Project

by Eddy Koh Wei Hen

Submission date: 14-Jan-2024 10:19PM (UTC-0800)
Submission ID: 2271216094
File name: DSA_Project.pdf (1.73M)
Word count: 5790
Character count: 27664

UTM

UNIVERSITI TEKNOLOGI MALAYSIA

FACULTY OF COMPUTING

SESSION 2023/2024

SEMESTER 1

SECJ2013-04 DATA STRUCTURE AND ALGORITHM

PROJECT - DATA STRUCTURE AND ALGORITHM

TOPIC: HOTEL BOOKING SYSTEM

LECTURER: DR. LIZAWATI BINTI MI YUSUF

Group Name: The Trio

NO	NAME	MATRIC NO
1	NUR HAFIZAH BINTI JAFRI	A22EC5022
2	EDDY KOH WEI HEN	A22EC0154
3	NURSYUHADA BINTI BADREN	A22EC0253

Table of Content

1. Project Overview

This project is an agent-based hotel booking system that uses classes to represent customers, a queue to manage reservations, and a stack to handle accepted booking requests. There are two characters that use the system, who are the agent and the hotel manager. The agent is the person who helps the customers to do a booking request and the booking list will pass to the hotel manager. The hotel manager will accept the booking request from the list passed by the agent. Customers are characterized by attributes such as name, age, IC number, phone number, room number, and check-in date. The queue class manages the reservation list and offers operations such as adding reservations, searching for customer data obtained in the reservation list, displaying the reservation list, and processing bookings by hotel agents. The stack class handles the list of accepted booking requests, allowing the hotel manager to accept new requests, cancel bookings, find customer data from the accepted list, and display the accepted list. The main function initializes customer data from a file named “Project.txt” and provides a user interface that allows individuals to act as either agents or hotel managers. A loop ensures continuous interaction until the user chooses to exit, with functions being called based on user choices. The program employs both queue and stack classes to manage reservations and accept booking requests.

Using classes in this project has several perks, significantly the idea of encapsulation, which is when classes effectively group relevant methods and related data attributes into separate objects, like the class Customer. Furthermore, class abstraction can be useful in concealing complex implementation details from other components so that users may interact with simplified interfaces. This abstraction helps developers to comprehend code simply by creating a separate division between the system's internal functions and its external interactions. Additionally, the main idea of code reusability is facilitated by the use of classes. Classes help organizations and improve the clarity of the codebase. Whether a class represents a customer or a particular system functionality such as the Queue or Stack, it always operates as a separate entity with a well-defined purpose. This modular architecture improves code comprehension and scalability simply by facilitating the easy addition of new features or functionalities through the creation or modification of classes.

2. Objectives

Our objectives for this project are:

a. To help the agent store customer information in the hotel booking reservation list using queue data structure concept
b. To help the hotel manager to manage the customers' booking request in accepted booking request list using stack data structure concept
c. Keep record of booking details, such as customers' name, age, IC number, phone number, room number and check-in date
d. To allow changeable functions such as push and pop function in stack data structure concept as well as enQueue and deQueue function in queue data structure concept

3. Synopsis of the project

Our agent-based hotel booking system uses two data structures concepts, which are queue and stack. The queue, which is used by the agent, is using pointers in a class called Queue. This design allows the queue to accommodate an unlimited number of customers that want to book a hotel, just like a real situation can. The queue operates on a first-come, first-served basis, meaning the first booking request received is the first one to be processed. Besides, the stack is used by the hotel manager and is using an array in a class called Stack. This design reflects the reality that a hotel has a limited number of rooms. The stack operates on a last-in, first-out basis, meaning if there's an issue with a room, the most recent booking can be canceled first. To be concluded, our agent-based hotel booking system uses a queue to handle customer requests and a stack to manage room availability. This setup ensures a fair process for customers and efficient management for the hotel.

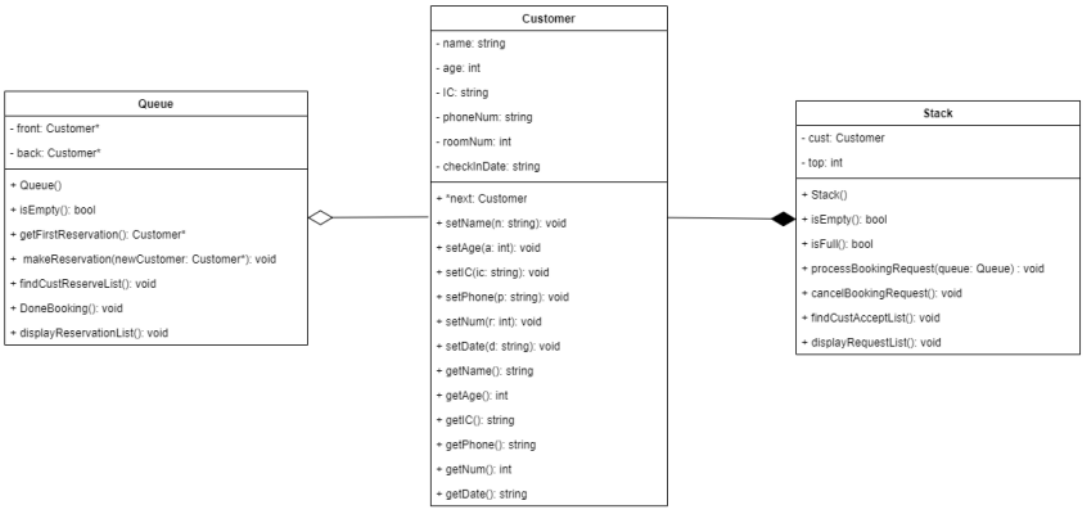
4. Design

Link for Class Diagram and Flow Chart:
https://app.diagrams.net/#G1qRs-q7iZpkrkau2ri2tI_dkmzvK0DKnU

4.1 Class Diagram

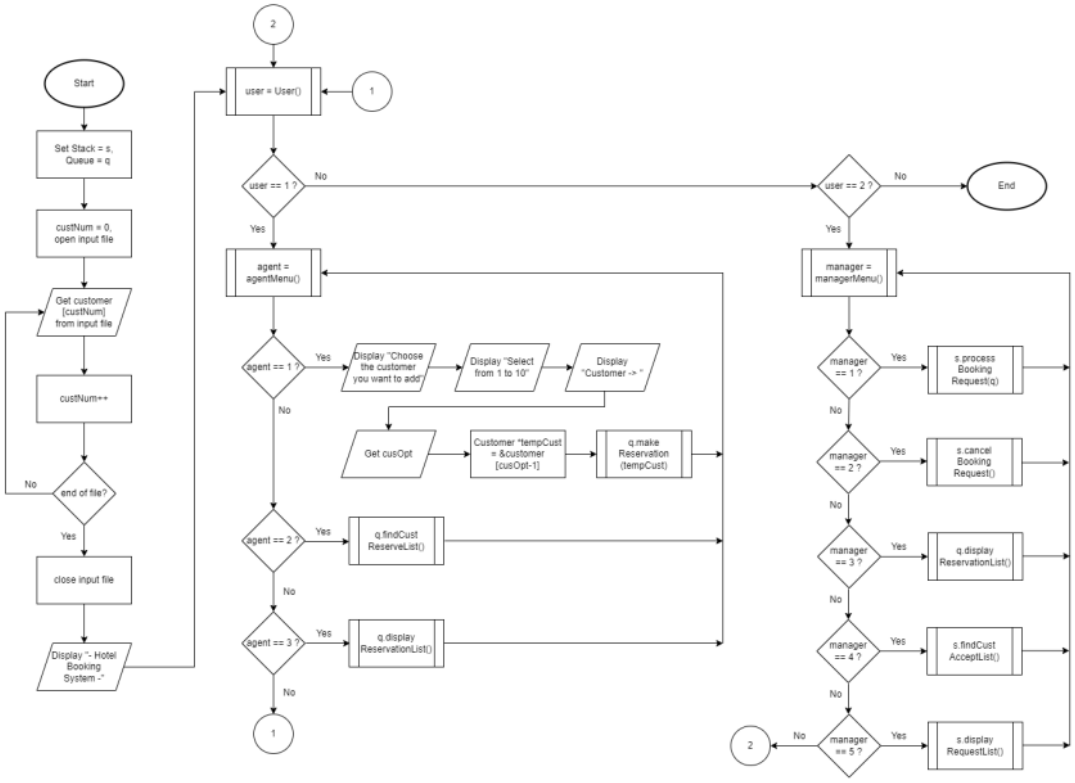
Description

Class Queue : Used by agent and hotel manager (Implement queue data structure concept)
Class Stack : Used by hotel manager (Implement stack data structure concept)
Class Customer: Store customer data via queue and stack

