

**PSP0201 - MINI IT PROJECT**

**Trimester 3, 2017/2018**

**Long Distance Bus Tickets Booking System**

**Lecturer: Mr. Neoh Kee Lin**

**Group: A02A**

**Group Members:**

|  |  |  |
| --- | --- | --- |
| **Student Name** | **ID** | **Task Distribution** |
| Kerk Chee Sheng | 1171100287 | Main / History / Select Bus Seat Module |
| Ang Jun Yuan | 1171101094 | Login / Registration / Authentication Module |
| Siah Ming Khai | 1171101374 | Calculation and Printing Module |
| Ong Shuoh Chwen | 1171102212 | Calendar and End Screen Module |

Table of Content

[Abstract 1](#_Toc515215711)

[Chapter 1: Introduction 2](#_Toc515215712)

[1.1 Introduction 2](#_Toc515215713)

[1.2 Problem Statement 3](#_Toc515215714)

[1.3 Objectives 4](#_Toc515215715)

[Chapter 2: Background Study 5](#_Toc515215716)

[2.1 Reasons and benefits to develop a long distance bus tickets booking system 5](#_Toc515215717)

[2.2 Existing long distance bus ticket booking system in Malaysia 6](#_Toc515215718)

[2.3 Proposed system 8](#_Toc515215719)

[Chapter 3: Methodology 10](#_Toc515215720)

[3.1 Destination and Priority Seats 10](#_Toc515215721)

[3.2 Ticket Price 13](#_Toc515215722)

[Chapter 4: Conclusion and Future Studies 15](#_Toc515215723)

[4.1 Conclusion 15](#_Toc515215724)

[4.2 Future Studies 16](#_Toc515215725)

[References 17](#_Toc515215726)

[Appendices A: Flowchart 19](#_Toc515215727)

[Appendix B: Pseudocode 27](#_Toc515215728)

[Appendix C: Data Dictionary 94](#_Toc515215729)

Abstract

A long distance bus ticket booking system is chosen to be developed. The program will enable the users to buy bus tickets that is within the upcoming months. There will be 3 buses available for each day at different time, all having the same seats layout. There are three departure and arrival locations. The user can freely choose wherever they want to go.

The user will have to enter their IC number before buying the bus ticket, so that the program will calculate the age of the user. If the user is a senior citizen which is 55 years old and older, the ticket fare will be 50% cheaper compared to the regular ticket. There are 8 priority seats for senior citizens in the bus, they can choose their desired seats on the bus. For normal ticket, the users (whose age under 55 years old) can choose their desired seats as well, but only limited to normal seats. The senior citizens can certainly choose to buy normal seats if they do not prefer priority seats.

After the users have chosen their desired seats, the program will calculate the price for the bus ticket and ask the users whether they want to buy the optional travel insurance. Once they made the payment, the program will print out the ticket as proof of payment. As the seat is occupied, the program will update its data and the seat will be red out as well. This program will be displayed with Graphical User Interface (GUI) so that the program will be more user-friendly.

Chapter 1: Introduction

1.1 Introduction

A long distance bus ticket booking system has chosen to be developed after the idea is being approved. Discussion is conducted among the team to identify and understand the concept, objectives and design of the whole program that will be developed. Besides, background studies are conducted as well to understand the pros and cons of the existing bus ticket booking system, in order for the team to apply enhancement to the program that yet to be develop. To make the program more unique and standalone, the team has decided to design a program that will provide priority seats to senior citizens, with condition that a proper IC number must be entered upon buying the ticket. The users are able to choose the seats freely and purchase the optional travel insurance. A survey will be conducted as well to know that which city has the highest demand for long distance bus. The ticket price for different trips will be different as well, where the market ticket price is observed and taken into consideration in this system. As soon as the background investigation are done, the development of the program is on its way. The program will undergo alpha and beta testing process to minimize the errors occur. Future plans and studies for the program will be discussed after the development of the program has completed, to further improve and enhance the program in the future.

1.2 Problem Statement

The existing long distance bus tickets booking program works good, but still there are some enhancement can be made. For example, the existing system did not give senior citizens discount in ticket price, let alone provide the priority seats for the senior citizen in this bus. Some of the elderly passenger are forced to purchase the rear seats since the front seats in the bus has been booked by other passengers. This may cause inconvenience to the elderly passenger as they have to walk further to reach their seat. Other than that, the elderly who sit at rear seats might experience uncomfortable.

Besides that, sometimes, there are often some mistakes made by the tickets counter’s clerk. It’s a common mistake that often perpetrated by the cashiers. From time to time, they tend to return wrong amount of changes back to customer. Either way, they may also calculate the total costs wrongly and the users may have to face the risk of overpay.

In addition, one of the problems that commonly faced by the passengers while buying tickets is that the company tends to include the optional travel insurance without granting the passenger’s permissions. The existing ticket booking system lacked the feature where the passengers can purchase an optional travel insurance.

1.3 Objectives

1. To develop a bus tickets booking program.

The main objective of this project is to develop a working system that enables user to book long distance bus tickets, without end up being in a long queue.

2. To develop a program that provides priority seats for the elderly.

The program developed will include buses with priority seats, offer them more comfortable experience while using the long distance bus service.

3. To develop a program that allows the user to choose whether to include the optional insurance.

The program will ask the user if they want to include an optional travel insurance for each individual passengers, right before the summary of bookings.

4. To develop a program that enables the user to view their past bookings through relogin.

The program enables users to view back their past bookings after relogged in, in case they had lost their tickets and wanted to reprint the tickets.

5. To develop a program that calculates whether the passenger is within the range of discount.

The system developed will ask the passengers to enter their IC number, then their age will be calculated based on the IC entered. If their age is 55 years old and above, the particular passenger will be eligible for selecting those priority seats.

Chapter 2: Background Study

2.1 Reasons and benefits to develop a long distance bus tickets booking system

A long distance bus ticket booking system is chosen to be developed. Nowadays, public transport is used by the society extensively. They tend to rarely buy a car due to economic and environmental pollution issues, especially students. Citizens in our country choose to take public transport because public transports can save fuel and toll fees ([Give Up Driving, Take Public Transport And Save Money - Is It Worth The Trouble?](http://malaysiandigest.com/news/576040-give-up-driving-take-public-transport-and-save-money-is-it-worth-the-trouble.html), 2015). Besides that, they also do not have to drive for a long period. In United States, public transport can also relieve the traffic congestion effectively and efficiently since it helps to reduce the amount of vehicles on the road (Public Transportation Relieves Traffic Congestion, 2011).

However, for those who want to buy the long distance bus ticket, they will need to queue up at the bus terminal’s counter. The queues are long especially during public holiday or cultural events (Bus, train tickets for Chinese New Year sold out, 2018). It was time-consuming when it comes to queue up to buy the tickets. So that, with the long distance bus ticket booking system chosen to be developed, they can buy the bus tickets that they want without queuing up at the counter. Besides that, they can also buy the bus tickets anytime, anywhere. They can save up time rather than spending time making reservations at the counter. This system also reduces traffic in bus terminal where one comes back and forth (Creel, 2017).

Furthermore, the long distance bus ticket booking system chosen to be developed can overcome human errors made by the clerks in the bus counter. The reason is sometimes, there are the mistakes made by the tickets counter’s clerks. They may calculate the total costs wrongly and the users may have to face the risk of overpay. (5 Advantages Of Booking Your Bus Ticket Online, 2016) Hence, this bus ticket booking system can reduce manpower and successfully solve the problems (What are the advantages and disadvantages of online booking systems?, 2018).

2.2 Existing long distance bus ticket booking system in Malaysia

There are two ways to purchase bus ticket in Malaysia, which is purchase bus tickets at the counter or purchase through an online system. For the first way, the customer has to go to the bus terminal’s counter to purchase the bus tickets. There are different bus companies’ counter in most of the bus terminals in Malaysia. They can directly ask for the ticket price and other enquiries over there and then purchase the tickets because there are workers that work at the counter. Most of the time, they have to purchase the bus tickets using cash. At some terminals such as Larkin Terminal in Johor and Terminal Bersepadu Selatan(TBS) in Kuala Lumpur, there are “centralised ticketing counters” where they can purchase the ticket from rather than going to individual bus company’s counter (What You Should Take Note of Before Buying Malaysia Bus Ticket, 2018). It is worth mentioning that the automatic bus ticket reservation system exists in the form of a selection menus alongside with an online system also available in most of the bus terminal in Malaysia.

To purchase the bus ticket through the online booking system, customers can visit the bus companies’ website. There is official website for most of the bus companies. They can choose the bus they prefer. For example, Mayang Sari Express, KKKL Express, Unititi Express and other else. However, there are also existing “centralised ticketing website” such as https://www.easybook.com/en-my and https://eticketing.tbsbts.com.my. These websites filter the best options and enable them to purchase bus tickets without comparing all over the internet. The advantages of online booking systems are that they can purchase the bus tickets anytime, anywhere. So that, they can save their time rather than spending time making reservations at the bus terminal’s counter. The online system can also reduce manpower use and human errors (What are the advantages and disadvantages of online booking systems?, 2018). Besides that, they need to make an online payment in order to purchase the bus tickets. After that, they need to print out the bus tickets by their own and show it to the tellers before board on the bus.

For both existing bus tickets booking system above in Malaysia, the customers can choose freely the destinations that they want to go. Moreover, these systems also provide an optional travel insurance. It allows them to buy the optional insurance for each ride. The insurance is RM4 per one ride. There are several benefits of purchasing this insurance. As an example, first, the insurance company, Chubb Insurance Malaysia will pay RM25,000 per passenger when accidental death or permanent disablement occurred. Second, the insurance company will pay RM2000 per passenger as the accidental medical expenses. Chubb Insurance Malaysia will also pay RM2000 per passenger as the emergency evacuation and repatriation (What is covered by my insurance?, 2018).

2.3 Proposed system

The long distance bus ticket booking system that is chosen to be developed will prepare priority seats for the senior citizen. There are some reasons to develop this system. In Malaysia, although most of the public transport such as bus, train, KLIA Transit, KTM and LRT included the priority seats for the elderly passengers, there are no priority seats prepared in a long distance bus. Furthermore, even the priority seats only prepared for the senior citizen, disabled person and pregnant woman, some of the passengers always ignore this rule (Young Male Ignored Giving Seat To 8 Months Pregnant Lady on Train, 2016). This problem happens to be a worldwide phenomenon, mostly every country is suffering the same problem. For example, in Singapore, a quarrel happened in the MRT. According to an article in The Straits Times (May 12), a young man, Joel Liang who is 23 years old refused to give the priority seat to an old man, Jason Wang who is 63 years old. This caused them to had a conflict on the MRT (Soh, 2011). Since there are no priority seats prepared in a long distance bus and the quarreling for seats need to be avoided, the long distance bus ticket booking system that is chosen to be developed will be enhanced with priority seats that are available for the senior citizen. The priority seats can only be book by the users whose age are above 55 years old. Without a doubt, they had the choice of choosing either the priority seat or the normal seat. No matter what kind of seat they chose, they can also enjoy up to fifty-percent discount on their particular ride, if they entered the I/C number beforehand. Besides that, as in methodology, a survey conducted also showed that most of them voted for the priority seats installation in long distance bus. This shows and proves that priority seats need to be prepared in long distance bus.

Moreover, the long distance bus ticket booking system that is chosen to be developed also allow the users to buy an optional travel insurance. Each of the ticket price will increase by RM4 if the users decide to include the travel insurance. The system that is chosen to be developed may include these following features. User will need to register before start to purchase tickets. The system will require the user to login, so that their previous and future purchase can be stored. In case that the users lost their tickets, they can easily reprint their tickets after they relogined into their account.

There are three departure and arrival location available in the long distance bus ticket booking system that is chosen to be developed. These locations are Melaka, Seremban and Butterworth. These locations are finalised through an online survey which inquired MMU students about the city that most of them frequently depart from and travel to. There are three departure time for buses in proposed system which are 8.00 a.m, 12.00 p.m and 4.00 p.m. These time are also finalised through the online survey conducted which inquired MMU students about the time that most of them frequently depart. Besides that, the proposed system also allows the users to confirm their information before they proceed to retrieve their tickets. The reason for this feature is to avoid the users for entering the wrong information and to allow them to choose the seats that they prefer. The long distance bus ticket booking system described certainly has its market values since it is better that the existing system.

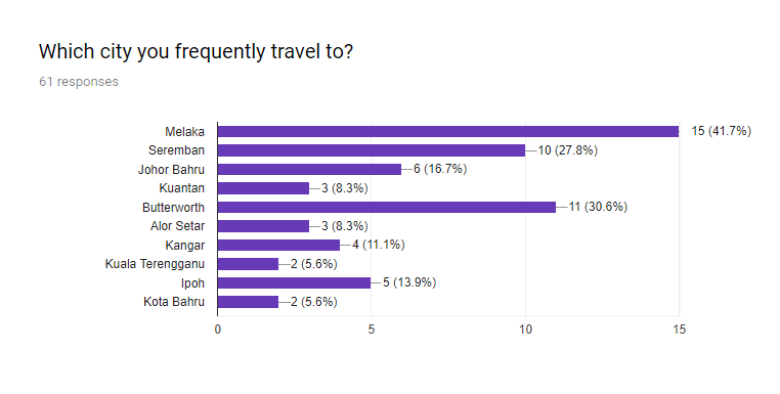
Chapter 3: Methodology

3.1 Destination and Priority Seats

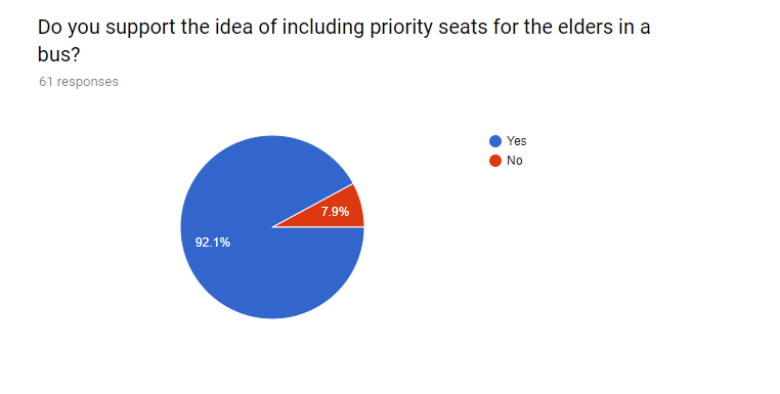
Even though there is an existing bus seat booking system available, but the system did not offer priority seats to the elderly, which is inconvenient to them. So, the program developed aimed to grant the elderly fifty-percent discount on their ticket price. The locations for departure and arrival has been chosen based on an online survey conducted through Google Form. The survey has been given to MMU students that are using the long distance bus service frequently.

