '3', Edit trip starting time: '4', Restore default settings: 'r'  
, Back: 'b', Exit program: 'e']  
Action: r  
Are you sure you want to restore to default settings?  
Type out 'YES' to confirm: YES  
Action selected: Restore default settings.  
Default settings restored.  
Program must be restarted.
```

Figure 6.12: Reset successful, prompted restart.

In the event that the user did not input the confirmation text correctly, the program will assume that the user has changed their mind about resetting the program and thus will redirect the user back to the admin actions menu as shown:

```
== Admin Panel ==
Which action would you like to take?
[Edit Timings: '1', Add Stations: '2', Remove stations: '3', Edit trip starting time: '4', Restore default settings: 'r'
, Back: 'b', Exit program: 'e']
Action: r
Are you sure you want to restore to default settings?
Type out 'YES' to confirm: test
Settings reset cancelled.

== Admin Panel ==
Which action would you like to take?
[Edit Timings: '1', Add Stations: '2', Remove stations: '3', Edit trip starting time: '4', Restore default settings: 'r'
, Back: 'b', Exit program: 'e']
Action:
```

Figure 6.13: Reset failed, redirected.

Edit Timings

```
-- Edit Timings --
Please select the route of which the timings you wish to edit:
KL-Thai via Butterworth (Route A): '1'
KL-Thai via Kelantan (Route B): '2'
KL-SG (Route C): '3'
[Back: 'b', Exit program: 'e']
Action:
```

Figure 6.14: Edit timings menu.

Figure 6.14 is the edit timings menu. The user will arrive on this menu upon selecting the edit timings option from the admin actions menu. Here, the user may choose a route to edit the stopover, turnaround, and travel timings. To choose a route, the user can input '1', '2', or '3', to edit timings of route A, B, or C respectively. The usual back and exit function are also present.

```
-- Edit Timings --
Index      Intervals      Timings (In minutes)
1          Kuala Lumpur      45
2          Kuala Lumpur to Butterworth 240
3          Butterworth      10
4          Butterworth to Kedah 60
5          Kedah            10
6          Kedah to Perlis  45
7          Perlis           30
8          Perlis to Thailand 15
9          Thailand         20
Please select the interval to edit.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.15: Edit timings route A menu.

The following is presented upon inputting '1', arriving on the menu for route A as shown in figure 6.15. In this menu, the timings are broken down into intervals as shown in the table. Stopover or turnaround times are represented with their station names, and travel duration between two stations are represented with a station-to-station listing. The user may choose the interval by entering the index number into the input section.

```
-- Edit Timings --
Index   Intervals                               Timings (In minutes)
1        Kuala Lumpur                           45
2        Kuala Lumpur to Butterworth            240
3        Butterworth                           10
4        Butterworth to Kedah                   60
5        Kedah                                 10
6        Kedah to Perlis                       45
7        Perlis                                30
8        Perlis to Thailand                     15
9        Thailand                              20
Please select the interval to edit.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New timing (in minutes):
```

Figure 6.16: Editing interval 1 demonstration.

For example, in figure 6.16, the first interval is selected by inputting '1'. This corresponds to the turnaround time at the Kuala Lumpur station. Then, the program prompts the user to input the new timings in minutes.

```
Please select the interval to edit.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New timing (in minutes): 50
Changes saved.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.17: New timings entered.

In this example, '50' is inputted as the new timings for the selected interval. The program prints a change confirmation text, and the user is then allowed to make another change. Once the changes are made, the user may wish to leave this menu by using the back function by inputting 'b'.

A program restart is not required for the change to take effect.

```
-- Edit Timings --  
Index      Intervals      Timings (In minutes)  
1          Kuala Lumpur      50  
2          Kuala Lumpur to Butterworth 240  
3          Butterworth      10  
4          Butterworth to Kedah 60  
5          Kedah      10  
6          Kedah to Perlis 45  
7          Perlis      30  
8          Perlis to Thailand 15  
9          Thailand      20  
Please select the interval to edit.  
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']  
Action: 10  
[!] Invalid selection, please try again.  
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.18: Selection outside of range.

