UTM
UNIVERSITI TEKNOLOGI MALAYSIA

UNIVERSITI TEKNOLOGI MALAYSIA, 81310, UTM JOHOR BAHRU, JOHOR, MALAYSIA

SECJ2013-04(DATA STRUCTURE AND ALGORITHM)

Section - 03

STUDENTS NAME	NO. MATRIC
DANIAL ERFAN SHAH BIN NOR AZAM SHAH	A22EC0151
MEGAT MUHAMMAD ZAFRAN BIN MEGAT MUAZZAM	A22EC0194
MUHAMMAD ARIF FIKRY BIN NOOR KHARIZAN	A22EC0203
ADAM FAHMI BIN MOHD ADNAN	A22EC0032

GROUP CAPYBARA

PROJECT

LECTURER'S NAME: DR LIZAWATI BINTI MT YUSUF

TABLE OF CONTENTS

Project Overview

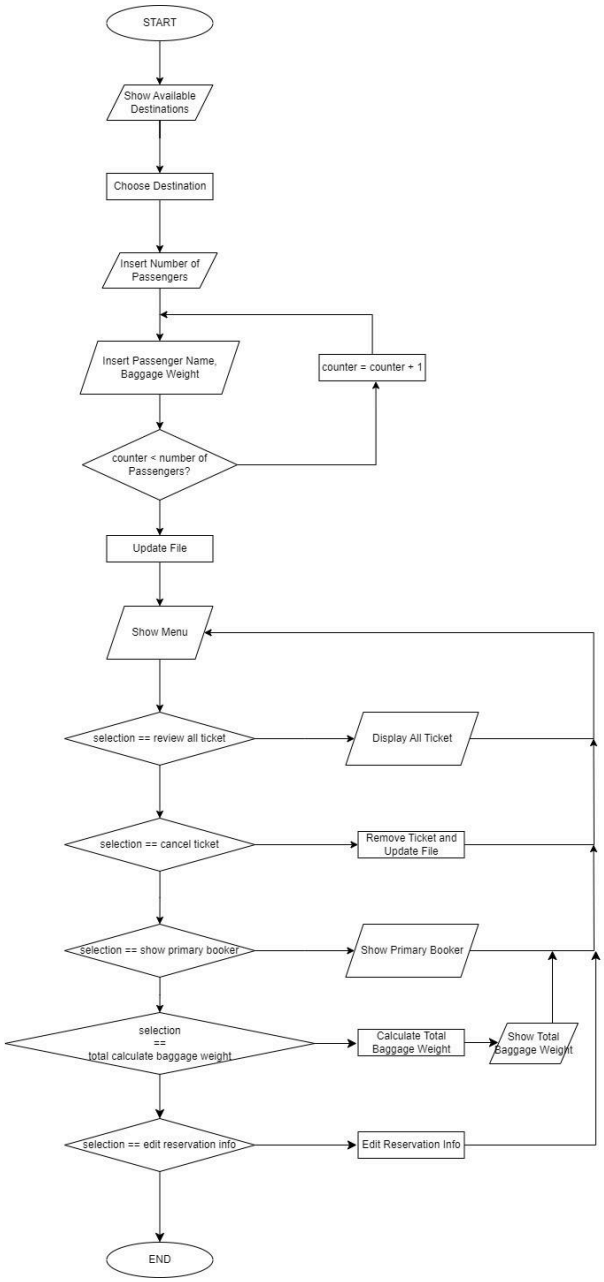
Airline reservation systems often have many features, such as booking, ticket review, and ticket cancellation. Our airline reservation system focuses on the user's point of view and not on the administrator's. Initially, the user will be given options such as selecting the location of the number of passengers to buy tickets. Moving on, the user will be given four options: review the purchased ticket details, delete all ticket details, display the primary booker or exit from the system. Throughout this process, we only apply queue implementation on all aspects, including adding, removing or returning data. Airline reservation systems often have many features, such as booking, ticket review, and ticket cancellation. Our airline reservation system focuses on the user's point of view and not on the administrator's. Initially, the user will be given options such as selecting the location of the number of passengers to buy tickets. Moving on, the user will be given six options: review the purchased ticket details, delete all ticket details, display the primary booker, show total weight of baggage, edit user information, or exit from the system. Throughout this process, we only apply queue implementation on all aspects, including adding, removing or returning data.