The survey conducted included two questions. The first question asked Questions that were asked in the survey included the cities that they frequently travel to. The second question asked the respondents whether they support the idea of adding priority seats to the bus system, which is still yet have been introduced yet in Malaysia.

Question 1: Which city you frequently travel to?

*Figure 3.1 Bar chart for results of Question 1*

Question 1 is asking the respondents about the city they frequently travel to. A few options are given to the respondents to choose from, mostly are the capital cities of each states in peninsular Malaysia. After some time, the three most voted cities will be chosen to be implemented inside the program. Out of the responses obtained from the survey, the city that received the highest votes is Melaka, with 15 respondents (41.7%). The second and third highest cities are Butterworth, with 11 respondents and (30.6%) and Seremban, with 10 respondents which is equivalent to 27.8%. As a result, Melaka, Butterworth and Seremban are implemented inside the program.

Question 2: Do you support the idea of including priority seats for the elders in a bus?

*Figure 3.2 Pie chart for results of Question 2*

Other than Question 1, the respondents are required to answer Question 2 as well. Question 2 is asking the respondents if they support the idea of including priority seats in a long distance bus ticket booking system. The pie chart above has shown that most respondents, 92.1% of them agreed and support the idea of implementing priority seats for the senior citizens in the bus system, while only 7.9% of respondents disagree with the idea. Therefore, the priority seats is decided to be developed and implemented in the program.

3.2 Ticket Price

|  |  |  |  |
| --- | --- | --- | --- |
| **Bus Company** | **Arrival** | **Departure** | **Price (Without Travel Insurance) / RM** |
| Transnasional • Business Club30 | Melaka  (Melaka Sentral) | Negeri Sembilan  (Seremban) | 7.50 |
| Express Kesatuan • SD31 | 8.80 |
| Sri Maju Group • AZRA • SVIP26 | 8.00 |
| Cepat & Cekap Express • 40 SEATERS (CP) | 7.50 |
| KKKL Sdn Bhd • Executive | Penang  (Butterworth) | Melaka  (Melaka Sentral) | 51.00 |
| Transnasional • Business Club30 | 47.70 |
| Cepat & Cekap Express • 30 SEATERS (CK) | 48.00 |
| YK HEE ENTEPRISE • A.CLASS • BW-SBN\_1800 | 55.00 |
| KKKL Sdn Bhd • Executive | Penang  (Butterworth) | Negeri Sembilan (Seremban) | 46.00 |
| Transnasional • Business Club30 | 40.50 |
| YK HEE ENTEPRISE • Jasmine • SVIP\_31 1045 | 50.00 |
| Jasmine Express • SD30S | 48.00 |

*Table 3.1 Bus Ticket Price in Existing System in Malaysia*

Table 3.1 above shows the bus ticket price in existing system in Malaysia. These prices have not included the travel insurance yet. All of the informations in the table are retrieved at May 4, 2018, from Easybook.com Pte Ltd. As shown in the table, from Melaka to Seremban, the bus ticket price is around RM7.50 to RM8.80. So that, average of the price, RM8.00 is set in the proposed system when the users choose departure and arrival as “Melaka to Seremban” or “Seremban to Melaka”. Besides that, from Butterworth to Melaka, the bus ticket price in existing system is around RM47.70 to RM55.00. The average price, RM50.00 is set in the proposed system when the users choose departure and arrival as “Butterworth to Melaka” or “Melaka to Seremban”. As for “Butterworth to Seremban” or “Seremban to Butterworth”, the price in proposed system is set as RM46.00 because the average price in existing system is around RM40.50 to RM50.00.

Chapter 4: Conclusion and Future Studies

4.1 Conclusion

By the end of the project, a bus ticket booking system that satisfied all the objectives has been developed. It took approximately 6 weeks to be developed. In the process, a lot of bug-testing and improvement has been done to ensure the system developed is in best condition as possible. The system has been improved from time to time, not only in design aspect, but also the logic aspect. Survey and research have been done in the process, such as the locations and the market price for long distance bus tickets. This is done so that the system developed is suitable for most of the users. The location chosen to be developed is based on the survey we have conducted, where most respondents among the cities Butterworth, Melaka and Seremban. Besides, the price of bus tickets and optional travel insurance has been adjusted a few times based on the research done to give the users a much more reasonable price.

As a conclusion, a lot of time and efforts are poured into the development of this program, in hope that this system will benefits the public and society, especially the senior citizens, in terms of conveniency. Other than that, the developers hope that the program will help reduce time-consuming queues among the busy cities, reduce the workload of the staffs in the counters.

4.2 Future Studies

In the future, the program will still be getting updates if there are any bugs found, as well as getting enhancement that fulfill the demand among the society. The program can still be enhanced in many ways. For example, implement with a more secure database to store user information rather than store it inside a Json file. As for now, the passengers’ account and data are stored in a Json file and can be modified manually without using the program. Furthermore, the developers felt that the security of the program must be strengthen in order to prevent the account of user being hacked. Besides that, the GUI can be improved so that the program looks way more attractive than now. In addition, the system is still capable of implementing kids’ ticket price and providing discounted seat price for the disabled in the future. With a proper IC entered, the program will calculate the price for children, same goes to the disabled people who have a card for disabilities**.** The developers believed that the program can be implement into an online system, so that users can access the program at anytime, anywhere.

References

*5 Advantages Of Booking Your Bus Ticket Online* (2016, September 16). Retrieved at May 1, 2018, from Princess Juliana International Airport Operating Company NV(PJIAE): <http://www.pjiae.com/5-advantages-of-booking-your-bus-ticket-online/>

*Bus, train tickets for Chinese New Year sold out*. (2018, February 7). Retrieved at April, 30 2018, from New Straits Times: <https://www.nst.com.my/news/nation/2018/02/332885/bus-train-tickets-chinese-new-year-sold-out>

*Butterworth, Penang, Malaysia to Seremban, Negeri Sembilan, Malaysia*. (2018). Retrieved at May, 2 2018, from Easybook.com Pte Ltd.: <https://www.easybook.com/en-my/bus/booking/butterworth-to-seremban>

Creel, B. (2017, August 15). *The Advantages of Booking A bus Ticket Online*. Retrieved at May, 1 2018, from Prime News Inc.: <http://www.enprimerplano.org/the-advantages-of-booking-a-bus-ticket-online/>

[*Give Up Driving, Take Public Transport And Save Money - Is It Worth The Trouble?*  (2015, October 28). Retrieved](http://malaysiandigest.com/news/576040-give-up-driving-take-public-transport-and-save-money-is-it-worth-the-trouble.html) at May 1, 2018, from Malaysian Digest: <http://malaysiandigest.com/news/576040-give-up-driving-take-public-transport-and-save-money-is-it-worth-the-trouble.html>

*Melaka Sentral, Malacca, Malaysia to Butterworth, Penang, Malaysia*. (2018). Retrieved at May, 2 2018, from Easybook.com Pte Ltd.: <https://www.easybook.com/en-my/bus/booking/melakasentral-to-butterworth>

*Melaka Sentral, Malacca, Malaysia to Seremban, Negeri Sembilan, Malaysia*. (2018). Retrieved at May 4, 2018, from Easybook.com Pte Ltd: <https://www.easybook.com/en-my/bus/booking/melakasentral-to-seremban>

*Public Transportation Relieves Traffic Congestion* (2011, January 20). Retrieved at May 1, 2018, from American Public Transportation Association: <http://www.apta.com/mediacenter/pressreleases/2011/Pages/112001_TTI_Report.aspx>

Soh, M. (2011, May 13). *Should reserved seats in public transport be exclusive?* Retrieved at May, 1 2018, from Youth.SG. : <https://www.youth.sg/Our-Voice/Your-Take/2011/5/Should-reserved-seats-in-public-transport-be-exclusive>

*What are the advantages and disadvantages of online booking systems?* (2018, January 15). Retrieved at May, 1 2018, from Quora: <https://www.quora.com/What-are-the-advantages-and-disadvantages-of-online-booking-systems>

*What is covered by my insurance?* (2018). Retrieved at May, 1 2018, from Chubb Corporation: <https://www.chubbtravelinsurance.com.my/en/learn/our-products/what-is-covered>

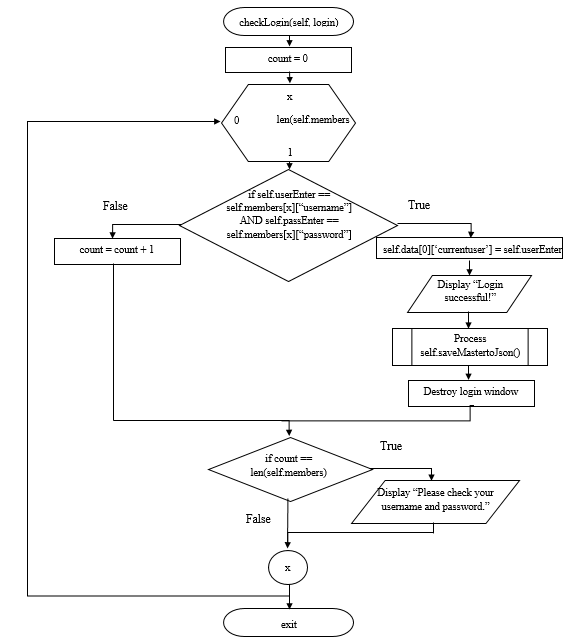
*What You Should Take Note of Before Buying Malaysia Bus Ticket* (2018, February 27). Retrieved at May 1, 2018, from Trevallog: <https://trevallog.com/take-note-before-buying-malaysia-bus-ticket/>

*Young Male Ignored Giving Seat To 8 Months Pregnant Lady on Train*. (2016, March 21). Retrieved at May 1, 2018, from World of Buzz: <https://www.worldofbuzz.com/young-male-ignored-giving-seat-to-8-months-pregnant-lady-on-train/>

