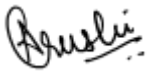
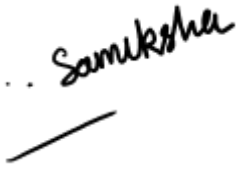
Declaration of Original Work for CE/CZ2002 Assignment

We hereby declare that the attached group assignment has been researched, undertaken, completed and submitted as a collective effort by the group members listed below.

We have honored the principles of academic integrity and have upheld Student Code of Academic Conduct in the completion of this work.

We understand that if plagiarism is found in the assignment, then lower marks or no marks will be awarded for the assessed work. In addition, disciplinary actions may be taken.

Name	Course (CE2002 or CZ2002)	Lab Group	Signature /Date
Bansal Arushi	CZ2002	SSP3	 13 November 2021
Poon Yan Xin Melise	CZ2002	SSP3	 13 November 2021
Sankar Samiksha	CZ2002	SSP3	 13 November 2021
Xing Kun	CZ2002	SSP3	 13 November 2021
Shreya Ramasubramanian	CZ2002	SSP3	 13 November 2021

Important notes:

1. Name must **EXACTLY MATCH** the one printed on your Matriculation Card.

1. Introduction

The Restaurant Reservation and Point of Sale System (RRPSS) is an application to computerise the process of making reservation, recording of orders and displaying of sale records. It is a system solely used by the restaurant staff.

This report consists of the diagrams used for the implementation of the system and also numerous design considerations and OO concepts used. Furthermore, test cases are also provided to ensure that the application runs smoothly without bugs.

2. Design Considerations

2.1 Approach taken

In designing our application, we have used multiple OOP design principles and object-oriented concepts that would be further discussed in the sections below.

To begin our design creation process, we constructed the class diagram to help in problem visualisation, communication and understanding of the assignment requirement. A sequence diagram is then constructed thereafter to visualise how the objects work together. With these diagrams constructed, it serves as an architecture skeleton for us to develop our code.

2.2 Design Principles

a. Single Responsibility Principle (SRP)

The Single Responsibility Principle states that there should never be more than one reason for a class to change. This implies that a class should only have one responsibility to ensure cohesion.

In our project, the *Table* class is an example. *Table* only manages all the existing tables and updates the available tables accordingly when the functions are called in the *mainapp*.

This enables us to eliminate rigidity as a change would not cause a cascade of subsequence changed in dependent modules.

b. Open-Closed Principle (OCP)

The Open-Closed Principle states that a module should be open for extension but closed for modification. This enables us to change what each module does without changing the source code of the module.

This is seen in our *MenuItems* and *SetPackage* classes whereby *SetPackage* is a subclass of *MenuItems*.

c. Liskov Substitution Principle (LSP)

This principle states that subclasses should be completely substitutable of superclass. The subclass should enhance its functionality, not reduce it.

In our application, this is also implemented in the *MenuItems* and the *SetPackage* class. The derived class, *SetPackage* is substitutable for its superclass, *MenuItems*, while retaining the methods of the class. This ensures that the pre-conditions of the *SetPackage* class are no stronger than the *MenuItems* class methods. Also, the post-conditions of the *SetPackage* class are no weaker than the *MenuItems* class methods. This enables *MenuItems* to continue to function properly when a derivative of *SetPackage* is passed into it.

d. Interface Segregation Principle (ISP)

In our application, we ensured that the classes do not depend on any interfaces. Hence, avoiding the usage of FAT interfaces. Hence, none of the classes depends on interfaces they do not use.

e. Dependency Injection Principle (DIP)

Dependency Injection Principle states that a high-level module should not depend on low level modules, classes should depend on abstractions.

As interfaces are not implemented in our application, this principle does not apply.

