College of Engineering

School of Computer Science & Engineering

CZ2002

Object Oriented Design & Programming

AY 2021/22 S1

<u>Declaration of Original Work for CE/CZ2002 Assignment</u>

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	Lab Group	Signature/Date
Jaiswal Arnav	CZ2002	DSAI3 Grp 1	[illegible] 14/11/2021
Goh Peng Aik	CZ2002	DSAI3 Grp 1	[illegible] 14/11/2021
Karanam Akshit	CZ2002	DSAI3 Grp 1	[illegible] 14/11/2021
Tan Yi Jie Joel	CZ2002	DSAI3 Grp 1	Joel 14/11/2021
Jain Aarushi	CZ2002	DSAI3 Grp 1	[illegible] 14/11/2021

1. Introduction

The functionality of our Console-based application, Restaurant Reservation and Point of Sale System (RRPSS), will be described and explained in this assignment report. It will also include a UML class diagram, sequence diagrams, and explanations of some of the design principles, considerations, and object-oriented concepts used in our program. RRPSS is a staff-oriented application. It allows employees to browse available tables and reservation periods as well as make reservations and orders directly through the app.

Here are the assumptions we made while developing this app.

1. Reservation can only be made in advance; no walk-ins are allowed. Reservation will be automatically removed after 30 minutes if a customer does not check in within 30 minutes from the reservation start time.

2. The currency will be in Singapore Dollar (SGD) and Good and Services Tax (GST) of 7% has been included in the order invoice.

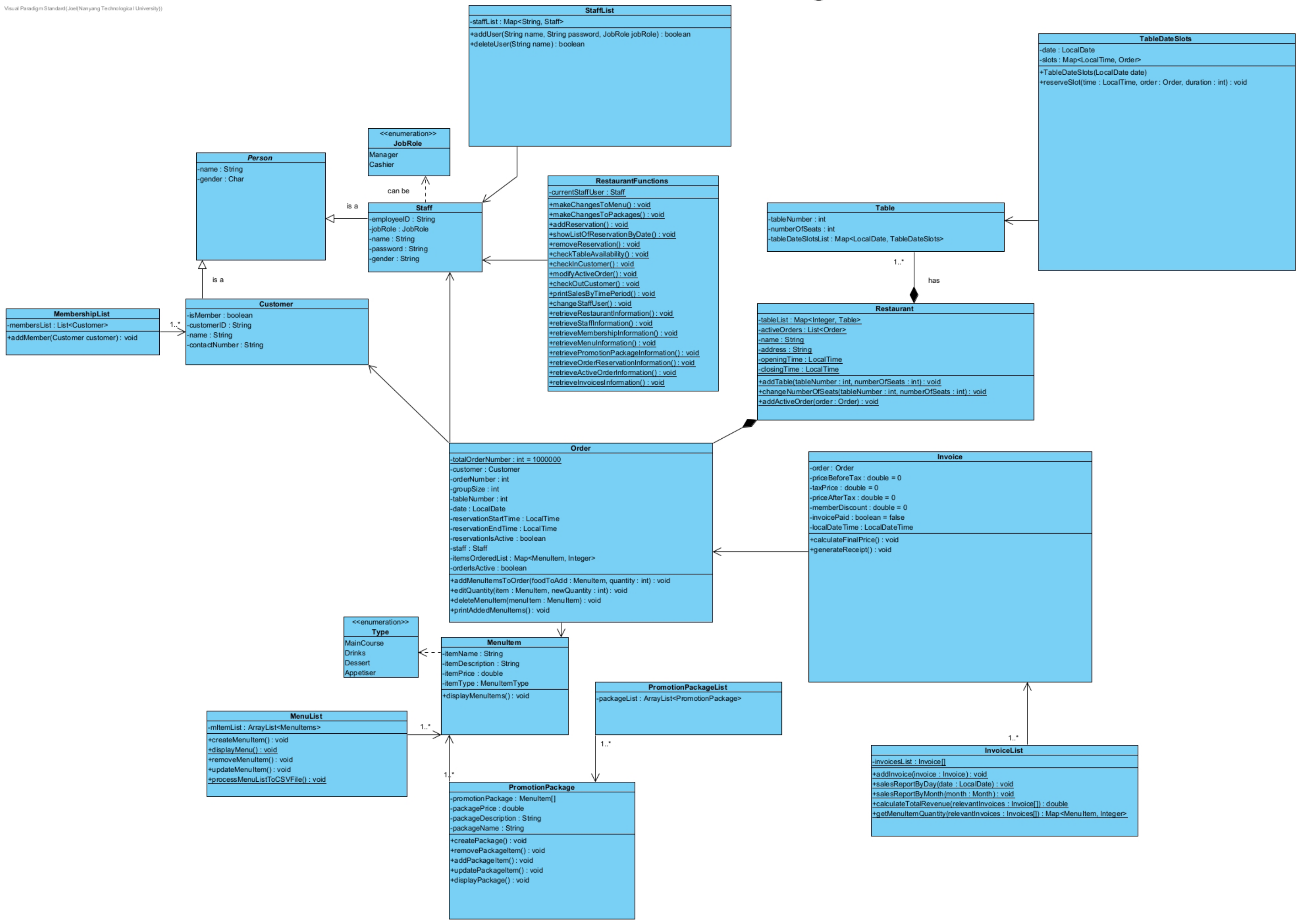
3. Once a customer checks-out, it is assumed that payment has been made and the table is vacated.