**Appendices A: Flowchart**

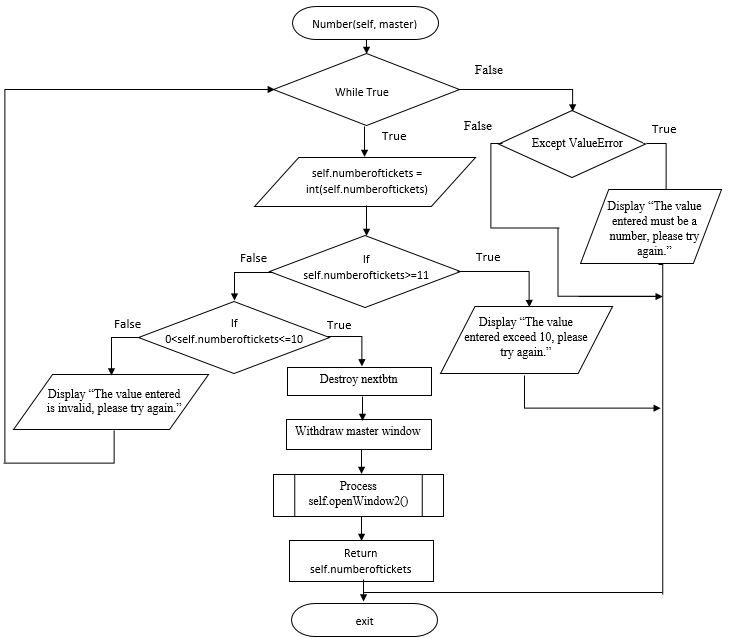
Flowcharts of Modules

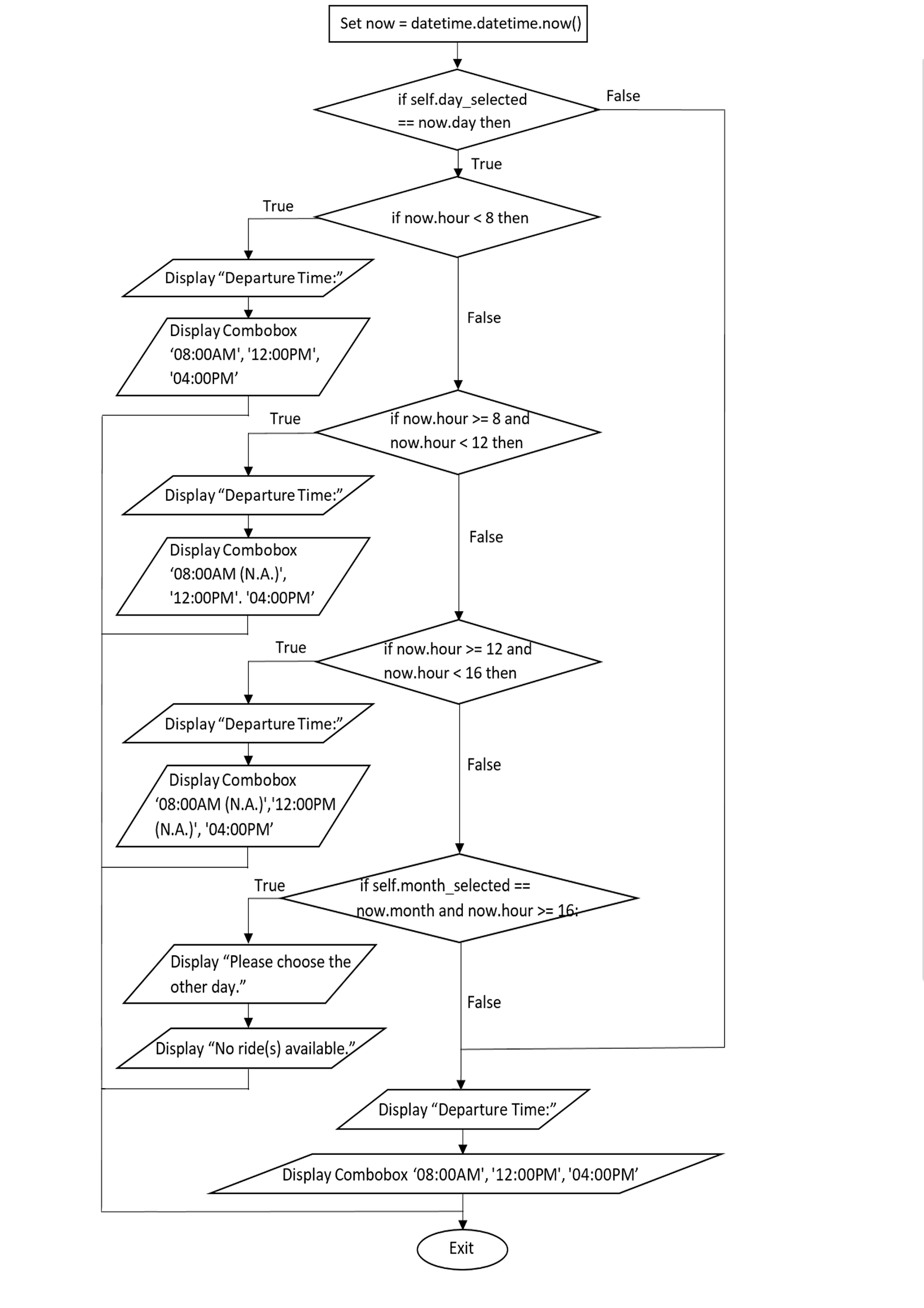
*Login Module*

**checkLogin(self, login)**

*Startup Module*

**number(self, master)**

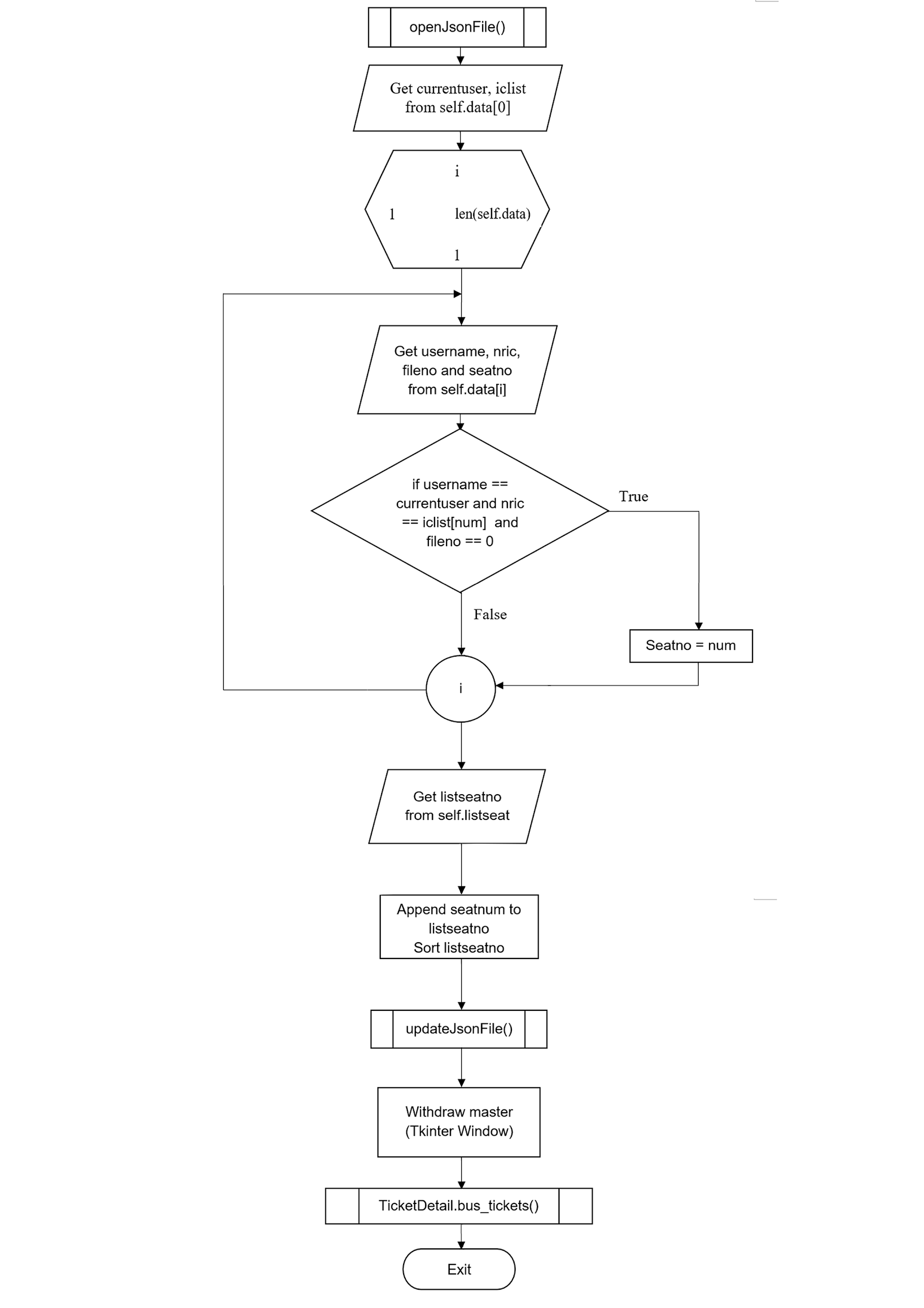


*Calendar Module*

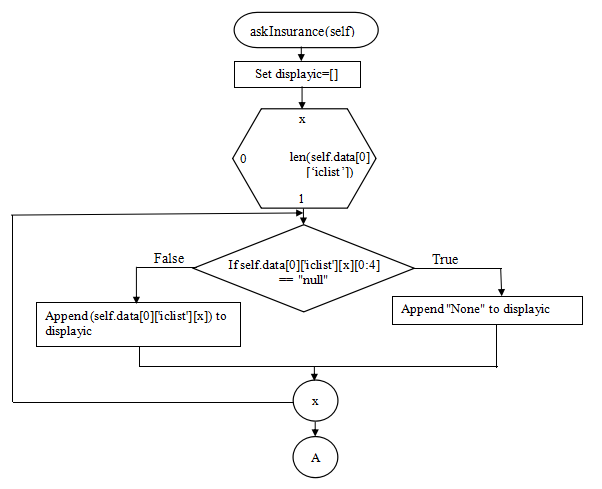
**departtimecombobox(self)**

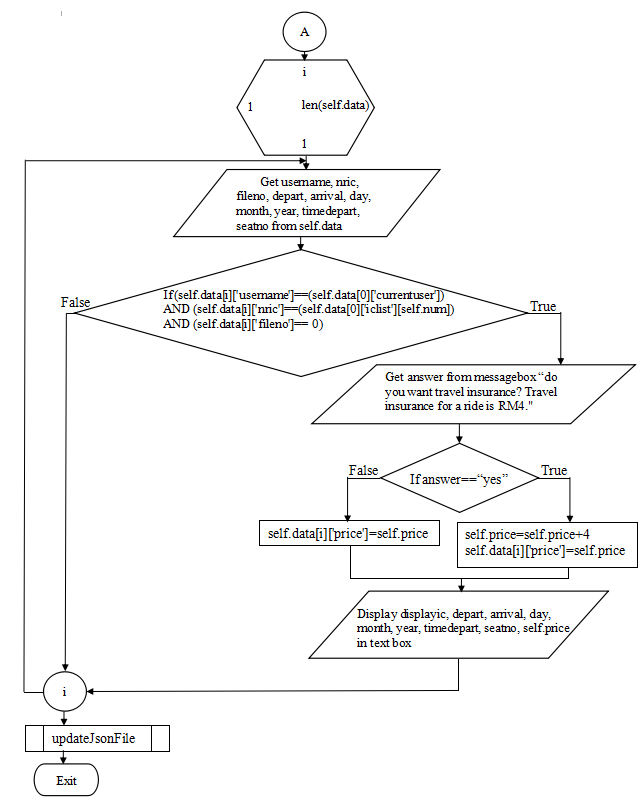
*SelectSeat Module*

**confirmbutton(self , master, seatnum)**

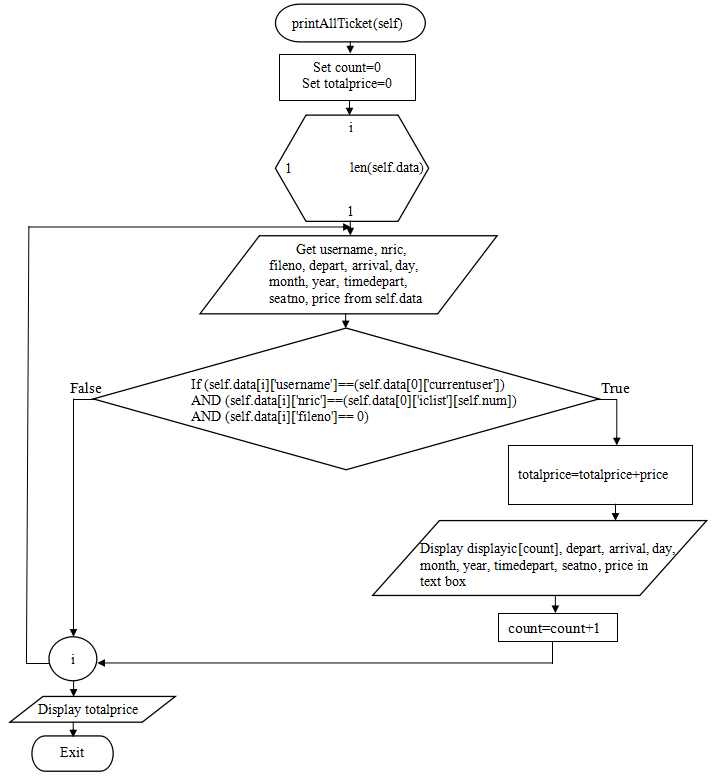


*TicketDetail Module*

**askInsurance(self)**



**printAllTicket(self)**



**Appendix B: Pseudocode**

**Pseudocode of all Modules**

***Main Module***

Import Login

Import StartUp

Import Calendar

if \_\_name\_\_ == '\_\_main\_\_' :

Process Login.loginRegister()

Process StartUp.startUp()

Process Calendar.Calendar()

***Login Module***

Import Json, OS

From tkinter import all(\*), messagebox

**Class loginRegister**

**\_\_init\_\_(self)**

Process self.openJsonFile()

Create login = Tk() as Tkinter window

Set window width to 500

Set window height to 400

Create self.userEnter as Username Entrybox

Create self.passEnter as Password Entrybox

Create self.passwordvbtn button with command: loginpasswordhideimg()

Create registerButton with command: openRegisterWindow(\*login)

Create loginButton with command: checkLogin(\*login)

Create exitButton with command: exit()

End

**openJsonFile(self)**

self.members = load member.json file

self.data= load Master.json file   
Return self.members, self.data

Exit

**enterclick(self, event=None)**

Process self.checkLogin(\*login)

**loginpasswordhideimg(self)**

Configure self.passwordvbtn button with command: loginpasswordviewimg()

Exit

**loginpasswordviewimg(self)**

Configure self.passwordvbtn button with command: loginpasswordhideimg()

Exit

**registerWindow(self, login)**

Create register = Toplevel() as Tkinter window

Set window width to 500

Set window height to 400

Create self.userEntry as Username Registration Entrybox

Create self.passEntry as Password Registration Entrybox

Create self.passwordvbtn button with command: registerpasswordhideimg()

Create self.passConfirm as Password Confirmation Entrybox

Create submitbtn button with command: registration(\*login, \*register)

Create backbtn button with command: openLoginPage(\*register, \*login)

Exit

**enterregclick(self, event)**

Process self.registration(\*login, \*register)

Exit

**registerpasswordhideimg(self)**

Configure self.passwordvbtn button with command: registerpasswordviewimg()

Exit

**registerpasswordviewimg(self)**

Configure self.passwordvbtn button with command: registerpasswordhideimg()

Exit

**openRegisterWindow(self, login)**

Withdraw login window

Process self.registerWindow(\*login)

Exit

**openLoginPage(self, registration, login)**

Destroy registration(Tkinter window)

Redraw login window

Exit

**checkLogin(self, login)**

count = 0

Loop: x = 0 to len(self.members) step 1

if self.userEnter == self.members[x][“username”] AND self.passEnter == self.members[x][“password”] then

self.data[0][‘currentuser’] = self.userEnter

Display “Login successful!” messagebox

Process self.saveMastertoJson()

Destroy login (Tkinter Window)

else

count = count + 1

endif

if count == len(self.members) then

Display “Please check your username and password” message

endif

Loop-End: x

Exit

**registration(self, login, register)**

count = 0

notmatch = 0

if len(self.userEntry) >= 9 then

Display “Username length cannot be more than 8 characters.” message

Return

endif

if len(self.userEntry) < 1 then

Display “Username cannot be blank.” message

Return

endif

if len(self.passEntry) <1 then

Display "Password cannot be blank." message

Return

endif

if self.passConfirm != self.passEntry then

Display “Both passwords did not match!” message

Return

endif

if len(self.members) == 0 then

member = {}

member[“username”] = self.userEntry

member[“password”] = self.passEntry

Append member to members

Display “You have been registered successfully!” message

Process self.saveMembertoJson(\*register, \*login)

Return

Loop: i = 0 to len(self.members) step 1

if self.userEntry == self.members[i][“username”] then

count = count + 1

endif

if self.userEntry != self.members[i][“username”] then

notmatch = notmatch + 1

endif

if self.userEntry != self.members[i][“username”] and notmatch == len(self.members) then

member = {}

member[“username”] = self.userEntry

member[“password”] = self.passEntry

Append member to members

Display “You have been registered successfully!” message

Process self.saveMembertoJson(\*register, \*login)

Return

endif

if count >= 1 then

Display “Username has been taken, please enter a new one.” message

Return

endif

Loop-End: i

Exit

**createFolder(self, register, login)**

Create folder with name self.userEntry

Exit

**saveMembertoJson(self, register, login)**

Save self.members to member.json file

Process createFolder(\*register, \*login)

Exit

**saveMastertoJson(self)**

Save self.data to Master.json file

Exit

***Startup module***

Import Copy, Json, history, collections, calendar, os

From tkinter import all(\*)

From tkinter import messagebox

From tkinter.ttk import Combobox

**Class startUp**

**\_\_init\_\_(self)**

self.z = 0

self.openJsonFile()

self.openWindow1()

End

**donothing(self)**

pass

Exit

**openJsonFile(self)**

self.data = load Master.json file

Return self.data

Exit

**openWindow1(self)**

global master

self.CurrentUser = self.data[0][‘currentuser’]

Create master = Tk() as Tkinter window

Set window width to 500

Set window height to 400

Display “Please enter the number of ticket you would like to purchase.”