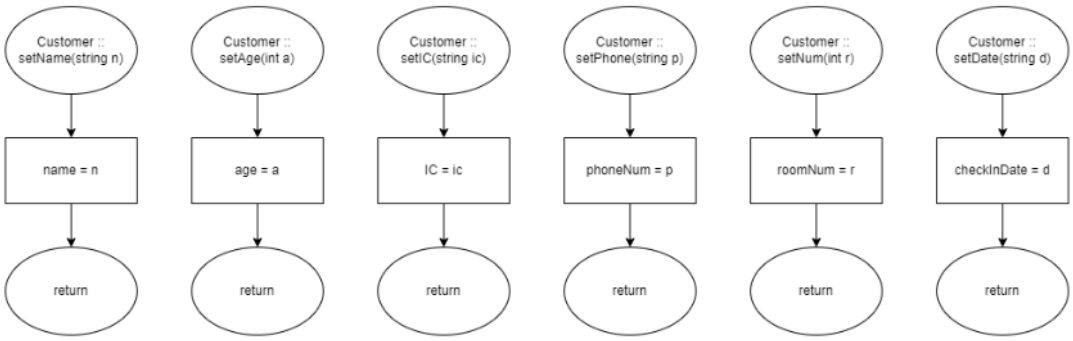


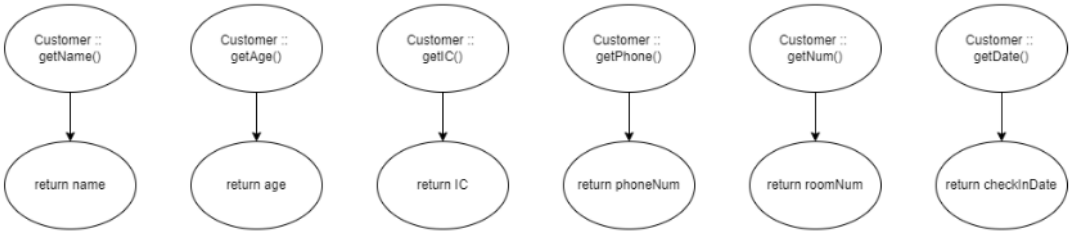
4.2 Flowchart

4.2.1 Main Body

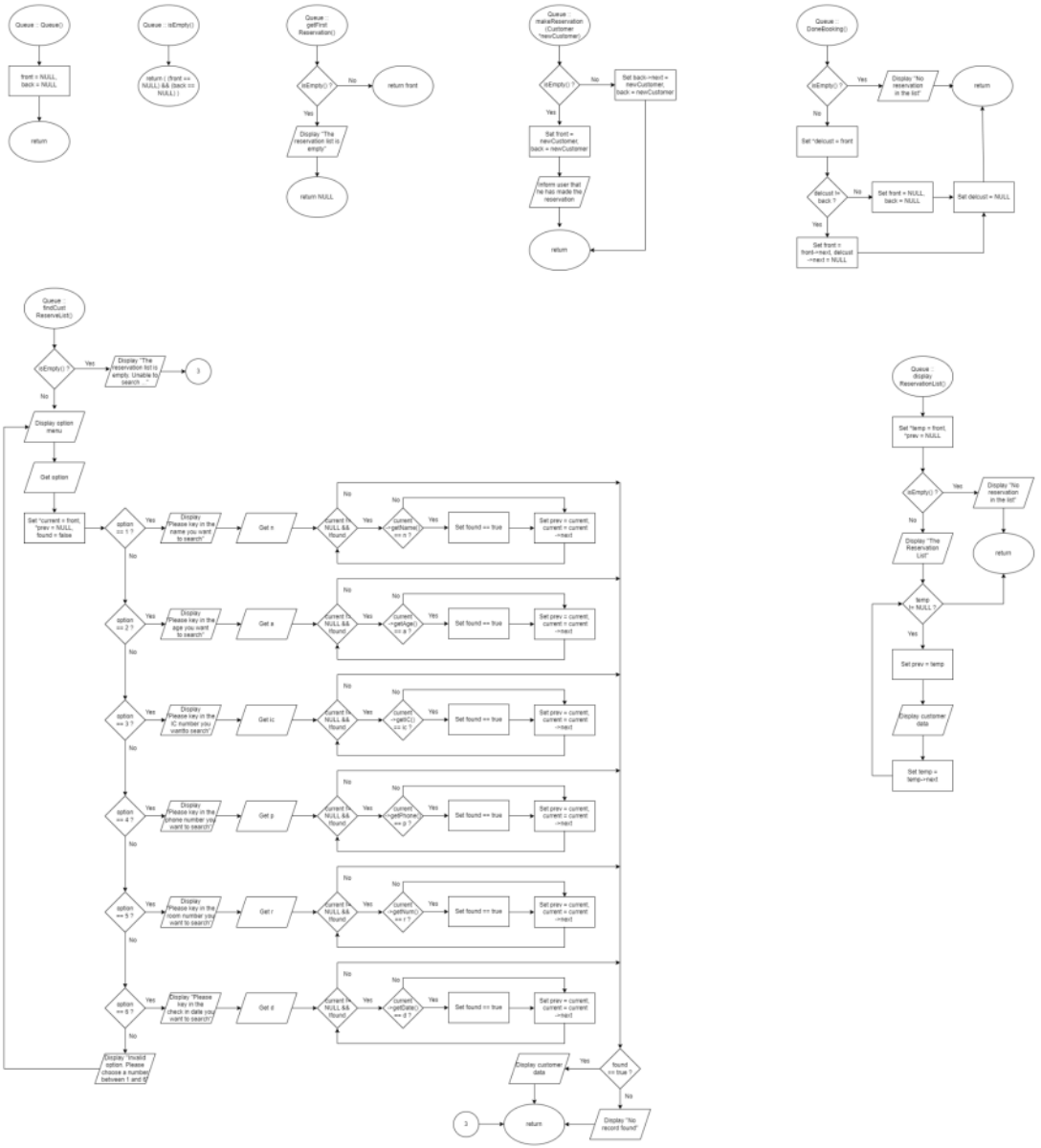


4.2.2 Function in Class Customer

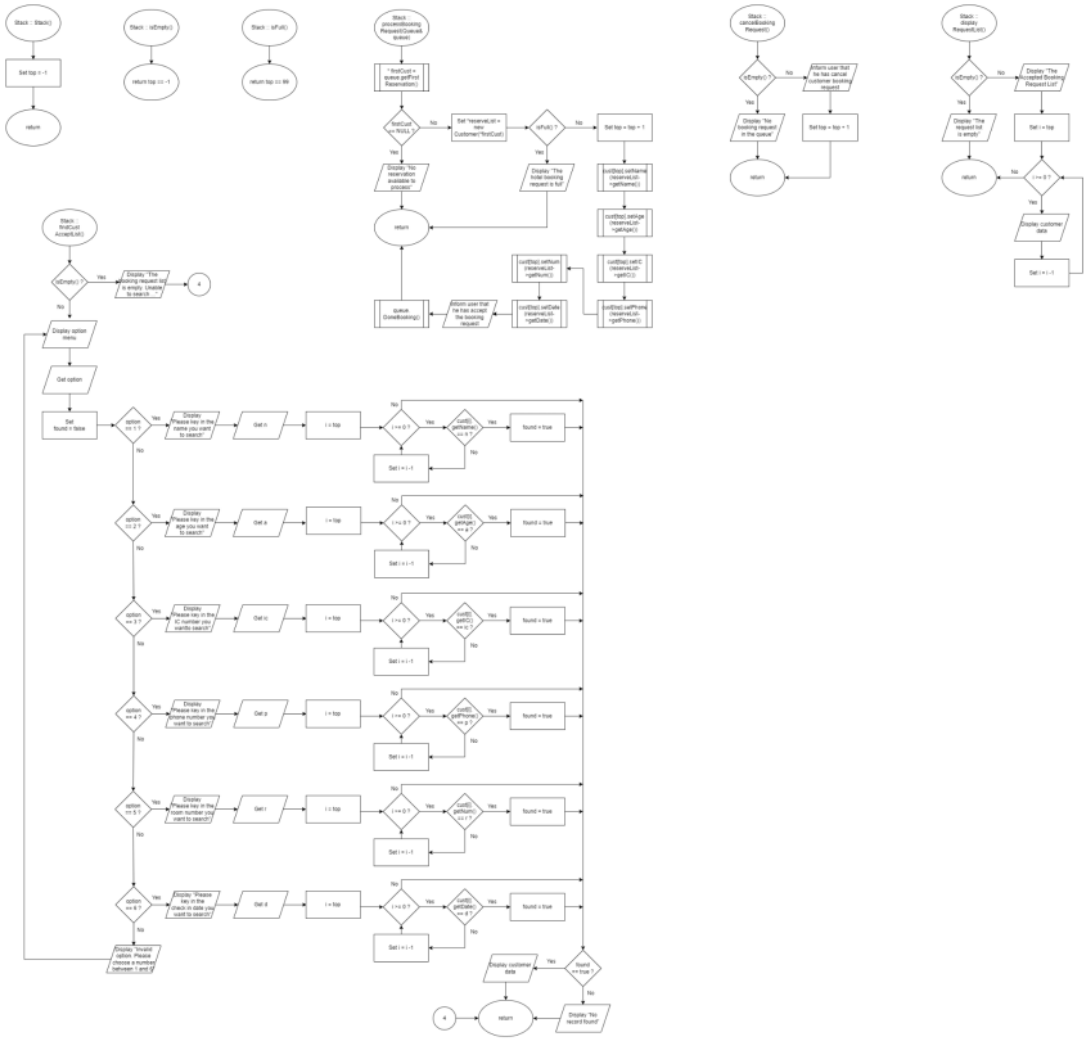




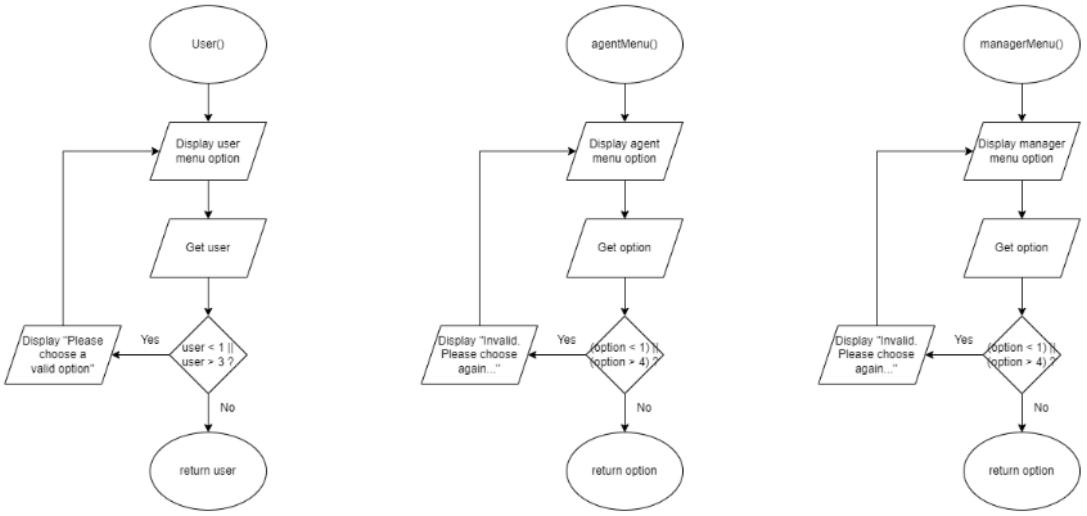
4.2.3 Function in Class Queue



4.2.4 Function in Class Stack



4.2.5 Stand Alone Function



5. Data structure concept implementation

In this project, our group has implemented two data structure concepts for the agent-based hotel booking system, which are the queue data structure concept and the stack data structure concept.

5.1 Queue Implementation

We have implemented a queue in the class Queue. The queue is in pointer style, in which all the functions inside the class are done using pointers. The class queue is used by the agent for this system. The advantage of implementing a pointer style is that it does not have a size limit. It acts as a reality where the agent just helps the customer book the hotel and does not have limitations on how many customers he wants to help. The concept also applies to agents because the queue is first in, first out, so when the hotel manager accepts the booking request, the first reservation will be deleted from the list. To conclude, first come, first served is applied.

First, we declare two private attributes for the class, which are *front and *back. For *front, it is used to point the first data in the queue and for *back, it is used to point the last data in the queue.

In public, we have declared several functions:

1. Queue()
    - It act as a constructor for the class Queue
    - When we declare a class Queue in main body, we initialize the *front and *back pointer point to NULL because we haven't insert any customer data inside the class

2. bool isEmpty()
    - The function is used to detect whether there is customer data inside the class or not
    - We detect the emptiness of the class by returning ((front == NULL) && (back == NULL))
    - If either one of the pointers is pointing to NULL, the function will return true
    - The function will return false if both pointers are not pointing to NULL

3. Customer* getFirstReservation()
    - The function is used to get the first customer data in the class Queue
    - We detect the emptiness of class using the isEmpty() function
    - If the class does not contain any customer data, it will display "The reservation list is empty"
    - If the class contains customer data, it will return the first customer data by using pointer *front

4. void makeReservation(Customer *newCustomer)
    - The function acts as enQueue() which means enter data into the class
    - The parameter of (Customer *newCustomer) is pointing a customer data from main body
    - We detect the emptiness of class using the isEmpty() function
    - If the class does not contain any customer data, the new customer data will be the first data insert in the class by pointing *front and *back to the data

- If the class contains customer data, the *next pointer (declared in class Customer) of the *back pointer will point to the new customer which means the new customer data will be arranged after the last data inside the class. Then, the *back pointer will point to the new inserted data to become it the last data inside the class

5. void findCustReserveList()
    - The function is used to search the customer data obtained in the class by specific keyword
    - We detect the emptiness of class using the isEmpty() function
    - If the class does not contain any customer data, it will display "The reservation list is empty. Unable to search …" then return to main body
    - The function will display option menu for user (agent) and get the option from user
    - We declare pointer *current equal to front (point to first data), *prev equal to NULL for purpose later
    - We also declare a "bool found = false" to check the presence of customer data inside the class. "False" means the data haven't found yet
    - Enter the switch case, if option is other than 1 to 6, it will looping to display option menu again and get user option
    - If user key in option = 1, the function will display "Please key in the name you want to search" and get the n (name) key in by user. The line "while((current != NULL) && (!found))" is used for looping if haven't got the same name inside the class as the name input by the user. It will continue looping if *current not equal to NULL and found = false. The pointer *current will start from first data inside the class, it will detect the current pointed data is same name as user key in or not, if same found = true, then *prev will point to *current to obtain the data pointed by *current and *current will point to its next data, as found = true, the loop will break. If not found the name same as user input, the function will continue looping until *current = NULL means the end of data inside the class, found will not equal to true as all the data not same as user input
    - For case just now, if obtain the same name (found == true), the function will start to display customer data using *prev pointer. The reason of using *prev instead of

