# TDS2111 Data Structures and Algorithms

Trimester 1, 2023/2024

## Restaurant Reservation System

Zero 1

Lab Section: 1B

Project Prepared by:

| | Name | Student ID | Major |
|---|---|---|---|
| 1 | Lee Chuan Tiew | 1211101782 | DCN |
| 2 | Justin Lau Li Ting | 1211101124 | DCN |
| 3 | Tan Wei Chuen | 1211201347 | BIA |
| 4 | Brandon Tan Chi Hong | 1211103334 | ST |

FACULTY OF INFORMATION SCIENCE & TECHNOLOGY

MULTIMEDIA UNIVERSITY

MALAYSIA

31 January 2024

## Table of Contents

***1.0 Introduction*** ........ ***3***

***2.0 Problem Statements*** ........ ***4***

***3.0 ADT Specification*** ........ ***5***

***4.0 Implementation Details*** ........ ***15***

***5.0 Program Screenshot*** ........ ***16***

***6.0 Task Declaration Forms*** ........ ***26***

***7.0 Conclusion*** ........ ***30***

# 1.0 Introduction

In the rapidly evolving technological landscape, effective management is pivotal to the success of any dining establishment. Recognizing this, our team at Zero 1 has embarked on an ambitious project to revolutionize the way restaurants manage their reservations. We have developed a Restaurant Reservation System using C++, designed to redefine the reservation experience for both patrons and restaurant management.

Our cutting-edge system is engineered to streamline the reservation process, offering a seamless, user-centric interface that simplifies the task of booking tables. Customers seeking to plan their dining experiences in advance will find our system exceptionally convenient, allowing them to reserve tables with ease and efficiency. This feature is especially beneficial in today’s fast-paced world, where time is a precious commodity.

Simultaneously, our system is a powerful tool for restaurant administrators. It offers unparalleled capabilities in managing table availability, tracking reservations in real time, and optimizing seating arrangements to maximize dining space. By automating these tasks, the system reduces the likelihood of human error, ensures a smoother operation, and enhances the overall efficiency of the restaurant.

Moreover, our system is designed with scalability and adaptability in mind, ensuring that it can evolve with the changing needs of the restaurant industry. Whether it's integrating with other management software or expanding to accommodate a growing customer base, our system is poised to meet the challenges of the future.

In essence, Zero 1’s Restaurant Reservation System is not just a technological advancement; it is a step towards redefining the dining experience in the modern era. By bridging the gap between technology and hospitality, we aim to empower restaurants to operate more efficiently while simultaneously elevating the customer experience to new heights.

# 2.0 Problem Statements

In today's restaurant industry, the lack of a fully integrated reservation system is a major challenge, leading to potential customer and revenue loss. Our team at Zero 1 has developed a comprehensive restaurant reservation system to address this gap.

Our solution is an all-in-one software that enables customers to book seats, order food in advance, and choose their dining time and date from home. This system is not just about reservations; it's a complete planning tool for the modern, fast-paced world.

Traditional reservation methods, like phone calls or third-party websites, are often inefficient and costly. Our user-friendly platform reduces the need for manual handling and eliminates third-party fees, making the reservation process more efficient and cost-effective.

The program is designed with a focus on user experience, featuring clear instructions for easy navigation and advanced error-checking to ensure order accuracy. This leads to a more enjoyable and satisfying dining experience for customers.

In essence, Zero 1's restaurant reservation system is a blend of convenience, efficiency, and innovation. It addresses the pain points of traditional methods and enhances customer experience, providing restaurants with a tool to optimize operations, reduce costs, and meet the high expectations of modern consumers.

# 3.0 ADT Specification

## <u>Admin ADT</u>

**Data items**

Std::string a_username

- Each admin needs to set a username before the reservation system starts. This username will be used for admin login purposes.

Std::string a_passwd

- Each admin needs to set a password before the reservation system starts. This password will be used for admin login purposes.

Int id_number

-Each reservation will be assigned by an id number automatically.

**Operations**

Static void AddReserve()

Results:

- Create a new reservation for the customers

Static void CancelReserve()

Results:

- Delete a reservation

Static void SearchReserve()

Results:

- Search the specific reservation by using the ID number

Static void DisplayReserve(const std::vector<Reservation> display_data)

Results:

- Displays the details of reservations from the provided vector.

Static void EditReserve()

Results:

- Edit the reservation details

Static void SortReserve()

Results:

- Sort reservation by using the ID number or date and time

Static void admin_page1()

Results:

- require a_username and a_passwd from the user for log in

Static void admin_page2()

Results:

- provide all the features such as add, display, cancel, edit, search, and sort for admin to manage the reservation

## Reservation ADT

Int id_number = 0

-Each reservation will be assigned an ID number automatically.

Std::string c_name

-Each customer needs to set a customer name to add to the reservation details

Std::string c_gender

-Each customer needs to give their gender to add to the reservation details.

Std::string c_email

-Each customer needs to provide their email address to ensure communication

Std::string c_contactNumber

-Each customer needs to provide their contact number to ensure communication

Std::string date

-Each customer needs to give a specific date to add to the reservation details

Std::string time

-Each customer needs to give a specific time to add to the reservation details

Std::string seat

-Each customer needs to choose whether they want a VIP room or a normal seat

**Operations**

Static void AddReserve()

Results:

- Create a new reservation for the customers

Static void CancelReserve()

Results:

- Delete a reservation

Static void SearchReserve()

Results:

- Search the specific reservation by using the ID number

Static void loadData()

Results:

- Reads reservation data from a file and populates the ‘rsvt_data’ vector.

Static void saveData()

Results:

- Writes reservation data from the ‘rsvt_data’ vector to a file.

Static void getNewId()

Results:

- Called upon every data creation and assign a new id number for the new reservation.

Static void getLastId()

Results:

- Called upon the everytime the system run, it will get the last id from the latest reservation details.

std::string Reservation::get_name() const

Results:

- Returns the name of the customer associated with the reservation.

std::vector<Reservation> rsvt_data

Description:

- a vector data structure that stores the reservations

const std::string FILENAME = "booking.txt"

Description:

- the name of the file where reservation data is stored or retrieved from

Void swap_index (int& x, int& y)

Results:

- swaps the value of two integers passed by reference

Static void welcome()

Description:

- a welcoming page for the user to choose to log in as admin or customer

Static void customer_page()

Results:

- shows 3 features such as add, search, and cancel for customers to manage their reservation

## Queue ADT

Data items

1. struct Queue Node

a. QueueItemType item

- Create an item of type int as QueueItemType is a type of int.

b. QueueNode *next

- Create a pointer next pointing to another QueueNode.

2. QueueNode* backPtr

- Create a pointer backPtr pointing to another QueueNode.

3. QueueNode* frontPtr

- Create a pointer frontPtr pointing to another QueueNode

**Operations**

1. Queue()

- Constructor. Creates a queue object with backPtr and frontPtr set to nullptr.

2. Queue (const Queue &Q)

- Copy constructor. Creates a queue object with member variables copied from Q.

3. ~Queue()

- Destructor. Dequeues the queue till it is empty.

4. bool Queue::isEmpty() const

- Checks if the Queue is empty.

- Returns true if the stack is empty otherwise returns false.

5. static void Queue::enqueue(const QueueItemType& newItem)

   - Adds an item to the queue.

   - If the queue is empty, add a new node with both frontPtr and backPtr pointing to it.

- If the queue is not empty, add a new node at the back of the queue and have backPtr pointing to it.

6. static void Queue::dequeue()

   - Removes the most front item from the queue.

   - If frontPtr equals to backPtr, set both to nullptr.