Create self.numoftickets as Number of Tickets Entrybox

Create self.history button with command: history.LostTicket()

Create self.nextbtn button with command: number(\*master)

Create logoutButton button with command: exit()

Exit

**entertowindow2(self, event)**

Process self.number(\*master)

Exit

**number(self, master)**

While true

try

self.numberoftickets = int(self.numoftickets)

if self.numberoftickets >= 11 then

Display “The value entered exceed 10, please try again.” messagebox

Return

endif

if 0 < self.numberoftickets <= 10 then

Destroy nextbtn

Withdraw master window

Process self.openWindow2()

Return self.numberoftickets

else

Display “The value entered is invalid, please try again.”

endif

except ValueError

Display “The value entered must be a number, please try again.”

Return

WhileEnd

Exit

**openWindow2(self)**

self.z = self.z + 1

Create window2 = Toplevel() as Tkinter window

Set window width to 500

Set window height to 400

Display “Please enter the IC of the passengers.”

Process self.NricEntry(window2)

Create self.nextbtn button with command: plusNricEntry(\*window2)

Exit

**entertowindow3(self, event)**

Process self.plusNricEntry(\*window2)

Exit

**NricEntry(self, window2)**

self.ic = []

Create self.minusButton button with command: minusEntry(window2)

Create self.plusButton button with command: plusEntry(window2)

if self.numberoftickets == 1 then

Create self.entry1 as IC Entry 1

endif

if self.numberoftickets == 2 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

endif

if self.numberoftickets == 3 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

endif

if self.numberoftickets == 4 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

endif

if self.numberoftickets == 5 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

endif

if self.numberoftickets == 6 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

Create self.entry6 as IC Entry 6

endif

if self.numberoftickets == 7 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

Create self.entry6 as IC Entry 6

Create self.entry7 as IC Entry 7

endif

if self.numberoftickets == 8 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

Create self.entry6 as IC Entry 6

Create self.entry7 as IC Entry 7

Create self.entry8 as IC Entry 8

endif

if self.numberoftickets == 9 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

Create self.entry6 as IC Entry 6

Create self.entry7 as IC Entry 7

Create self.entry8 as IC Entry 8

Create self.entry9 as IC Entry 9

endif

if self.numberoftickets == 10 then

Create self.entry1 as IC Entry 1

Create self.entry2 as IC Entry 2

Create self.entry3 as IC Entry 3

Create self.entry4 as IC Entry 4

Create self.entry5 as IC Entry 5

Create self.entry6 as IC Entry 6

Create self.entry7 as IC Entry 7

Create self.entry8 as IC Entry 8

Create self.entry9 as IC Entry 9

Create self.entry10 as IC Entry 10

endif

Exit

**minusEntry(self, window2)**

if self.numberoftickets > 10 then

self.numberoftickets == 10

endif

if 1 < self.numberoftickets < 11 then

self.numberoftickets = self.numberoftickets - 1

endif

if self.numberoftickets == 1 then

Destroy self.entry2 Entrybox

endif

if self.numberoftickets == 2 then

Destroy self.entry3 Entrybox

endif

if self.numberoftickets == 3 then

Destroy self.entry4 Entrybox

endif

if self.numberoftickets == 4 then

Destroy self.entry5 Entrybox

endif

if self.numberoftickets == 5 then

Destroy self.entry6 Entrybox

endif

if self.numberoftickets == 6 then

Destroy self.entry7 Entrybox

endif

if self.numberoftickets == 7 then

Destroy self.entry8 Entrybox

endif

if self.numberoftickets == 9 then

Destroy self.entry10 Entrybox

endif

if self.numberoftickets == 10 then

Destroy self.entry10 Entrybox

endif

Exit

**plusEntry(self, window2)**

if self.numberoftickets > 10 then

self.numberoftickets == 10

endif

if self.numberoftickets == 1 then

Create self.entry2 as IC Entry 2

endif

if self.numberoftickets == 2 then

Create self.entry3 as IC Entry 3

endif

if self.numberoftickets == 3 then

Create self.entry4 as IC Entry 4

endif

if self.numberoftickets == 4 then

Create self.entry5 as IC Entry 5

endif

if self.numberoftickets == 5 then

Create self.entry6 as IC Entry 6

endif

if self.numberoftickets == 6 then

Create self.entry7 as IC Entry 7

endif

if self.numberoftickets == 7 then

Create self.entry8 as IC Entry 8

endif

if self.numberoftickets == 8 then

Create self.entry9 as IC Entry 9

endif

if self.numberoftickets == 9 then

Create self.entry10 as IC Entry 10

endif

if 0 < self.numberoftickets < 10 then

self.numberoftickets = self.numberoftickets + 1

endif

Exit

**plusNricEntry(self, window2)**

if self.numberoftickets == 1 then

Append self.entry1 to self.ic

endif

if self.numberoftickets == 2 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

endif

if self.numberoftickets == 3 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

endif

if self.numberoftickets == 4 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

endif

if self.numberoftickets == 5 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

endif

if self.numberoftickets == 6 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

Append self.entry6 to self.ic

endif

if self.numberoftickets == 7 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

Append self.entry6 to self.ic

Append self.entry7 to self.ic

endif

if self.numberoftickets == 8 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

Append self.entry6 to self.ic

Append self.entry7 to self.ic

Append self.entry8 to self.ic

endif

if self.numberoftickets == 9 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

Append self.entry6 to self.ic

Append self.entry7 to self.ic

Append self.entry8 to self.ic

Append self.entry9 to self.ic

endif

if self.numberoftickets == 10 then

Append self.entry1 to self.ic

Append self.entry2 to self.ic

Append self.entry3 to self.ic

Append self.entry4 to self.ic

Append self.entry5 to self.ic

Append self.entry6 to self.ic

Append self.entry7 to self.ic

Append self.entry8 to self.ic

Append self.entry9 to self.ic

Append self.entry10 to self.ic

endif

Process self.confirmation(\*window2)

**confirmation(self, window2)**

blanknum = 0

times = 1

Minimize window2 Tkinter window

Copy self.ic to self.newic

self.yearlist = []

self.monthlist = []

self.daylist = []

self.iclist = []

While “” in self.newic

Remove “” in self.newic

WhileEnd

Loop: s = 0 to len(self.newic) step 1

if len(set(self.newic) != len(self.newic) then

Display “There are duplicate IC entered!” messagebox

self.ic = []

self.newic = []

Redraw window2 Tkinter window

Return

Loop-End: s

Loop: i = 0 to len(self.ic) step 1

if self.ic[i] == “” then

blanknum = blanknum + 1

endif

Loop-End: i

Loop: x = 0 to len(self.ic) step 1

if self.ic[x] != “” then

try

check1 = int(self.ic[x])

except ValueError

Display “The IC entered must be digits!” messagebox

Redraw window2 Tkinter window

self.ic = []

self.newic = []

Return

endif

Loop-End: x

Loop: c = 0 to len(self.ic) step 1

if self.ic[c] == “” then

continue

endif

if len(self.ic[c]) != 12 then

Display “The IC entered is invalid!” messagebox

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

endif

Loop-End: c

if blanknum > 0 then

if blanknum > 1 then

verb = “are”

endif

if blanknum == 1 then

verb = “is”

endif

Get confirmation from messagebox “‘blanknum’ entry ‘verb’ blank, do you wish to continue? (Ticket without inserting IC will not getting priority seat!”

if confirmation == True then

Loop: a = 0 to len(self.ic) step 1

if self.ic[a] == “” then

self.ic[a] = “null\_‘times’”

times = times + 1

endif

Loop-End: a

endif

if confirmation == False then

Redraw window2 Tkinter window

self.ic = []

self.newic = []

Return

endif

Loop: f = 0 to len(self.ic) step 1

Append 3rd and 4th digit from each object from self.ic to self.monthlist

Loop-End: f

Loop: h = 0 to len(self.ic) step 1

if self.ic[h][0:4] == “null” then #check first 4 characters

Append 1 to self.daylist

else

Append the 5th and 6th digits from each object from self.ic to self.daylist

endif

Loop-End: h

Loop: g = 0 to len(self.monthlist) step 1

if str(self.monthlist[g] == “ll” then

continue

else

if int(self.monthlist[g]) >= 13 OR int(self.monthlist[g]) <=0 then

Display “The IC entered is invalid!” messagebox

self.daylist = []

self.monthlist = []

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

endif

endif

Loop: j = 0 to len(self.daylist) step 1

if int(self.monthlist[g]) == 1 OR int(self.monthlist[g]) == 3 OR int(self.monthlist[g]) == 5 OR int(self.monthlist[g]) == 7 OR int(self.monthlist[g]) == 8 OR int(self.monthlist[g]) == 10 OR int(self.monthlist[g]) == 12 then

if int(self.daylist[j]) >= 32 OR int(self.daylist[j]) <=0 then

Display “The IC entered is invalid!” messagebox

self.daylist = []

self.monthlist = []

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

endif

else

if int(self.daylist[j]) >= 31 OR int(self.daylist[j]) <=0 then

Display “The IC entered is invalid!” messagebox

self.daylist = []

self.monthlist = []

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

endif

endif

Loop-End: j

Loop-End: g

Loop: b = 0 to len(self.ic) step 1

Append 1st and 2nd digits from each object from self.ic to self.iclist

Loop-End: b

Loop: p = 0 to len(self.iclist) step 1

if str(self.iclist[p]) == “nu” then

Append “nu” to self.yearlist

else

if int(self.iclist[p]) >= 18 then

Append (1900 + int(self.iclist[p])) to self.yearlist

else

Append (2000 + int(self.iclist[p])) to self.yearlist

endif

endif

Loop-End: p

Loop: r = 0 to len(self.yearlist) step 1

if str(self.yearlist[r] == “nu” then

continue

endif

if calendar.isleap(self.yearlist[r]) == True then

if int(self.monthlist[r]) == 2 then

if 1 <= int(self.daylist[r]) < 30 then

continue

endif

if int(self.daylist[r] >= 30 then

Display “The IC entered is invalid!” messagebox

self.daylist = []

self.monthlist = []

self.yearlist = []

self.iclist = []

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

else

continue

else

if int(self.monthlist[r]) == 2 then

if 1 <= int(self.daylist[r] <= 28 then

continue

endif

if int(self.daylist[r]) > 28 then

Display “The IC entered is invalid!”

self.daylist = []

self.monthlist = []

self.yearlist = []

self.iclist = []

self.newic = []

self.ic = []

Redraw window2 Tkinter window

Return

endif

endif

endif

Loop-End: r

Process checkPriority(\*window2)

Exit

**checkPriority(self, window2)**

self.age = []

self.priority = []

Loop: e = 0 to len(self.yearlist) step 1

if str(self.yearlist[e]) == “nu” then

Append 0 to self.age

else

int(self.yearlist[e]) >= 2018 then

Append (2018 - int(self.yearlist[e])) to self.age

endif

Loop-End: e

Loop: z = 0 to len(self.age) step 1

if int(self.age[z]) >= 55 then

Append 1 to self.priority

else

Append 0 to self.priority

endif

Loop-End: z

Destroy nextbtn button

Process openWindow3(\*window2)

Exit

**openWindow3(self, window2)**

global window3

Destroy window2(Tkinter Window)

Create window3 = Toplevel() as Tkinter window

Set window width to 500

Set window height to 400

Display “Please select departure and arrival city for all the passengers.”