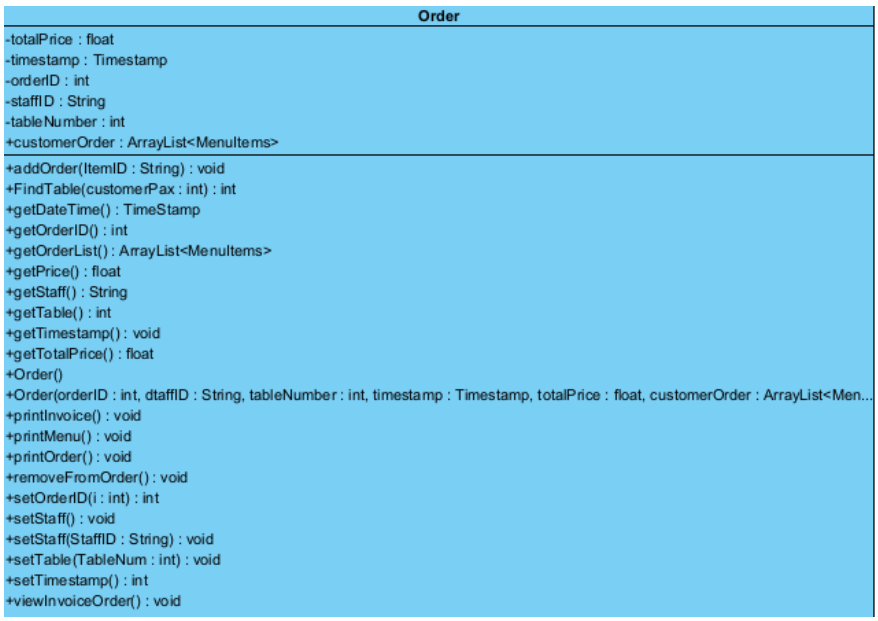
2.3 Object-Oriented Concepts

a. Abstraction

Abstraction denotes the essential characteristics of an object that distinguishes it from other kinds of objects. This is evident in the classes of our application created, whereby only the necessary attributes and methods are defined. This provide crisply defined conceptual boundaries, relative to the perspective of the viewer, reducing the complexity of the code.

b. *Encapsulation*

Encapsulation builds a barrier to protect an object's private data. Hence, the object's private data can only be accessed through public methods like the getters and the setters. This hides the implementation of the class from users.

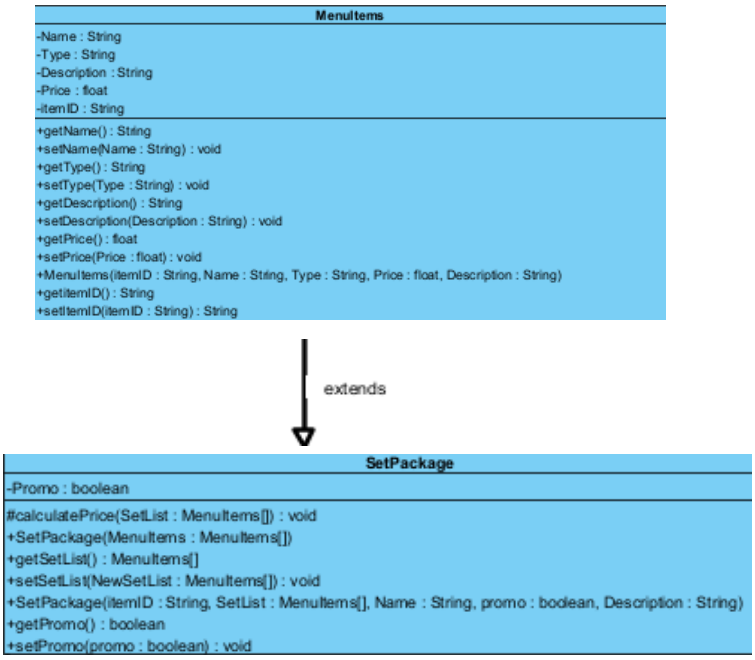


Encapsulation is used in multiple classes of our application. For example, in the *Order* class, it has numerous attributes *timestamp, orderID, staffID, tableNumber, totalPrice* are all private. Thus, getters and setters are implemented to access them.

c. *Inheritance*

Inheritance allows it to derive new classes from existing objects. The new classes would be able to absorb the attributes and methods of the existing class, thereby also adding new capabilities in the new classes.

Inheritance is implemented in our design. For example, *MenuItems* is the parent class of *SetPackage*.



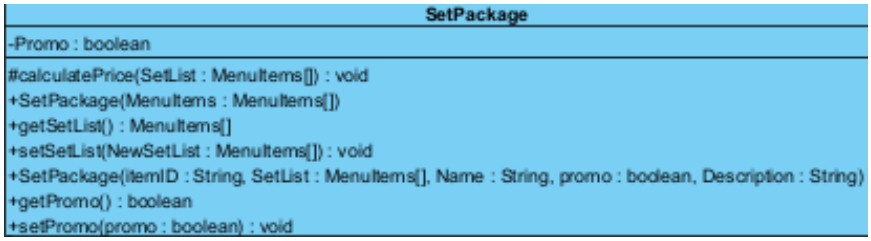
Inheritance enables code-reuse and reduce programming effort in implementing new classes. As seen in the class diagram, a generalisation relationship is used to indicate the inheritance between *MenuItems* and *SetPackage.*