   - Else, set frontPtr to point to the next node it was pointing to.

7. QueueItemType Queue::getFront() const

- Returns the item of the most front node if the queue is not empty.

## Stack ADT

1. struct StackNode

a. StackItemType item

- Create an item of type int as StackItemType is a type of int.

b. StackNode* next

- Create a pointer next pointing to another StackNode.

2. StackNode* topPtr

- Create a pointer topPtr pointing to another StackNode.

**Operations**

1. Stack()

- Constructor. Creates a stack object with topPtr set to nullptr.

2. Stack(const Stack& S)

- Copy constructor. Creates a stack object with member variables copied from S.

3. ~Stack()

- Destructor. Pop the stack till it is empty.

4. bool Stack::isEmpty() const

- Checks if the stack is empty.

- Returns true if the stack is empty otherwise returns false.

5. static void Stack::push(const StackItemType& newItem)

- Adds an item to the top of a stack.

- Add a new node and have topPtr point to it.

6. static void Stack::pop()

- Removes items on the top of a stack if the stack is not empty.

- Set topPtr to point to the next node it was pointing to.

7. StackItemType Stack::getTop() const

- Returns the item of topPtr if the stack is not empty.

## Sort_by_name ADT

Function

static void sort_by_name(const std::vector<Reservation> rsvt_data, int first, int last, Queue& result_queue, Stack& result_stack)

a. Parameters:

const std::vector<Reservation> rsvt_data: Vector of Reservation objects representing the reservation data to be sorted.

- int first: Integer representing the index of the first element in the current partition.
- int last: Integer representing the index of the last element in the current partition.
- Queue& result_queue: Reference to a Queue object used to store the sorted reservation indices in ascending order.
- Stack& result_stack: Reference to a Stack object used to store the sorted reservation indices in descending order.

b. Description:

- This function implements the quicksort algorithm to sort the reservation data by the customer's name.
- It uses the result_queue to store the sorted indices in ascending order and result_stack to store them in descending order.

c. Preconditions:

- first and last must be valid indices within the range of the rsvt_data vector.
- The result_queue and result_stack should be initialized.

d. Postconditions:

- The result_queue contains the indices of the reservations in ascending order of customer names.
- The result_stack contains the indices of the reservations in descending order of customer names.

e. Algorithm:

- The function uses the quicksort algorithm to partition and sort the reservation indices based on the customer's name.

## Sort_by_name_partition ADT

Function

static void sort_by_name_partition(const std::vector<Reservation> record, int first, int last, int& pivotIndex, std::vector<int>& result)

a. Parameters:

- const std::vector<Reservation> record: Vector of Reservation objects representing the reservation data.
- int first: Integer representing the index of the first element in the current partition.
- int last: Integer representing the index of the last element in the current partition.
- int& pivotIndex: Reference to an integer representing the index where the partitioning is based.
- std::vector<int>& result: Reference to a vector of integers used to store the sorted reservation indices.

b. Description:

- This function performs a partition step in the quicksort algorithm to sort the reservation indices based on the customer's name.
- It updates the pivotIndex and modifies the result vector accordingly.

c. Preconditions:

- first and last must be valid indices within the range of the record vector.
- pivotIndex should be initialized before calling the function.
- The result vector should be initialized.

d. Postconditions:

- The elements in the range [first, last] are partitioned based on the customer's name, and the pivotIndex is updated.
- The result vector contains the sorted reservation indices after the partition step.

# 4.0 Implementation Details

For the data structure justification, we have use Vector (“std::vector<Reservation>”) instead of array. The reason is vector are used to store the reservations (“rsvt_data”). Vectors can provide dynamic array functionality to allow easy addition, removal, and access of reservation data. Sequential access to elements is beneficial for operations like sorting, where elements need to be compared and swapped.

Besides, we are using Queue in our project. We used queue in the sorting process (“SortReserve”) to store the sorted results in ascending order. Queues follow the First-In-First-Out (FIFO) principle, which aligns well with processing elements in the order they are sorted.

Moreover, we are using Stack in our project. We used stack in the sorting process (“SortReserve”) to store the sorted results in descending order. Stacks follow the Last-In-First-Out (LIFO) principle, which is useful for reversing the order of sorted elements.

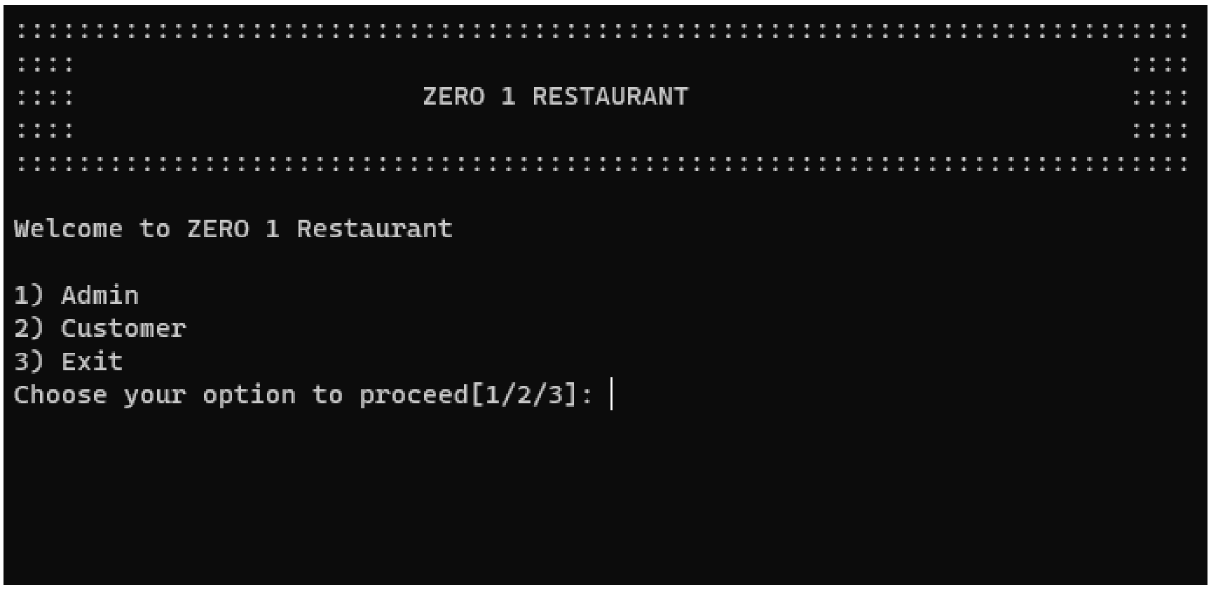
For the algorithm justification, we have used QuickSort and we apply in the code using (“sort_by_name_partition” and “sort_by_name”). We chose to use QuickSort for sorting reservations by customer name. The benefit of using QuickSort is it has an average-case time complexity of O (n log n), making it efficient for sorting large datasets. And QuickSort is an in-place sorting algorithm, which means that it doesn’t require additional memory.

Besides that, we have use Linear Search and we apply in the code using (“SearchReserve”). The reason why we use Linear Search is because it is used for searching reservations based on booking ID. It is straightforward to implement and effective for small to medium-sized datasets. In the context of reservations, the dataset is not expected to be extremely large, making linear search a reasonable choice.

Furthermore, we have use File I/O for Data Persistence and we apply in the code using (“loadData” and “saveData”). It is because File I/O operations are used to load and save reservation data from and to a text file ("booking.txt"). This can allow the program to persist data between runs, providing a simple and durable storage solution.There are some other details we have added which is Static Variables. For static variables, we have applied in code using (“getNewId” and “getLastId”). The use of static variables can ensure that the id for each new reservation is unique across the entire program run.