Create nextThreeButton button with command: confirm()

Display “Please select your departure city.”

self.selection1 = Display Combobox “Butterworth”, “Melaka”, “Seremban”

Display “Please select your arrival city.”

self.selection2 = Display Combobox “Butterworth”, “Melaka”, “Seremban”

Exit

**enterclicked(self, event)**

Process self.confirm()

**confirm(self)**

self.depart = []

self.arrival = []

if self.selection1 == “” OR self.selection2 == “” then

Display “The departure city or arrival city cannot be blank!” messagebox

Return

endif

if self.selection1 != self.selection2 then

Display “The departure city and arrival city has been confirmed!” messagebox

else

Display “The departure city and arrival city cannot be the same!” messagebox

Return

endif

self.depart = self.selection1

self. arrival = self.selection2

Process savePassengerToFile()

Exit

**savePassengerToFile(self)**

Loop: k = 0 to len(self.ic) step 1

passengerdata = {}

passengerdata['username'] = self.CurrentUser

passengerdata['datepurchased']= ""

passengerdata['nric' ] = self.ic[k]

passengerdata['age'] = self.age[k]

passengerdata['depart'] = self.depart

passengerdata['arrival'] = self.arrival

passengerdata['priority'] = self.priority[k]

passengerdata['year'] =""

passengerdata['month'] = ""

passengerdata['day'] = ""

passengerdata['timedepart'] = ""

passengerdata['seatno'] = ""

passengerdata["price"]= 0

passengerdata["fileno"]=0

Append passengerdata to self.data

Loop-End: k

self.data[0]['iclist']=self.ic

self.data[0]['num']=0

self.data[0]['depart']= self.depart

self.data[0]['arrival']=self.arrival

self.data[0]['prioritylist']=self.priority

Process updateJsonFile()

Destroy window3(Tkinter Window)

Destroy master(Tkinter Window)

Exit

**updateJsonFile(self)**

Save self.data to Master.json

Exit

***History Module***From tkinter import all(\*)  
From os import walk

Import os , json

**Class LostTicket**  
  
**\_\_init\_\_(self)**Start

Global master

Process openJsonFile()

Get currentuser from self.data[0]

self.CurrentUser= currentuser

Create master=Toplevel() as Tkinter Window  
Set window height to 400  
Set window width to 500

Create backbtn button with command : Close master(Tkinter WIndow)

Process readText(\*master)  
Process DisplayHistory()  
End

**openJsonFile(self)**

self.data= load Master.json file   
Return self.data

Exit

**DisplayHistory(self)**

Set self.frames=[]

Set self.buttons=[]  
If len(self.listfiles) > 4 then

Create nextpagebtn button with command : Process displaynextpage()

Set number = 4  
Else

Set number = len(self.listfiles)  
endif

Loop : f = 0 to number step 1  
 Create Fbutton=Frame(master)

Append Fbutton to self.frames

Loop-End : f

Loop : i = 0 to number step 1   
 Create historybtn in self.frames[i] with command : lambda j = i : Process aaaaaasdisplaytextbox(\*j, \*master)

Append historybtn to self.buttons

Loop-End : i

Exit

**readtext(self)**

Set (self.listfiles, self.namefiles, self.datepurchase, self.filenum, self.timedepart, self.destination, self.seats) = []

Get filenames from file directory ('Desktop\\ITProject\\Users',self.CurrentUser)

Append filenames to self.listfiles

If self.listfiles == []:  
 Display ‘Sorry! No record found!’

Return  
Endif

Loop : x = 0 to len(self.listfiles) step 1

count=filenamenew[-1]

Append count to self.filenum

filenamenew = Replace “.txt” with “” from self.listfiles[x]  
 filenamenew2 = Replace “\_[self.CurrentUser]\_[count]” with “” from filenamenew

Append filenamenew2 to self.namefiles

date = Replace “\_” with “/” from filenamenew2  
 Append date to self.datepurchase

Loop-End : x  
Loop : num = 0 to len(self.listfiles) step 1  
 numseat=0

Loop : z = 1 to len(self.data) step 1

Get username, datepurchased and fileno from self.data[z]  
 If username == self.Currentuser and datepurchased == self.namefiles[num] and fileno fileno == int(self.filenum[num]) then

Numseat + = 1  
 Timedeparture = Get timedepart from self.data[z]

Arrival = Get arrival from self.data[z]

Loop-End : z

Append numseat to self.seats  
 Append timedeparture to self.timedepart  
 Append arrival to self.destination

Loop-End : num

Loop : c = 0 to len(self.destination) step 1  
 if self.destination[c] == “Melaka” then

Self.destination[c] = “MEL”

endif

if self.destination[c] == “Seremban” then

Self.destination[c] = “SRM”

endif

if self.destination[c] == “Butterworth” then

Self.destination[c] = “BUT.”

endif

Loop-End : c

Exit

**self.backpagebutton(self,mater)**  
Destroy backpagebtn button  
Process DisplayHistory()  
Exit

**displaynextpage(self, master)**

Destroy self.buttons[0] , self.buttons[1], self.buttons[2], self.buttons[3]  
Create backpagebtn button with command : Process backpagebutton()

Loop : i = 4 to len(self.listfiles) step 1

Create historybtn in self.frames[i-4] with command : lambda j = i :Process aaaaaaadisplaytextbox(\*j)

Append historybtn to self.buttons  
Loop-End : i   
Exit  
  
**displaytextbox(self, j)**Close master(Tkinter window)  
Create detail=Toplevel() as Tkinter window

Set window height to 400  
Set window width to 500

Create Scrollbar on right side at Y- axis   
F\_contents = Read [self.listfiles[j]].txt file from file directory ('Desktop\\ITProject\\Users',self.CurrentUser)  
Display F\_contents in Textbox  
Create backclosetextbtn button with command : Process closetextwindow()  
Exit

**closetextwindow(self,detail)**

Close detail(Tkinter window)

Process \_\_init\_\_()  
Exit

***Calendar Module***Import json , Calendar , TicketDetail   
From tkinter import all(\*)

From tkinter import messagebox

From tkinter import Combobox

From datetime import date

**Class Calendar  
callbackCalendar()**

Redraw root(Tkinter window)

**\_\_init\_\_(self)**

Start

Process loadfile()

Global root

Set self.data = {}

now = datetime.datetime.now()

Set self.cfm = 0

Set self.wid = []  
Set self.day\_selected = now.day  
Set self.month\_selected = self.month  
Set self.year\_selected = self.year  
Process clocktitle(\*self)

Process tdydate(\*self)

Process clocktimer(\*self)

Process calendarselection(\*self)

Create root = Tk() as Tkinter window   
Set window height to 400  
Set window width to 500  
End

**loadfile(self)**self.filedata = Load Master.json file   
Return

Exit

**clocktitle()**

Display “Current Time”

Exit

**clickedconfirm(self)**

self.cfm += 1

Global self.timedepart.get()

if self.timedepart.get() == '08:00AM (N.A.)' then

Display "This ride(08:00AM) is not available. Please choose other available ride(s)."

message

self.cfm = 0

Display “Time Selected” + “ ”

endif

if self.timedepart.get() == '12:00PM (N.A.)' then

Display "This ride(12:00PM) is not available. Please choose other available ride(s)."

message

self.cfm = 0

Display “Time Selected” + “ ”

endif

if self.timedepart.get() == '08:00AM' or self.timedepart.get() == '12:00PM' or self.timedepart.get() == '04:00PM' then

Display “Time Selected” + self.timedepart.get()

endif

exit

**tdydate(self)**

todaydate = date.today()

Display “Today’s Date”

currentdate = todaydate

Display currentdate

exit

**clocktimer(self)**

Display current time from system

Process self.refreshcombobox()

exit

**calendarselection(self)**

Global selectedtime  
Display current month and year selected from system

days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

Loop : z = 0 to len(days) step 1

Display days[z][:3]

Loop-End : z   
Loop : w = 0 to len() step 1

Loop : d = 0 to len(week) step 1

if day then  
 Create buttonspawn = frame(root)  
 Append buttonspawn to self.frame

endif  
 Loop-End : d

Loop-End : w

Process self.disablebuttons()

Display “Date Selected: [self.day\_selected] [self.month\_selected] [self.year\_selected]”

Display "Time Selected:" + " "

Process self.departtimecombobox()

exit

**refreshcombobox(self)**

currenttime = current time from system

if (currenttime == '08:00:00') or (currenttime == '12:00:00') or (currenttime == '16:00:00') then

Process self.departtimecombobox()  
 Display "Time Selected:" + " "

endif

exit

**departtimecombobox(self)**

Set now = datetime.datetime.now()

if self.day\_selected == now.day then

if now.hour < 8 then

Display “Departure Time:”

Display Combobox ‘08:00AM', '12:00PM', '04:00PM’

endif

if now.hour >= 8 and now.hour < 12 then

Display “Departure Time:”

Display Combobox ‘08:00AM (N.A)', '12:00PM', '04:00PM’

endif

if now.hour >= 12 and now.hour < 16 then

Display “Departure Time:”

Display Combobox ‘08:00AM (N.A.)', '12:00PM (N.A.)', '04:00PM’

endif

if self.month\_selected == now.month and now.hour >= 16:

Display “Please choose the other day.”

Display “No ride(s) available.”

endif

Else

Display “Departure Time:”

Display Combobox ‘08:00AM', '12:00PM', '04:00PM’

Endif

Exit

**disablebuttons(self)**

if now.year > self.year then

Disable daybutton[0:27]

if self.month == 2 then

if calendar.isleap(self.year) == True then

Disable daybutton[28]

endif

endif

if (self.month == 4) or (self.month == 6) or (self.month == 9) or (self.month == 11) then

Disable daybutton[28:29]

endif

if (self.month == 1) or (self.month == 3) or (self.month == 5) or (self.month == 7) or

(self.month == 8) or (self.month == 10) or (self.month == 12) then

Disable daybutton[28:30]

endif

endif

if now.year == self.year then

if (now.month > self.month) then

Disable daybutton[0:27]

if self.month == 2 then

if calendar.isleap(self.year) == True then

Disable daybutton[28]

endif

endif

if (self.month == 4) or (self.month == 6) or (self.month == 9) or (self.month == 11) then

Disable daybutton[28:29]

endif

if (self.month == 1) or (self.month == 3) or (self.month == 5) or (self.month == 7) or

(self.month == 8) or (self.month == 10) or (self.month == 12) then

Disable daybutton[28:30]

endif

endif

if now.month == self.month then

if now.day == 2 then

Disable daybutton[0]

endif

if now.day == 3 then

Disable daybutton[0:1]

endif

if now.day == 4 then

Disable daybutton[0:2]

endif

if now.day == 5 then

Disable daybutton[0:3]

endif

if now.day == 6 then

Disable daybutton[0:4]

endif

if now.day == 7 then

Disable daybutton[0:5]

endif

if now.day == 8 then

Disable daybutton[0:6]

endif

if now.day == 9 then

Disable daybutton[0:7]

endif

if now.day == 10 then

Disable daybutton[0:8]

endif

if now.day == 11 then

Disable daybutton[0:9]

endif

if now.day == 12 then

Disable daybutton[0:10]

endif

if now.day == 13 then

Disable daybutton[0:11]

endif

if now.day == 14 then

Disable daybutton[0:12]

endif

if now.day == 15 then

Disable daybutton[0:13]

endif

if now.day == 16 then

Disable daybutton[0:14]

endif

if now.day == 17 then

Disable daybutton[0:15]

endif

if now.day == 18 then

Disable daybutton[0:16]

endif

if now.day == 19 then

Disable daybutton[0:17]

endif

if now.day == 20 then

Disable daybutton[0:18]

endif

if now.day == 21 then

Disable daybutton[0:19]

endif

if now.day == 22 then

Disable daybutton[0:20]

endif

if now.day == 23 then

Disable daybutton[0:21]

endif

if now.day == 24 then

Disable daybutton[0:22]

endif

if now.day == 25 then

Disable daybutton[0:23]

endif

if now.day == 26 then

Disable daybutton[0:24]

endif

if now.day == 27 then

Disable daybutton[0:25]

endif

if now.day == 28 then

Disable daybutton[0:26]

endif

if now.day == 29 then

Disable daybutton[0:27]

endif

if now.day == 30 then

Disable daybutton[0:28]

endif

if now.day == 31 then

Disable daybutton[0:29]

endif

endif

exit

**changevaluetostring(self)**

Global timedepart

if (self.day\_selected == now.day) and (now.hour > 16) then

Display "This date([self.month\_selected] [self.day\_selected] [self.year\_selected]) is not

available. Please choose other available date(s)."

else

self.num = int(self.filedata[0]['num'])

endif

if self.day\_selected == 1 then

self.day = "1"

endif

if self.day\_selected == 2 then

self.day = "2"

endif

if self.day\_selected == 3 then

self.day = "3"

endif

if self.day\_selected == 4 then

self.day = "4"

endif

if self.day\_selected == 5 then

self.day = "5"

endif

if self.day\_selected == 6 then

self.day = "6"

endif

if self.day\_selected == 7 then

self.day = "7"

endif

if self.day\_selected == 8 then

self.day = "8"

endif

if self.day\_selected == 9 then

self.day = "9"

endif

if self.day\_selected == 10 then

self.day = "10"

endif

if self.day\_selected == 11 then

self.day = "11"

endif

if self.day\_selected == 12 then

self.day = "12"

endif

if self.day\_selected == 13 then

self.day = "13"

endif

if self.day\_selected == 14 then

self.day = "14"

endif

if self.day\_selected == 15 then

self.day = "15"

endif

if self.day\_selected == 16 then

self.day = "16"

endif

if self.day\_selected == 17 then

self.day = "17"

endif

if self.day\_selected == 18 then

self.day = "18"

endif

if self.day\_selected == 19 then

self.day = "19"

endif

if self.day\_selected == 20 then

self.day = "20"

endif

if self.day\_selected == 21 then

self.day = "21"

endif

if self.day\_selected == 22 then

self.day = "22"

endif

if self.day\_selected == 23 then

self.day = "23"

endif

if self.day\_selected == 24 then

self.day = "24"

endif

if self.day\_selected == 25 then

self.day = "25"

endif

if self.day\_selected == 26 then

self.day = "26"

endif

if self.day\_selected == 27 then

self.day = "27"

endif

if self.day\_selected == 28 then

self.day = "28"

endif

if self.day\_selected == 29 then

self.day = "29"

endif

if self.day\_selected == 30 then

self.day = "30"

endif

if self.day\_selected == 31 then

self.day = "31"

endif

if self.month\_selected == 1 then

self.mth = "1"

endif

if self.month\_selected == 2 then

self.mth = "2"

endif

if self.month\_selected == 3 then

self.mth = "3"

endif

if self.month\_selected == 4 then

self.mth = "4"

endif

if self.month\_selected == 5 then

self.mth = "5"

endif

if self.month\_selected == 6 then

self.mth = "6"

endif

if self.month\_selected == 7 then

self.mth = "7"

endif

if self.month\_selected == 8 then

self.mth = "8"

endif

if self.month\_selected == 9 then

self.mth = "9"

endif

if self.month\_selected == 10 then

self.mth = "10"

endif

if self.month\_selected == 11 then

self.mth = "11"

endif

if self.month\_selected == 12 then

self.mth = "12"

endif

self.td = self.timedepart.get()