using *current is because while *current pointing name is same as user input, found change to true but the loop is continued, *prev have equal to *current and *current has pointed to the next data. Therefore, the *prev has pointed to the name same as user input and *current has pointed to other data. If not obtain the same name which means found remain false, the function will display “No record found”

- The same concept is apply to option = 2 to 6 and 2 is for searching age, 3 for searching IC number, 4 for searching phone number, 5 for searching room number and 6 for searching check in date

6. void DoneBooking()
    - The function acts as deQueue() which means delete the first data in the class
    - We detect the emptiness of class using the isEmpty() function
    - If the class does not contain any customer data, the function will display “No reservation in the list”
    - If the class contains customer data, we declare a *delcust pointer = *front pointer for purpose later
    - (delcust != back) means there are many customer data in the class since at first we insert the first data by point *front and *back to the new inserted data, to insert again data *front remain and *back point to the new inserted data.
    - If the class contains more than one customer data, we set *front pointer point to the next data inside the class and the *next pointer of the *delCust point to NULL to break the line for *delCust pointed customer data between the queue
    - If the class contains only one customer data, we just set *front and *back point to NULL to delete the customer data from the queue
    - *delCust point to NULL to prevent the system confuse which customer data is in the queue

7. void displayReservationList()

- We declare pointer of *temp = *front and *prev = NULL for purpose later
- We detect the emptiness of class using the isEmpty() function

- If the class does not contain any customer data, the function will display “The reservation list is empty”
- If the class contains customer data, the *prev will point to *temp pointed data. The customer data will be display by using *prev pointer. After display the customer data, *temp will point to the next data inside the queue. The display process will continue until *temp is point to NULL which means the end of data inside the class

5.2 Stack Implementation

We have implemented a stack in the class Stack. The stack is in array style in which all the customer data is stored using an array. The class stack is used by the hotel manager for this system. The implementation of an array style is due to the size limit. It acts as a reality where the hotel has a limited number of rooms and cannot accept all the booking requests if the room is full. The concept also applies to the hotel manager because the stack is last in first out, it modifies the situation of if the room that customer booked has faced a problem like the room has been booked by previous customer, the hotel manager who accepted the customer booking request, can cancel the booking request immediately.

First, we declare two private attributes for the class which are cust[100] and top. For cust[100], it means the hotel can maximum store 100 customer data and the hotel has only 100 rooms. For the top attribute, it is declared to know current customer data the hotel has.

In public, we have declared several functions:

1. Stack()
    - It act as a constructor for the class Stack
    - When we declare a class Stack in the main body, we initialize the top equal to -1 because we haven’t inserted any customer data inside the class. As the array is counted start from 0, if we insert one data, top will become 0 and the data can be stored in the array

2. bool isEmpty()
    - The function is used to detect whether there is customer data inside the class or not
    - We detect the emptiness of the class by returning (top == -1)
    - If top equal to -1, the function will return true
    - The function will return false if the top is a positive number or 0

3. bool isFull()
    - The function is used to detect whether the customer data inside the class is full or not
    - We detect the fullness of the class by returning (top == 99), the reason of top == 99 instead of top == 100 is because the array starts from 0
    - If top equal to 99, the function will return true
    - The function will return false if the top is not equal to 99

4. void processBookingRequest(Queue& queue)
    - The function acts as push() which means enter data into the class
    - The parameter (Queue& queue) is used as we want to obtain customer data from class Queue. “&” is used to update the data change for class Queue
    - We detect the fullness of class using the isFull() function, if it is full, we cannot insert any data and the system will display “The hotel booking request is full”
    - If the system is not full, the value of top will increase by one for purpose storing the customer data into the array at assigned position
    - The customer data is being inserted inside the array of stack via mutator in class Customer
    - The queue.DoneBooking() function is called to delete the inserted data from class Queue. It modify the real situation of the accepted booking request will be delete from the reservation list

5. void cancelBookingRequest()
    - The function acts as pop() which means pop out the data from stack
    - The isEmpty() is used to detect the emptiness of class

