**APPENDIX B:**

## 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 honoured 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 |
|---|---|---|---|
| ANNAMALAI TANISHA | CE2002 | SE1 | [illegible] 16 Apr 2022 |
| CHAN EU CHING | CE2002 | SE1 | [illegible] 13 Apr 2022 |
| CHUA JIN SHENG JASON | CE2002 | SE1 | [illegible] 13 Apr 2022 |
| ONG MENG HOW RYAN | CE2002 | SE1 | [illegible] 13 Apr 2022 |
| TULIP MAJUMDER | CE2002 | SE1 | [illegible] 17 Apr 2022 |

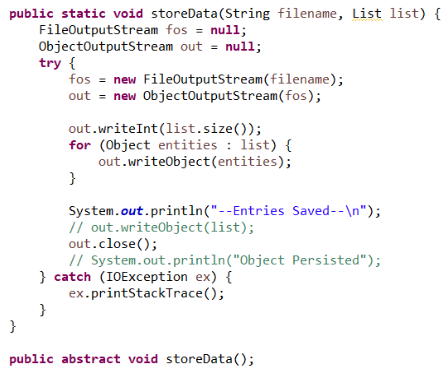
Important notes:

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

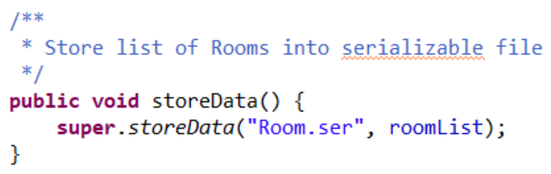
## Design Considerations

**Principles Used:**

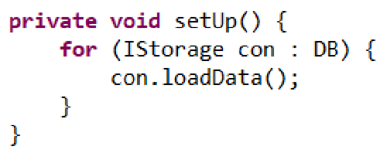
- Abstraction
    - StandardUI superclass defines abstract methods showSelection() and mainMenu(), and implemented methods getUserChoice(), getUserString(), and getUserYN() which are shared with subclasses CheckInOutUI, GuestUI, MenuUI, OrderUI, RoomUI, and ReservationUI.
    - SerializeDB superclass defines methods loadData() and storeData() which are shared with subclasses GuestController, Menu, OrderController, ReservationController, and RoomController.
- Encapsulation
    - Data encapsulation in entities. Access to an entity object's private data can only be done through the accessors and mutators. This helps to maintain data consistency.
- Single Responsibility Principle
    - Each control class interacts with its respective object classes
- Open-Closed Principle + Liskov Substitution Principle
    - All controllers are an extension of SerializeDB superclass containing implemented storeData(filename, List) and loadData(filename) methods, and implement the IStorage interface. If a new controller is added, only abstract methods storeData() and loadData() must be implemented to pass in the required values to the superclass' methods. loadData() and storeData() methods can be easily called during loop of setup and winddown.



```java
public static void storeData(String filename, List list) {
    FileOutputStream fos = null;
    ObjectOutputStream out = null;
    try {
        fos = new FileOutputStream(filename);
        out = new ObjectOutputStream(fos);

        out.writeInt(list.size());
        for (Object entities : list) {
            out.writeObject(entities);
        }

        System.out.println("--Entries Saved--\n");
        // out.writeObject(list);
        out.close();
        // System.out.println("Object Persisted");
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

public abstract void storeData();
```



```java
/**
 * Store list of Rooms into serializable file
 */
public void storeData() {
    super.storeData("Room.ser", roomList);
}
```



```java
private void setUp() {
    for (IStorage con : DB) {
        con.loadData();
    }
}
```

- Interface Segregation Principle
    - Creating interfaces for different purposes and implementing it when needed. Since our controller also contains entities to be stored, it will implement both IController (for CRUD) and IStorage (store and load from file).

- High cohesion between user interface and respective control classes
- Reusability
    1) Interface - IStorage
        - Controller storing data that needs to persist after the App is terminated
    2) Interface - IController and ControllerUI (Assigns common functionality to classes that function independently of one another)
        - RoomController, GuestController, ReservationController, OrderController, Menu implement interface IController which contain abstract methods: create(), read(), delete(), update(Object entities, int choice, String value)
        - RoomUI, GuestUI and ReservationUI, MenuUI implement ControllerUI which contain abstract methods: create(), delete(), update(), readOneDets().
    3) Superclass - SerializeDB
        - To implement logic for all data that needs to be serialised.
    4) Superclass - StandardUI
        - Subclasses CheckInOutUI, GuestUI, MenuUI, OrderUI, RoomUI, ReservationUI inherit methods getUserChoice(), getUserString(), getUserYN(), and redefine abstract methods showSelection() and mainMenu()

**Approach taken:**

- Main Structure used: Entity-Boundary-Controller
    - Entities used to contain attributes and accessor, mutator methods of each object
    - Boundary/User Interface to take in user input and pass information to the controller
    - Controller used to execute commands from Boundary and handles the logic and data manipulation of entities

- Proposed 2 relevant feature/functions how they can be implemented to MINIMISE impacts
    - Adding a new way to store and load data. This follows the polymorphism principle. This can be done by an object that contains data(in our case, controllers with an array list of entities) implementing the storeData() and

loadData() function in IStorage. In MainUI, the object can be added to the array list of IStorage. The object can then implement its own storing and loading method to its own file(text file, json file).

- Added a checkInOut class as a mediator across roomController, orderController, and reservationController. This class reduces inter-controller communication, allowing controllers to run their single responsibility of managing their respective entities and reduces coupling of classes.

**Classes Used:**

1. Main Class
   - Main Interface (Boundary class) : Handles input from the user using the main menu.
2. Boundary Class
   - Boundary classes for respective menu options (ReservationUI, OrderUI, RoomUI, GuestUI) that sends commands to respective Control Class.
3. Controller Class
   - GuestController: contains the methods to create, remove, update and delete guests.
   - ReservationController: contains the methods to create, remove, update and delete reservations. Expired reservations are removed with a built in function
   - RoomController: contains the methods to create, remove, update and delete rooms in the hotel.
   - Menu: contains the methods to create, remove, update and delete items from the catalogue. One can also view the whole catalogue of the whole menu available.
   - OrderController: contain the methods to create, remove, update and delete items from room service order.
4. Mediator Class
   - CheckInOut → Communicates between ReservationController, RoomController, and OrderController for Check-In, Check-Out, and payment processes
5. Entity Class
   - Guest: stores guestID, name, address, contact, country, gender, nationality
   - Reservation: stores reservationID, guestID, roomID, room type, check in and check out date, number of children and adults reservation status.