Get timedepart from self.filedata[0]

if self.td == "08:00AM" and self.cfm >= 0 then  
 timedepart = self.timedepart.get()

endif

if self.td == "12:00PM" and self.cfm >= 0 then

timedepart = self.timedepart.get()

endif

if self.td == "04:00PM" and self.cfm >= 0 then

timedepart = self.timedepart.get()

endif

if (self.td == '08:00AM (N.A.)' or self.td == '12:00PM (N.A.)') then

Display "Departure time is not available. Please try again." message self.cfm = 0

return

endif

if self.td == "" or self.cfm == 0 then

Display "Departure time is not selected. Please try again." message return

endif

day = self.year\_selected

if self.cfm == 0 then

return

Endif

exit

**updateJsonFile(self)**  
Save self.filedata into Master.json file  
Exit

***SelectSeat Module***Import json , Calendar , TicketDetail   
From tkinter import all(\*)

**Class Seatlayout**

**callbackSeat()**

Redraw master(Tkinter window)

**\_\_init\_\_(self)**

Start

Process openJsonFile()  
Global num

Global master

Set (self.times\_1A, self.times\_1B, self.times\_1B, self.times\_1C, self.times\_1D, self.times\_2A,self.times\_2B, self.times\_2C, self.times\_2D, self.times\_3A, self.times\_3B, self.times\_3C, self.times\_3D, self.times\_4A, self.times\_4B, self.times\_4C, self.times\_5A, self.times\_5B, self.times\_5C, self.times\_5D, self.times\_6A, self.times\_6B, self.times\_6C, self.times\_6D, self.times\_7A, self.times\_7B, self.times\_7C, self.times\_7D) =0

Set self.onetime = 0  
Create master=Toplevel() as Tkinter window   
Set window height to 400  
Set window width to 500  
Get num from self.data  
if num == 0 then

Create Back Button with command : Process backbutton(\*master)  
endif  
Process createseatlayout(\*master)  
End

**backbutton(self, master, k)**Process openJsonFile()  
Set number = 0

Get year , month, day, timedepart from self.data  
year = “”  
month = “”  
day = “”

timedepart = “”

Process updateJsonFile()   
Withdraw master(Tkinter Window)

If k == 0 :

Display “Sorry, it is full! Please change to another date OR depart time.” message  
Process Calendar.Calendar.callbackCalendar()  
Exit  
  
**openJsonFile(self)**self.data= load Master.json file   
self.listseat= load CheckListSeat.json  
Return self.data, self.listseat

Exit

**createseatlayout(self, master)**

Create all seat buttons with command for each buttons  
(button1A, button1B, button1C, button1D, button2A, button2B, button2C, button2D, button3A,

button3B, button3C, button3D, button4A, button4B, button4C, button4D, button5A, button5B,

button5C, button5D, button6A, button6B, button6C, button6D, button7A, button7B, button7C,

button7D)  
Display “A”, “B”, “C”, “D”, “1”, “2”, “3”, “4”, “5”, “6”, “7”  
Process checkseatavailable(\*master)

**checkseatavailable(self, master)**

Get priority

if priority == 0 then

Disable all priority seat buttons(1A,1B,1C,1D,2A,2B,2C,2D)

endif

Get year , month, day, depart, arrival and timedepart from self.data[0]

Get listseatno from self.listseat

list=listseatno

Loop : y = 0 to len(list) step 1

if list[y] == "1A" then

Disable button1A

endif

if list[y] == "2A" then

Disable button2A

endif

if list[y] == "3A" then

Disable button3A

endif

if list[y] == "4A" then

Disable button4A

endif

if list[y] == "5A" then

Disable button5A

endif

if list[y] == "6A" then

Disable button6A

endif

if list[y] == "7A" then

Disable button7A

endif

if list[y] == "8A" then

Disable button8A

endif

if list[y] == "1B" then

Disable button1B

endif

if list[y] == "2B" then

Disable button2B

endif

if list[y] == "3B" then

Disable button3B

endif

if list[y] == "4B" then

Disable button4B

endif

if list[y] == "5B" then

Disable button5B

endif

if list[y] == "6B" then

Disable button6B

endif

if list[y] == "7B" then

Disable button7B

endif

if list[y] == "1C" then

Disable button1C

endif

if list[y] == "2C" then

Disable button2C

endif

if list[y] == "3C" then

Disable button3C

endif

if list[y] == "4C" then

Disable button4C

endif

if list[y] == "5C" then

Disable button5C

endif

if list[y] == "6C" then

Disable button6C

endif

if list[y] == "7C" then

Disable button7C

endif

if list[y] == "1D" then

Disable button1D

endif

if list[y] == "2D" then

Disable button2D

endif

if list[y] == "3D" then

Disable button3D

endif

if list[y] == "4D" then

Disable button4D

endif

if list[y] == "5D" then

Disable button5D

endif

if list[y] == "6D" then

Disable button6D

endif

if list[y] == "7D" then

Disable button7D

Endif

Loop-End : y

Set self.forcepriority = 0

normalseatlist = ["3A","3B","3C","3D","4A","4B","4C","4D","5A","5B","5C","5D","6A","6B","6C","6D","7A","7B","7C","7D"]

disabled = 0

Loop : c = 0 to len(list) step 1  
 Loop : b = 0 to len(normalseatlist) step 1  
 If list[c] == normalseatlist[b] then  
 Disabled += 1

endif

Loop-End : b  
Loop-End : c

Set priorityseatlist=["1A","1B","1C","1D","2A","2B","2C","2D"]

prioritydisabled=0

Loop : c = 0 to len(list) step 1  
 Loop : b = 0 to len(priorityseatlist) step 1  
 If list[c] == priorityseatlist [b] then  
 prioritydisabled += 1

endif

Loop-End : b  
Loop-End : c

Set prioritynum=0

Set prioritylist=self.data[0]['prioritylist']

Loop : j = 0 to len(priortylist) step 1

if prioritylist[j] == 1 :

prioritynum +=1

Loop-End : j

If num==0:

if list == ["1A","1B","1C","1D","2A","2B","2C","2D","3A","3B","3C","3D","4A","4B","4C","4D","5A","5B","5C","5D","6A","6B","6C","6D","7A","7B","7C","7D"] then

set k=0

Process backbutton(\*master,\*k)

return

endif

seatavailable= 28-disabled-prioritydisabled

self.data[0]['seatavailable']=seatavailable

self.updateJsonFile()

if prioritynum > 0 :

if seatavailable < len(self.data[0]['iclist']) :

k = 0

self.backbutton(master,k)

return

if prioritynum == 0 :

if ((disabled == 20) or ((20-disabled) < len(self.data[0]['iclist']))):

k = 0

self.backbutton(master,k)

return

Loop : t = 1 to len(self.data) step 1:

Get nric, username, fileno from self.data[t]

Get iclist, currentuser from self.data[0]

If (iclist[num] == nric and

currentuser==username and

fileno == 0 and

prioritydisabled == 8):

Set self.forcepriority=0

Break

endif

if iclist[num] == nric and currentuser==username and

priority == 1 and fileno == 0) and

(self.data[0]['seatavailable'] == len(self.data[0]['iclist']))):

Set self.forcepriority=1

break

endif

Loop-end : t

Exit

**button\_1A(self, master)**

seatnum = “1A”  
if (float(self.times\_1A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_1A += 1  
 self.onetime -=1  
 Configure button1A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_1A +=1

self.onetime +=1

Configure button 1A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_2A(self, master)**

seatnum = “2A”  
if (float(self.times\_2A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_2A += 1  
 self.onetime -=1  
 Configure button2A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_2A +=1

self.onetime +=1

Configure button 2A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_3A(self, master)**

seatnum = “3A”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_3A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_3A += 1  
 self.onetime -=1  
 Configure button3A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_3A +=1

self.onetime +=1

Configure button 3A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
Exit

**button\_4A(self, master)**

seatnum = “4A”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_4A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_4A += 1  
 self.onetime -=1  
 Configure button4A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_4A +=1

self.onetime +=1

Configure button 4A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_5A(self, master)**

seatnum = “5A”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_5A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_5A += 1  
 self.onetime -=1  
 Configure button5A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_5A +=1

self.onetime +=1

Configure button 5A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_6A(self, master)**

seatnum = “6A”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_6A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_6A += 1  
 self.onetime-=1  
 Configure button6A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_6A +=1

self.onetime +=1

Configure button 6A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_7A(self, master)**

seatnum=“7A”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_7A % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_7A+=1  
 self.onetime-=1  
 Configure button7A color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_7A +=1

self.onetime +=1

Configure button 7A color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_1B(self, master)**

seatnum = “1B”

if (float(self.times\_1B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_1B += 1  
 self.onetime -=1  
 Configure button1B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_1B +=1

self.onetime +=1

Configure button 1B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_2B(self, master)**

seatnum = “2B”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_2B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_2B += 1  
 self.onetime -=1  
 Configure button2B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_2B +=1

self.onetime +=1

Configure button 2B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_3B(self, master)**

seatnum = “3B”  
if (float(self.times\_3B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_3B += 1  
 self.onetime -=1  
 Configure button3B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_3B +=1

self.onetime +=1

Configure button 3B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_4B(self, master)**

seatnum = “4B”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_4B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_4B += 1  
 self.onetime -=1  
 Configure button4B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_4B +=1

self.onetime +=1

Configure button 4B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_5B(self, master)**

seatnum = “5B”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_5B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_5B += 1  
 self.onetime -=1  
 Configure button5B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_5B +=1

self.onetime +=1

Configure button 5B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_6B(self, master)**

seatnum = “6B”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_6B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_6B += 1  
 self.onetime -=1  
 Configure button6B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_6B +=1

self.onetime +=1

Configure button 6B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_7B(self, master)**

seatnum = “7B”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_7B % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_7B += 1  
 self.onetime -=1  
 Configure button7B color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_7B +=1

self.onetime +=1

Configure button 7B color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_1C(self, master)**

seatnum = “1C”  
if (float(self.times\_1C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_1C += 1  
 self.onetime -=1  
 Configure button1C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_1C +=1

self.onetime +=1

Configure button 1C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_2C(self, master)**

seatnum = “2C”  
if (float(self.times\_2C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_2C += 1  
 self.onetime -=1  
 Configure button2C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_2C +=1

self.onetime +=1

Configure button 2C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_3C(self, master)**

seatnum = “3C”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_3C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_3C += 1  
 self.onetime -=1  
 Configure button3C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_3C +=1

self.onetime +=1

Configure button 3C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_4C(self, master)**

seatnum = “4C”  
if (float(self.times\_4C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_4C += 1  
 self.onetime -=1  
 Configure button4C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_4C +=1

self.onetime +=1

Configure button 4C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_5C(self, master)**

seatnum = “5C”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_5C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_5C += 1  
 self.onetime -=1  
 Configure button5C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_5C +=1

self.onetime +=1

Configure button 5C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_6C(self, master)**

seatnum = “6C”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_6C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_6C += 1  
 self.onetime -=1  
 Configure button6C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_6C +=1

self.onetime +=1

Configure button 6C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_7C(self, master)**

seatnum = “7C”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_7C % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_7C += 1  
 self.onetime -=1  
 Configure button7C color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_7C +=1

self.onetime +=1

Configure button 7C color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_1D(self, master)**

seatnum = “1D”

if (float(self.times\_1D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_1D += 1  
 self.onetime -=1  
 Configure button1D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_1D +=1

self.onetime +=1

Configure button 1D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_2D(self, master)**

seatnum = “2D”  
if (float(self.times\_2D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_2D += 1  
 self.onetime -=1  
 Configure button2D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_2D +=1

self.onetime +=1

Configure button 2D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_3D(self, master)**

seatnum = “3D”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_3D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_3D += 1  
 self.onetime -=1  
 Configure button3D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_3D +=1

self.onetime +=1

Configure button 3D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_4D(self, master)**

seatnum = “4D”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_4D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_4D += 1  
 self.onetime -=1  
 Configure button4D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_4D +=1

self.onetime +=1

Configure button 4D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_5D(self, master)**

seatnum = “5D”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_5D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_5D += 1  
 self.onetime -=1  
 Configure button5D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_5D +=1

self.onetime +=1

Configure button 5D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_6D(self, master)**

seatnum = “6D”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_6D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_6D += 1  
 self.onetime -=1  
 Configure button6D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_6D +=1

self.onetime +=1

Configure button 6D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**button\_7D(self, master)**

seatnum = “7D”

if self.forcepriority == 1:

Display 'You must choose priority seat!'+'\n'+ 'Because no more available

seat for your friend(s)/family member(s)' message  
if (float(self.times\_7D % 2) != 0 ) and (self.onetime == 1) Then  
 Destroy nextbtn button  
 self.times\_7D += 1  
 self.onetime -=1  
 Configure button7D color to ‘gold’

else  
 if self.onetime == 0 then

self.times\_7D +=1

self.onetime +=1

Configure button 7D color to ‘green’

Create nextbtn with command : Process confirmbutton(\*master,\*seatnum)  
else

Display “You can only select 1 seat!!” message  
endif

endif  
exit

**confirmbutton (self, master, seatnum)**

Process openJsonFile()

Get currentuser, iclist from self.data[0]

Loop : i = 1 to len(self.data) step 1

Get username, nric, fileno and seatno from self.data[i]  
 if username == currentuser and nric == iclist[num] and fileno == 0 then

Seaatno = seatnum

Endif

Loop-End : i

Get listseatno from self.listseat

Append seatnum to listseatno  
Sort listseatno  
Process updateJsonFile()  
Withdraw master(Tkinter Window)  
Process TicketDetail.bus\_tickets()

Exit  
  
**updateJsonFile(self)**  
Save self.data into Master.json file  
Save self.listseat into CheckListSeat.json file  
Exit

***TicketDetail Module***

Import json, SelectSeat

From tkinter import all(\*)

**Class bus\_tickets**

**\_\_init\_\_(self)**

Global CurrentUser

Process openJsonFile()