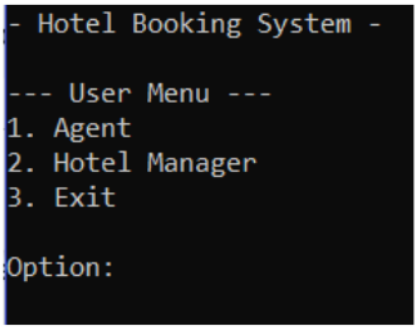
- If the class is empty, the system will display “No booking request in the queue” and do nothing
- If the class has customer data, the function will pop out the last data in the stack

6. void findCustAcceptList()
    - The function is used to search the customer data obtained in the class by specific keyword
    - The concept of this function is same as void findCustReserveList() in class Queue just the pointer style has change to array style

7. void displayRequestList()
    - The function is used to display all the customer data in the class Stack
    - The concept of this function is same as void displayReservationList() in class Queue just the pointer style has change to array style

6. Source Code Demonstration about Data Structure Concept Employed and User Guide

The following diagram show the interface of Agent-Based Hotel Booking System

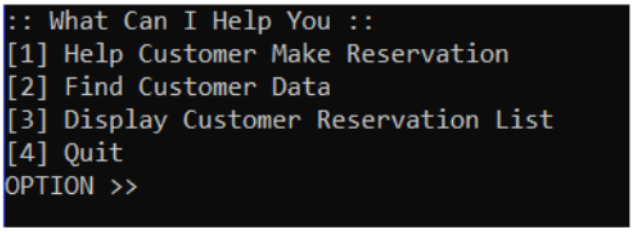


Scene 1: Main menu

Scene 1:

The user needs to choose the option by inserting values 1 to 3. If the user inputs the value 1, the system will show the agent interface. If the user inputs the value 2, the system will show the hotel manager interface. If the user inputs the value 3, it will quit the program. If the user inputs the option value other than 1 to 3, the system will inform the user to choose the option again.

The following diagram shows the menu of the agent interface after the user inputs the value 1 and the user will act as an agent

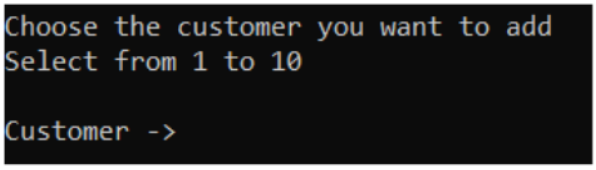


Scene 2: Agent Menu interface

Scene 2:

The agent needs to choose the option by inserting values 1 to 4. If the agent inputs the value 1, he will help the customer to make a reservation. If the agent inputs the value 2, the system will help him to search the specific customer data from the customer reservation list. If the agent inputs the value 3, the system will display all the customer and the customer data currently obtained in the customer reservation list. If the agent inputs the value 4, the system will back to the main menu. If the agent inputs a value other than 1 to 4, the system will inform the user to choose the option again.

The following diagram shows the interface after agent inputs value 1 in the agent menu

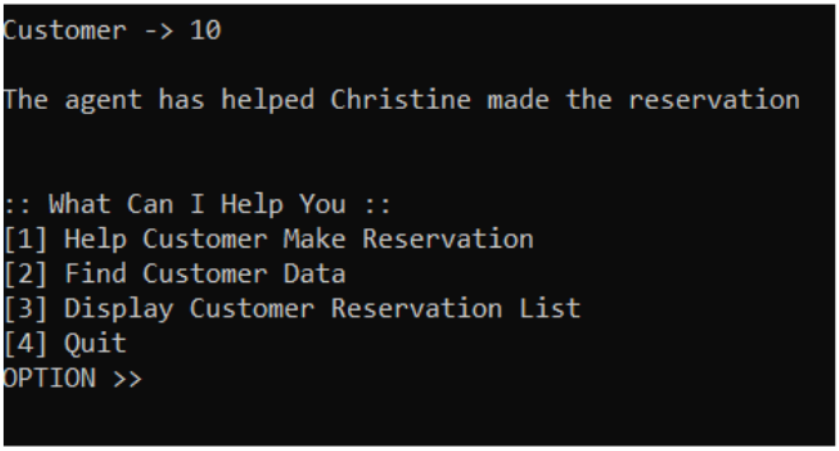


Scene 3: Interface after agent inputs value 1 in the agent menu

Scene 3:

We provide 10 customers for the system user to test inserting the customer data so the system user does not need to key in one by one. All the customer data is obtained from the input file. For the code behind this scene, it implements the queue data structure concept and acts as enQueue() of queue concept. The agent can choose from Customer 1 until 10 to get different customers and insert them into the customer reservation list.

The following diagram show the output after agent inputs the customer he chose

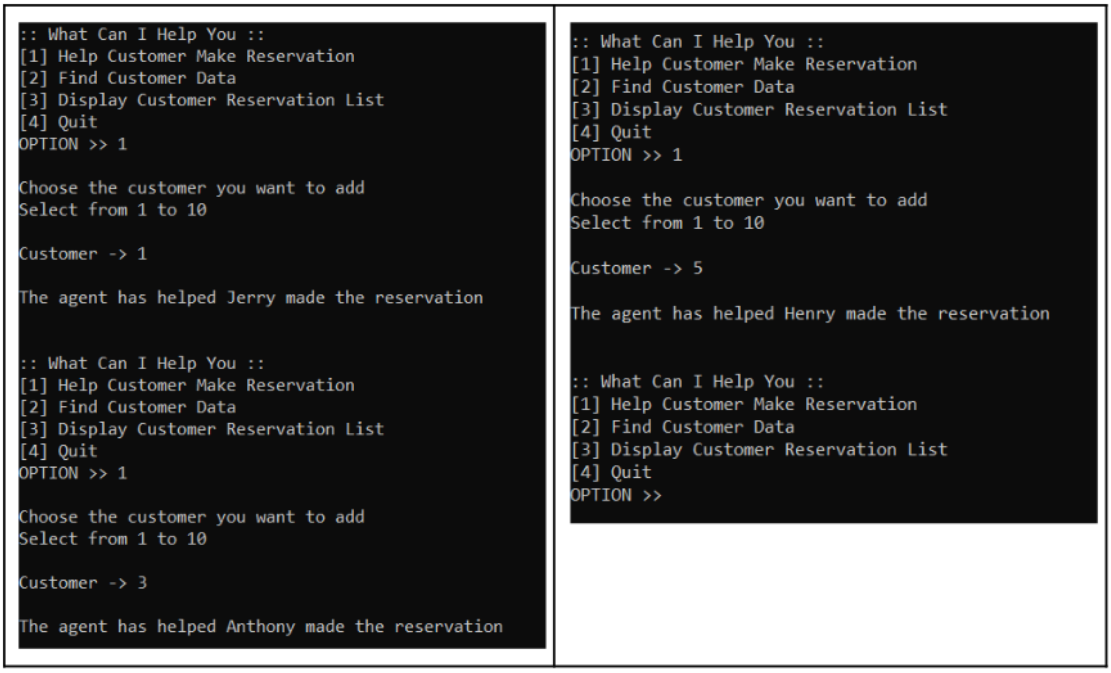


Scene 4: Output after agent inputs customer data

Scene 4:

The scene shows the agent has input Customer 10 data and customer data has been stored in the customer reservation list (Queue). The Customer 10 names Christine. After inserting the customer data, the system will go back to the agent menu interface and the agent needs to choose the option again.

The following diagram show the output after the agent continues to input several customer data

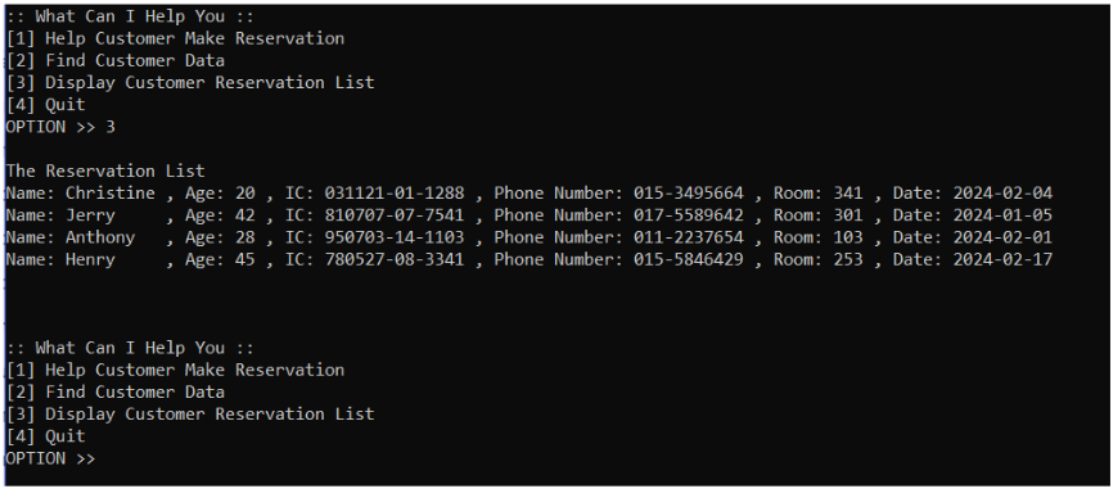


Scene 5: Agent inserts several customer data

Scene 5:

The scene shows the agent continues to insert the customer data. The customer data is stored and arranged in a queue. The first customer in the queue is Christine and the last person in the queue is Henry.