  - Room: stores roomID, guestID. Details of Room include price, type, view, smoking allowed, WiFi and status.
  - Order: contains a list of items, storing roomID and orderID, date, remarks and status of order.
  - Item: stores itemID, name , item description, item price and item type (Appetiser, Entree, Side, Dessert)

6. Serializable files to store information:
   - Each Controller has an arraylist of entities. Entities with their attributes are stored into their serializable files.
7. Enums
   - Attributes are kept in enums

**Assumptions Made**

- Currency is in SGD.
- Time zone is set at UTC +8 SGT.
- Payment is always successful (Provided credit card details are always correct)
- No need to log in
- No graphical interface for selection of rooms
- Guest will only make one booking at any Check-In Date
- Waitlisted reservations will always be confirmed when room is available
- Booking reservations will take note of room type but will not assign room. Guests are only assigned rooms upon check in, subject to room availability.
- Rooms will not be out of service for a lengthy period of time
- Guest details (contact, creditcard, …) are always entered correctly
- Room service can only be made from an “Occupied” room

## **Documentation of Java API**

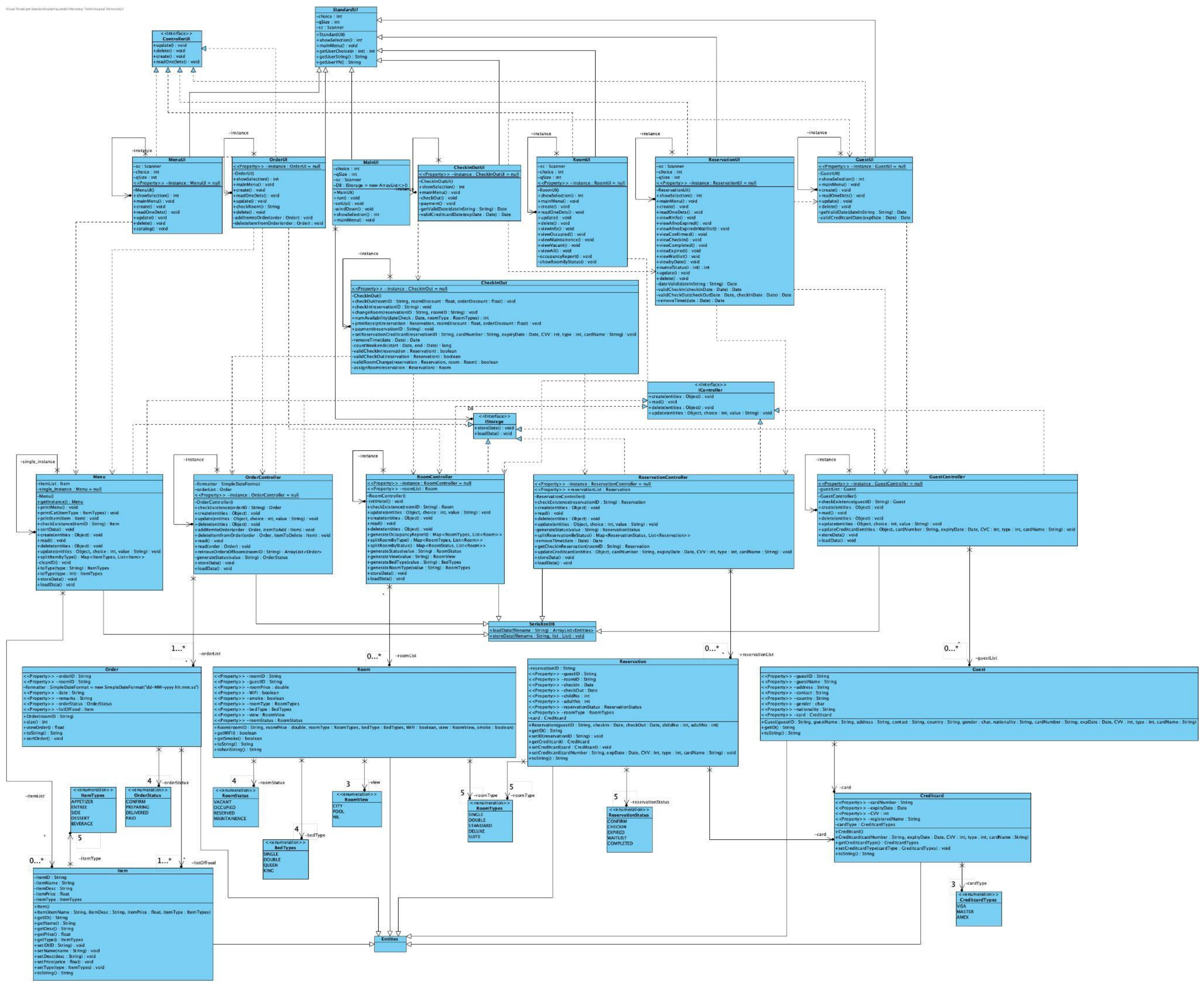
The documentation of the Java API can be found in index.html under the src/docs folder.