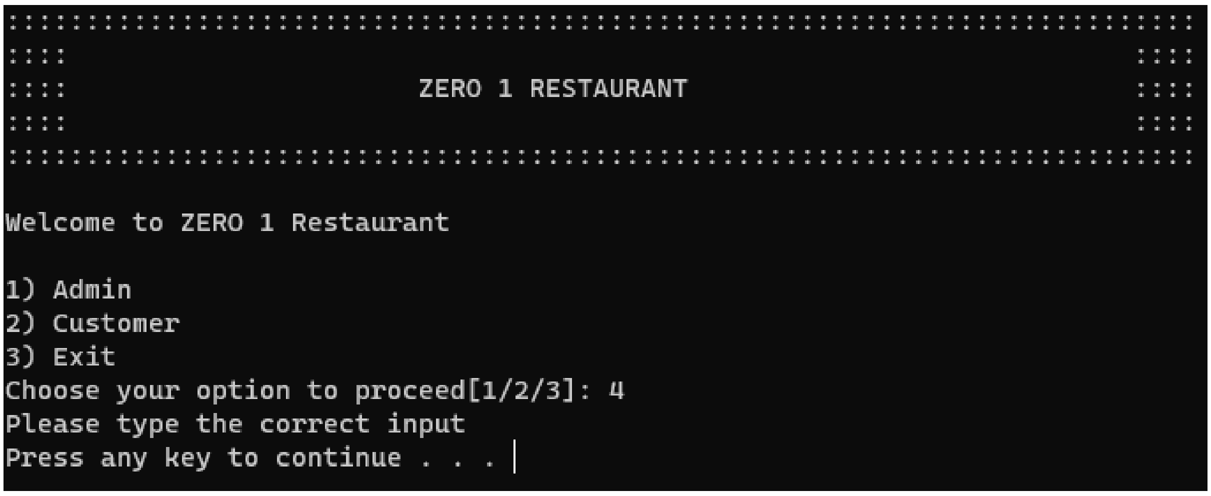
# 5.0 Program Screenshot

## User Login Page



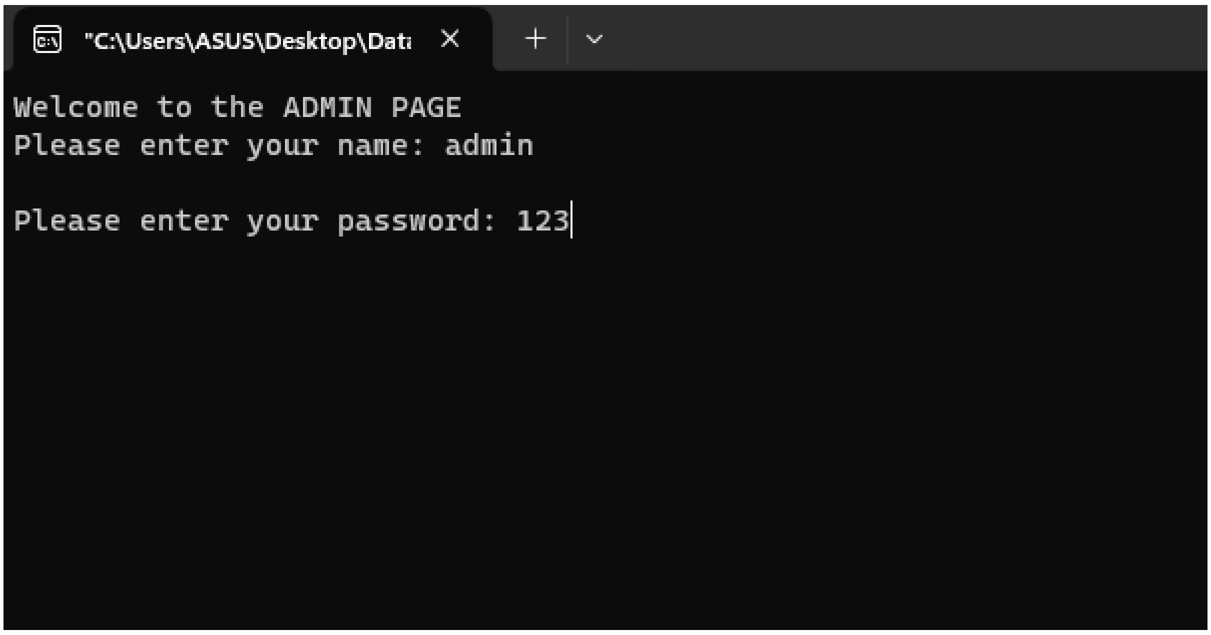
```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::                                                                  ::::
::::                      ZERO 1 RESTAURANT                           ::::
::::                                                                  ::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

Welcome to ZERO 1 Restaurant

1) Admin
2) Customer
3) Exit
Choose your option to proceed[1/2/3]:
```

This is the welcoming page for our restaurant reservation system. The user can choose to use login as admin or customer by entering the correct input. If the user enters 1, then the system will bring the user to admin page and if enter 2, it will bring the user to customer page.



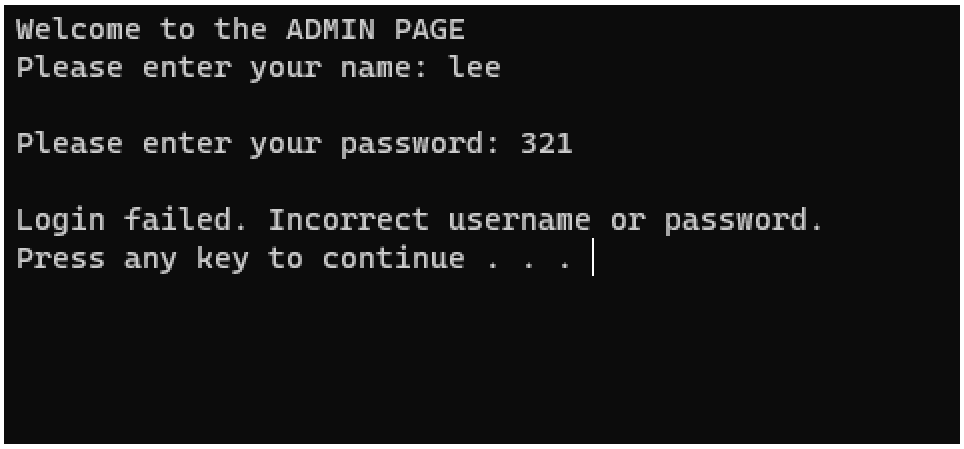
```
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
::::                                                                  ::::
::::                      ZERO 1 RESTAURANT                           ::::
::::                                                                  ::::
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

Welcome to ZERO 1 Restaurant

1) Admin
2) Customer
3) Exit
Choose your option to proceed[1/2/3]: 4
Please type the correct input
Press any key to continue . . .
```

Besides that, if enter 3, the system will be terminated, else the system will return “Please type the correct input” and loop again the welcoming page until the user enters the correct input.



```
Welcome to the ADMIN PAGE
Please enter your name: admin

Please enter your password: 123
```