This menu is also fully equipped with error handling. As shown in figure 6.18, the possible index inputs are from 1 to 9. If the user inputs a number outside of the range, an error message is displayed, and the user is prompted to try again.

```
Please select the interval to edit.  
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']  
Action: 1  
New timing (in minutes): test  
[!] Invalid selection, please try again.  
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.19: Timing not an integer.

Error handling is also present on the new timings input section. Since the new timings are meant for minutes which are in integer form only, if a user inputs anything other than an integer, an error message is displayed, and the user is asked to try again.

The examples shown for the edit timings function were for KL-Thai Route A. However, the menus, functionality, and method of operation is exactly the same for KL-Thai Route B, and KL-SG route. With the only difference being the lists used when running the function.

Add Station

```
-- Add Stations --
KL-Thai via Butterworth (Route A): '1'
KL-Thai via Kelantan (Route B): '2'
KL-SG (Route C): '3'
[Back: 'b', Exit program: 'e']
Action:
```

Figure 6.20: Add station menu.

Figure 6.20 is the add station menu. The user will arrive on this menu upon selecting the add station option from the admin actions menu. Here, the user may select a route to add a station. To choose a route, the user can input '1', '2', or '3', to add stations on route A, B, or C respectively. The usual back and exit function are also present.

```
-- Add Stations --
Index   Stations
1        Kuala Lumpur
2        Butterworth
3        Kedah
4        Perlis
Please select the station after which the new station will be placed.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.21: Add station route A menu.

Upon inputting '1' to select the stations for KL-Thai route A, the menu in figure 6.21 is presented. In this menu, all the stations on the route are listed. To add a station, the user must select the station which will precede the newly added station. For example, to add a station between the Kuala Lumpur station and Butterworth station, the user must select Kuala Lumpur station, which is '1' in the input section.

```
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New Station Name:
```

Figure 6.22: Input new station name.

Upon selecting the location of the new station, the user is prompted to input the new station's name as shown in figure 6.22. The name can be in whatever form the user chooses. For the proceeding example, Lumut was chosen as it resides between Kuala Lumpur and Butterworth.

Next, the user is prompted to input the travel duration from the previous station to the new station. The input is in minutes and the maximum allowable time is limited to under the original travel duration between the old stations. This will be displayed as shown in figure 6.23 below:

```
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New Station Name: Lumut
Please set the travel duration from previous station to the new station. Max: 240
Time in minutes:
```

Figure 6.23: Input travel duration from previous station.

Upon entering the travel duration from the previous station to the newly added station, the program automatically readjusts the travel time from the new station to the next station accordingly.

For example, the original travel time between KL and Butterworth is 240 minutes, upon adding a station, Lumut, in between the two. The travel time from KL station to Lumut station is set at 120 minutes. The program then automatically adjusts the travel time from Lumut to Butterworth by subtracting the original duration (240), with the new duration (120), to generate the duration from the new station to the next station (120). This change is not displayed to the user in the add station menu and will only be reflected in the schedules.

```
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New Station Name: Lumut
Please set the travel duration from previous station to the new station. Max: 240
Time in minutes: 120
Please set the stopover time at the new station.
Time in minutes:
```

Figure 6.24: Input new station stopover time.

Next, the user is asked to enter the new station's stopover time in minutes. After which the changes are saved and written into the files in the background. A confirmation text will be printed as follows:

```
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
New Station Name: Lumut
Please set the travel duration from previous station to the new station. Max: 240
Time in minutes: 120
Please set the stopover time at the new station.
Time in minutes: 10
Changes saved
```

Figure 6.25: Add station complete.