## **Video Link**

Link: https://www.youtube.com/watch?v=rAE63JVDagY&ab_channel=Uztra

## **UML Class Diagram**

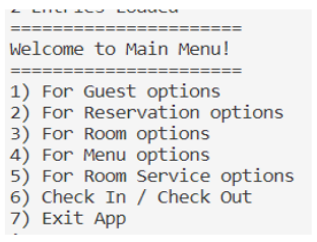
* Please refer to UML_Class_Diagram.jpg in the folder for a clearer image



- Dashed lines with an arrowhead represent a dependency relationship. Dependent class methods can be called by using class.
- Solid lines represent an Association (‘has-a’) relationship. Each of the control classes are associated with their respective entities (RoomController has-a Room). Each of the entities are associated with their relevant enumerators (e.g. Room Entity associated to (has-a) RoomView, RoomTypes, BedTypes etc)

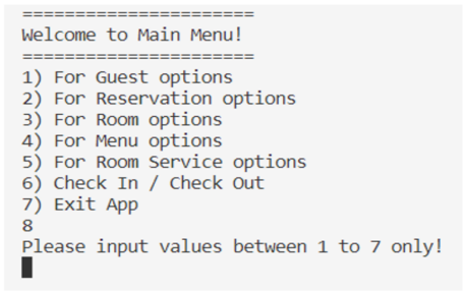
## **Test Cases and Results**

Main Menu



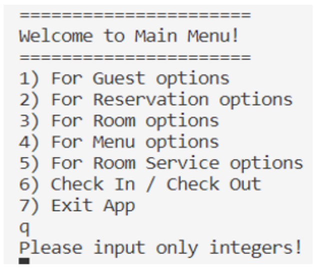
```
2 Entries Loaded
======================
Welcome to Main Menu!
======================
1) For Guest options
2) For Reservation options
3) For Room options
4) For Menu options
5) For Room Service options
6) Check In / Check Out
7) Exit App
```

Input out of range



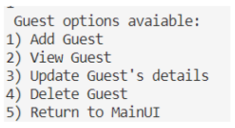
```
======================
Welcome to Main Menu!
======================
1) For Guest options
2) For Reservation options
3) For Room options
4) For Menu options
5) For Room Service options
6) Check In / Check Out
7) Exit App
8
Please input values between 1 to 7 only!
```

Random Char Input



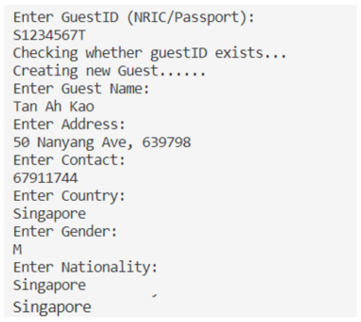
```
======================
Welcome to Main Menu!
======================
1) For Guest options
2) For Reservation options
3) For Room options
4) For Menu options
5) For Room Service options
6) Check In / Check Out
7) Exit App
q
Please input only integers!
```

Guest options

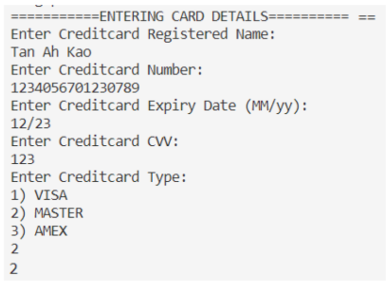


```
 Guest options avaiable:
1) Add Guest
2) View Guest
3) Update Guest's details
4) Delete Guest
5) Return to MainUI
```

Adding Guest

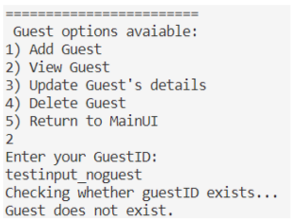


```
Enter GuestID (NRIC/Passport):
S1234567T
Checking whether guestID exists...
Creating new Guest......
Enter Guest Name:
Tan Ah Kao
Enter Address:
50 Nanyang Ave, 639798
Enter Contact:
67911744
Enter Country:
Singapore
Enter Gender:
M
Enter Nationality:
Singapore
Singapore
```



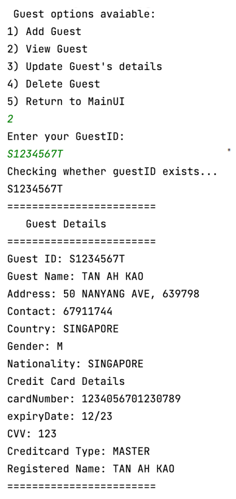
```
===========ENTERING CARD DETAILS========== ==
Enter Creditcard Registered Name:
Tan Ah Kao
Enter Creditcard Number:
1234056701230789
Enter Creditcard Expiry Date (MM/yy):
12/23
Enter Creditcard CVV:
123
Enter Creditcard Type:
1) VISA
2) MASTER
3) AMEX
2
2
```

Invalid guest ID input



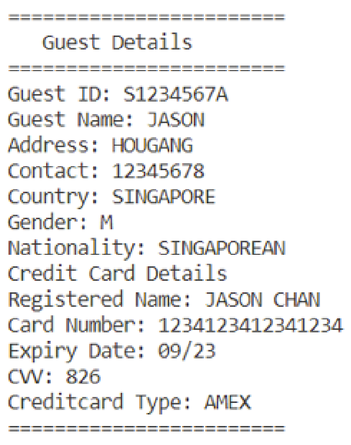
```
=======================
 Guest options avaiable:
1) Add Guest
2) View Guest
3) Update Guest's details
4) Delete Guest
5) Return to MainUI
2
Enter your GuestID:
testinput_noguest
Checking whether guestID exists...
Guest does not exist.
```

Guest found



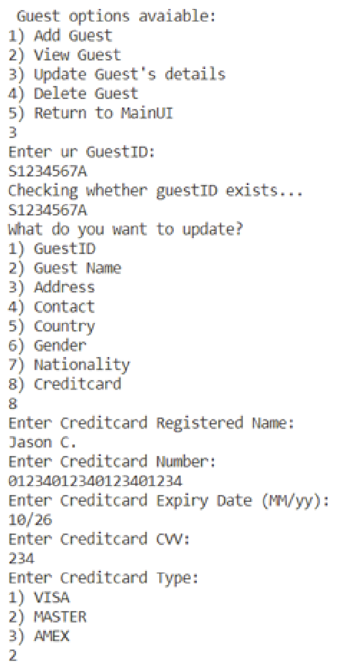
```
 Guest options avaiable:
1) Add Guest
2) View Guest
3) Update Guest's details
4) Delete Guest
5) Return to MainUI
2
Enter your GuestID:
S1234567T
Checking whether guestID exists...
S1234567T
========================
   Guest Details
========================
Guest ID: S1234567T
Guest Name: TAN AH KAO
Address: 50 NANYANG AVE, 639798
Contact: 67911744
Country: SINGAPORE
Gender: M
Nationality: SINGAPORE
Credit Card Details
cardNumber: 1234056701230789
expiryDate: 12/23
CVV: 123
Creditcard Type: MASTER
Registered Name: TAN AH KAO
========================
```