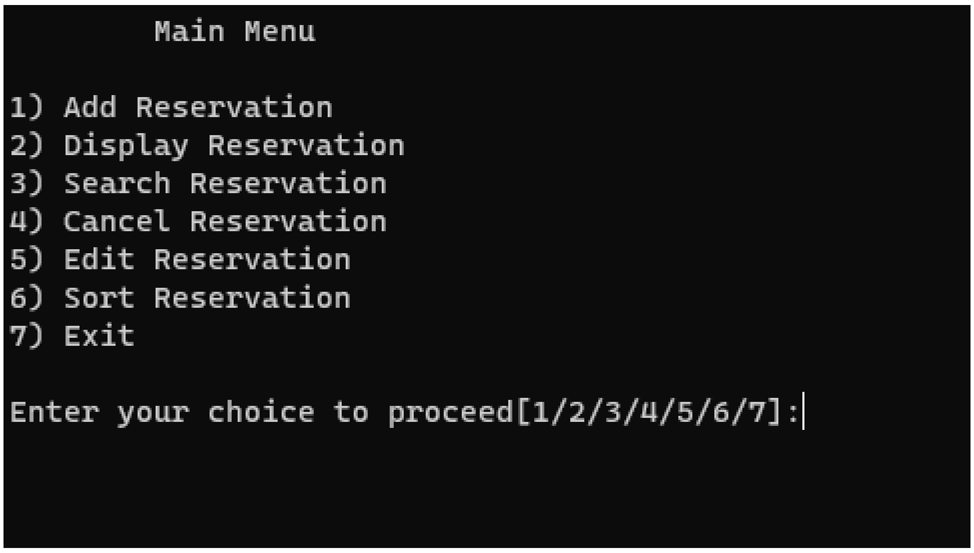
After the user goes to admin page, the system required to enter the correct name and password for login purposes.



```
Welcome to the ADMIN PAGE
Please enter your name: lee

Please enter your password: 321

Login failed. Incorrect username or password.
Press any key to continue . . .
```

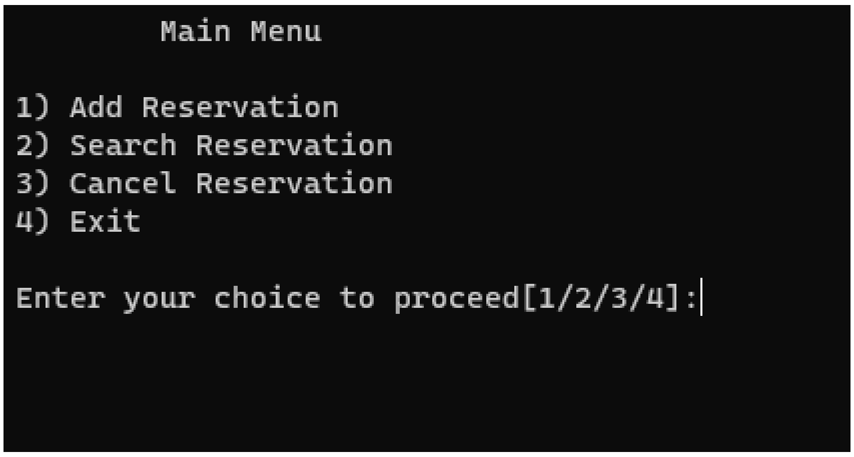
If the user enters the invalid name and password, the system will show "Login failed. Incorrect username or password." And loop again the admin page until the user enters the correct data.

## Main Menu Page



```
        Main Menu

1) Add Reservation
2) Display Reservation
3) Search Reservation
4) Cancel Reservation
5) Edit Reservation
6) Sort Reservation
7) Exit

Enter your choice to proceed[1/2/3/4/5/6/7]:
```

After the user successfully logs in through the admin page, the system will bring the user to the main menu page. The main menu consists of 6 functions for the user to do the management. The functions consist of add, display, search, cancel, edit, and sort reservation. Besides that, if the user wants to log out the admin page, just enter 7 to exit from the main menu in the admin page and go back to the welcoming page.

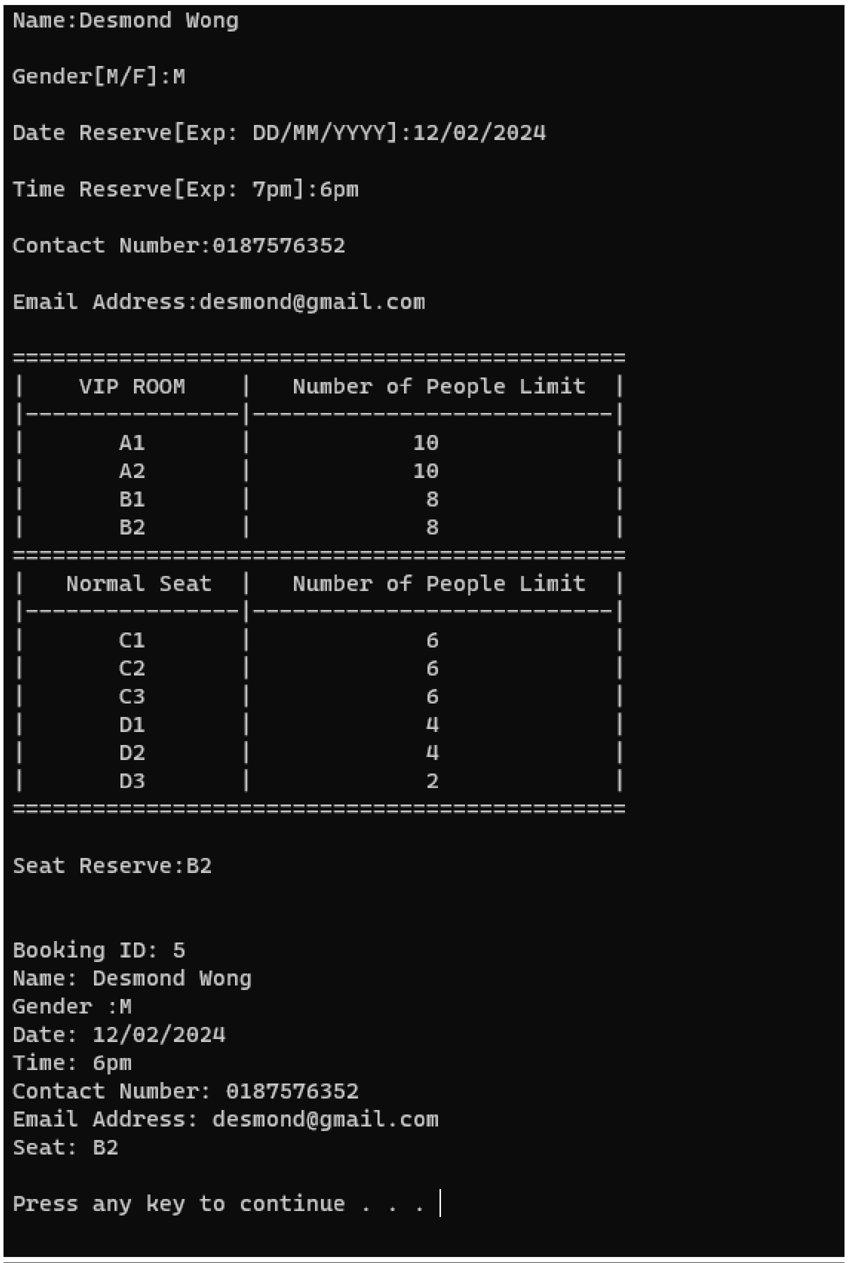


```
        Main Menu

1) Add Reservation
2) Search Reservation
3) Cancel Reservation
4) Exit

Enter your choice to proceed[1/2/3/4]:
```

This the main menu page for customer. The user can directly go to the main menu page if login as customer. There are not required any name or password. But the features can be used in customer page are limited, only add, search, and cancel reservation. Besides that, if the user wants to log out, they also just need to enter 4 to exit from the customer page and return to the welcoming page. The functionality of the 3 functions in the customer page is the same as the functionality in admin page.