4. Customers who are members are entitled to a 5% discount. A customer can only declare his membership status at the point of reservation.

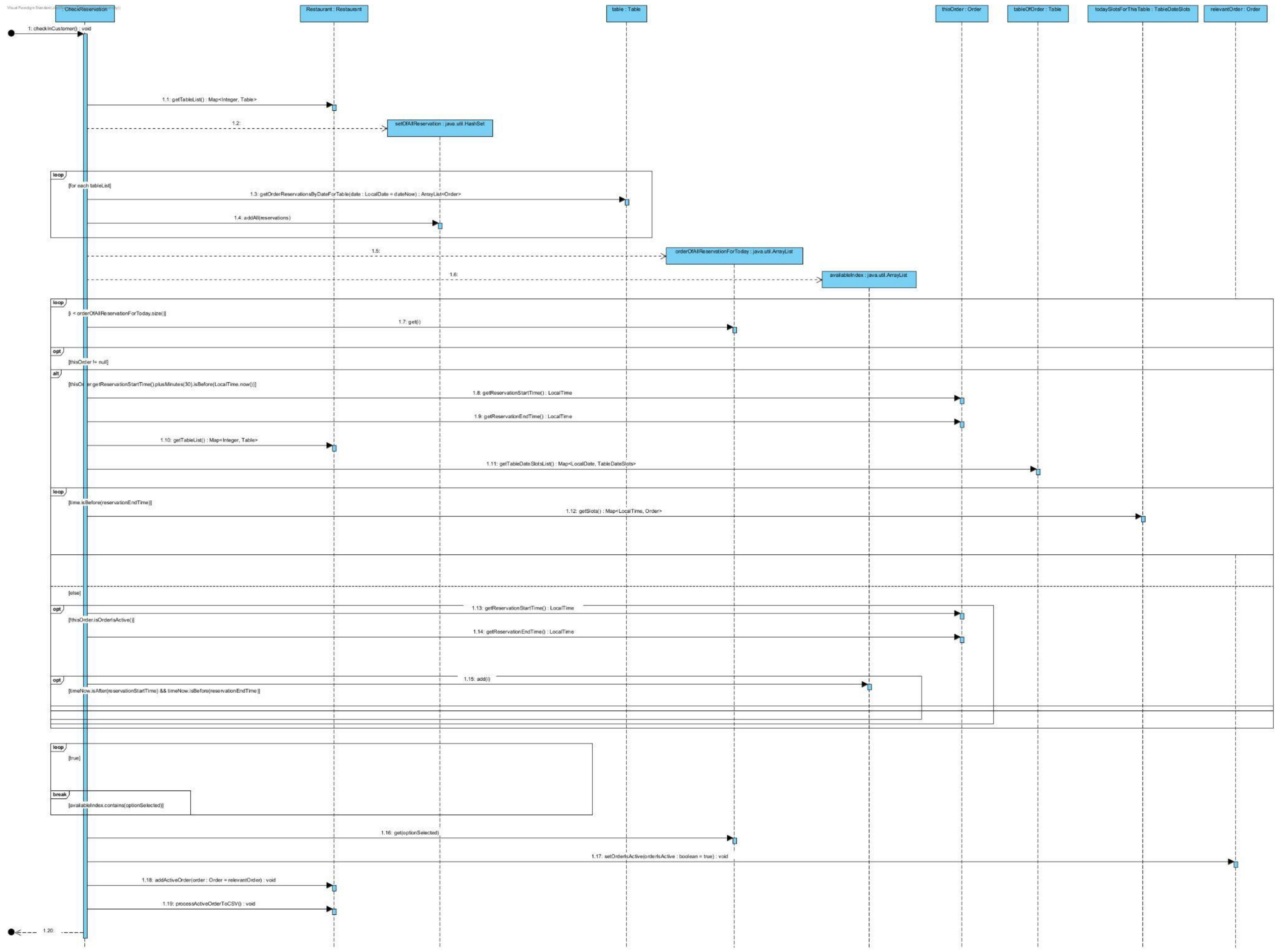
5. Application is secured and has been already logged in by a relevant authority.

6. Customers can check-in 20 minutes before the start of the reservation, provided the table that was reserved is empty at that time