Design of Project

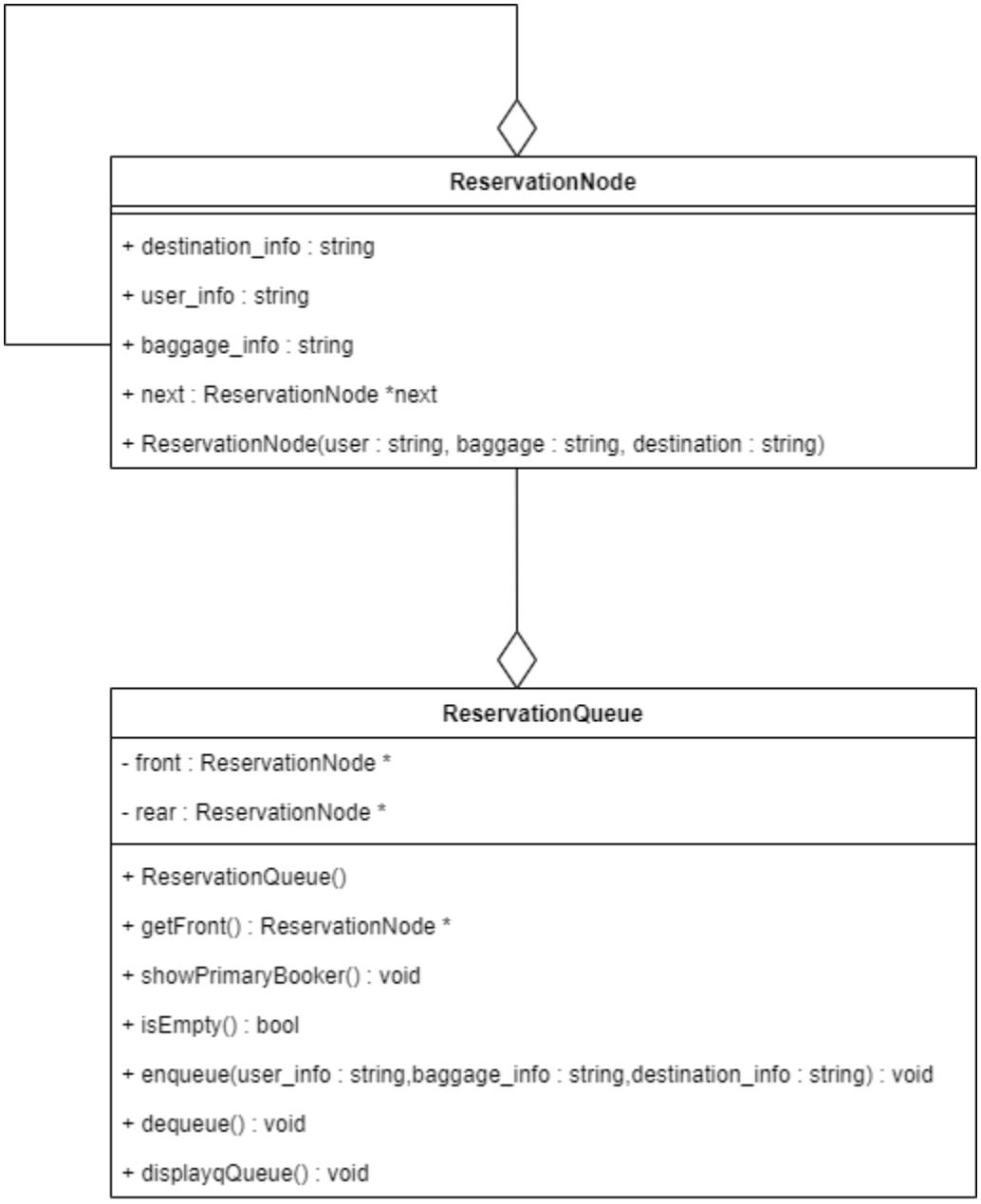
Pseudocode

```
1.0 START
    1.1 SHOW Destinations
    1.2 CHOOSE Destinations
    1.3 INPUT Number of Passengers
        1.3.1 INPUT Every Passengers Name, Baggage Weight
        1.3.2 RECORD Data in file
    1.4 SHOW Menu
    1.5 START WHILE
    1.5 WHILE Selection ⩾ 1 AND Selection ⩽ 3
        1.6 START IF
        1.7 IF Selection == Review All Ticket
            1.7.1 SHOW All Destination, Passengers Name, Baggage Weight
        1.8 ELSE IF Selection == Cancel Ticket
            1.8.1 REMOVE ALL Passengers Name, Baggage Weight, Destination
            1.8.2 Record Data in file
        1.9 ElSE IF Selection == Show Primary Booker
            1.9.1 SHOW Primary Booker
        2.0 ELSE IF Selection == Calculate All Total Baggage Weight
            2.0.1 CALCULATE Total Baggage Weight
        2.1 ELSE IF Selection == Edit Reservation Info
            2.1.1 EDIT Reservation Info
        2.2 END IF
    1.6 END WHILE
2.0 END
```