## Features Page

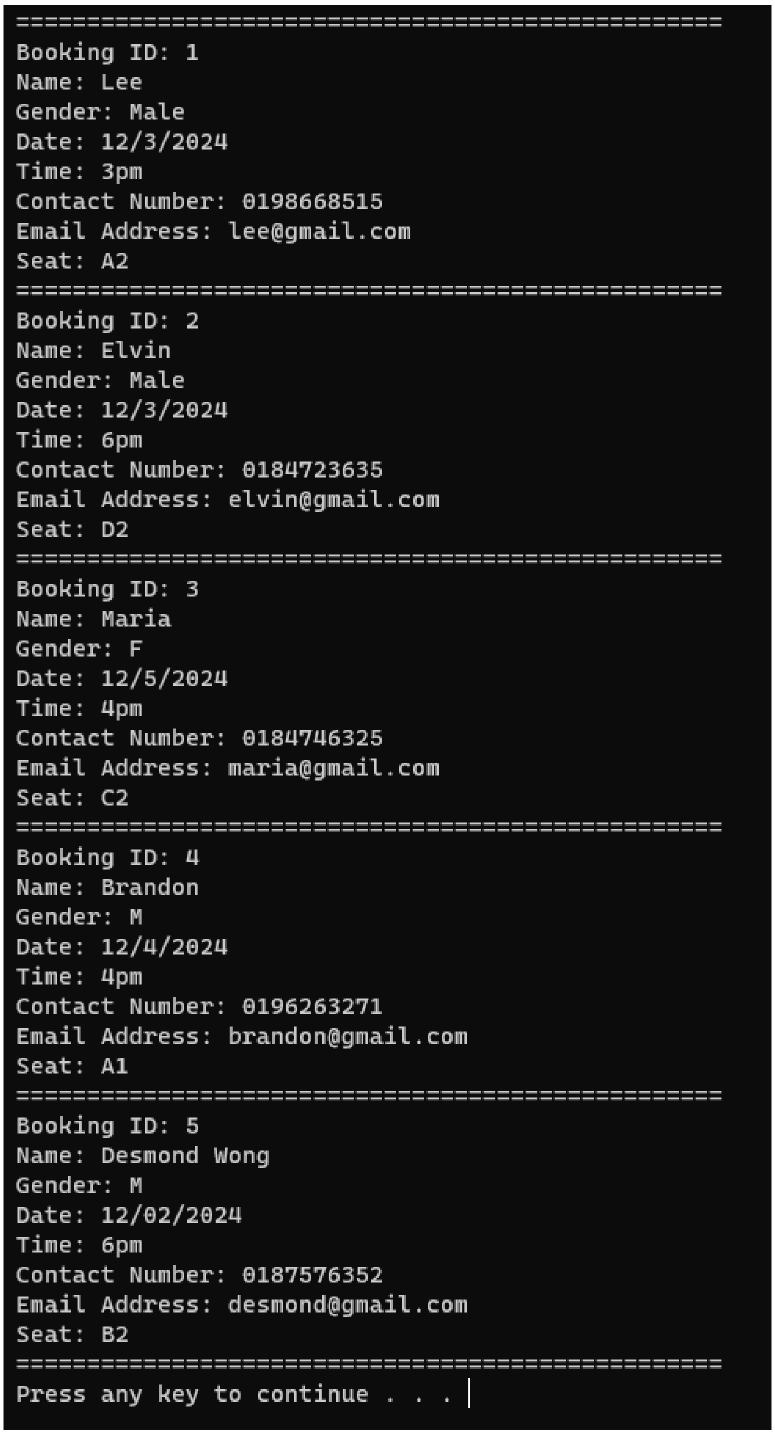
### *Add Reservation Function*



```
Name:Desmond Wong

Gender[M/F]:M

Date Reserve[Exp: DD/MM/YYYY]:12/02/2024

Time Reserve[Exp: 7pm]:6pm

Contact Number:0187576352

Email Address:desmond@gmail.com

==============================================
|    VIP ROOM    |   Number of People Limit  |
|----------------|---------------------------|
|       A1       |            10             |
|       A2       |            10             |
|       B1       |             8             |
|       B2       |             8             |
==============================================
|   Normal Seat  |   Number of People Limit  |
|----------------|---------------------------|
|       C1       |             6             |
|       C2       |             6             |
|       C3       |             6             |
|       D1       |             4             |
|       D2       |             4             |
|       D3       |             2             |
==============================================

Seat Reserve:B2


Booking ID: 5
Name: Desmond Wong
Gender :M
Date: 12/02/2024
Time: 6pm
Contact Number: 0187576352
Email Address: desmond@gmail.com
Seat: B2

Press any key to continue . . .
```

This the add reservation function. When the users go to this function, they need to provide their data to finish the reservation. The data that needs to be provided consists of name, gender, date reserve, time reserve, contact number, email address and seat. After providing all the required data, the system will show the booking details of the user and assign a booking ID for the user.

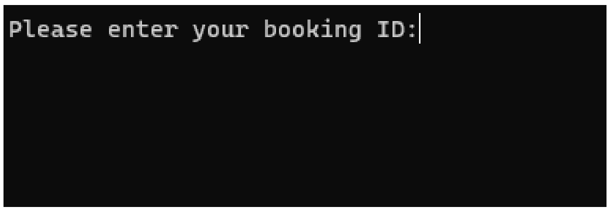
## *Display Reservation Function*

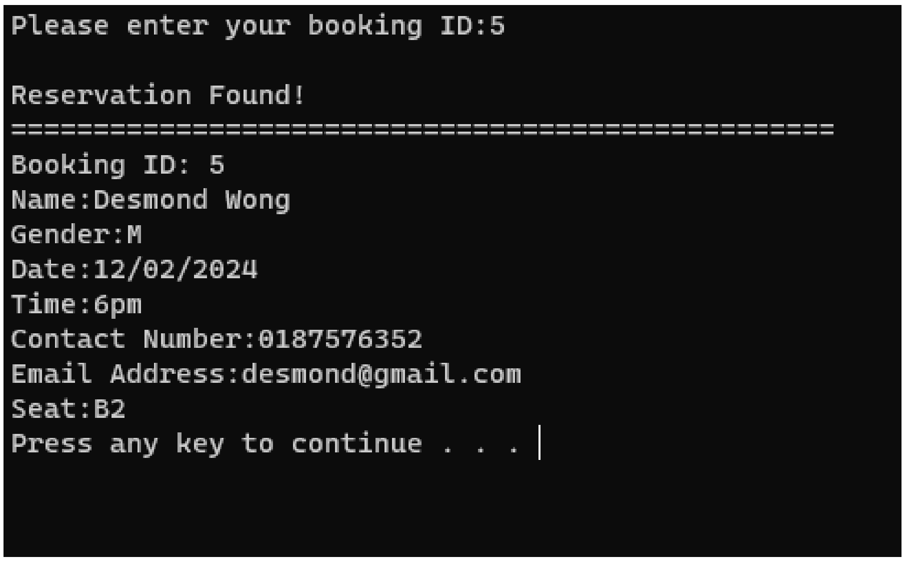


```
==================================================
Booking ID: 1
Name: Lee
Gender: Male
Date: 12/3/2024
Time: 3pm
Contact Number: 0198668515
Email Address: lee@gmail.com
Seat: A2
==================================================
Booking ID: 2
Name: Elvin
Gender: Male
Date: 12/3/2024
Time: 6pm
Contact Number: 0184723635
Email Address: elvin@gmail.com
Seat: D2
==================================================
Booking ID: 3
Name: Maria
Gender: F
Date: 12/5/2024
Time: 4pm
Contact Number: 0184746325
Email Address: maria@gmail.com
Seat: C2
==================================================
Booking ID: 4
Name: Brandon
Gender: M
Date: 12/4/2024
Time: 4pm
Contact Number: 0196263271
Email Address: brandon@gmail.com
Seat: A1
==================================================
Booking ID: 5
Name: Desmond Wong
Gender: M
Date: 12/02/2024
Time: 6pm
Contact Number: 0187576352
Email Address: desmond@gmail.com
Seat: B2
==================================================
Press any key to continue . . .
```