Set CurrentUser=self.data[0]['currentuser']

Create window=Toplevel() as Tkinter window

Set window height to 400

Set window width to 500

Create Scrollbar on right side at Y-axis

Process calPriceWithoutInsurance()

End

**openJsonFile(self)**

self.data=load Master.json file

self.listseat=load CheckListSeat.json file

Return self.data, self.listseat

Exit

**calPriceWithoutInsurance(self)**

Set self.num=self.data[0][‘num’]

Loop: i:1 to len(self.data) step1

Get username, nric, depart, arrival, priority, fileno from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][self.num])AND

(self.data[i]['fileno']==0) then

If depart==“Melaka” then

If arrival==“Seremban” then

If priority==1 then

self.price=8\*(50/100)

else

self.price=8

endif

endif

endif

If depart==“Melaka” then

If arrival==“Butterworth” then

If priority==1 then

self.price=50\*(50/100)

else

self.price=50

endif

endif

endif

If depart==“Butterworth” then

If arrival==“Melaka” then

If priority==1 then

self.price=50\*(50/100)

else

self.price=50

endif

endif

endif

If depart==“Butterworth” then

If arrival==“Seremban” then

If priority==1 then

self.price=48\*(50/100)

else

self.price=48

endif

endif

endif

If depart==“Seremban” then

If arrival==“Melaka” then

If priority==1 then

self.price=8\*(50/100)

else

self.price=8

endif

endif

endif

If depart==“Seremban” then

If arrival==“Butterworth” then

If priority==1 then

self.price=48\*(50/100)

else

self.price=48

endif

endif

endif

Loop-End:i

Process askInsurance()

Exit

**askInsurance(self)**

Global displayic

Set displayic=[]

Loop: x:0 to len(self.data[0][‘iclist’]) step1

If self.data[0]['iclist'][x][0:4] == "null"

Append "None" to displayic

else

Append (self.data[0]['iclist'][x]) to displayic

endif

Loop: i:1 to len(self.data) step1

Get username, nric, fileno, depart, arrival, day, month, year, timedepart, seatno from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][self.num])AND

(self.data[i]['fileno']== 0) then

Get answer from messagebox “do you want travel insurance? Travel insurance for a ride is RM4."

If answer==“yes” then

self.price=self.price+4

self.data[i]['price']=self.price

Display displayic, depart, arrival, day, month, year, timedepart, seatno, self.price in text box

else

self.data[i]['price']=self.price

Display displayic, depart, arrival, day, month, year, timedepart, seatno, self.price in text box

endif

endif

Loop-End: i

Process updateJsonFile

Create nextbtn with command : Process self.next\_button()

Create backbtn with command : Process self.back\_button()

Exit

**next\_button(self)**

self.num=self.num+1

Set self.data[0]['num']=self.num

Process updateJsonFile()

If (self.num>=len(self.data[0]['iclist']) then

Destroy nextbtn, backbtn

Create confirmbtn with command : Process self.close\_window

Process printAllTicket()

else

Destroy window(Tkinter window)

Process SelectSeat.seatlayout.callbackSeat()

endif

Exit

**back\_button(self)**

Global seatno

Loop: i:1 to len(self.data) step1

Get username, nric, fileno from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][self.num])AND

(self.data[i]['fileno']== 0) then

seatno=self.data[i]['seatno']

self.data[i]['seatno']=""

endif

Loop-End: i

Loop: i:0 to len(self.listseat) step1

Get year, month, day, timedepart, depart, arrival from self.listseat

If (self.listseat[i]['year']==(self.data[0]['year'])AND

(self.listseat[i]['month']==(self.data[0]['month'])AND

(self.listseat[i]['day']==(self.data[0]['day'])AND

(self.listseat[i]['timedepart']==(self.data[0]['timedepart'])AND

(self.listseat[i]['depart']==(self.data[0]['depart'])AND

(self.listseat[i]['arrival']==(self.data[0]['arrival']) then

Remove seatno (self.listseat[i]['listseatno'])

endif

Loop-End: i

Destroy window(Tkinter Window)

Process SelectSeat.seatlayout()

Exit

**printAllTicket(self)**

Set count=0

Set totalprice=0

Loop: i:1 to len(self.data) step1

Get username, datepurchased, nric, fileno, depart, arrival, day, month, year, timedepart, seatno, price from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][count])AND

(self.data[i]['fileno']== 0) then

totalprice=totalprice+price

Display displayic[count], depart, arrival, day, month, year, timedepart, seatno, price

self.nowdate=self.data[i]['datepurchased']

count=count+1

endif

Loop-End: i

Display "Total Price: RM [totalprice]"

Exit

**close\_window(self)**

Change file directory to ('Desktop\\ITProject')

Process endscreen.Endscreen()

Exit

**updateJsonFile(self)**

Save self.data into Master.json file

Save self.listseat into CheckListSeat.json file

Exit

**saveDataToTextFile(self)**

Set listfiles=[]

Change file directory to ('Desktop\\ITProject\\Users',CurrentUser)

Get filenames from file directory ('Desktop\\ITProject\\Users',CurrentUser)

Loop: x:0 to len(filenames) step1

Append filenames[x] into listfiles

Loop-End: x

Set times=1

Loop: y:0 to len(listfiles) step1

If listfiles[y]=[self.nowdate]\_[CurrentUser]\_[times].txt file then

times=times+1

endif

Loop-End: y

Create [self.nowdate]\_[CurrentUser]\_[times].txt file

Set turn=0

Set self.num=self.data[0]['num']

Loop: i:1 to len(self.data) step1

Get username, nric, fileno, depart, arrival, day, month, year, timedepart seatno, price from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][turn])AND

(self.data[i]['fileno']== 0) then

Write displayic[turn], depart, arrival, day, month, year, timedepart seatno, price in [self.nowdate]\_[CurrentUser]\_[times].txt file

turn=turn+1

endif

Loop-End: i

Set count=0

Loop: i:1 to len(self.data) step1

Get username, nric, fileno from self.data

If (self.data[i]['username']==(self.data[0]['currentuser'])AND

(self.data[i]['nric']==(self.data[0]['iclist'][count])AND

(self.data[i]['fileno']== 0) then

self.data[i]['fileno']=times

count=count+1

endif

Change file directory to ('Desktop\\ITProject')

Process updateJsonFile()

Exit

***End(Last) Screen Module***  
From tkinter import all(\*)**Class Endscreen**

**\_\_init\_\_(self)**

Start

Set root.bind("<Button-1>", self.leftclick)

Set root.bind("<Button-2>", self.middleclick)

Set root.bind("<Button-3>", self.rightclick)

Process self.exitdisplay()  
Create root = Tk() as Tkinter window   
Set window height to 400  
Set window width to 500  
End

**exitdisplay(self)**

Start

Display “Click anywhere to exit the program."

End

**leftclick()**

Exit the program.

**middleclick()**

Exit the program.

**rightclick()**

Exit the program.

**Appendix C: Data Dictionary**

***Data dictionary of all modules***

|  |  |  |
| --- | --- | --- |
| **Name** | **Submodules** | **Description** |
| Login | - | Enable users to login or register |
| Startup | - | To insert IC of passengers and choose location |
| History | Startup | To display all the records of user’s past purchased tickets |
| Calendar | SelectSeat | To choose departure time and date |
| SelectSeat | Calendar, TicketDetail | To choose the bus seat number |
| TicketDetail | SelectSeat, Endscreen | To calculate bus tickets price and print the tickets |
| Endscreen | - | To exit the program |

***Data Dictionary of Login Module***

**Functions**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| checkLogin | Check if the input username and password is correct for login |
| createfolder | Create a folder for member that registered successfully |
| enterclick | Enable users to execute checkLogin function by clicking Enter key |
| enterregclick | Enable users to execute registration function by clicking Enter key |
| loginpasswordhideimg | Enable user to see the password that entered in passEnter |
| loginpasswordviewimg | Enable user to hide the password entered in passEnter |
| openJsonFile | Load member.json file and Master.json file |
| openLoginPage | Open login window |
| openRegisterWindow | Open register window |
| registerpasswordhideimg | Enable user to see the password that entered in passEntry |
| registerpasswordviewimg | Enable user to hide the password that entered in passEntry |
| registerwindow | Open registration window |
| registration | Check if the input username and password combination is available for registration |
| saveMastertoJson | Save data to Master.json file |
| saveMembertoJson | Save members to member.json file |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| backbtn | object | Display backimg and executes openLoginPage upon clicking |
| backimg | image | Retrieve backtxt.png |
| count | integer | To count so that display error message when reach the end of members list |
| data | dictionary | Save the username that are currently logged in |
| exitbtn | object | Exit program upon clicking |
| exitimg | image | Retrieve exit.png |
| login | object | Login window |
| loginbtn | object | Display loginimg and executes checkLogin function upon clicking |
| loginimg | image | Retrieve login.png |
| logintitleimg | image | Retrieve logintitle1.png |
| logintitleimg2 | image | Retrieve logintitle2.png |
| members | dictionary | Save the username and password of previous and future users |
| notmatch | integer | To make sure the username and password is only saved to the file if there are not any identical username registered before |
| passConfirm | string | Input password for double confirmation in registration |
| passEnter | string | Input password for login purpose |
| passEntry | string | Input password for registration purpose |
| passwordhimg | image | Retrieve passhide.png |
| passwordhregimg | image | Retrieve passhide.png |
| passwordvbtn | object | Display passwordvimg and executes loginpasswordhideimg function upon clicking |
| passwordvimg | image | Retrieve passview.png |
| passwordvregbtn | object | Display passwordvimg and executes registerpasswordhideimg function upon clicking |
| passwordvregimg | image | Retrieve passview.png |
| register | object | Register window |
| registerbtn | object | Display registerimg and executes openRegisterWindow function upon clicking |
| registerimg | image | Retrieve register.png |
| submitbtn | object | Display submitimg and executes registration function upon clicking |
| submitimg | image | Retrieve submit.png |
| userEnter | string | Input username for login purpose |
| userEntry | string | Input username for registration purpose |

***Data Dictionary of Startup module***

**Functions**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| checkNRIC | Check the validity of IC entered by user |
| checkPriority | Check if passengers is eligible for priority seats |
| confirmLocation | Check if the location selected for departure and arrival is valid |
| donothing | Pass when executes |
| enterclicked | Enable user to execute confirmLocation function by clicking Enter key |
| entertowindow2 | Enable user to execute number function by clicking Enter key |
| entertowindow3 | Enable user to execute plusNricEntry function by clicking Enter key |
| minusEntry | Remove one entry box |
| NricEntry | Generate entry boxes for user to input IC |
| number | Check the validity of the input numoftickets entered by users |
| openJsonFile | Load Master.json file |
| openWindow1 | Open master window |
| openWindow2 | Open window2 window |
| openWindow3 | Open window3 window |
| plusEntry | Add one entry box |
| plusNricEntry | Save entry1, entry2, entry3, entry4, entry5, entry6, entry7, entry8, entry9, entry10 to list ic |
| savePassengerToFile | Save passengerdata to data |
| updateJsonFile | Save data to Master.json file |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| age | list | Save the age of each passengers |
| answer | string | Confirmation before exit the program |
| arrival | list | Save the value selected from selection 2 |
| blanknum | integer | Count the number of blank input entered in entry1, entry2, entry3, entry4, entry5, entry6, entry7, entry8, entry9, entry10 |
| confirmation | string | Confirmation if user entered nothing for entry1, entry2, entry3, entry4, entry5, entry6, entry7, entry8, entry9, entry10 |
| CurrentUser | string | The name of user that is currently logged in |
| data | dictionary | Save ic, depart, arrival |
| daylist | list | Save fifth and sixth digits from each items in list ic |
| depart | list | Save the value selected from selection 1 |
| entry1, entry2, entry3, entry4, entry5, entry6, entry7, entry8, entry9, entry10 | string | Input IC entered by user |
| history | object | Display ViewImg and executes history.LostTicket function upon clicking |
| ic | list | Save entry1, entry2, entry3, entry4, entry5, entry6, entry7, entry8, entry9, entry10 |
| iclist | list | Save first and second digits from each items in list ic |
| logoutButton | object | Display logoutimg and exit the program upon clicking |
| logoutimg | image | Retrieve logout.png |
| master | object | Window where the user input number of tickets to be purchase |
| minusButton | object | Display photo and executes minusEntry function upon clicking |
| monthlist | list | Save third and fourth digits from each items in list ic |
| newic | list | An identical list that is copied from ic |
| nextbtn | object | Display nextbuttonImg and executes number function upon clicking |
| nextbuttonImg | image | Retrieve nexttxt.png |
| nextbuttonImg2 | image | Retrieve nexttxt.png |
| numberoftickets | integer | Number of tickets to be purchase |
| numoftickets | string | Input number of tickets that the users want to purchase |
| passengerdata | dictionary | Save CurrentUser, ic, age, depart, arrival, priority |
| photo | image | Retrieve minus2.png |
| photo1 | image | Retrieve plus.png |
| plusButton | object | Display photo1 and executes plusEntry function upon clicking |
| priority | list | Save the value 0 if the passenger is not eligible for priority seats, value 1 for passengers who are eligible |
| selection1 | object | A combobox with choices of “Butterworth”, “Melaka” and “Seremban” to be selected as departure city |
| selection2 | object | A combobox with choices of “Butterworth”, “Melaka” and “Seremban” to be selected as arrival city |
| times | integer | Attached behind “null” when saving blank input in list ic |
| ViewImg | image | Retrieve view.png |
| welcomeImg | image | Retrieve welcome.png |
| window2 | object | Window where user input IC |
| window3 | object | Window where location for departure and arrival is selected |
| yearlist | list | Save the yearborn of each passenger |
| z | integer | Index to make sure “Do you want to cancel your booking?” message only shown starting from window2 onwards when clicking close |

***Data Dictionary of History Module  
Functions***

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| backpagebutton | Go to the previous page |
| closetextwindow | To close the text box window |
| DisplayHistory | Create master window and display layout that contains history buttons |
| displaynextpage | Display the next page layout that contains history buttons |
| displaytextbox | Create a window for text box |
| openJsonFile | Load Master.json file |
| readText | To check numbers of files for that username and get all the values for displaying them on the buttons for better appearance |