The subclass, *SetPackage* would be able to use the methods in *MenuItems* such as calling the constructor of *MenuItems* and the *setPrice* method in *MenuItems* by using the super keyword.

```
public SetPackage(String itemID, MenuItems[] SetList, String Name,boolean promo, String Description) {
        super(itemID,Name,"Set Item",(float)0,Description);
        this.setPromo(promo);
        this.SetList = SetList;
        super.setPrice(calculatePrice(SetList));
}
```

Furthermore, the method *calculatedPrice* in the *SetPackage* class has a visibility modifier of protected.

```
protected float calculatePrice(MenuItems[] SetList) {
        float price = (float) 0;
        int i;
        for (i =0; i < SetList.length; i++) {
                price += SetList[i].getPrice();
        }
        price = price * (float)0.9;
        if (getPromo()) {
                price = price * (float)0.85;
        }
        return price;
}
```



This implies that this method is only visible to methods in the same class, the methods of subclasses or any classes in the same package. This is denoted in the UML class diagram here as '#'.

d. Polymorphism

Polymorphism refers to the ability of an object reference being referred to different types. This facilitates the adding of new classes into the application with minimal modifications to the system's code, thereby creating a system that is extendable.

e. Usage relationship

A usage is a relationship in which one element requires another element for its full implementation or operation. For our application, the *mainapp* class uses the *SecurityAccess* class. The *SecurityAccess* must be an independent class not needing to know how the constructors, setters and getters in the other classes (*MenuItems*, *SetPackage* and *Order*) works. This enables the *SecurityAccess* class to be easily upgradable as the other classes would not be changed while upgrading the code.

Additionally, *SecurityAccess* class is also kept separate from other classes to prevent any accidental access by rebellious staff members or other users. Furthermore, users without the password should not be able to access *SecurityAccess*, hence preventing unauthorised users from accessing these functions in the class.

3. Assumptions

During the implementation of the application, these are the assumptions made:

- Only Staffs with a valid staff access code would be able to access the whole system, including altering the Menu items and making a reservation.
- The restaurant is only open for business every day from 10am to 10pm. Any orders made after that time would be unsuccessful. Any reservation made after 8pm would not be accepted.

4. Exception Handling

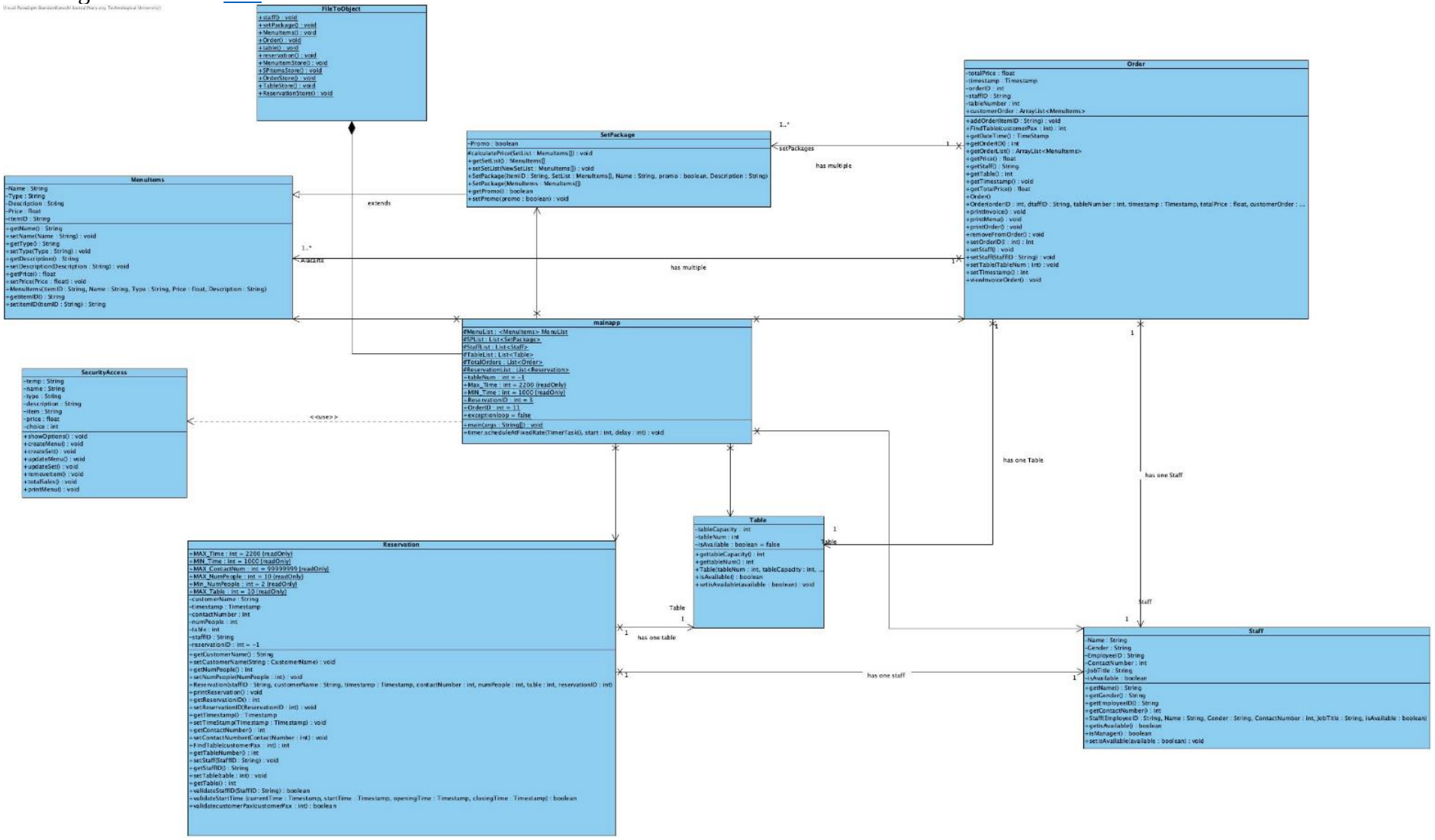
Exception are runtime anomalies that a program may detect. Exception handlers may catch an exception to recover from the problem.