Flow Chart


START
Show Available Destinations
Choose Destination
Insert Number of Passengers
Insert Passenger Name, Baggage Weight
counter = counter + 1
counter < number of Passengers?
Update File
Show Menu
selection == review all ticket
Display All Ticket
selection == cancel ticket
Remove Ticket and Update File
selection == show primary booker
Show Primary Booker
selection == total calculate baggage weight
Calculate Total Baggage Weight
Show Total Baggage Weight
selection == edit reservation info
Edit Reservation Info
END

UML Diagram


ReservationNode
+ destination_info : string
+ user_info : string
+ baggage_info : string
+ next : ReservationNode *next
+ ReservationNode(user : string, baggage : string, destination : string)
ReservationQueue
- front : ReservationNode *
- rear : ReservationNode *
+ ReservationQueue()
+ getFront() : ReservationNode *
+ showPrimaryBooker() : void
+ isEmpty() : bool
+ enqueue(user_info : string,baggage_info : string,destination_info : string) : void
+ dequeue() : void
+ displayqQueue() : void

Data Structure Code Implementation

Here, our code primarily focuses on the Queue Linked List concept. Queue is an operation where the first element inserted into the list will also be the first to be removed from the list. This scenario is also usually known as FIFO (First In First Out) method. Our queue is operated in a linear concept instead of a circular queue.

Enqueue Method

This method adds a new element at the very first position of the Queue.

```
void enqueue(string user_info, string baggage_info, string destination_info)
{
    ReservationNode *new_reservation = new ReservationNode(user_info, baggage_info, destination_info);
    if (isEmpty())
    {
        front = rear = new_reservation;
    }
    else
    {
        rear→next = new_reservation;
        rear = new_reservation;
    }
}
```

If the queue is empty, the front and rear values will be the same as the newly inserted Node.

Apart from that, if the queue is not empty, the rear pointer will be appointed to the new node, and the rear value will change to a new value, which is a new node.

Dequeue Method

```
void dequeue()
{
    if (isEmpty())
    {
        cout << "Queue is empty. No reservations to dequeue." << endl;
    }
    else
    {
        ReservationNode *temp = front;
        front = front->next;
        cout << endl;
        cout << "Removed Ticket " << temp->user_info << " together with " << temp->baggage_info << " kg of baggage";
        cout << endl;
        delete temp;
        cout << "Reservation removed successfully." << endl;
    }
}
```

As stated before, whoever is inserted first into the list will also be the one who will be removed from the queue.

Therefore, if the queue is not empty, we will create a new temporary node with the same value as the front node.