2. UML Class Diagram


StaffList
-staffList : Map<String, Staff>
+addUser(String name, String password, JobRole jobRole) : boolean
+deleteUser(String name) : boolean
TableDateSlots
-date : LocalDate
-slots : Map<LocalTime, Order>
+TableDateSlots(LocalDate date)
+reserveSlot(time : LocalTime, order : Order, duration : int) : void
<<enumeration>>
JobRole
Manager
Cashier
can be
Person
-name : String
-gender : Char
is a
Staff
-employeeID : String
-jobRole : JobRole
-name : String
-password : String
-gender : String
RestaurantFunctions
-currentStaffUser : Staff
+makeChangesToMenu() : void
+makeChangesToPackages() : void
+addReservation() : void
+showListOfReservationByDate() : void
+removeReservation() : void
+checkTableAvailability() : void
+checkInCustomer() : void
+modifyActiveOrder() : void
+checkOutCustomer() : void
+printSalesByTimePeriod() : void
+changeStaffUser() : void
+retrieveRestaurantInformation() : void
+retrieveStaffInformation() : void
+retrieveMembershipInformation() : void
+retrieveMenuInformation() : void
+retrievePromotionPackageInformation() : void
+retrieveOrderReservationInformation() : void
+retrieveActiveOrderInformation() : void
+retrieveInvoicesInformation() : void
Table
-tableNumber : int
-numberOfSeats : int
-tableDateSlotsList : Map<LocalDate, TableDateSlots>
1..*
has
is a
MembershipList
-membersList : List<Customer>
+addMember(Customer customer) : void
1..*
Customer
-isMember : boolean
-customerID : String
-name : String
-contactNumber : String
Restaurant
-tableList : Map<Integer, Table>
-activeOrders : List<Order>
-name : String
-address : String
-openingTime : LocalTime
-closingTime : LocalTime
+addTable(tableNumber : int, numberOfSeats : int) : void
+changeNumberOfSeats(tableNumber : int, numberOfSeats : int) : void
+addActiveOrder(order : Order) : void
Order
-totalOrderNumber : int = 1000000
-customer : Customer
-orderNumber : int
-groupSize : int
-tableNumber : int
-date : LocalDate
-reservationStartTime : LocalTime
-reservationEndTime : LocalTime
-reservationIsActive : boolean
-staff : Staff
-itemsOrderedList : Map<MenuItem, Integer>
-orderIsActive : boolean
+addMenuItemsToOrder(foodToAdd : MenuItem, quantity : int) : void
+editQuantity(item : MenuItem, newQuantity : int) : void
+deleteMenuItem(menuItem : MenuItem) : void
+printAddedMenuItems() : void
Invoice
-order : Order
-priceBeforeTax : double = 0
-taxPrice : double = 0
-priceAfterTax : double = 0
-memberDiscount : double = 0
-invoicePaid : boolean = false
-localDateTime : LocalDateTime
+calculateFinalPrice() : void
+generateReceipt() : void
<<enumeration>>
Type
MainCourse
Drinks
Dessert
Appetiser
MenuItem
-itemName : String
-itemDescription : String
-itemPrice : double
-itemType : MenuItemType
+displayMenuItems() : void
PromotionPackageList
-packageList : ArrayList<PromotionPackage>
MenuList
-mItemList : ArrayList<MenuItems>
+createMenuItem() : void
+displayMenu() : void
+removeMenuItem() : void
+updateMenuItem() : void
+processMenuListToCSVFile() : void
1..*
1..*
1..*
1..*
PromotionPackage
-promotionPackage : MenuItem[]
-packagePrice : double
-packageDescription : String
-packageName : String
+createPackage() : void
+removePackageItem() : void
+addPackageItem() : void
+updatePackageItem() : void
+displayPackage() : void
InvoiceList
-invoicesList : Invoice[]
+addInvoice(invoice : Invoice) : void
+salesReportByDay(date : LocalDate) : void
+salesReportByMonth(month : Month) : void
+calculateTotalRevenue(relevantInvoices : Invoice[]) : double
+getMenuItemQuantity(relevantInvoices : Invoices[]) : Map<MenuItem, Integer>

3. UML Sequence Diagram



: CheckReservation
Restaurant : Restaurant
table : Table
thisOrder : Order
tableOfOrder : Table
todaySlotsForThisTable : TableDateSlots
relevantOrder : Order
1: checkInCustomer() : void
1.1: getTableList() : Map<Integer, Table>
1.2:
setOfAllReservation : java.util.HashSet
loop
[for each tableList]
1.3: getOrderReservationsByDateForTable(date : LocalDate = dateNow) : ArrayList<Order>
1.4: addAll(reservations)
1.5:
orderOfAllReservationForToday : java.util.ArrayList
1.6:
availableIndex : java.util.ArrayList
loop
[i < orderOfAllReservationForToday.size()]
1.7: get(i)
opt
[thisOrder != null]
alt
[thisOrder.getReservationStartTime().plusMinutes(30).isBefore(LocalTime.now())]
1.8: getReservationStartTime() : LocalTime
1.9: getReservationEndTime() : LocalTime
1.10: getTableList() : Map<Integer, Table>
1.11: getTableDateSlotsList() : Map<LocalDate, TableDateSlots>
loop
[time.isBefore(reservationEndTime)]
1.12: getSlots() : Map<LocalTime, Order>
[else]
opt
[!thisOrder.isOrderIsActive()]
1.13: getReservationStartTime() : LocalTime
1.14: getReservationEndTime() : LocalTime
opt
[timeNow.isAfter(reservationStartTime) && timeNow.isBefore(reservationEndTime)]
1.15: add(i)
loop
[true]
break
[availableIndex.contains(optionSelected)]
1.16: get(optionSelected)
1.17: setOrderIsActive(orderIsActive : boolean = true) : void
1.18: addActiveOrder(order : Order = relevantOrder) : void
1.19: processActiveOrderToCSV() : void
1.20:

4. Design Principles, Considerations & Object-Oriented Concepts

a. Design Considerations & Principles

The flow of the Entity-Boundary-Control (EBC) design is reflected in the way we designed our program. The EBC design is a type of stereotype that aids users in comprehending the code flow. We opted to build a single control class POSApp for the RRPSS module because the application's logic is quite sophisticated. The main controller class will then assign responsibilities to the other control classes. The following sections will go over these design considerations in further detail.

i. Data Access

We used the CSV format to store and retrieve data in our data files by making use of BufferedReader. For instance, we utilize a single RestaurantFunctions class, which implements retrieve functions, to read and write data that is stored as CSV collection in our application.

ii. High Cohesion

We adhere to the Single Responsibility Principle to attain High Cohesion. Every class should only be responsible for one thing. In the larger scheme of things, this means that the program must be divided into three tiers of modules: border classes, control classes, and entity classes. Our border class, for example, oversees showing and accepting the user's selection. The control class will then receive and interpret this information to correctly contact the correct entity using a data access object. As a result, there will be no all-encompassing class. The benefit of having a high cohesion is that we can isolate functionality into various classes, lowering the complexity of each class and improving system maintainability while reducing unwanted bugs.

iii. Object-Oriented Encapsulation

Encapsulation is a property that keeps a class's data and inner implementation hidden from other classes. We maintain the data integrity of a class's object by appropriately setting access modifiers for classes, methods, and attributes, so that other classes cannot directly access and edit another's class data. A method is set as a private method if it is not used by any other class. All of this reduces the impact on other classes, providing for easy reusability and data integrity, and making unit testing of the encapsulated code easier.

b. SOLID Principles

i. Single Responsibility Principle

The single responsibility principle states that a class should only have one responsibility and that there should not be a "God" class that performs all functions. This entails ensuring that our UI classes just provide UI functionalities but do not perform any logical processing.

ii. Open-Closed Principle

Concrete classes are used to implement the bulk of our top-level modules. We utilized abstract classes or interfaces to ensure that if a feature or a new class is needed, the base class will not need to be changed, thereby adhering to the "closed for modification" principle. Furthermore, because new concrete classes can easily implement our top-level abstract classes and interfaces, the program can be "opened for modification" quickly and easily.

iii. Liskov Substitution Principle

When creating a class that will inherit or override features from a superclass or abstract class, we always keep in mind that our subclass should never be more restrictive than the superclass's behavior. Because many of our classes rely on abstraction to conform to the dependency inversion concept, this is significant because if the subclass methods are more restrictive than the superclass or abstract class, abstraction may fail. A class associated with the superclass or abstract class's subclass may not know how to handle the more restrictive method if the Liskov substitution principle is not followed.

iv. Interface Segregation Principle

The interface segregation principle is an extension of the principle of single responsibility for interfaces. By ensuring that we do not force any controller class to depend on methods that it may not use, we adhere to the interface segregation principle.

v. Dependency Inversion Principle

We should avoid permitting classes from different tiers to be tightly coupled and reliant on concretion across layers. This is done by ensuring that our higher-level modules are not being dependent on lower-level modules, but instead on abstraction.