Similar to all other menus, this function also checks for errors such as selecting a non-existent station or the user entering a string where an integer is supposed to be inputted as shown below:

```
Please set the travel duration from previous station to the new station. Max: 240
Time in minutes: test
[!] Invalid selection, please try again.
[Select Interval: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.26: Error handling.

The showcase for the add station function has thus far been from route A. However, the same applies for the other routes as well as they all function the same. The only difference being that different lists are used.

Remove Station

```
-- Remove Stations --  
KL-Thai via Butterworth (Route A): '1'  
KL-Thai via Kelantan (Route B): '2'  
KL-SG (Route C): '3'  
[Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.27: Remove station menu.

Figure 6.27 shows the remove station menu of the program. The user will arrive on this page upon selecting the remove station function in the admin actions menu. As the name suggests, the user gets to remove stations in this menu. To choose a route, the user can input '1', '2', or '3', to remove stations from route A, B, or C respectively. The usual back and exit function are also present.

```
-- Remove Stations --  
Index      Stations  
1           Kuala Lumpur  
2           Butterworth  
3           Kedah  
4           Perlis  
5           Thailand  
Please select the station that you wish to remove.  
[Select Station: 'Index Number', Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.28: Remove station route A menu.

Upon inputting '1' to select the stations for KL-Thai route A, the menu in figure 6.28 is shown. In this menu, all the stations on the route are listed. To remove a station, the user must select the station according to its index number. For example, to remove the Butterworth station, the user must input '2'.

```
Please select the station that you wish to remove.  
[Select Station: 'Index Number', Back: 'b', Exit program: 'e']  
Action: 2  
Changes saved.
```

Figure 6.29: Station successfully removed.

Upon a successful removal, a changes saved message is shown as in figure 6.29.

In addition, the program also automatically adjusts the travel duration in the background. For example, the travel duration from Kuala Lumpur station to Butterworth station is 240 minutes and the travel duration from Butterworth station to Kedah station is 60 minutes. When the Butterworth station is removed, the program combines the original 2 travel durations to form a single longer travel duration. Resulting in a travel duration between Kuala Lumpur station and Kedah station of 300 minutes. This change is made to reflect the real world more accurately as travel durations don't suddenly change upon the removal of the station.

```
Please select the station that you wish to remove.  
[Select Station: 'Index Number', Back: 'b', Exit program: 'e']  
Action: 1  
[!] Cannot remove hub station, please try again.  
[Select Station: 'Index Number', Back: 'b', Exit program: 'e']  
Action: 4  
[!] Cannot remove terminus station, please try again.  
[Select Station: 'Index Number', Back: 'b', Exit program: 'e']  
Action:
```

Figure 6.30: Function limitations.

The station remove function has some limitations, namely the hub station which is Kuala Lumpur station, as well as the terminus stations, Thailand, and Singapore stations, cannot be removed. This limitation is due to how the program handles the timings removal logic. If the user tries to remove the stations, error messages such as in figure 6.30 will be shown and the user will be asked to try again.

Edit Trip Starting Time

```
-- Edit Trip Starting Times --
Index   Trains                Starting Time
1        KL-Thai Route A Train 1      05:15
2        KL-Thai Route A Train 2      12:15
3        KL-Thai Route B Train 1      07:15
4        KL-SG Route C Train 1        00:15
5        KL-SG Route C Train 2        04:15
6        KL-SG Route C Train 3        06:15
7        KL-SG Route C Train 4        08:15
8        KL-SG Route C Train 5        12:15
Note that the actual start time accounts for the starting station's stopover time.
[Edit start time: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.31: Edit trip start time menu.

The menu shown above is the edit trip start time menu. The user can arrive at this page upon selecting the edit trip starting time option in the admin actions menu. In this page, the user gets to change the initial start time of all the trains from Kuala Lumpur station.

There are a total of 8 trains and each of the trains are represented by one line in the table. The user can select which train's start time they wish to adjust by inputting its index number. The train's start time is displayed on the side in 24-hour format.

```
[Edit start time: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
Please enter the new starting time.
Start hour: 6
Start minute:
```

Figure 6.32: Changing start time.