Later, the front value will be appointed to the one next to it. Before deleting the temporary node, we will display the value of the user, as well as its baggage weight.

Then, if the queue is empty, there will be no operation since there is no single element to remove.

Display Queue Method

```
void displayQueue()
{
    if (isEmpty())
    {
        cout << "Queue is empty." << endl;
    }
    else
    {
        ReservationNode *current = front;
        cout << "\nDestination: " << current->destination_info << endl;

        while (current != nullptr)
        {
            cout << "Name: " << current->user_info << ", Baggage Info: " << current->baggage_info << endl;
            current = current->next;
        }

        cout << endl;
    }
}
```

Our approach to the display queue is to create a new temporary node known as current that will be appointed to the very front of the list.

Then, we first display the destination of the list since every element in the list will have the same values anyway.

After that, a while loop will traverse the list until it reaches a node with a NULL value.

If the queue is empty, it will display the error message and do nothing.

Get Front Method and Show Primary Booker Method

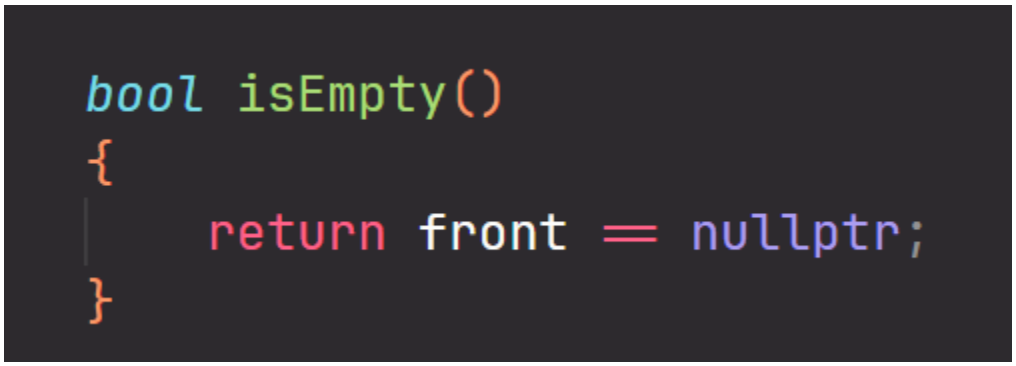
```
ReservationNode *getFront()
{
    return front;
}

void showPrimaryBooker()
{

    if (isEmpty())
    {
        cout << "There are no reservations made\n";
    };
    cout << "The primary booker is " << front→user_info << " with the baggage of " << front→baggage_info << endl
         << endl;
}
```

Both of these methods have similar functionality.

If you look at the getFront method, it is basically to return the front node.

Looking at the primary booker, it is to show the value of the front node.

Is Empty method



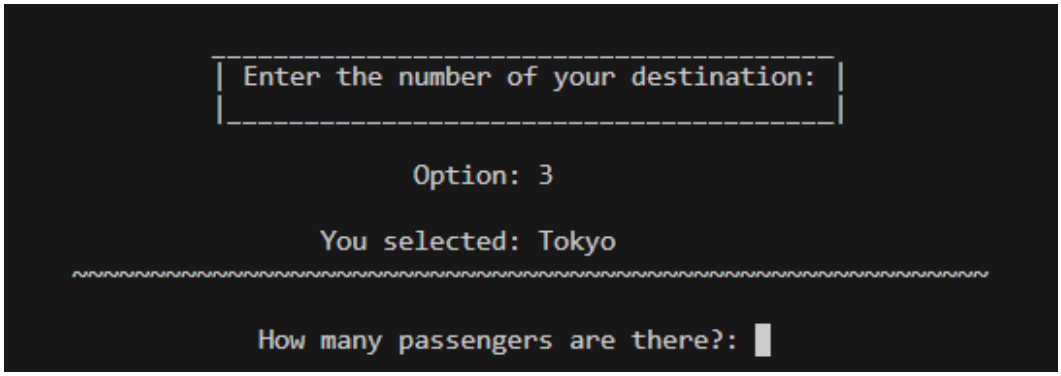
This method is relatively straightforward. It will return the true or false value if the front value is NULL.