5. Video Demo Link

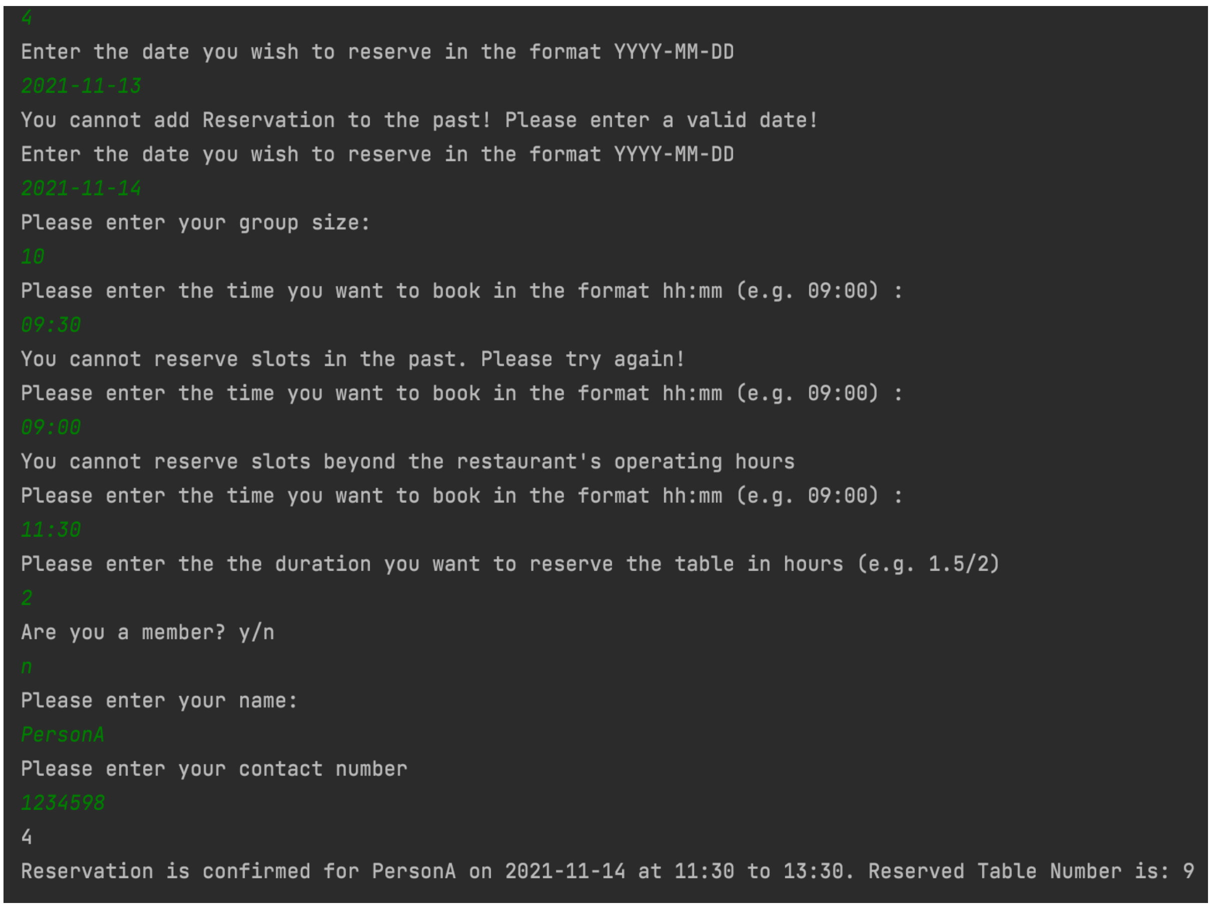
https://youtu.be/NnjJYRnQhQE

6. Test Cases

All test cases are independent from each other (i.e. you need to restart the app if needed), please follow the instructions given at each case before starting to test the cases. There are a total of 7 additional test cases here that were not covered in the video demonstration

Test Case 1: Inputting date/time of past & inputting time beyond restaurant opening hours.

Current Date: 2021-11-14   Current Time: 11:25



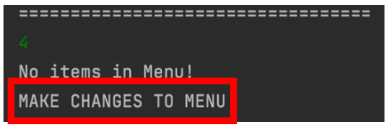
```
4
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-13
You cannot add Reservation to the past! Please enter a valid date!
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-14
Please enter your group size:
10
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
09:30
You cannot reserve slots in the past. Please try again!
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
09:00
You cannot reserve slots beyond the restaurant's operating hours
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
11:30
Please enter the the duration you want to reserve the table in hours (e.g. 1.5/2)
2
Are you a member? y/n
n
Please enter your name:
PersonA
Please enter your contact number
1234598
4
Reservation is confirmed for PersonA on 2021-11-14 at 11:30 to 13:30. Reserved Table Number is: 9
```

Entering a date in the past
Entering a time in the past
Entering a time beyond the restaurants operating hours (Can be changed via the Restaurant.csv file).

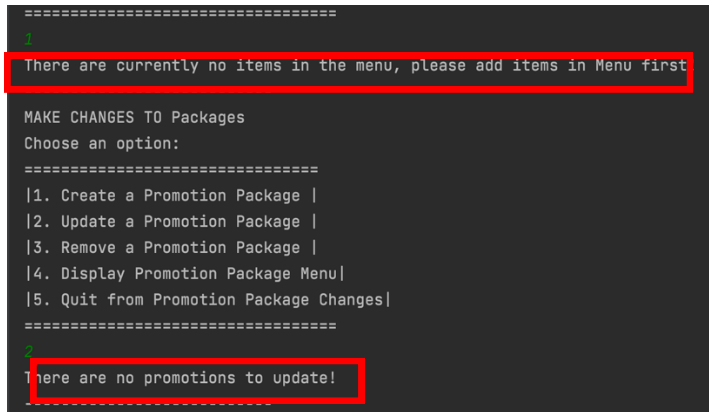
When these wrong inputs are entered, the reservation does not go to the next step, instead it would ask the user to re-input until an appropriate input is entered.

Input: 2021-11-12,09:30,22:00,11:30,2,PersonA,12345897
*Please adjust according to the current time

Test Case 2: Trying to add PromoPackage when in fact there are no MenuItems in the Menu