The following diagram show the result after agent inputs value 3 in the agent menu

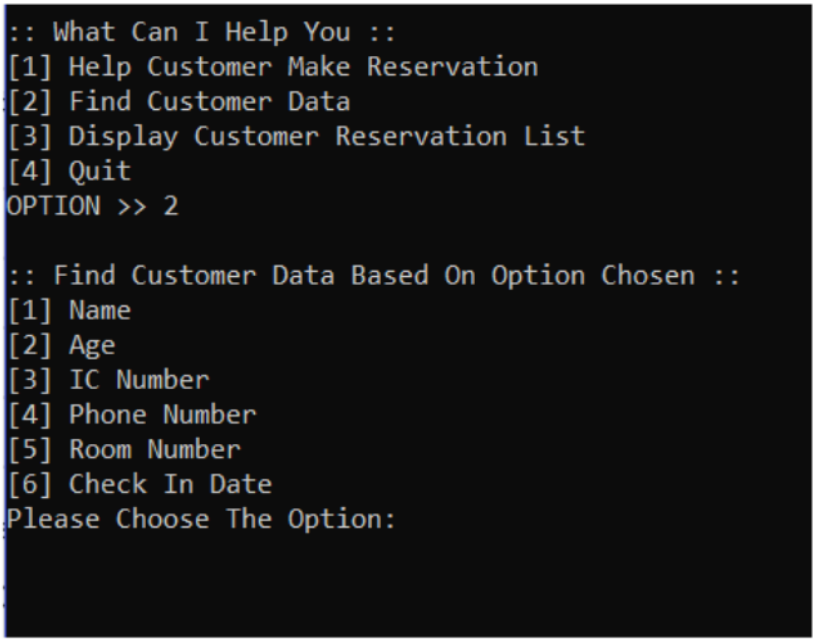


Scene 6: Result after agent inputs value 3 in the agent menu

Scene 6:

The scene shows the result after agent inputs value 3 in the agent menu. The system will start displaying the first customer data until the end of customer data in the queue. The system will return to the agent menu interface after displaying the customer data.

The following diagram show the output after agent inputs value 2 in the agent menu

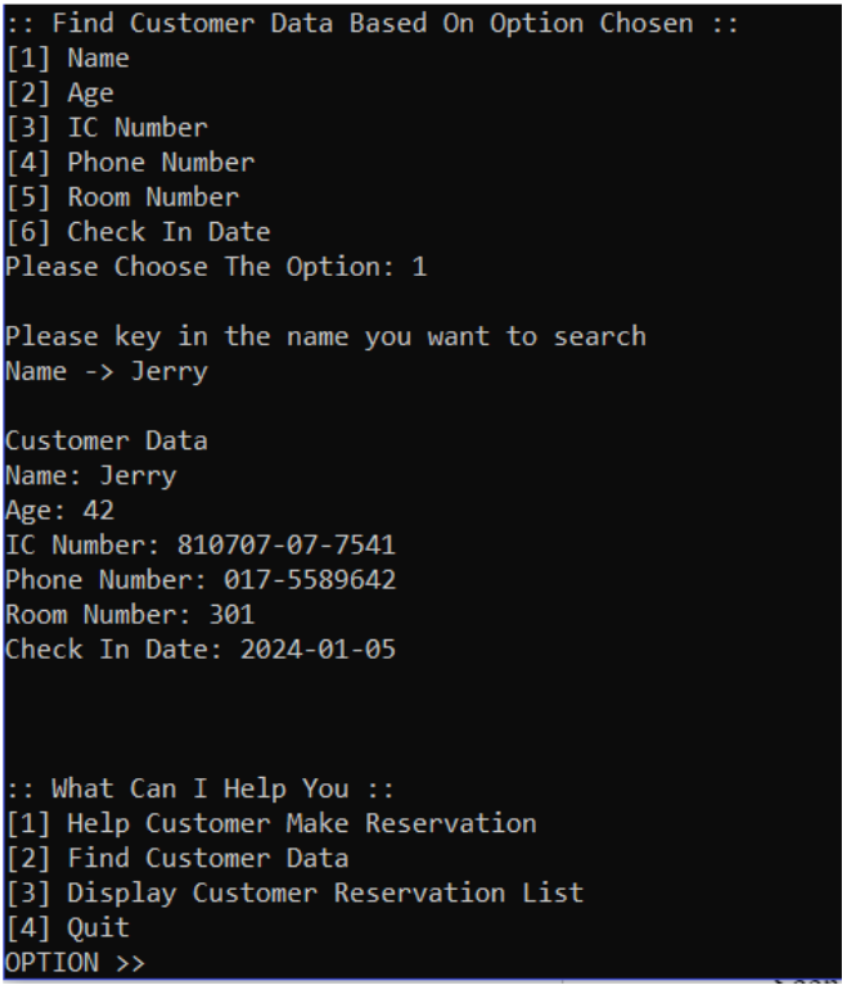


Scene 7: Output after agent inputs value 2 in the agent menu

Scene 7:

The scene shows the option that helps the agent to find customer data based on keywords. The feature is used to find the customer data that has already been obtained in the customer reservation list. If agent inputs option value 1 means he wants to search the specific customer data based on name. If agent inputs option value 2 means he wants to search the specific customer data based on age. If agent inputs option value 3 means he wants to search the specific customer data based on IC number. If agent inputs option value 4 means he wants to search the specific customer data based on phone number. If agent inputs option value 5 means he wants to search the specific customer data based on room number. If agent inputs option value 6 means he wants to search the specific customer data based on check in date. If agent inputs option value other than 1 to 6, the system will inform agent to inputs option again. The system will help the agent to find the customer data that already obtained the customer reservation list (Queue). The system will find from the first customer that agent inserts until the last customer that agent inserted.

The following diagram show the output after agent inputs name to find specific customer data

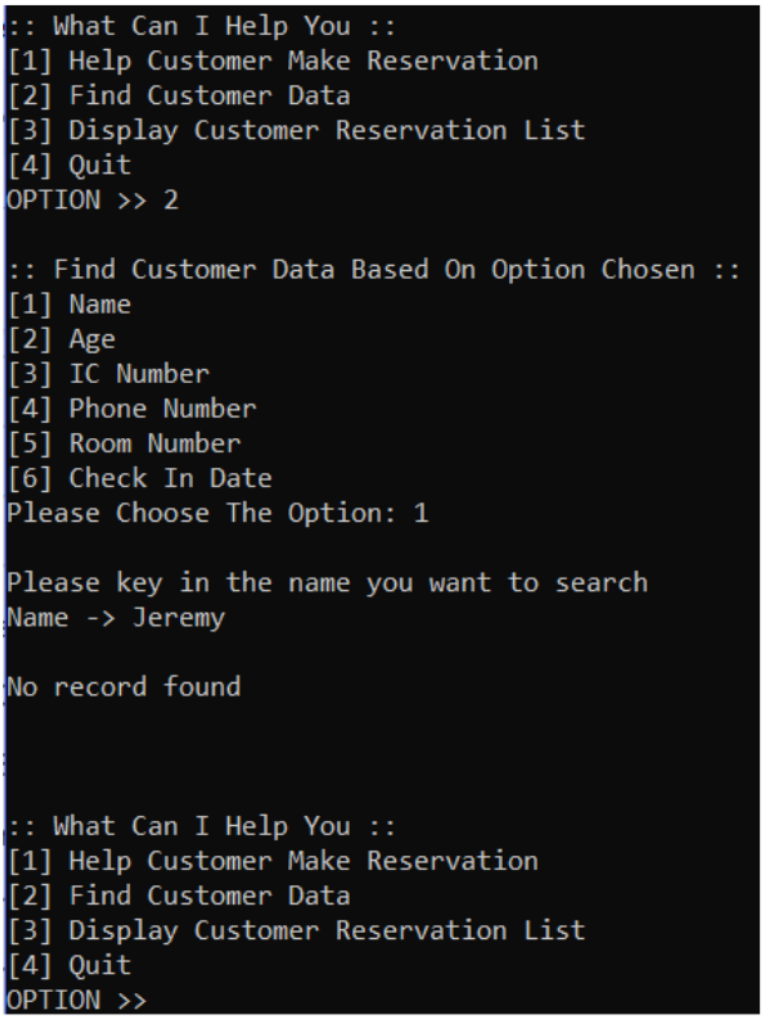


Scene 8: Output after agent inputs name to search specific customer data

Scene 8:

The following scene shows that the agent wants to search customer data based on customer name. When the agent inputs a name that has already been obtained in the customer reservation list, the system will display the specific customer data based on the name that agent inputs. Then, the system will return to the agent menu interface.

The following diagram show the output after agent inputs the name that not obtained in the customer reservation list

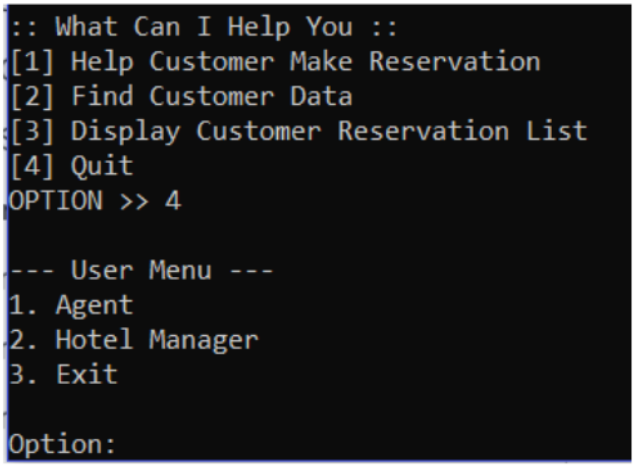


Scene 9: Output after agent inputs the name that not obtained in customer reservation list

Scene 9:

The system displays “No record found” to the agent because there is no customer data of name Jeremy inside the customer reservation list. Then the system will return to the agent menu interface.