.

User Manual Guide

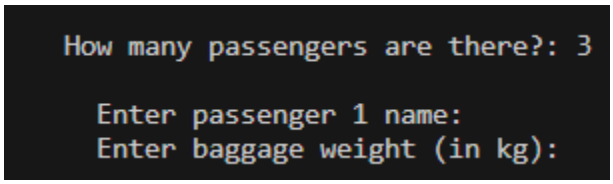
Screen 1: Airplane Reservation Main Menu

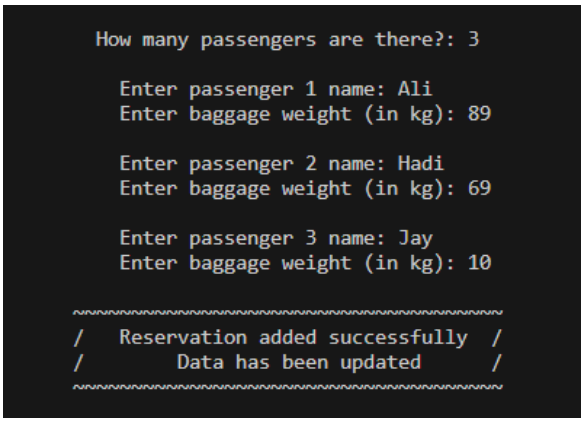
Screen 1: Starting the code will greet the user with an interface that will First, greet the user then it will ask for the DESTINATION LOCATION that the user wants to go. The user will be given a range of integer value from 0 to 5. Any input that the user will enter will give the same output such as New York, Paris, Tokyo, London and Syndney but the data file will store all the different destination location to each user. Entering a value that is more than 5 will result in an error and an error message will be display to notice the user that they entered an invalid value. The cursor will automatically be at the Option, so the user only needs to enter their respected location.



Screen 2: Total Number of Passenger Option

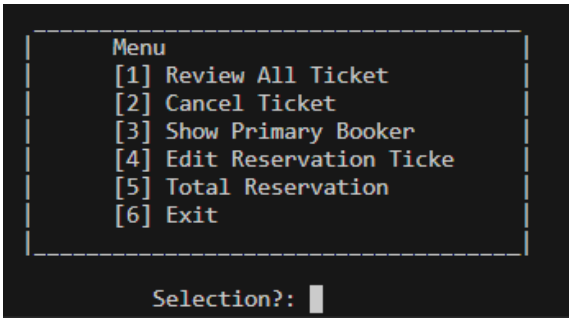
Screen 2: After entering an integer value from 0 to 5, the code will display a menu for the user to key-in the total number of passenger that they wish to book with us. There is no limitation number of passenger as the code will store all information that the user wishes to store inside our system. The code will also display the selected location as a confirmation to the user that they entered the correct location and they are able to redo the process if they accidentally entered the wrong value.





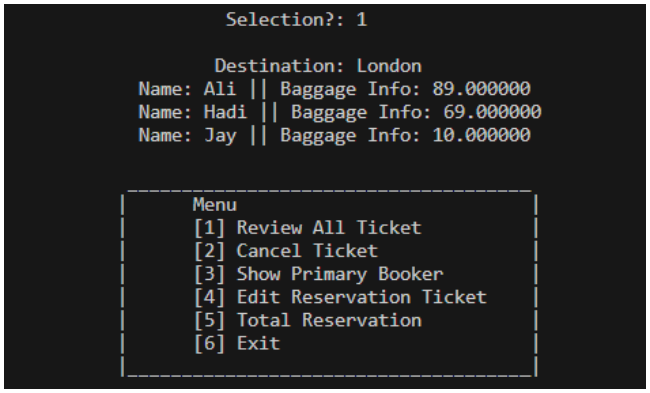
Screen 3: Information Entering Option

Screen 3: When the user entered a set of number of passengers they wish to book an airplane, as an example 3. The user will be showed a message that is asking the user to enter the passenger's name and their baggage weight. This process will repeat depending on the amount of passengers that the user entered at Screen 2. After the user entered the final information, the system will output a message that confirms to the user that all the information has been stored into the system's file.