## Updating Guest: Before



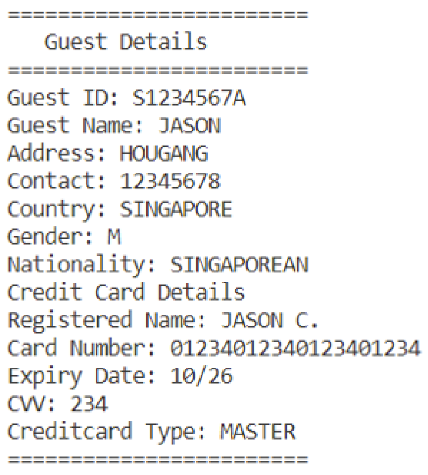
```
========================
   Guest Details
========================
Guest ID: S1234567A
Guest Name: JASON
Address: HOUGANG
Contact: 12345678
Country: SINGAPORE
Gender: M
Nationality: SINGAPOREAN
Credit Card Details
Registered Name: JASON CHAN
Card Number: 1234123412341234
Expiry Date: 09/23
CVV: 826
Creditcard Type: AMEX
========================
```

## Changing Details: E.g. credit card



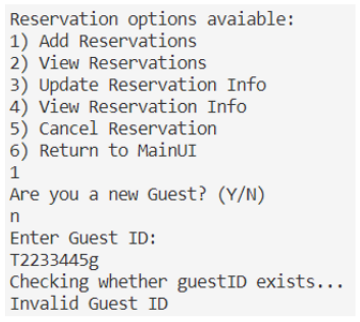
```
 Guest options avaiable:
1) Add Guest
2) View Guest
3) Update Guest's details
4) Delete Guest
5) Return to MainUI
3
Enter ur GuestID:
S1234567A
Checking whether guestID exists...
S1234567A
What do you want to update?
1) GuestID
2) Guest Name
3) Address
4) Contact
5) Country
6) Gender
7) Nationality
8) Creditcard
8
Enter Creditcard Registered Name:
Jason C.
Enter Creditcard Number:
01234012340123401234
Enter Creditcard Expiry Date (MM/yy):
10/26
Enter Creditcard CVV:
234
Enter Creditcard Type:
1) VISA
2) MASTER
3) AMEX
2
```

## After



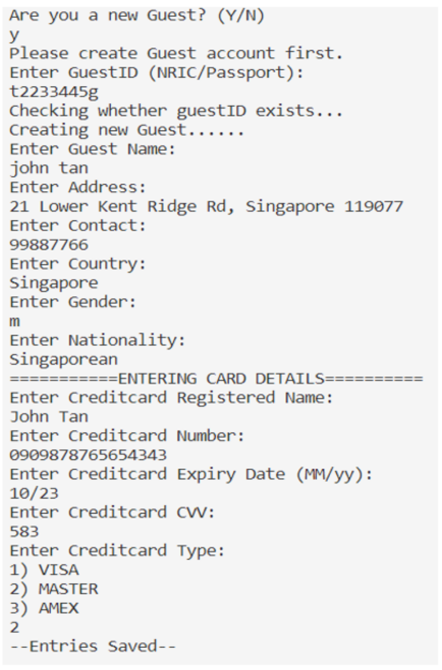
```
========================
   Guest Details
========================
Guest ID: S1234567A
Guest Name: JASON
Address: HOUGANG
Contact: 12345678
Country: SINGAPORE
Gender: M
Nationality: SINGAPOREAN
Credit Card Details
Registered Name: JASON C.
Card Number: 01234012340123401234
Expiry Date: 10/26
CVV: 234
Creditcard Type: MASTER
========================
```

## Reservation Options + Invalid Guest Input



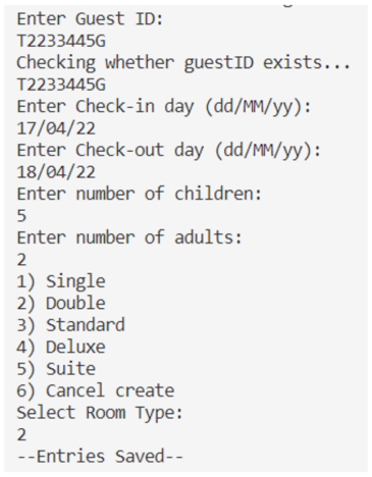
```
Reservation options avaiable:
1) Add Reservations
2) View Reservations
3) Update Reservation Info
4) View Reservation Info
5) Cancel Reservation
6) Return to MainUI
1
Are you a new Guest? (Y/N)
n
Enter Guest ID:
T2233445g
Checking whether guestID exists...
Invalid Guest ID
```

## Entry of new guest from Reservation



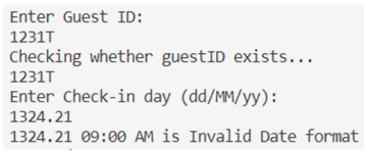
```
Are you a new Guest? (Y/N)
y
Please create Guest account first.
Enter GuestID (NRIC/Passport):
t2233445g
Checking whether guestID exists...
Creating new Guest......
Enter Guest Name:
john tan
Enter Address:
21 Lower Kent Ridge Rd, Singapore 119077
Enter Contact:
99887766
Enter Country:
Singapore
Enter Gender:
m
Enter Nationality:
Singaporean
===========ENTERING CARD DETAILS==========
Enter Creditcard Registered Name:
John Tan
Enter Creditcard Number:
0909878765654343
Enter Creditcard Expiry Date (MM/yy):
10/23
Enter Creditcard CVV:
583
Enter Creditcard Type:
1) VISA
2) MASTER
3) AMEX
2
--Entries Saved--
```