In our application, multiple methods have exception handling implemented. For example, in the *mainapp* class, an exception would be triggered in the try catch block when a user enters an integer that is not withing the range of 1-7.

```
while (c >= 0){
    System.out.println("---------------------------STAFF ACCESS---------------------------");
    System.out.println("| (1) Create An Order                                              |");
    System.out.println("| (2) Place/Remove An item from Order                              |");
    System.out.println("| (3) View an Order                                                |");
    System.out.println("| (4) Reservation (Make, Remove, Check)                            |");
    System.out.println("| (5) Print Order Invoice                                          |");
    System.out.println("| (6) Check Table Availability                                     |");
    System.out.println("| (7) Manager Access Only (Menu Item, Promotion, Sales Revenue)    |");
    System.out.println("| (8) Close Shop                                                   |");
    System.out.println("--------------------------------------------------------------------");
    try{
        c = Integer.parseInt(sc.nextLine());
    }catch(Exception e){
        c = 9;
    }
```

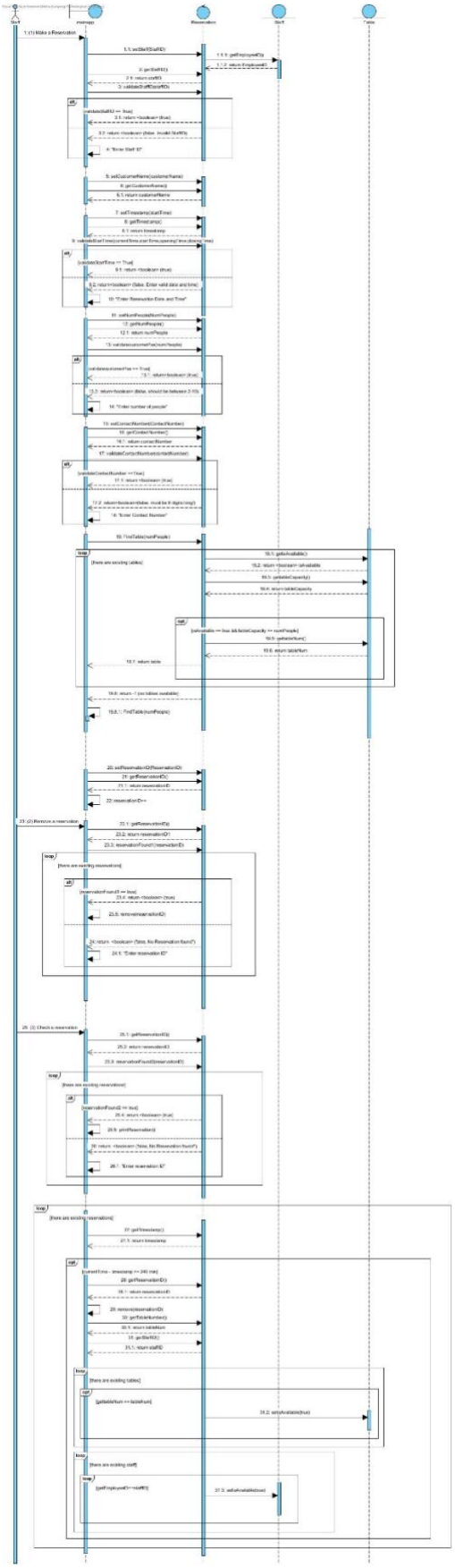
5. UML Class Diagram

Full diagram is in this link.



6. UML Sequence Diagram

The full diagram is in this link.



7. Test Cases

Case 1: Creating an order

With reservation	Enter choice: (1): Create Order (Made Reservation) (2): Create Order (Walk-in) (3): Exit 1 Have you reserved a table? Enter true or false. true Please enter your Reservation ID: 3 Reservation found! Your reservation is active! Staff E1 will be helping you! You are allocated to Table 1! OrderID:11
Without reservation	Enter choice: (1): Create Order (Made Reservation) (2): Create Order (Walk-in) (3): Exit 2 Enter number of people to be seated in the table 2 Staff E2 will be helping you! You are allocated to Table 2! OrderID:12 2
Removing of reservation/s upon 'period expiry' (adjusted the period of expiry to 2 minutes instead of 240 minutes to demonstrate)	Enter choice: (1): Make a Reservation (2): Remove a Reservation (3): Check a Reservation (4): Exit 3 Please enter your Reservation ID: 3 Reservation Details are as follows: Reservation ID:3 Name:C1 Time Of Reservation :2021-11-13 15:20:00.0 Contact Number:12345678 Number of People:3 Table:3 StaffID:E1 Enter choice: (1): Make a Reservation (2): Remove a Reservation (3): Check a Reservation (4): Exit ---------- |NOTICE:Reservation with ID 3 has been deleted due to reservation period expiry at 2021-11-13 15:22:01.152| ---------- 3 Please enter your Reservation ID: 3 No reservation with ID 3 found!