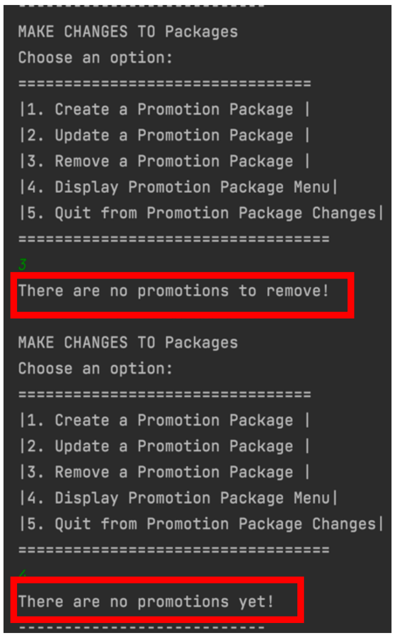


```
==================================
4
No items in Menu!
MAKE CHANGES TO MENU
```



```
=================================
1
There are currently no items in the menu, please add items in Menu first

MAKE CHANGES TO Packages
Choose an option:
================================
|1. Create a Promotion Package |
|2. Update a Promotion Package |
|3. Remove a Promotion Package |
|4. Display Promotion Package Menu|
|5. Quit from Promotion Package Changes|
=================================
2
There are no promotions to update!
---------------------------
```



```
---------------------------
MAKE CHANGES TO Packages
Choose an option:
================================
|1. Create a Promotion Package |
|2. Update a Promotion Package |
|3. Remove a Promotion Package |
|4. Display Promotion Package Menu|
|5. Quit from Promotion Package Changes|
=================================
3
There are no promotions to remove!

MAKE CHANGES TO Packages
Choose an option:
================================
|1. Create a Promotion Package |
|2. Update a Promotion Package |
|3. Remove a Promotion Package |
|4. Display Promotion Package Menu|
|5. Quit from Promotion Package Changes|
=================================
4
There are no promotions yet!
---------------------------
```

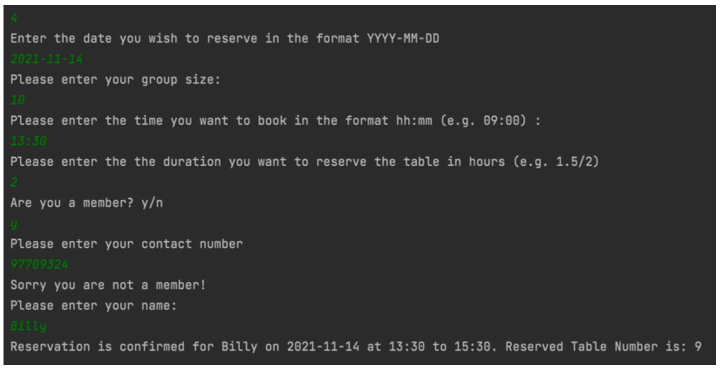
* Please take note that the menu.csv file and package.csv file should be empty for this to work properly! Please save the contents of the csv file elsewhere before removing its contents, as it may be needed for few other test cases. *

As you can see here, when the menu is empty (first picture on the right), all the functions in the "Make Changes to Packages" do not work, since there are no available MenuItems to create the packages.

Input: 1,4,5,1,2,3,4,5

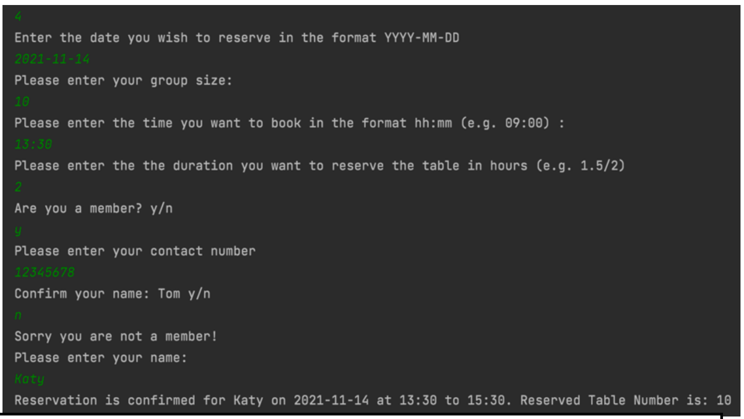
Test Case 3: During reservation, the customer proclaims he is a member when in fact he is not.

3.1 Mobile number does not exist in MembershipList



```
4
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-14
Please enter your group size:
10
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
13:30
Please enter the the duration you want to reserve the table in hours (e.g. 1.5/2)
2
Are you a member? y/n
y
Please enter your contact number
97709324
Sorry you are not a member!
Please enter your name:
Billy
Reservation is confirmed for Billy on 2021-11-14 at 13:30 to 15:30. Reserved Table Number is: 9
```

Mobile number does exist in MembershipList but Name is different



```
4
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-14
Please enter your group size:
10
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
13:30
Please enter the the duration you want to reserve the table in hours (e.g. 1.5/2)
2
Are you a member? y/n
y
Please enter your contact number
12345678
Confirm your name: Tom y/n
n
Sorry you are not a member!
Please enter your name:
Katy
Reservation is confirmed for Katy on 2021-11-14 at 13:30 to 15:30. Reserved Table Number is: 10
```

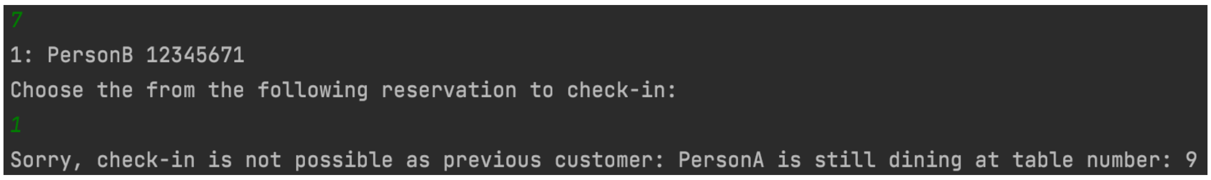
Input 3.1: 2021-11-12,10,13:30,2,y,97709324,Billy
Input 3.2: 2021-11-12,10,13:30,2,y,12345678,n,Katy
*Please adjust according to the current time