## Adding new Reservation



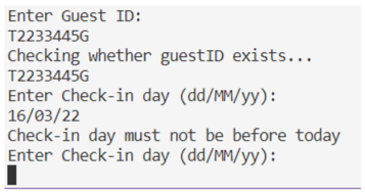
```
Enter Guest ID:
T2233445G
Checking whether guestID exists...
T2233445G
Enter Check-in day (dd/MM/yy):
17/04/22
Enter Check-out day (dd/MM/yy):
18/04/22
Enter number of children:
5
Enter number of adults:
2
1) Single
2) Double
3) Standard
4) Deluxe
5) Suite
6) Cancel create
Select Room Type:
2
--Entries Saved--
```

Invalid Date format



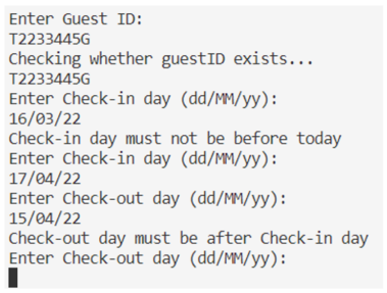
```
Enter Guest ID:
1231T
Checking whether guestID exists...
1231T
Enter Check-in day (dd/MM/yy):
1324.21
1324.21 09:00 AM is Invalid Date format
```

Invalid Check in date input



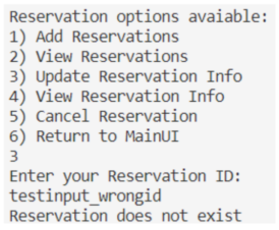
```
Enter Guest ID:
T2233445G
Checking whether guestID exists...
T2233445G
Enter Check-in day (dd/MM/yy):
16/03/22
Check-in day must not be before today
Enter Check-in day (dd/MM/yy):
```

Invalid Check out date input



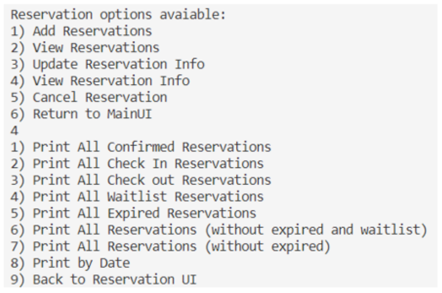
```
Enter Guest ID:
T2233445G
Checking whether guestID exists...
T2233445G
Enter Check-in day (dd/MM/yy):
16/03/22
Check-in day must not be before today
Enter Check-in day (dd/MM/yy):
17/04/22
Enter Check-out day (dd/MM/yy):
15/04/22
Check-out day must be after Check-in day
Enter Check-out day (dd/MM/yy):
```

Invalid Reservation input



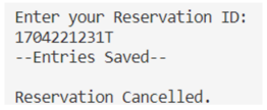
```
Reservation options avaiable:
1) Add Reservations
2) View Reservations
3) Update Reservation Info
4) View Reservation Info
5) Cancel Reservation
6) Return to MainUI
3
Enter your Reservation ID:
testinput_wrongid
Reservation does not exist
```

Update Reservation options



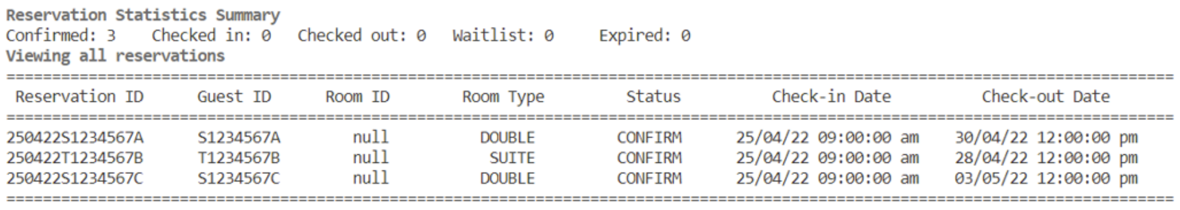
```
Reservation options avaiable:
1) Add Reservations
2) View Reservations
3) Update Reservation Info
4) View Reservation Info
5) Cancel Reservation
6) Return to MainUI
4
1) Print All Confirmed Reservations
2) Print All Check In Reservations
3) Print All Check out Reservations
4) Print All Waitlist Reservations
5) Print All Expired Reservations
6) Print All Reservations (without expired and waitlist)
7) Print All Reservations (without expired)
8) Print by Date
9) Back to Reservation UI
```

Deleting Reservation



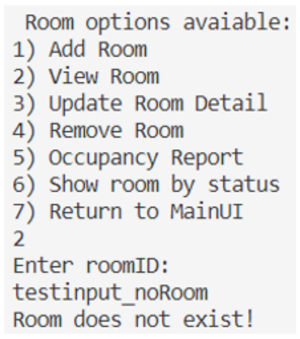
```
Enter your Reservation ID:
1704221231T
--Entries Saved--

Reservation Cancelled.
```

Print All Reservations *[expected; details created above]*



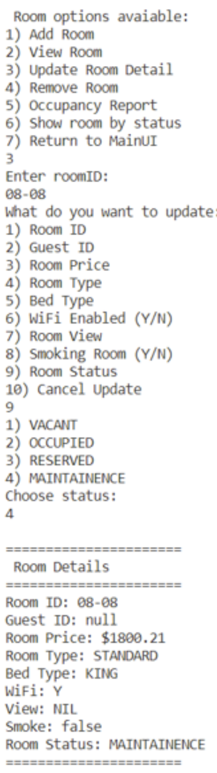
```
Reservation Statistics Summary
Confirmed: 3    Checked in: 0   Checked out: 0   Waitlist: 0    Expired: 0
Viewing all reservations
==========================================================================================================
 Reservation ID       Guest ID      Room ID       Room Type       Status          Check-in Date          Check-out Date
==========================================================================================================
250422S1234567A      S1234567A       null          DOUBLE        CONFIRM     25/04/22 09:00:00 am    30/04/22 12:00:00 pm
250422T1234567B      T1234567B       null           SUITE        CONFIRM     25/04/22 09:00:00 am    28/04/22 12:00:00 pm
250422S1234567C      S1234567C       null          DOUBLE        CONFIRM     25/04/22 09:00:00 am    03/05/22 12:00:00 pm
==========================================================================================================
```