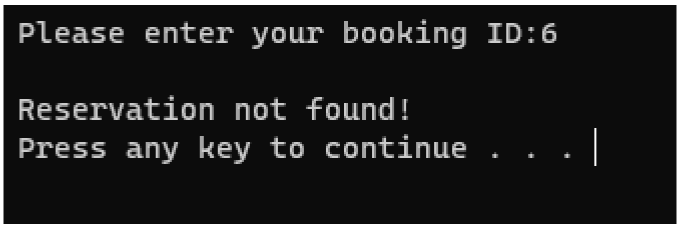
This is the display reservation function. This function will display all the reservation details to help the users to manage the reservation wisely.

## *Search Reservation Function*


```
Please enter your booking ID:
```

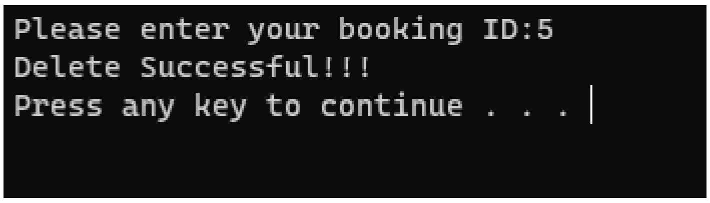

```
Please enter your booking ID:5

Reservation Found!
==================================================
Booking ID: 5
Name:Desmond Wong
Gender:M
Date:12/02/2024
Time:6pm
Contact Number:0187576352
Email Address:desmond@gmail.com
Seat:B2
Press any key to continue . . .
```

This is the search reservation function. This function can help to find the specific reservation details faster. The user needs to input the booking ID, then the system will automatically help to find the corresponding reservation details.


```
Please enter your booking ID:6

Reservation not found!
Press any key to continue . . .
```

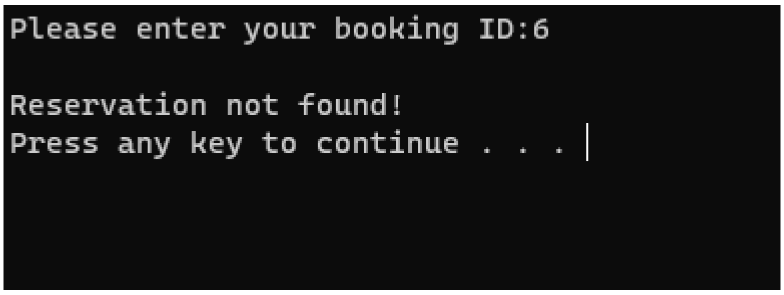
But when the entered booking ID is not tally, the system will show “Reservation not found!”.

## *Cancel Reservation Function*



```
Please enter your booking ID:5
Delete Successful!!!
Press any key to continue . . .
```

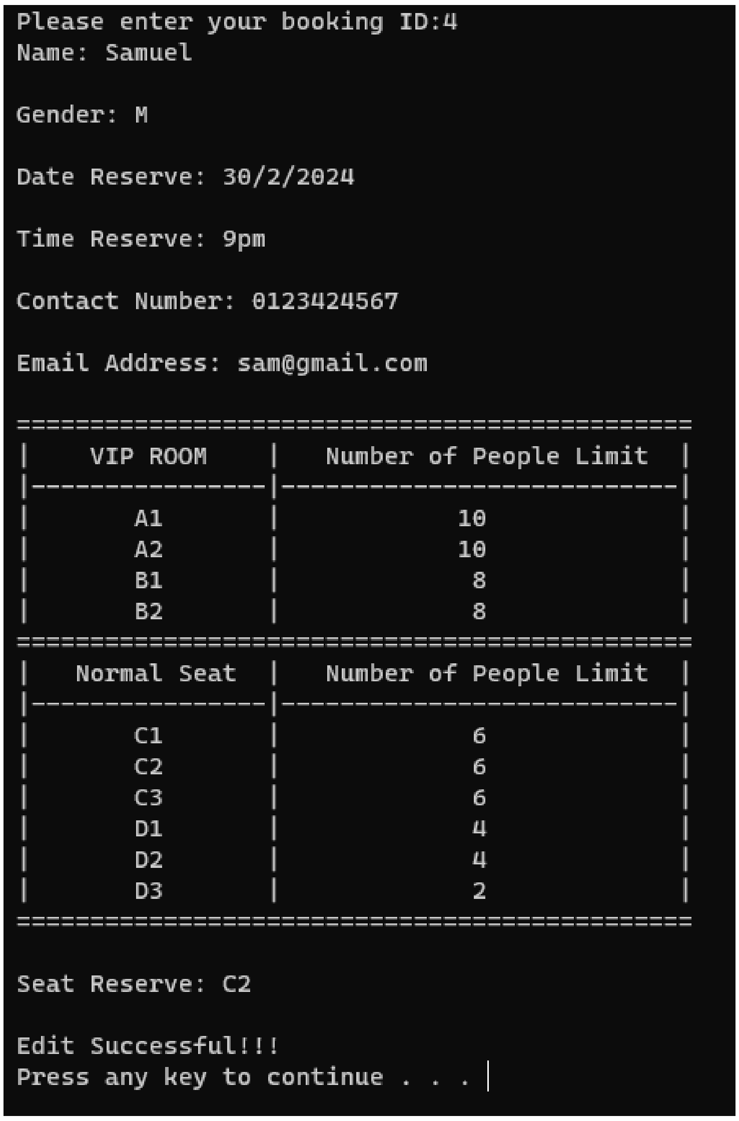
This is the cancel reservation function. The user just needs to enter the booking ID then the corresponding reservation will be deleted. To prove this, user can go to display function to check the result of the cancellation.



```
Please enter your booking ID:6

Reservation not found!
Press any key to continue . . .
```

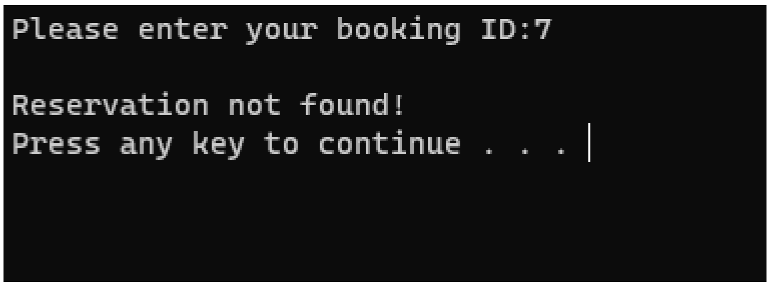
Moreover, when the entered booking ID is not tally, the system will also show “Reservation not found!”.