*Refer to project zip file to see the contents of MembersList.csv

In Scenario 3.1, the customer claims he is a member, but his mobile number does not exist in MembershipList (refer to the csv file). Hence, he will not receive membership discount upon checkout

In Scenario 3.2, the customer claims he is a member, and his mobile phone number exists in MembershipList. However, when the staff asks the Customer his name, he realizes that the name doesn't match and hence enters 'n'. So, this customer will not receive membership discount upon checkout as well

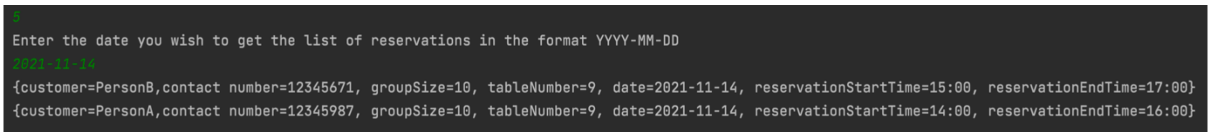
Test Case 4: Customer wants to check-in, but the reserved table is still occupied by a previous customer.



```
7
1: PersonB 12345671
Choose the from the following reservation to check-in:
1
Sorry, check-in is not possible as previous customer: PersonA is still dining at table number: 9
```

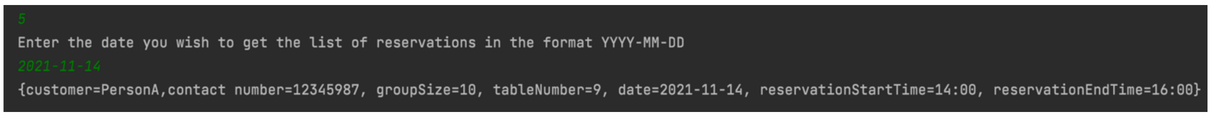
Test Case 5: Reservations that are 'active' are not automatically removed after 30 minutes, while only those 'inactive' orders are removed from active reservation.

5.1 Active Reservation at 15:25



```
5
Enter the date you wish to get the list of reservations in the format YYYY-MM-DD
2021-11-14
{customer=PersonB,contact number=12345671, groupSize=10, tableNumber=9, date=2021-11-14, reservationStartTime=15:00, reservationEndTime=17:00}
{customer=PersonA,contact number=12345987, groupSize=10, tableNumber=9, date=2021-11-14, reservationStartTime=14:00, reservationEndTime=16:00}
```

5.2 Active Reservation at 15:30



```
5
Enter the date you wish to get the list of reservations in the format YYYY-MM-DD
2021-11-14
{customer=PersonA,contact number=12345987, groupSize=10, tableNumber=9, date=2021-11-14, reservationStartTime=14:00, reservationEndTime=16:00}
```

NOTE: DO THE FOLLOWING STEPS BEFORE TESTING THIS CASE!

orderReservation.csv -> Replace with the following lines (adjust the date and time accordingly):

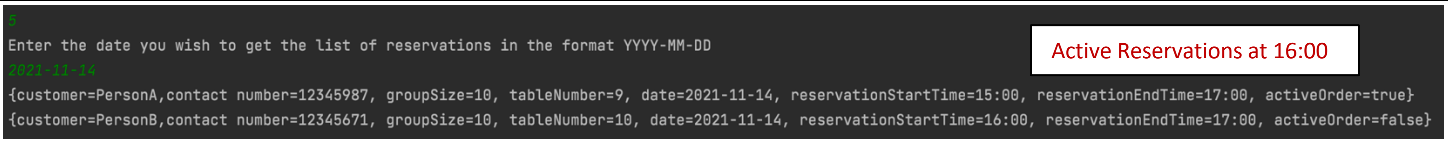
```
    PersonA;12345987;2021-11-14;9;10;14:00;16:00;
    PersonB;12345671;2021-11-14;10;10;15:00;17:00;2
```

activeOrder.csv -> Replace with the following lines

```
    1;{};{}
```

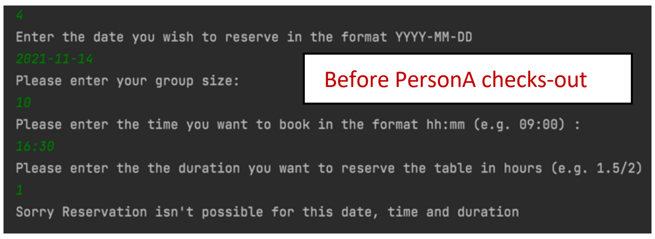
orderNumber.csv -> Replace the value in the first line with 2.

It is seen that personA has a reservation from 14:00-16:00 while PersonB has a reservation from 15:00-1700. PersonA has already checked-in and is considered an 'active order'. Once the order is active, the reservation will only be removed when PersonA checks out. However, since PersonB hasn't checked-in before 15:30, his reservation is automatically cancelled as shown in the second screenshot.