Room Options +
Invalid Room Input



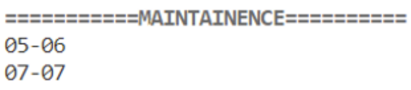
```
 Room options avaiable:
1) Add Room
2) View Room
3) Update Room Detail
4) Remove Room
5) Occupancy Report
6) Show room by status
7) Return to MainUI
2
Enter roomID:
testinput_noRoom
Room does not exist!
```

Change status to
“Under Maintenance”



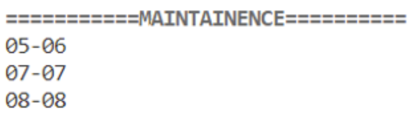
```
 Room options avaiable:
1) Add Room
2) View Room
3) Update Room Detail
4) Remove Room
5) Occupancy Report
6) Show room by status
7) Return to MainUI
3
Enter roomID:
08-08
What do you want to update:
1) Room ID
2) Guest ID
3) Room Price
4) Room Type
5) Bed Type
6) WiFi Enabled (Y/N)
7) Room View
8) Smoking Room (Y/N)
9) Room Status
10) Cancel Update
9
1) VACANT
2) OCCUPIED
3) RESERVED
4) MAINTAINENCE
Choose status:
4

====================
 Room Details
====================
Room ID: 08-08
Guest ID: null
Room Price: $1800.21
Room Type: STANDARD
Bed Type: KING
WiFi: Y
View: NIL
Smoke: false
Room Status: MAINTAINENCE
====================
```

Update Room:
Before

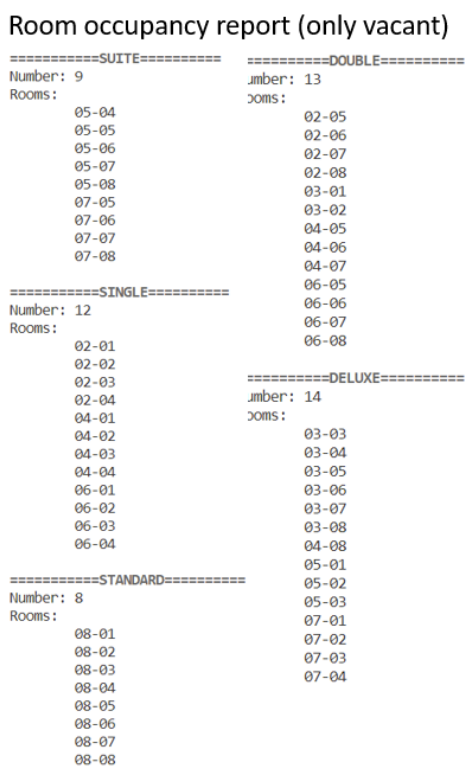


```
===========MAINTAINENCE==========
05-06
07-07
```

After

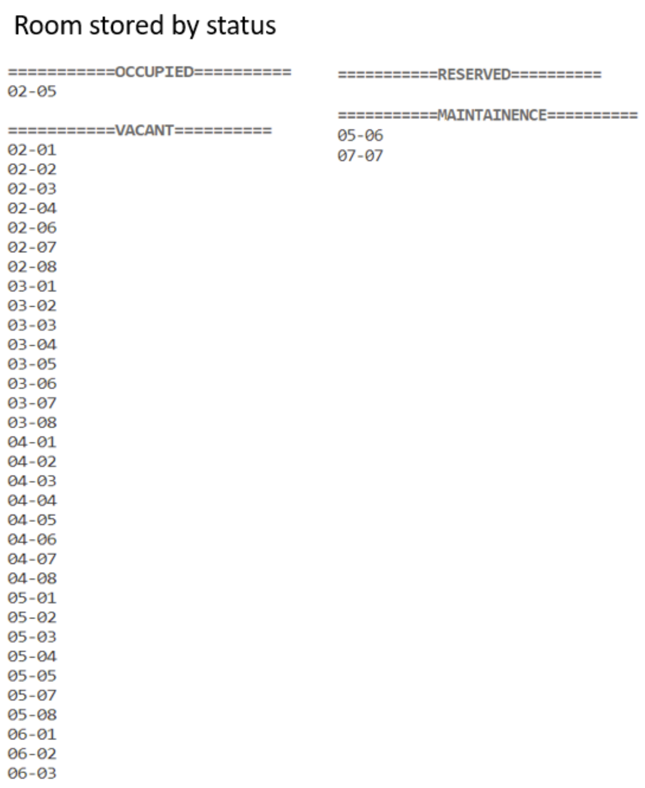


```
===========MAINTAINENCE==========
05-06
07-07
08-08
```