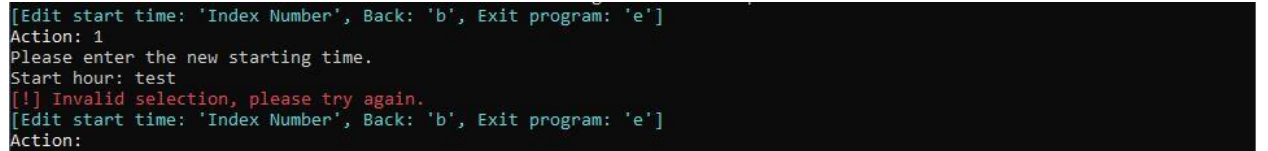
Upon selecting a train to edit the start time, the user is prompted to input the start hour first, followed by the starting minute as shown in Figure 6.32. Do note that the starting time does account for the hub station's turnaround time. For example, the default turnaround time for Kuala Lumpur station is 45 minutes and the 1st train on the KL-Thai Route A has a starting time of 05:15. The actual departure time is a sum of the turnaround time and the trip start time, meaning that the train will depart Kuala Lumpur station at 06:00.

```
[Edit start time: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
Please enter the new starting time.
Start hour: 6
Start minute: 15
Changes saved.
```

Figure 6.33: Edit complete.

Upon a successful edit, a confirmation message is shown as in figure 6.33.

As always, this menu is also fully equipped with error handling. When the program prompts the user for new hour and minute values of the start time, the user can only input integers. Inputting anything else will result in an error message printed and the user is asked to try again as shown in figure 6.34 below:



```
[Edit start time: 'Index Number', Back: 'b', Exit program: 'e']
Action: 1
Please enter the new starting time.
Start hour: test
[!] Invalid selection, please try again.
[Edit start time: 'Index Number', Back: 'b', Exit program: 'e']
Action:
```

Figure 6.34: Error handling.

7.0 Conclusion

In summary, the program is considered a success as it achieves all its original objectives and more. The program can be broken down into two user roles, a customer/passenger role, and the admin role. Each with their unique capabilities.

The customer role can view all the schedules and timetables in a clear and easily understandable manner. The admin role enables the user to make changes to the timetables and schedules such as editing all the timings, adding, and removing stations, edit the starting times of each of the trains, as well as resetting the program settings if needed. In addition, all the inputs in the program are fully equipped with error handling to deal with mistaken or unintended inputs. Lastly, all the menus also feature a back and exit function, allowing the user to go back a page or exit the program whenever they wish.

Unfortunately, there are also some limitations of the program that was unable to be addressed within the timeframe of the assignment. The first is the inability to add a station before or after the last station of the route. Next is the inability to remove the first and last station of a route. Both limitations are due to how the program handles the timing calculations. A solution to this problem has been identified but is unable to be implemented in the limited timeframe of the assignment.

In conclusion, given more experience and knowledge in programming in Python, more capabilities and optimizations can be added and made to the program. However, as it stands, the program still fully achieved all its intended functions and beyond.

My knowledge in programming has increased significantly since undertaking this task and I am honored to be given this opportunity to explore Python programming to the best of my abilities.

8.0 References

1. Mortensen, P. joeld. (2008, Nov 13). *How do I print colored text to the terminal?*. Retrieved from stackoverflow: <https://stackoverflow.com/questions/287871/how-do-i-print-colored-text-to-the-terminal>
2. Python. (n.d.). *Python Documentation*. Retrieved from Python website: <https://docs.python.org/3/>
3. W3 Schools. (n.d.). *Python Functions*. Retrieved from W3 Schools website: https://www.w3schools.com/python/python_functions.asp
4. KTM. (n.d.). *KTMB*. Retrieved from KTM website: <https://www.ktmb.com.my/traintime.html>
5. Lim, A. (2022, Aug 24). *KL-Singapore HSR high-speed rail*. Retrieved from paultan.org: <https://paultan.org/2022/08/24/kl-singapore-hsr-high-speed-rail-malaysia-looking-to-revive-project-as-soon-as-possible-talks-ongoing/>