## *Edit Reservation Function*



```
Please enter your booking ID:4
Name: Samuel

Gender: M

Date Reserve: 30/2/2024

Time Reserve: 9pm

Contact Number: 0123424567

Email Address: sam@gmail.com

==============================================
|    VIP ROOM    |   Number of People Limit  |
|----------------|---------------------------|
|       A1       |            10             |
|       A2       |            10             |
|       B1       |             8             |
|       B2       |             8             |
==============================================
|   Normal Seat  |   Number of People Limit  |
|----------------|---------------------------|
|       C1       |             6             |
|       C2       |             6             |
|       C3       |             6             |
|       D1       |             4             |
|       D2       |             4             |
|       D3       |             2             |
==============================================

Seat Reserve: C2

Edit Successful!!!
Press any key to continue . . .
```

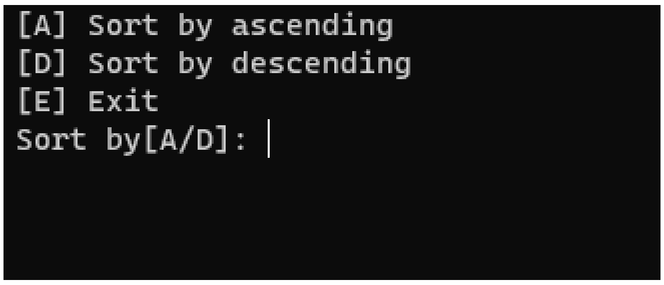
This is the edit reservation function. The user needs to enter the booking ID, then can edit the details of the corresponding reservation. To prove this, the user may go to display function to check the result of the edition.



```
Please enter your booking ID:7

Reservation not found!
Press any key to continue . . .
```

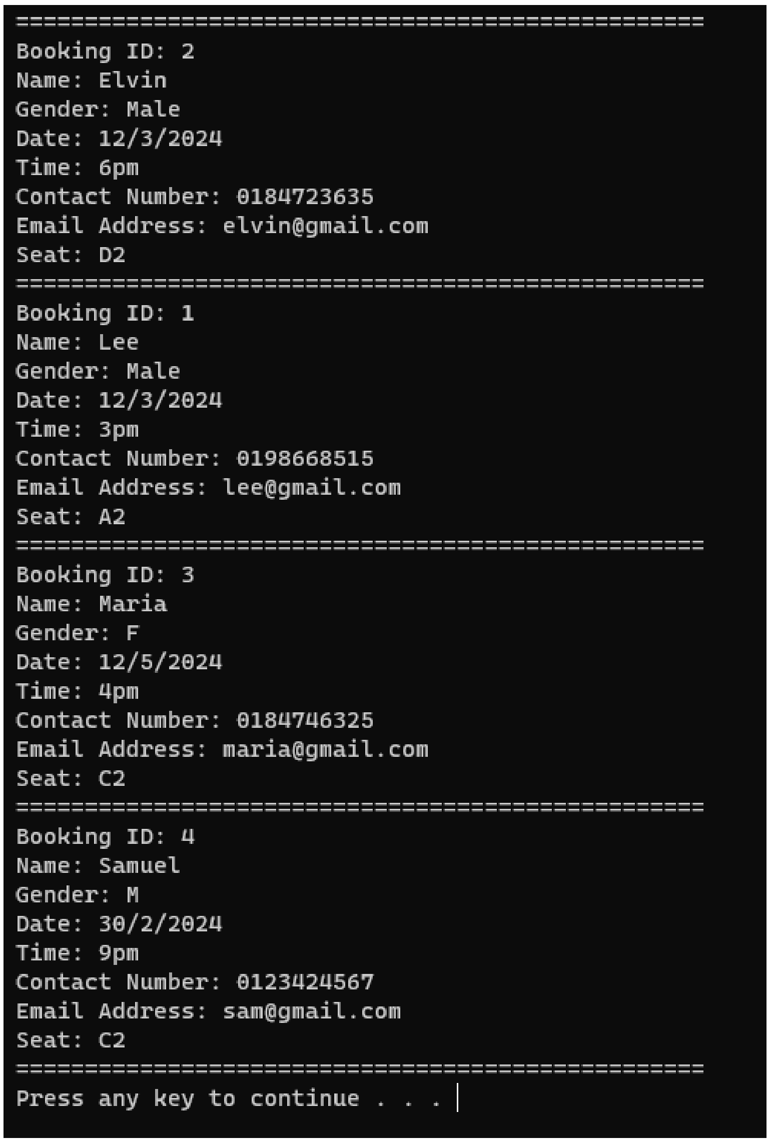
Furthermore, when the entered booking ID is not tally, the system will also show “Reservation not found!”.

## *Sort Reservation Function*



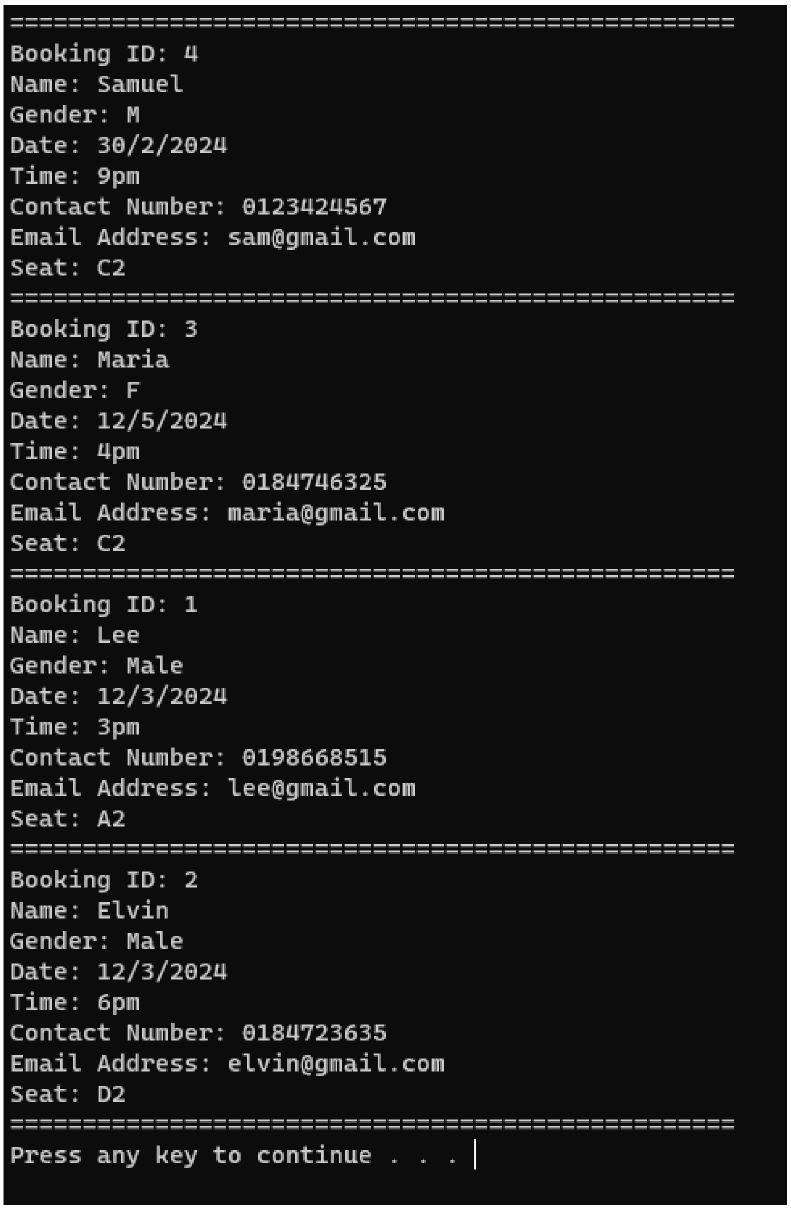
```
[A] Sort by ascending
[D] Sort by descending
[E] Exit
Sort by[A/D]:
```

In this sorting function, the system will sort the reservation by using name detail of the reservation. The user can choose to sort the reservation in ascending or descending order. The user can enter A or a to sort the reservation in ascending order or enter D or d to sort the reservation in descending order.