***Variables***

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| backbtn | object | Display backImg and close the window upon clicking |
| backclosetextbtn | object | Display backImg and executes closetextwindow function |
| backImg | image | Retrieve backtxt.png |
| buttons | list | To store all the generated history buttons / historybtn |
| combo | image | Retrieve combo.png for better appearance |
| count | string | To store the number value of namefiles from filenamenew  Eg. filenamenew= ‘25\_06\_2018\_1’  count= ‘1’ |
| CurrentUser | string | The name of user that is currently logged in |
| data | dictionary | About all passenger’s informations from Master.json file |
| date | string | To format and store the date from ‘namefiles’ for better appearance Eg. namefiles= [‘25\_06\_2018’]  date= ‘25/06/2018’ |
| datepurchase | list | To store the formated dates or ‘date’ |
| destination | list | To store the arrival name for each text files (.txt) for better appearance |
| detail | object | Create Tkinter window for text box |
| Fbutton | object | A frame that contains a button only |
| filenamenew | string | To store the file names without ‘.txt’ Eg. filenamenew= ‘25\_06\_2018\_user123\_1’ |
| filenamenew2 | string | To store the file name without username and number filenamenew2= ‘25\_06\_2018’ |
| filenum | list | To store the number of filenames or count |
| frames | list | To store all the generated frame objects / Fbutton |
| f\_contents | string | To store texts from text files( .txt) |
| historybtn | object | Display ‘combo’ and executes displaytextbox function upon clicking |
| label | object | Display Pbimg for better appearance |
| listfiles | list | To store all the filename of text files(.txt) |
| master | object | Create Tkinter window |
| namefiles | list | To store the date from the name of text file(.txt)  Eg. 25\_06\_2018\_user123\_1.txt  namefiles= [‘25\_06\_2018’] |
| nextImg | object | Retrieve next.png |
| nextpagebtn | object | Display nextImg and executes displaynextpage function upon clicking |
| number | integer | To initialize number of button displayed on a page |
| numseat | integer | To store numbers of seat purchased |
| PBimg | image | Retrieve pastbooking.png |
| seats | list | To store numseat for each text files (.txt) for better appearance |
| text | object | Text box that display f\_contents |
| timedepart | list | To store the depart time for each text files (.txt) for better appearance |

***Data Dictionary of Calendar Module***

**Functions**

|  |  |
| --- | --- |
| **Function name** | **Description** |
| calendarselection | Display the calendar and date selected. |
| callbackCalendar | Force Tkinter window to withdraw |
| changevaluetostring | Convert day and month to string. |
| clear\_calendar | Remove daybutton one by one in a automatic loop |
| clickedconfirm | To check whether the time user selected is available or not. |
| clocktimer | Display frame1 and process departtimecombobox |
| clocktitle | Display frame2 |
| closebuttonwindow | Exit the program if “OK” is clicked by user |
| departtimecombobox | Display frame3. Display timedepart based on system time. |
| disablebuttons | Disable daybuttons before today. |
| go\_nextbtn | Move forward to next month in the calendar |
| go\_prevbtn | Go back to previous month in the calendar |
| openJsonFile | Load Master.json file |
| outlineframe\_calendar | A border displayed behind the calendar |
| refreshcombobox | Create a border outside the calendar |
| save\_to\_file | Save filedata and run Seatlayout module. |
| saveselection | Save selected date(day, month, year, name of day, name of month) into dictionary(values) |
| tdydate | Display datetitle and currentdate |
| updateJsonFile | Save filedata into Master.json file |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| answer | object | Display “Do you want to cancel your booking?” |
| cfm | integer | A loop variable to ensure user clicked Confirm button |
| clock | integer | Display time2 |
| confirmbtn | object | Display confirmimg and executes confirmbtn upon clicking |
| confirmimg | image | Retrieve confirm.png |
| currentdate | string | Display todaydate |
| currentuser | string | The name of user that is currently logged in |
| datetitle | string | Display “Today’s Date” |
| day | string | Convert selected day to string |
| daybutton | list | A list of day of a month range from day 1 to 31 from [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30] |
| day\_selected | integer | Get selected day |
| days | list | A list of the name of the days from ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'] |
| departtimetitle | string | Display “Departure Time:” |
| filedata | dictionary | Save month, day, year, timedepart |
| frame | object | Contain leftbtn, header, rightbtn and calendar |
| frame1 | object | Contain datetitle and currentdate |
| frame2 | object | Contain timetitle and clock |
| frame3 | object | Contain departtimetitle, Combobox and comfirmbtn |
| header | string | Display the selected month and year |
| leftbtn | object | Display leftimg and executes go\_prevbtn upon clicking |
| leftimg | image | Retrieve left.png |
| month | integer | Current system’s month |
| month\_selected | integer | Get selected month |
| mth | string | Convert selected month to string |
| nextbtn | object | Display nextimg and executes changevaluetostring upon clicking |
| nextimg | image | Retrieve next.png |
| outlinef | object | A border displayed behind the calendar |
| rightbtn | object | Display rightimg and executes go\_nextbtn upon clicking |
| rightimg | image | Retrieve right.png |
| selecteddate | string | Display selected date |
| selectedtime | string | Display selected departure time |
| todaydate | string | Retrieve system’s current date |
| time1 | string | A variable for time2 to compare |
| time2 | string | Retrieve system’s current time |
| timedepart | object | A combobox with departure time values that differs based on conditions. Values that listed in combobox: '08:00AM', '08:00AM (N.A.)', '12:00PM', '12:00PM (N.A.)', '04:00PM' |
| timetitle | string | Display “Current Time” |
| values | dictionary | Save the values of day, month, year, name of day, name of month |
| year\_selected | integer | Get year selected |

***Data Dictionary of SelectSeat Modul*e**

**Functions**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| backbutton | Go to the previous module ( Calendar) and clear the values in Master.json file |
| callbackSeat | To redraw the master Tkinter window |
| checkseatavailable | To check numbers of seat taken and disable priority seat buttons if the user is not eligible for priority seats |
| closebuttonwindow | Exit the program if “OK” is clicked by user |
| confirmbutton | To save the seat number value into self.data and self.listseat and run TicketDetail Module |
| createseatlayout | Display the bus seat layout that contains seat buttons |
| button\_1A, button\_2A, button\_3A, button\_4A, button\_5A, button\_6A, button\_7A, button\_1B, button\_2B, button\_3B, button\_4B, button\_5B, button\_6B, button\_7B, button\_1C, button\_2C, button\_3C, button\_4C, button\_5C, button\_6C, button\_7C, button\_1D, button\_2D, button\_3D, button\_4D, button\_5D, button\_6D, button\_7D | Functions for each seat buttons and set the seat number value when the button is selected |
| openJsonFile | Load Master.json and CheckListSeat.json files |
| updateJsonFile | Save self.data into Master.json file and self.listseat into CheckListSeat.json |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| A,B,C,D | object | Labels that display “A”, “B”, “C”, “D” |
| answer | string | Confirmation before exit the program |
| backbtn | object | Display backImg and executes backbutton function upon clicking |
| backImg | image | Retrieve backtxt.png |
| button1A, button2A, button3A, button4A, button5A, button6A, button7A, button1B, button2B, button3B, button4B, button5B, button6B, button7B, button1C, button2C, button3C, button4C, button5C, button6C, button7C, button1D, button2D, button3D, button4D, button5D, button6D, button7D | object | Display seatImg and executes its own function upon clicking  List of buttons’ function names :  (button\_1A, button\_2A, button\_3A, button\_4A, button\_5A, button\_6A, button\_7A, button\_1B, button\_2B, button\_3B, button\_4B, button\_5B, button\_6B, button\_7B, button\_1C, button\_2C, button\_3C, button\_4C, button\_5C, button\_6C, button\_7C, button\_1D, button\_2D, button\_3D, button\_4D, button\_5D, button\_6D, button\_7D) |
| confirm | integer | To confirm if the bus exists on that day |
| data | dictionary | About all passenger’s informations from Master.json file |
| disabled | value | To know the number of seat taken for that bus |
| door | image,  object | Retrieve door.png and display on window for better appearance |
| driver | image,  object | Retrieve driver.png and display on window for better appearance |
| CurrentUser | string | Current username from Master.json file |
| displayic | string | To store either “None” or CurrentUser |
| empty ,empty2 | object | To create a blank row for better appearance on window |
| forcepriority | integer | To force the passenger to choose priority seat rather than normal seat because number of bus seats are limited when forcepriority = 1 |
| IcLabel | object | A label that display ‘displayic’ on the top right of window |
| layout | object | A frame that contains all the buttons and labels |
| listseat | dictionary | About all seat numbers taken of each buses from CheckListSeat.json file |
| master | object | Create Tkinter window |
| newlistseat | list | Create a new list if the bus on that day does not exist and save into CheckListSeat.json file |
| normalseatlist | list | A list that stores all the normal seat numbers |
| nextbtn | object | Display nextimg and executes confirmbutton function upon clicking |
| nextimg | image | Retrieve nexttxt.png |
| num | integer | a index for loop to know the order of passengers |
| num1, num2, num3, num4, num5, num6, num7 | object | Labels that display “1”, “2”, “3”, “4”, “5”, “6”, “7” in layout |
| onetime | integer | To know user whether user already selected the seat button or not. |
| priorityseatlist | list | To store all the priority seat numbers |
| prioritynum | integer | To know the number of passengers that are eligible for priority seat |
| seatnum | string | To store the seat number that user selected |
| seatavailable | integer | To know the number of seats that are available for that bus |
| times\_1A, times\_2A, times\_3A, times\_4A, times\_5A, times\_6A, times\_7A, times\_1B, times\_2B, times\_3B, times\_4B, times\_5B, times\_6B, times\_7B, times\_1C, times\_2C, times\_3C, times\_4C, times\_5C, times\_6C, times\_7C, times\_1D, times\_2D, times\_3D, times\_4D, times\_5D, times\_6D, times\_7D | integer | To know the number of clicks on seat button by User |

***Data Dictionary of TicketDetail Modul*e**

**Functions**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| askInsurance | To ask the user to include the optional travel insurance |
| back\_button | Go to the previous module (SelectSeat) and clear the values in Master.json file |
| calPriceWithoutInsurance | To calculate bus ticket price without insurance |
| close\_window | To go to next module (Endscreen) |
| closebuttonwindow | Exit the program if “OK” is clicked by user |
| escbutton | Executes closebuttonwindow function |
| next\_button | Executes printAllTicket function |
| openJsonFile | Load Master.json and CheckListSeat.json files |
| printAllTicket | To print all ticket information |
| saveDataToTextFile | To write all tickets information in a text file |
| updateJsonFile | Save self.data into Master.json file and self.listseat into CheckListSeat.json |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| answer | string | Confirmation to include insurance / before exit the program |
| backbtn | object | Display backImg and executes back\_button function upon clicking |
| confirmbtn | object | Display confirmImg and executes close\_window function |
| count | integer | A variable to loop self.data[0][‘iclist’] in Master.json |
| CurrentUser | string | The name of user that is currently logged in |
| data | dictionary | About all passenger’s informations from Master.json file |
| displayic | string | To store either “None” or CurrentUser |
| listfiles | list | To store all the filename of text files(.txt) |
| listseat | dictionary | About all seat numbers taken of each buses from CheckListSeat.json file |
| nextbtn | object | Display nextImg and executes next\_button function upon clicking |
| num | integer | a index for loop to know the order of passengers |
| price | real numbers | To calculate the value of the bus ticket price |
| times | integer | A variable to make sure all the filename of text files(.txt) do not duplicate |
| totalprice | real numbers | To sum up all the value of price |
| turn | integer | A variable to loop self.data[0][‘iclist’] in Master.json |
| window | object | Create Tkinter window |

***Data Dictionary of Endscreen Modul*e**

**Functions**

|  |  |
| --- | --- |
| **Function Name** | **Description** |
| display | Display enddisplay |
| exitdisplay | Display “Click anywhere to exit the program." |
| leftclick | Exit the program upon clicking Button-1 |
| middleclick | Exit the program upon clicking Button-2 |
| rightclick | Exit the program upon clicking Button-3 |

**Variables**

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Data Type** | **Description** |
| endimg | image | Retrieve endscreendis.png |
| enddisplay | object | Label that display endimg |

**Data Dictionary of Master Json file**

|  |  |  |
| --- | --- | --- |
| **Key** | **Data Type** | **Description** |
| age | string | To store the age of passenger |
| arrival | string | The arrival chosen by user |
| currentuser | string | The name of user that is currently logged in |
| day | string | The date of day chosen by user |
| depart | string | The departure chosen by user |
| fileno | integer | To store the text file (.txt) number |
| iclist | list | To store all the IC entered by user |
| month | string | The date of month chosen by user |
| nric | string | To store the ic of the passenger |
| price | real numbers | To store the price of ticket purchased by the passenger |
| priority | integer | To know whether the passenger is eligible for priority seat Eg. 1 = Eligible , 0 = Not eligible |
| prioritylist | list | To store all the 'priority' value |
| seatavailable | integer | To know the number of seats that are available on that bus on that day |
| seatno | string | The bus seat number taken by the passenger |
| timedepart | string | The departure time chosen by user |
| year | string | The date of year chosen by user |

**Data Dictionary of CheckListSeat Json file**

|  |  |  |
| --- | --- | --- |
| **Key** | **Data Type** | **Description** |
| arrival | string | The arrival chosen by user |
| day | string | The date of day chosen by user |
| depart | string | The departure chosen by user |
| listseatno | list | To store all the seat number taken of the bus on that day |
| month | string | The date of month chosen by user |
| year | string | The date of year chosen by user |

**Data Dictionary of Member Json file**

|  |  |  |
| --- | --- | --- |
| **Key** | **Data Type** | **Description** |
| password | string | To store password of the username |
| username | string | To store the username |