The following diagram show the interface after agent quit the agent menu interface

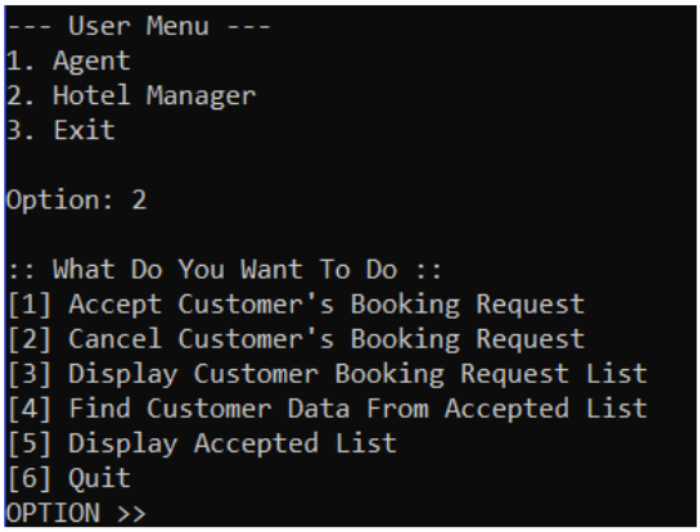


Scene 10: Interface after agent quit from agent menu interface

Scene 10:

When the agent quits from the agent menu interface, the system will return to the main menu interface. The agent will become a normal user. The user can choose the option either he is an agent or hotel manager or he wants to exit the system.

The following diagram shows the menu of the hotel manager interface after the user inputs the value 2 and the user will act as a hotel manager

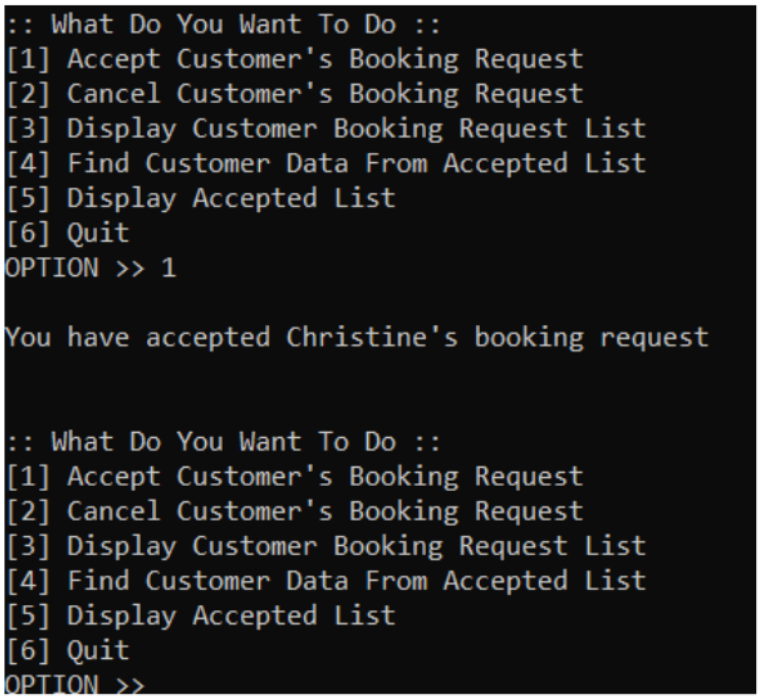


Scene 11: Hotel Manager Menu Interface

Scene 11:

The agent needs to choose the option by inserting values 1 to 6. If the hotel manager inputs the value 1, he will accept the customer's booking request from the agent. If the hotel manager inputs the value 2, he cancels the booking request from the accepted booking request list. If the hotel manager inputs the value 3, the system will display the current customer booking request list which is the customer reservation list from the agent. If the hotel manager inputs the value 4, the system will help him to search the specific customer data from the accepted booking request list. If the hotel manager inputs the value 5, the system will display the current accepted booking request list. If the agent inputs a value other than 1 to 6, the system will inform the user to choose the option again.

The following diagram show the output after hotel manager inputs value 1 in hotel manager menu interface



Scene 12: Output after hotel manager inputs value 1

Scene 12:

The system will help the hotel manager to accept the first customer from the customer reservation list (Queue) from the agent. It follows the concept of queue, in which the data is first in first out. At the beginning, the first customer data that was inserted by the agent is Christine's data so she was the first customer accepted by the hotel. After the customer booking request is accepted by the hotel manager, the system will automatically delete Christine's data from the reservation list as the customer has succeeded in booking the hotel. Then the system will return to the hotel manager menu interface.

The following diagram show the hotel manager inputs value 3 to display customer booking request list (Customer reservation list from agent)

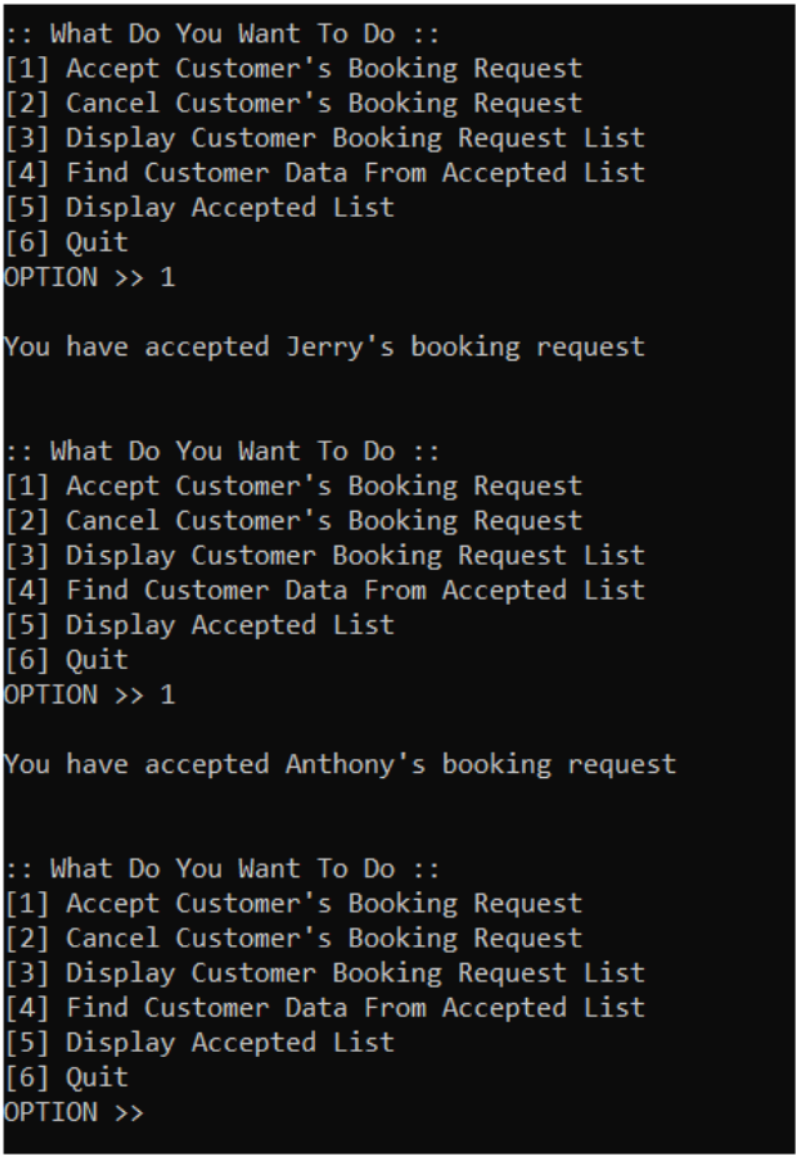
```
:: What Do You Want To Do ::
[1] Accept Customer's Booking Request
[2] Cancel Customer's Booking Request
[3] Display Customer Booking Request List
[4] Find Customer Data From Accepted List
[5] Display Accepted List
[6] Quit
OPTION >> 3

The Reservation List
Name: Jerry     , Age: 42 , IC: 810707-07-7541 , Phone Number: 017-5589642 , Room: 301 , Date: 2024-01-05
Name: Anthony   , Age: 28 , IC: 950703-14-1103 , Phone Number: 011-2237654 , Room: 103 , Date: 2024-02-01
Name: Henry     , Age: 45 , IC: 780527-08-3341 , Phone Number: 015-5846429 , Room: 253 , Date: 2024-02-17



:: What Do You Want To Do ::
[1] Accept Customer's Booking Request
[2] Cancel Customer's Booking Request
[3] Display Customer Booking Request List
[4] Find Customer Data From Accepted List
[5] Display Accepted List
[6] Quit
OPTION >>
```

Scene 13: Result after hotel manager inputs value 3 in hotel manager menu

Scene 13:

As Christine has succeeded to book the hotel, Christine's data is deleted from the customer's booking request list (Customer reservation list from agent). The system will display the remaining customer data that are waiting for the hotel to accept their booking request. After displaying the reservation list, the system will return to the hotel manager menu interface.