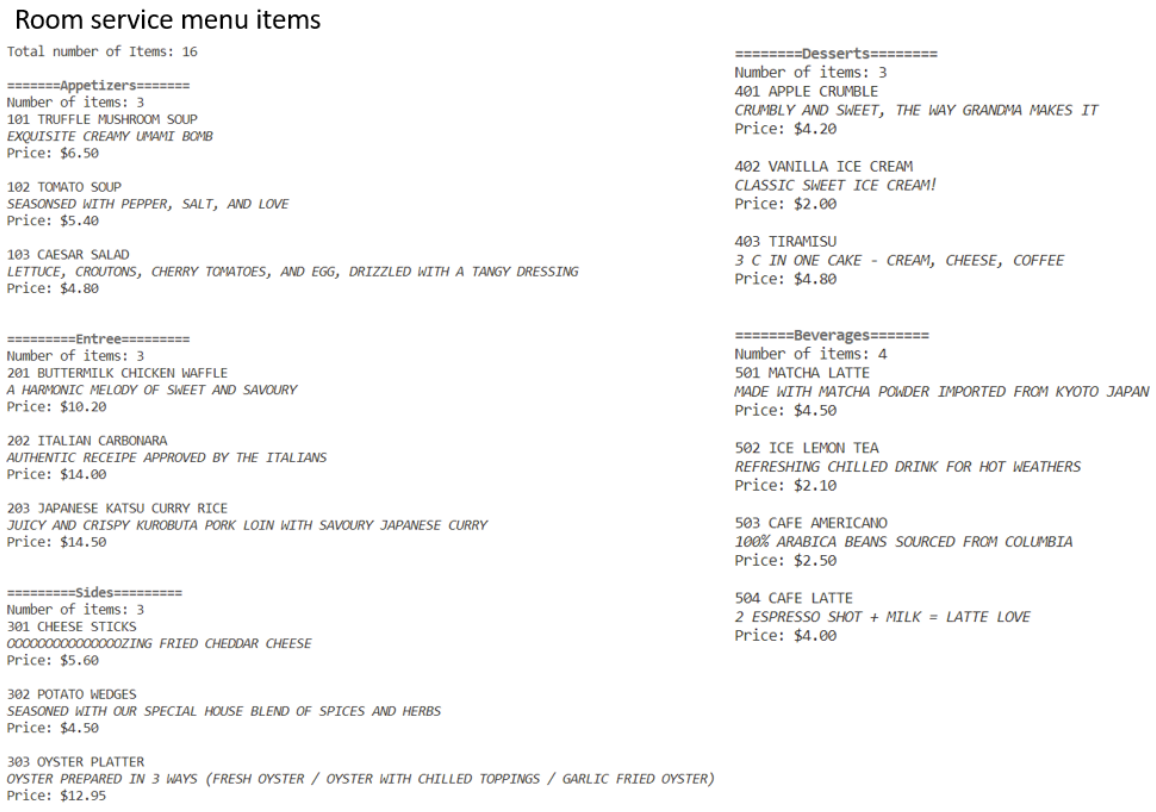
Room occupancy report (only vacant)

```
===========SUITE==========        ==========DOUBLE==========
Number: 9                         umber: 13
Rooms:                            ooms:
        05-04                             02-05
        05-05                             02-06
        05-06                             02-07
        05-07                             02-08
        05-08                             03-01
        07-05                             03-02
        07-06                             04-05
        07-07                             04-06
        07-08                             04-07
                                          06-05
==========SINGLE==========                06-06
Number: 12                                06-07
Rooms:                                    06-08
        02-01
        02-02                     ==========DELUXE==========
        02-03                     umber: 14
        02-04                     ooms:
        04-01                             03-03
        04-02                             03-04
        04-03                             03-05
        04-04                             03-06
        06-01                             03-07
        06-02                             03-08
        06-03                             04-08
        06-04                             05-01
                                          05-02
===========STANDARD==========              05-03
Number: 8                                 07-01
Rooms:                                    07-02
        08-01                             07-03
        08-02                             07-04
        08-03
        08-04
        08-05
        08-06
        08-07
        08-08
```



Room stored by status

```
===========OCCUPIED==========        ===========RESERVED==========
02-05
                                     ===========MAINTAINENCE==========
===========VACANT==========          05-06
02-01                                07-07
02-02
02-03
02-04
02-06
02-07
02-08
03-01
03-02
03-03
03-04
03-05
03-06
03-07
03-08
04-01
04-02
04-03
04-04
04-05
04-06
04-07
04-08
05-01
05-02
05-03
05-04
05-05
05-07
05-08
06-01
06-02
06-03
```

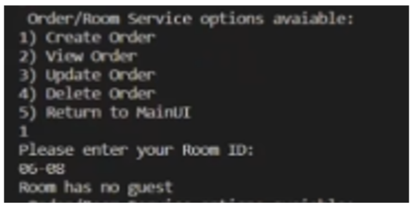


Room service menu items

```
Total number of Items: 16

=======Appetizers=======
Number of items: 3
101 TRUFFLE MUSHROOM SOUP
EXQUISITE CREAMY UMAMI BOMB
Price: $6.50

102 TOMATO SOUP
SEASONSED WITH PEPPER, SALT, AND LOVE
Price: $5.40

103 CAESAR SALAD
LETTUCE, CROUTONS, CHERRY TOMATOES, AND EGG, DRIZZLED WITH A TANGY DRESSING
Price: $4.80


=========Entree=========
Number of items: 3
201 BUTTERMILK CHICKEN WAFFLE
A HARMONIC MELODY OF SWEET AND SAVOURY
Price: $10.20

202 ITALIAN CARBONARA
AUTHENTIC RECEIPE APPROVED BY THE ITALIANS
Price: $14.00

203 JAPANESE KATSU CURRY RICE
JUICY AND CRISPY KUROBUTA PORK LOIN WITH SAVOURY JAPANESE CURRY
Price: $14.50


=========Sides=========
Number of items: 3
301 CHEESE STICKS
OOOOOOOOOOOOOOOZING FRIED CHEDDAR CHEESE
Price: $5.60

302 POTATO WEDGES
SEASONED WITH OUR SPECIAL HOUSE BLEND OF SPICES AND HERBS
Price: $4.50

303 OYSTER PLATTER
OYSTER PREPARED IN 3 WAYS (FRESH OYSTER / OYSTER WITH CHILLED TOPPINGS / GARLIC FRIED OYSTER)
Price: $12.95
```

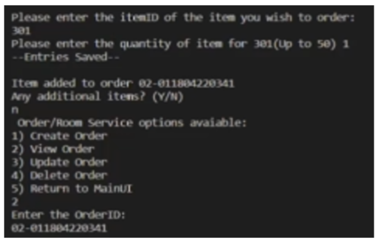
```
========Desserts========
Number of items: 3
401 APPLE CRUMBLE
CRUMBLY AND SWEET, THE WAY GRANDMA MAKES IT
Price: $4.20

402 VANILLA ICE CREAM
CLASSIC SWEET ICE CREAM!
Price: $2.00

403 TIRAMISU
3 C IN ONE CAKE - CREAM, CHEESE, COFFEE
Price: $4.80


=======Beverages=======
Number of items: 4
501 MATCHA LATTE
MADE WITH MATCHA POWDER IMPORTED FROM KYOTO JAPAN
Price: $4.50

502 ICE LEMON TEA
REFRESHING CHILLED DRINK FOR HOT WEATHERS
Price: $2.10

503 CAFE AMERICANO
100% ARABICA BEANS SOURCED FROM COLUMBIA
Price: $2.50

504 CAFE LATTE
2 ESPRESSO SHOT + MILK = LATTE LOVE
Price: $4.00
```