Case 2: Add/remove item to order

Add non-existing item	Enter the menu item you want to order from the menu: AC0 Enter the quantity 1 Invalid menu item/quantity value Enter the menu item you want to order from the menu:
Add item	Enter the menu item you want to order from the menu: AC3 Enter the quantity 1 Item(s) added successfully This is your updated order Your Order: Order 1: ID:AC3 Name:Quorn Burger Type:Main Description:Healthy vegetarian and delicious. Price:7.550000
Remove item	Your Order: Order 1: ID:AC1 Name:Fish Burger Type:Main Description:Burger made with a fish patty. Price:6.550000 Order 2: ID:AC2 Name:Chicken Burger Type:Main Description:Deep fried chicken patty with lettuce, tomato, cheese with specially made sauce. Price:6.000000 Select an item to remove from order and enter the number: 1 Item removed successfully This is your updated order Your Order: Order 1: ID:AC2 Name:Chicken Burger Type:Main Description:Deep fried chicken patty with lettuce, tomato, cheese with specially made sauce. Price:6.000000

Case 3: Creating reservation

Full reservation	4 Enter choice: (1): Make a Reservation (2): Remove a Reservation (3): Check a Reservation (4): Exit 1 Enter Staff ID: E1 Enter Name of Customer: Melise Enter a Reservation Date and Time: Format: yyyy-MM-dd HH:mm:ss Booking Hours: 10:00:00 - 20:00:00 2021-11-15 11:30:00 Entered date and time are valid! Enter number of people to be there 4 Enter Customer Contact Number(8 digits): 34876656 No Tables Available! Reservation cannot be made the moment! Enter choice: (1): Make a Reservation (2): Remove a Reservation
Available reservation	Enter choice: (1): Make a Reservation (2): Remove a Reservation (3): Check a Reservation (4): Exit 1 Enter Staff ID: E1 Enter Name of Customer: FYA Enter a Reservation Date and Time: Format: yyyy-MM-dd HH:mm:ss Booking Hours: 10:00:00 - 20:00:00 2021-11-13 12:00:00 Entered date and time are valid! Enter number of people to be there 3 Enter Customer Contact Number(8 digits): 12345678 Table 4 has been blocked for you! Reservation Made! Your reservation ID is:3