The following diagram show the hotel manager continues to accept customers' booking request



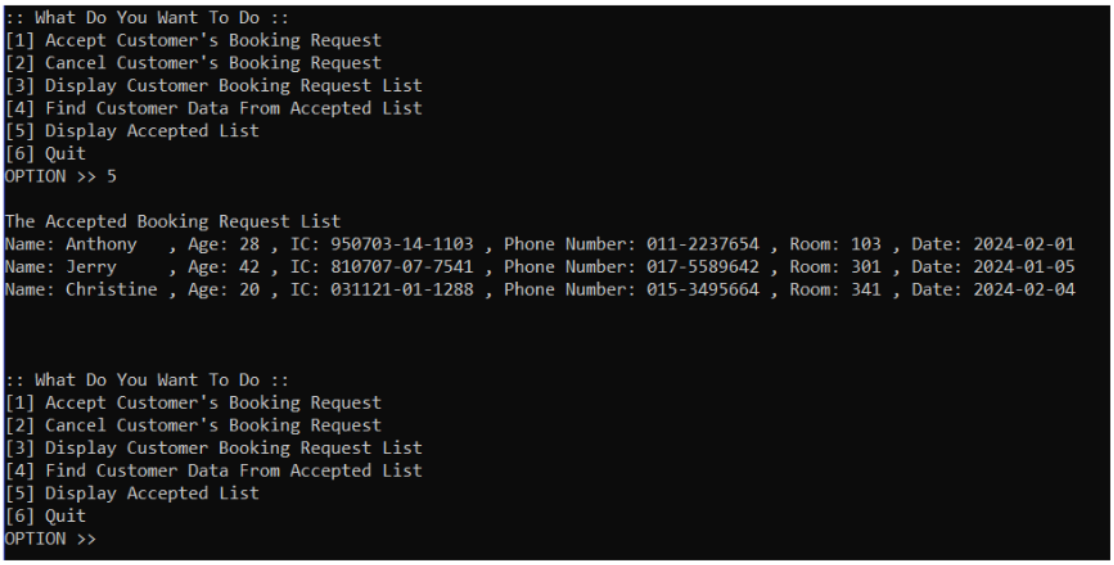
Scene 14: Output after hotel manager continues to accept customers' booking request

Scene 14:

After Christine's booking request has been accepted, the turn for the next customer to be accepted is Jerry because he is the next customer who wants to book the hotel after Christine. After Jerry's

booking request has been accepted, the turn for the next customer to be accepted is Anthony. The system follows the concept of a queue in which the first data has been deleted from the queue, the next data that will be deleted from the queue is the second data and so on. After Jerry's and Anthony's booking requests have been accepted, their data also will be deleted from the customer booking request list (Customer reservation list from the agent). For the accepted booking request, the list is following the concept of stack in which the last customer data inserted will be arranged at the top of the list.

The following diagram show the result after hotel manager inputs value 5 in hotel manager menu

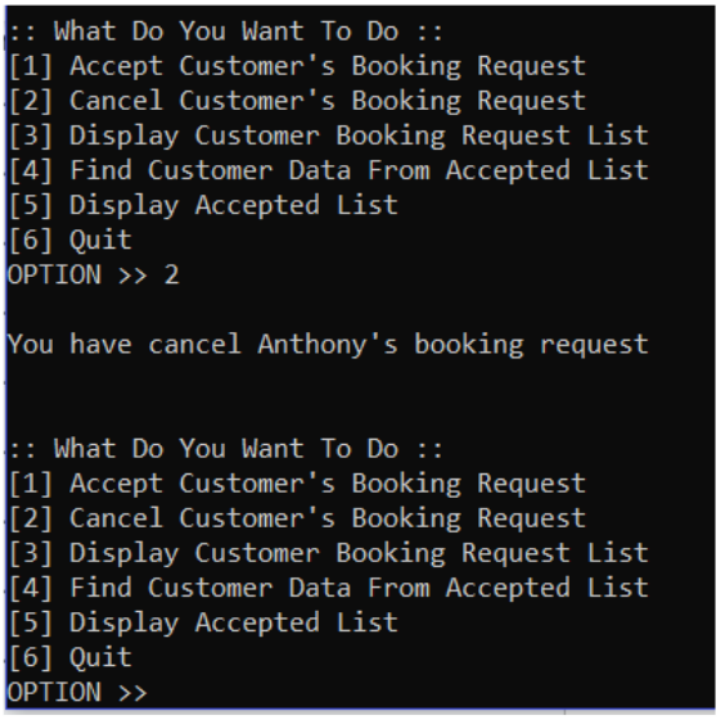


Scene 15: Result after hotel manager inputs value 5 in hotel manager menu

Scene 15:

The scene shows the result after the hotel manager inputs value 5 in the hotel manager menu. The system will start displaying the top customer data until the end of customer data in the stack. As the list applies to the concept of stack, the last customer inserted into the list will be arranged at the top of the list so Anthony's data will be displayed first. The system will return to the hotel manager menu interface after displaying the customer data.

The following diagram show the output after hotel manager inputs values 2 in hotel manager menu

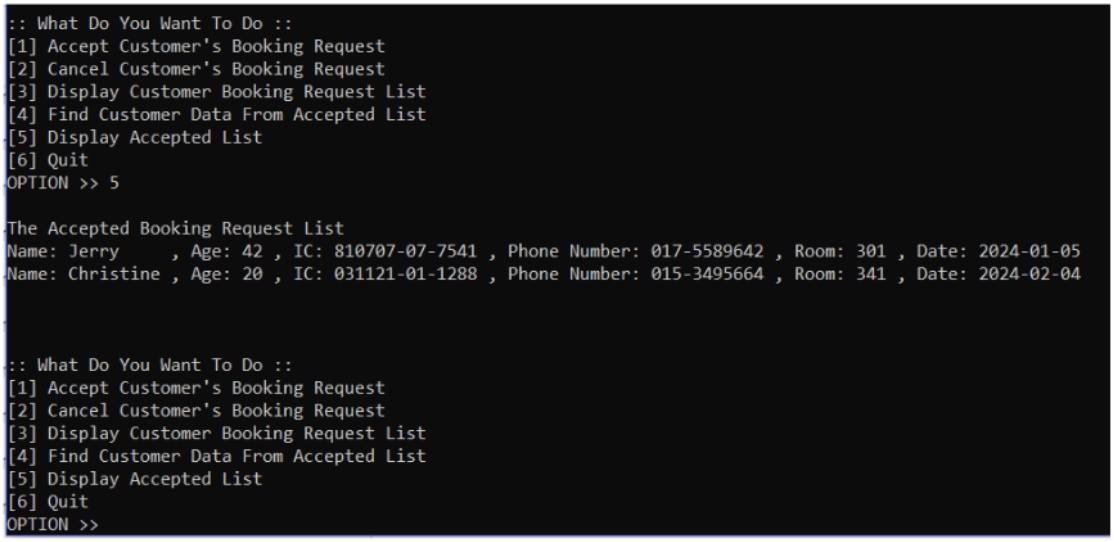


Scene 16: Output after hotel manager inputs values 2 in hotel manager menu

Scene 16:

The cancellation of booking requests has applied to the concept of stack in which the last data entering the stack will be the first one to pop out from the stack. We modify the situation of if the hotel manager has accepted one customer's booking request and found that the room that the customer booked has faced problems like the damage of furniture, the breakdown of the air conditioner or the room has been booked by a previous customer. He can undo the accepted booking request list. Therefore, in this situation, the last customer's booking request is Anthony's so if the hotel manager undo his action, Anthony's booking request will be canceled. After cancellation is done, the system will return to the hotel manager menu interface.

The following diagram show the result after hotel manager inputs value 5 in hotel manager menu

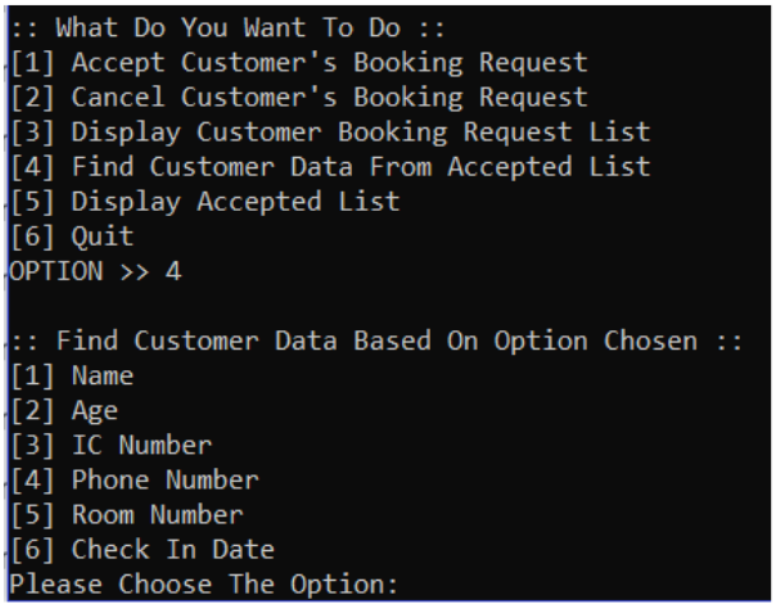


Scene 17: Result after hotel manager inputs value 5 in hotel manager menu

Scene 17:

The scene shows the result after the hotel manager inputs value 5 in the hotel manager menu. The system will start displaying the top customer data until the end of customer data in the stack. As Anthony's booking request has been canceled, his data will be deleted from the accepted booking request list. The system will return to the hotel manager menu interface after displaying the customer data.