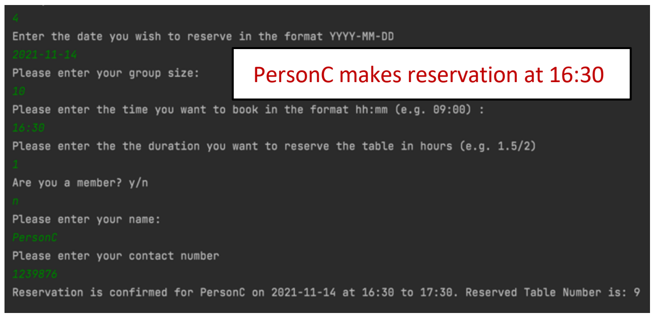


```
5
Enter the date you wish to get the list of reservations in the format YYYY-MM-DD
2021-11-14
{customer=PersonA,contact number=12345987, groupSize=10, tableNumber=9, date=2021-11-14, reservationStartTime=15:00, reservationEndTime=17:00, activeOrder=true}
{customer=PersonB,contact number=12345671, groupSize=10, tableNumber=10, date=2021-11-14, reservationStartTime=16:00, reservationEndTime=17:00, activeOrder=false}
```

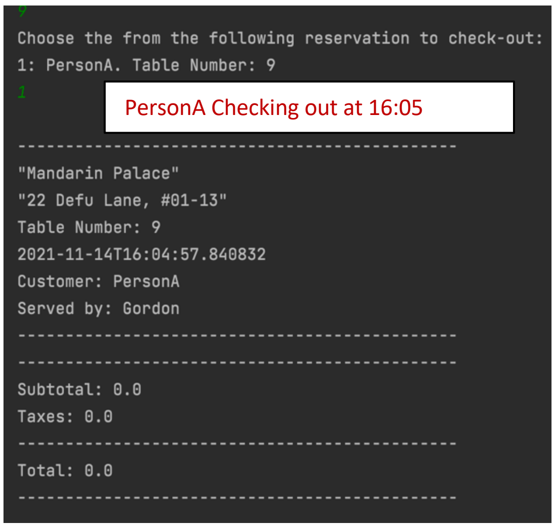
Active Reservations at 16:00



```
4
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-14
Please enter your group size:
10
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
16:30
Please enter the the duration you want to reserve the table in hours (e.g. 1.5/2)
1
Sorry Reservation isn't possible for this date, time and duration
```



```
4
Enter the date you wish to reserve in the format YYYY-MM-DD
2021-11-14
Please enter your group size:
10
Please enter the time you want to book in the format hh:mm (e.g. 09:00) :
16:30
Please enter the the duration you want to reserve the table in hours (e.g. 1.5/2)
1
Are you a member? y/n
n
Please enter your name:
PersonC
Please enter your contact number
1239876
Reservation is confirmed for PersonC on 2021-11-14 at 16:30 to 17:30. Reserved Table Number is: 9
```



```
9
Choose the from the following reservation to check-out:
1: PersonA. Table Number: 9
1
---------------------------------------------
"Mandarin Palace"
"22 Defu Lane, #01-13"
Table Number: 9
2021-11-14T16:04:57.840832
Customer: PersonA
Served by: Gordon
---------------------------------------------
---------------------------------------------
Subtotal: 0.0
Taxes: 0.0
---------------------------------------------
Total: 0.0
---------------------------------------------
```

NOTE: DO THE FOLLOWING STEPS BEFORE TESTING THIS CASE!

orderReservation.csv -> Replace with the following lines (adjust the date and time accordingly):

```
    PersonA;12345987;2021-11-14;9;10;15:00;17:00;1
    PersonB;12345671;2021-11-14;10;10;16:00;17:00;2
```

activeOrder.csv -> Replace with the following lines

```
    1;{};{}
```

orderNumber.csv -> Replace the value in the first line with 2.

It is seen that personA has a reservation from 15:00-17:00 while PersonB has a reservation from 16:00-1700. PersonA has already checked-in and is considered an ‘active order’, while PersonB hasn’t checked-in yet. PersonC wants to make a reservation for a group-size of 10 at 16:30, but he is unable to do so, since all tables that can accommodate a group size of 10 has been reserved/occupied. However, personB checks out early at 16:05, and when person tries to make the reservation again, he is now successful. (Refer to the screenshots above for detailed steps)

Input1: 5,2021-11-14 Input2: 4,2021-11-14,10,16:30,1 Input3: 9,1 Input4: 4,2021-11-14,10,16:30,1,n,PersonC,123456789

Test Case 7: In the event of a blackout/information all information should be retrieved back.

Use the original CSV Files in the ‘storeddata’ sub-package in the source code, while making relevant changes to the date/time in the orderReservation.csv/activeOrder.csv. In the case of any blackout, all active orders, orderReservation, orderInvoices are all retrieved back from the respective CSV Files. All the required information is retained, and all the functions work as expected. Since, it is difficult to show this information via screenshots, these are the steps we took to check if all works fine:

Populate the CSV Files
Look at the Active Reservations -> All added reservations are present.
Look at the active orders -> All added items/promo packages are initialized as expected
All previous order invoices have been retrieved, and the sales and revenue report can be generated as well.