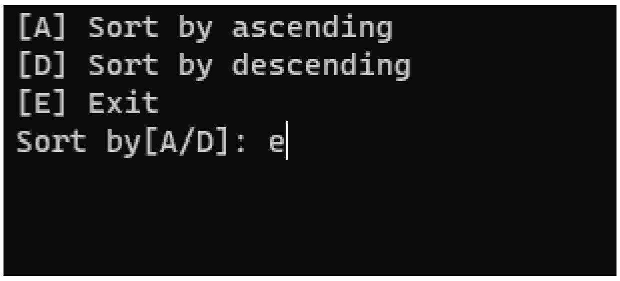
```
==================================================
Booking ID: 2
Name: Elvin
Gender: Male
Date: 12/3/2024
Time: 6pm
Contact Number: 0184723635
Email Address: elvin@gmail.com
Seat: D2
==================================================
Booking ID: 1
Name: Lee
Gender: Male
Date: 12/3/2024
Time: 3pm
Contact Number: 0198668515
Email Address: lee@gmail.com
Seat: A2
==================================================
Booking ID: 3
Name: Maria
Gender: F
Date: 12/5/2024
Time: 4pm
Contact Number: 0184746325
Email Address: maria@gmail.com
Seat: C2
==================================================
Booking ID: 4
Name: Samuel
Gender: M
Date: 30/2/2024
Time: 9pm
Contact Number: 0123424567
Email Address: sam@gmail.com
Seat: C2
==================================================
Press any key to continue . . .
```

This is the result of sort by ascending order.



```
====================================================
Booking ID: 4
Name: Samuel
Gender: M
Date: 30/2/2024
Time: 9pm
Contact Number: 0123424567
Email Address: sam@gmail.com
Seat: C2
====================================================
Booking ID: 3
Name: Maria
Gender: F
Date: 12/5/2024
Time: 4pm
Contact Number: 0184746325
Email Address: maria@gmail.com
Seat: C2
====================================================
Booking ID: 1
Name: Lee
Gender: Male
Date: 12/3/2024
Time: 3pm
Contact Number: 0198668515
Email Address: lee@gmail.com
Seat: A2
====================================================
Booking ID: 2
Name: Elvin
Gender: Male
Date: 12/3/2024
Time: 6pm
Contact Number: 0184723635
Email Address: elvin@gmail.com
Seat: D2
====================================================
Press any key to continue . . .
```

This is the result of sort by descending order.



```
[A] Sort by ascending
[D] Sort by descending
[E] Exit
Sort by[A/D]: e
```

If want to exit from this function, needs to enter e or E.

# 6.0 Task Declaration Forms

**Task Declaration Form**

Each group member, including the group leader, must individually complete and sign this form.
The signed forms (from all members) are to be appended to the final report for submission.

**Restaurant Reservation System**

by

**1B Zero 1**

| | | |
|---|---|---|
| **Group Member's Name** | : | Lee Chuan Tiew |
| **Group Member's ID** | : | 1211101782 |

For the purpose of completing this project, I have performed the following tasks:

- Discuss and distribute the work to my group member.

- Source code compilation.

- Testing the system to find the error and discuss with my member to solve it.

- Record the demonstration video for our system.

1. I hereby declare that I have assessed the final submission and I take the full responsibility should there be any incompleteness, omissions, delays, or non-submission.

2. I also certify that not part of this project is copied from any other student's work, or from any other sources, except where acknowledgement is made.

| | | |
|---|---|---|
| Group Member's Signature | : | Tien |
| Group Member's Name | : | Lee Chuan Tiew |
| Group Member's ID | : | 1211101782 |
| Date | : | 26 January 2024 |

**Task Declaration Form**

Each group member, including the group leader, must individually complete and sign this form.
The signed forms (from all members) are to be appended to the final report for submission.

**Restaurant Reservation System**

by

**1B Zero 1**

| | | |
|---|---|---|
| **Group Member's Name** | : | Justin Lau Li Ting |
| **Group Member's ID** | : | 1211101124 |

For the purpose of completing this project, I have performed the following tasks:

- Discuss with other members to decide on various task.
- Meet with other group members to solve problem/bug meet during the coding process.
- Documentation Compilation & report review
- PowerPoint completion & materials check

1. I hereby declare that I have assessed the final submission and I take the full responsibility should there be any incompleteness, omissions, delays, or non-submission.

2. I also certify that not part of this project is copied from any other student's work, or from any other sources, except where acknowledgement is made.

| | | |
|---|---|---|
| Group Member's Signature | : | [illegible] |
| Group Member's Name | : | Justin Lau Li Ting |
| Group Member's ID | : | 1211101124 |
| Date | : | 26 January 2024 |

**Task Declaration Form**

Each group member, including the group leader, must individually complete and sign this form.
The signed forms (from all members) are to be appended to the final report for submission.

**Restaurant Reservation System**

by

**1B Zero 1**

| | | |
|---|---|---|
| **Group Member's Name** | : | Tan Wei Chuen |
| **Group Member's ID** | : | 1211201347 |

For the purpose of completing this project, I have performed the following tasks:

- Done tasks given by leader
- Discuss with other member about the project
- Participate all the meeting
- Check the error of the program
- Check the error of the report

1. I hereby declare that I have assessed the final submission and I take the full responsibility should there be any incompleteness, omissions, delays, or non-submission.

2. I also certify that not part of this project is copied from any other student's work, or from any other sources, except where acknowledgement is made.

| | | |
|---|---|---|
| Group Member's Signature | : | Tan |
| Group Member's Name | : | Tan Wei Chuen |
| Group Member's ID | : | 1211201347 |
| Date | : | 26 January 2024 |

**Task Declaration Form**

Each group member, including the group leader, must individually complete and sign this form.
The signed forms (from all members) are to be appended to the final report for submission.

**Restaurant Reservation System**

by

**1B Zero 1**

| | | |
|---|---|---|
| **Group Member's Name** | : | Brandon Tan Chi Hong |
| **Group Member's ID** | : | 1211103334 |

For the purpose of completing this project, I have performed the following tasks:

- I have done the implementation details.
- I have done the problem statements.
- I have done the task given by group leader.
- I have check all the parts of final report.

1. I hereby declare that I have assessed the final submission and I take the full responsibility should there be any incompleteness, omissions, delays, or non-submission.

2. I also certify that not part of this project is copied from any other student's work, or from any other sources, except where acknowledgement is made.

| | | |
|---|---|---|
| Group Member's Signature | : | [illegible] |
| Group Member's Name | : | Brandon Tan Chi Hong |
| Group Member's ID | : | 1211103334 |
| Date | : | 26 January 2024 |

# 7.0 Conclusion

In conclusion, the Restaurant Reservation System stands out as a hallmark of efficiency and user-friendliness in the realm of reservation management. This system not only provides an intuitive platform for customers to manage their bookings but also equips administrators with robust tools to oversee and organize reservations effectively. Its success lies in the seamless integration of tailored features, facilitating a streamlined and well-organized booking process. Moreover, the judicious selection of data structures and algorithms underpinning this system deserves commendation. These technological choices have been expertly aligned with the core requirements, striking an optimal balance between performance, simplicity, and user adaptability. Consequently, this system doesn't just meet expectations; it redefines the standards for reservation management in the restaurant industry.