Case 4: Release table upon payment

Making order	Availability of table during order
Enter choice: (1): Create Order (Made Reservation) (2): Create Order (Walk-in) (3): Exit 2 Enter number of people to be seated in the table 3 Staff E1 will be helping you! You are allocated to Table 3! OrderID:12	Table Availability: Table Number: 1 Availability: true Table Number: 2 Availability: true Table Number: 3 Availability: false Table Number: 4 Availability: true Table Number: 5 Availability: true Table Number: 6 Availability: true Table Number: 7 Availability: true Table Number: 8 Availability: true Table Number: 9 Availability: true Table Number: 10 Availability: true
During payment	**Availability of table after payment**
Please enter your Order ID: 12 View your Order Invoice: Are you a member? Enter 'true' if yes and 'false' is no false Here is your final invoice: +----------2021-11-13 17:35:37.257----------+ |----------| |----------Sally's Burger Town----------| |----------50 Nanyang Avenue----------| |----------639798----------| |----------Tel: 98765432 ----------| |----------Staff ID: E1 Table No.:3----------| | 2 Vegetarian option 13.337999 | | 2 Onion Ring 4.500000 | | Total Price : SGD$17.84 | | Total GST : SGD$1.25 | | Total Service Tax : SGD$1.78 | | Total Member's Discount : SGD$0.00 | | Total Payable Amount: SGD $20.87 | |---------- Thank you for Visiting! ----------| |---------- Please do come again! ---------- | +----------+	Table Availability: Table Number: 1 Availability: true Table Number: 2 Availability: true Table Number: 3 Availability: true Table Number: 4 Availability: true Table Number: 5 Availability: true Table Number: 6 Availability: true Table Number: 7 Availability: true Table Number: 8 Availability: true Table Number: 9 Availability: true Table Number: 10 Availability: true

Case 5: Manager access

<table>
<tr><td></td><td>Process</td><td>After</td></tr>
<tr><td>Create menu item</td><td>Enter the new name
Check
Enter the new type
Drinks
Enter the new description
refreshing
Enter the new price
2
Successfully added the MenuItems!</td><td>ID: AC15 Name: Check Type: Drinks Price: 2.000000
Description: refreshing</td></tr>
<tr><td>Remove menu item</td><td colspan="2">Are you removing Menu or Set Package?
(1) Menu
(2) Set Package
1
Which Item would you like to remove?
Enter the ItemID:
AC15
Removed Successfully!</td></tr>
<tr><td>Update set package</td><td>Which Item would you like to update? Enter the itemID:
SP4
Enter the new name
check
Enter the new description
something
Enter in the list of Menuitems(type z to stop):
Enter MenuItem ID press z to exit:
AC1
Enter MenuItem ID:
AC2
Enter MenuItem ID:
z
Would you like to add Promotion on this set package? Enter '1' for ye
0
Successfully updated!</td><td>ID: SP4 Name: check Type: Set Item Price: $5.202000
Description: something</td></tr>
<tr><td>Check sales report (for November)</td><td colspan="2">6
Do you want to see revenue for a day or for a month?
(1) Day
(2) Month
2
Enter month (int):
11
Sales Revenue
ItemId ItemName Qty Total Price
AC1 Fish Burger 2 SGD$13.10
AC2 Chicken Burger 1 SGD$6.00
AC3 Quorn Burger 1 SGD$7.55
AC4 French Fries 2 SGD$6.80
AC5 Onion Ring 0 SGD$0.00
AC6 Hashbrown 3 SGD$7.50
AC7 Sprite 2 SGD$2.50
AC8 Coke 2 SGD$2.50
AC9 Orange Juice 0 SGD$0.00
AC10 Apple Juice 0 SGD$0.00
AC11 Chocolate Cake 0 SGD$0.00
AC12 IceCream 0 SGD$0.00
AC13 Brownie 0 SGD$0.00
AC14 Apple Pie 0 SGD$0.00
SP1 Package A 0 SGD$0.00
SP2 Package B 1 SGD$12.39
SP3 Vegetarian option 1 SGD$13.34
SP4 Holiday Meal 4 SGD$37.49
Total: SGD$109.17
Total GST: SGD$7.64
Total service tax: SGD$10.92
Total Revenue: SGD$128.49</td></tr>
</table>

8. Video

This is the link to access the demonstration video.

https://youtu.be/yr4fOC-rus8