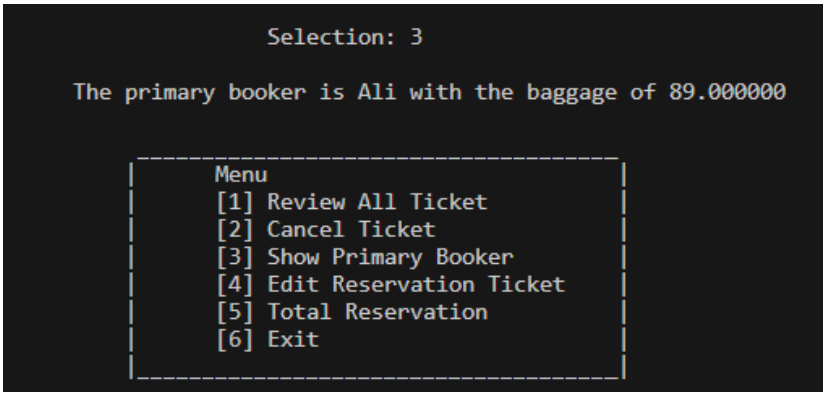
Screen 4: User Menu

Screen 4: After entering all the necessary information, the system will output a menu to the user that can interact with. Users are given an integer range of 1 to 4 in which if the user enter 1, it will display all the information that user has entered at Screen 3. Entering 2, will trigger the delete mechanism in the system and will delete all the information that was entered earlier. Entering 3, will display the Primary Booker, Primary Booker in this context is the First Passenger's detail as seen in Screen 3. Next, entering 4 will enter the information editor in our system. Entering 5, will display the total weight of all the passenger's luggage. Lastly entering 6 will kill the system



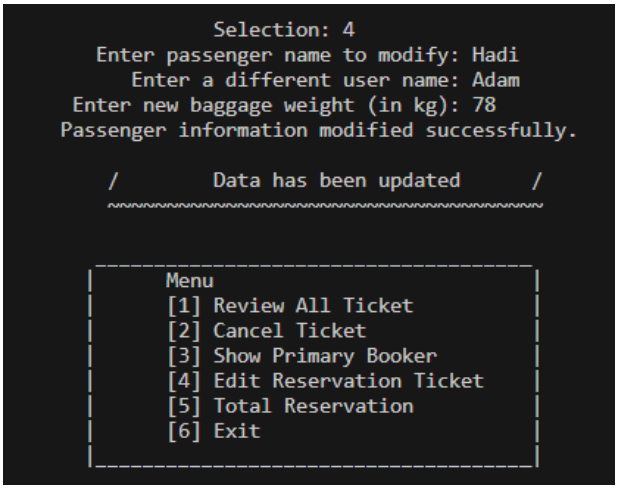
Screen 5: Review All Ticket Menu

Screen 5: Picking option 1 will display all the passenger's information such as Name and Luggage Info to the User. This information can be used by the User to recheck all the information if they are all correct



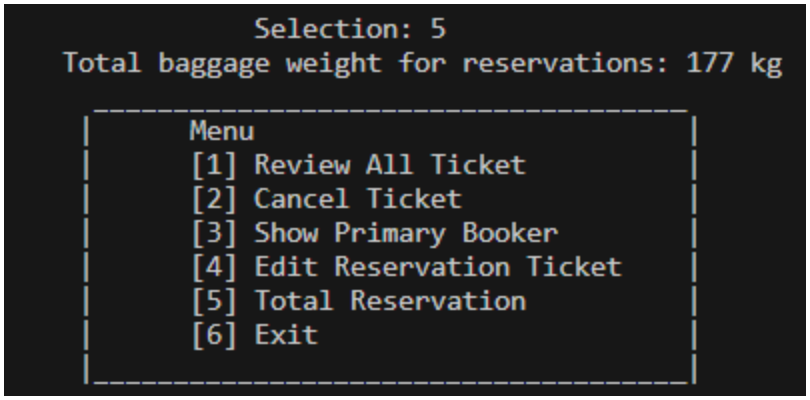
Screen 6: Show Primary Booker Menu

Screen 6: Entering 3 will display the information regarding the Primary Booker. As the data entered in Screen 3 the first passenger is Ali with the luggage weight of 89 KG. So this menu option shows whose data is entered first of the list.