The following diagram show the output after hotel manager inputs value 4 in the hotel manager menu



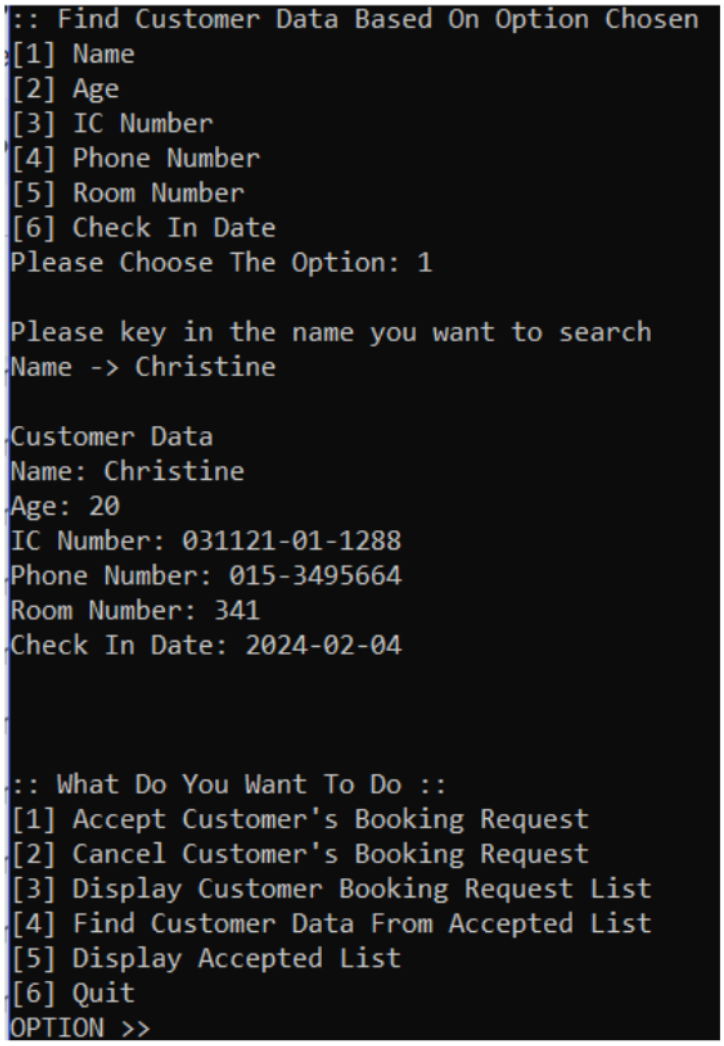
Scene 18: Output after hotel manager inputs value 4 in the hotel manager menu

Scene 18:

The scene shows the option that helps the hotel manager to find customer data based on keywords. The concept of this feature is the same as the "Find Customer Data" feature in the agent menu. The feature is used to find the customer data that has already been obtained in the accepted booking request list. If the hotel manager inputs option value 1 means he wants to search the specific customer data based on name. If the hotel manager inputs option value 2 means he wants to search the specific customer data based on age. If the hotel manager inputs option value 3 means he wants to search the specific customer data based on IC number. If the hotel manager inputs option value 4 means he wants to search the specific customer data based on phone number. If the hotel manager inputs option value 5 means he wants to search the specific customer data based on room number. If the hotel manager inputs option value 6 means he wants to search the specific customer data based on check in date. If the hotel manager inputs an option value other than 1 to 6, the system will inform the hotel manager to input the option again. The system will help the hotel manager to find the customer data that has already been obtained in the accepted booking request

list (Stack). The system will find from the last customer that the hotel manager inserts until the first customer that the hotel manager inserts.

The following diagram show the output after the hotel manager inputs name to find specific customer data

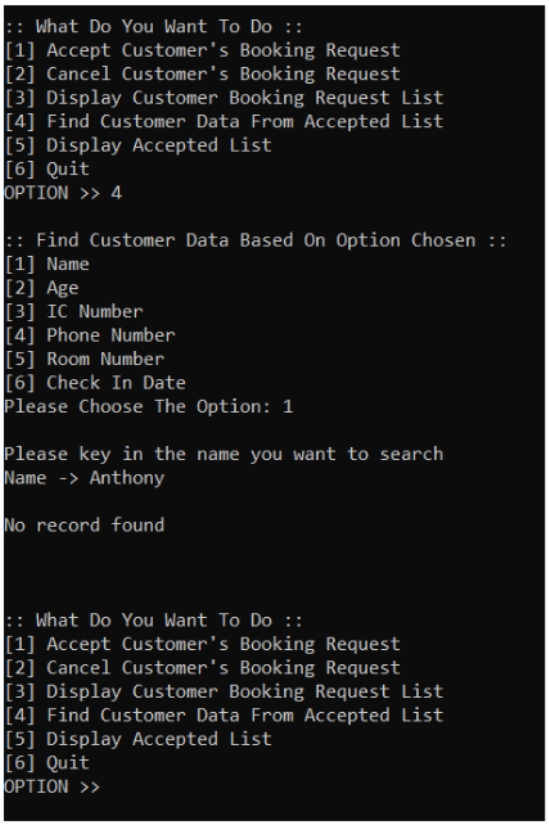


Scene 19: Output after hotel manager inputs name to search specific customer data

Scene 19:

The following scene shows that the hotel manager wants to search customer data based on customer names. When the hotel manager inputs a name that has already been obtained in the accepted booking request list, the system will display the specific customer data based on the name that the manager inputs. Then, the system will return to the hotel manager menu interface.

The following diagram show the output after hotel manager inputs the name that not obtained in the accepted booking request list

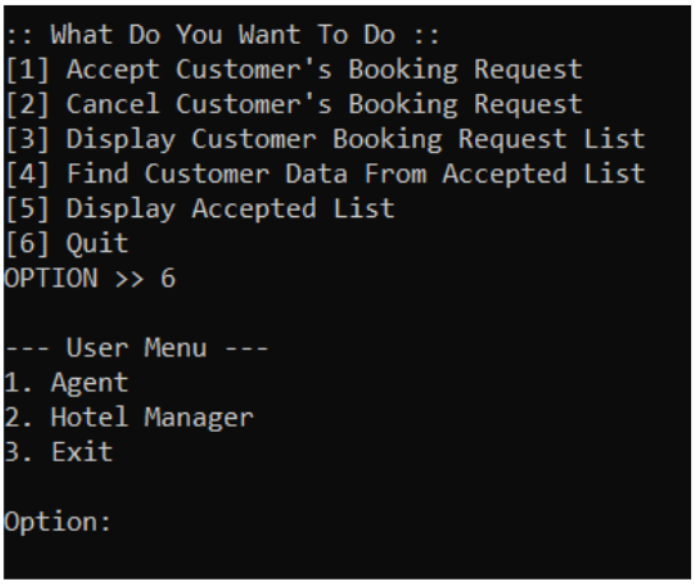


Scene 20: Output after hotel manager inputs the name that not obtained in accepted booking request list

Scene 20:

The system displays “No record found” to the hotel manager since Anthony’s data has been deleted from the list so there is no customer data for Anthony inside the accepted booking request list. Then the system will return to the hotel manager menu interface.

The following diagram show the interface after hotel manager quit the hotel manager menu interface

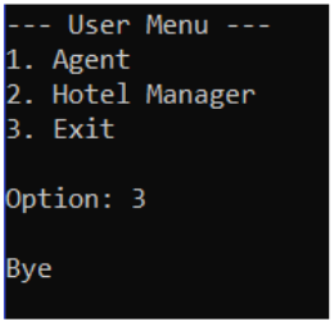


Scene 21: Interface after hotel manager quit from hotel manager menu interface

Scene 21:

When the hotel manager quits from the hotel manager menu interface, the system will return to the main menu interface. The hotel manager will become a normal user. The user can choose the option either he is an agent or hotel manager or he wants to exit the system.

The following diagram show that the user has exit the agent-based hotel booking system



Scene 22: End the system

Scene 22:

The scene shows the user has exited the agent-based hotel booking system so the program has ended.

7. Summary

Our project was about data structures and algorithms, with a focus on the application in a hotel booking system. The project is a continuation of two previous assignments. In the first assignment, we implemented sorting and searching algorithms, learning about their complexities and how they can be used to manage hotel data effectively. In the second assignment, we implemented linked lists. We used this data structure to improve the efficiency and flexibility of our hotel booking system. In this project, we used queues and stacks. The queue, created using a pointer in the Queue class, was used by the agent to handle customer requests in a fair, first-come-first-serve manner. The stack, designed in an array style within the Stack class, was used by the hotel manager to manage room availability, with the most recent booking being the first to be canceled if necessary. Lastly, teamwork is important during the project. We worked together effectively. This not only improved our understanding of data structures and algorithms concepts but also enhanced our teamwork and problem-solving skills. Overall, this project was a practical experience in using data structures and algorithms to solve real-world problems.

Project

ORIGINALITY REPORT

1%	0%	0%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to AUT University Student Paper	1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		