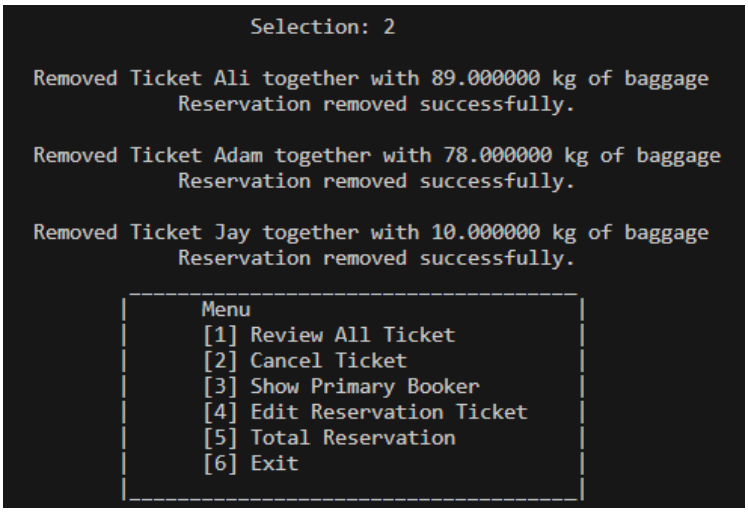
Screen 7: Edit Reservation Ticket Menu

Screen 7: Picking 4 will trigger the function of the system to find the information on whose data is inputted by the user to be modified. In this case, the user will enter any name from Screen 3 and they are able to rename the name as in this case Hadi is changed to Adam and the luggage weight is changed from 69 to 78. The system will give a confirmation message to notify the user that the change is successfully modified in the system files.



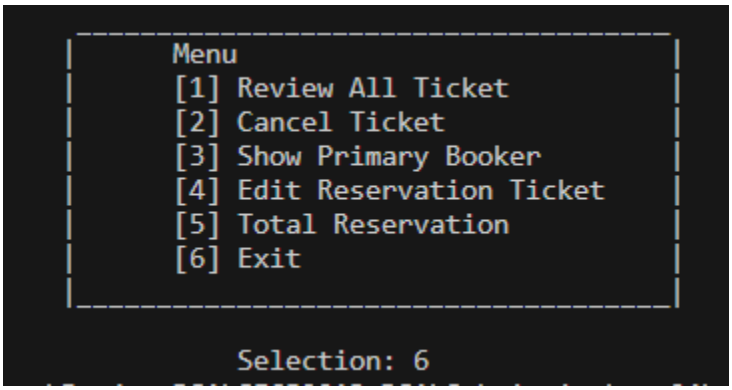
Screen 8: Total Reservation Menu

Screen 8: Option 5 will display the total luggage weight of the User's imputed passenger information. The data is calculated from all the Passenger's information inside the system's datafile. This data is important to the Crews as they need to make sure the total luggage weight does not exceed maximum cargo limit of the airplane.



Screen 9: Cancel Ticket

Screen 9: Lastly option 2 will delete all the passengers information in the system's data file. This menu will delete all the information that was previously entered at Screen 3 if the User accidentally made an abundance of mistakes and they are not willing to modify all of it individually. Users will be notified that all the information is successfully deleted as a confirmation. Thus deleting all the information and reentering the information is the efficient way to go.



Screen 10: Exit

Screen 10: Selecting 6 is rather straightforward as it kills the system and ends any activity that will be done to the code. Rest assure that all the information entered are kept in the data file if the User did not decide to cancel their ticket before exiting the system.