Order options, unoccupied room input



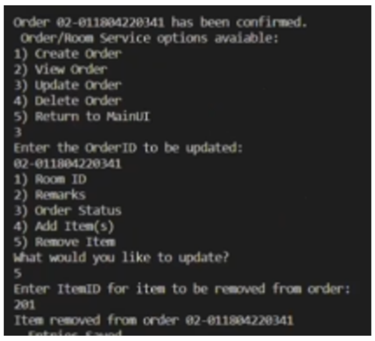
```
 Order/Room Service options avaiable:
1) Create Order
2) View Order
3) Update Order
4) Delete Order
5) Return to MainUI
1
Please enter your Room ID:
06-08
Room has no guest
```

Create order



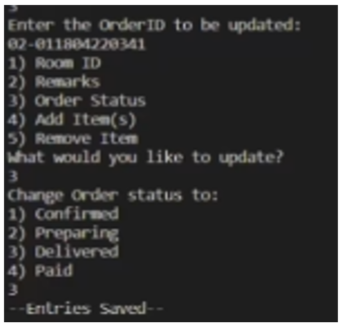
```
Please enter the itemID of the item you wish to order:
301
Please enter the quantity of item for 301(Up to 50) 1
--Entries Saved--

Item added to order 02-011804220341
Any additional items? (Y/N)
n
 Order/Room Service options avaiable:
1) Create Order
2) View Order
3) Update Order
4) Delete Order
5) Return to MainUI
2
Enter the OrderID:
02-011804220341
```

Remove item from order



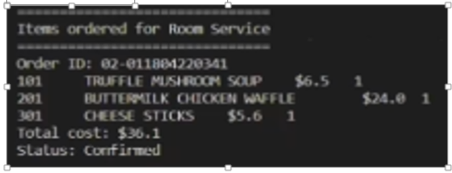
```
Order 02-011804220341 has been confirmed.
 Order/Room Service options avaiable:
1) Create Order
2) View Order
3) Update Order
4) Delete Order
5) Return to MainUI
3
Enter the OrderID to be updated:
02-011804220341
1) Room ID
2) Remarks
3) Order Status
4) Add Item(s)
5) Remove Item
What would you like to update?
5
Enter ItemID for item to be removed from order:
201
Item removed from order 02-011804220341
```

Update order status



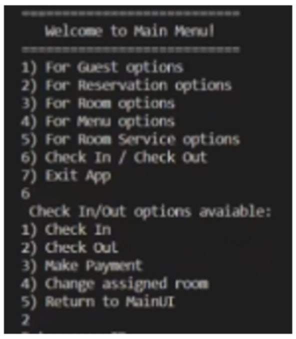
```
Enter the OrderID to be updated:
02-011804220341
1) Room ID
2) Remarks
3) Order Status
4) Add Item(s)
5) Remove Item
What would you like to update?
3
Change Order status to:
1) Confirmed
2) Preparing
3) Delivered
4) Paid
3
--Entries Saved--
```

View order



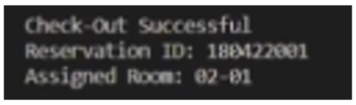
```
==============================
Items ordered for Room Service
==============================
Order ID: 02-011804220341
101     TRUFFLE MUSHROOM SOUP    $6.5   1
201     BUTTERMILK CHICKEN WAFFLE        $24.0  1
301     CHEESE STICKS    $5.6   1
Total cost: $36.1
Status: Confirmed
```

Check in/out options

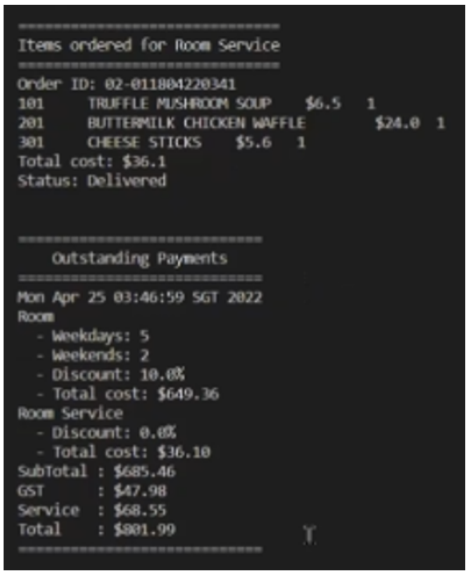


```
=========================
  Welcome to Main Menu!
=========================
1) For Guest options
2) For Reservation options
3) For Room options
4) For Menu options
5) For Room Service options
6) Check In / Check Out
7) Exit App
6
 Check In/Out options avaiable:
1) Check In
2) Check Out
3) Make Payment
4) Change assigned room
5) Return to MainUI
2
```

Check out success + bill

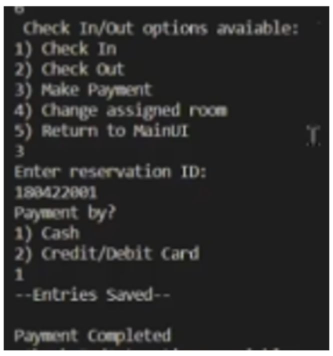


```
Check-Out Successful
Reservation ID: 180422001
Assigned Room: 02-01
```



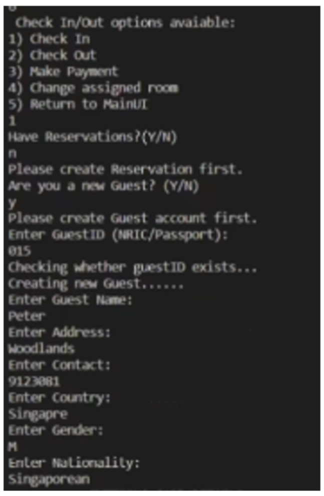
```
==============================
Items ordered for Room Service
==============================
Order ID: 02-011804220341
101     TRUFFLE MUSHROOM SOUP    $6.5   1
201     BUTTERMILK CHICKEN WAFFLE        $24.0  1
301     CHEESE STICKS    $5.6   1
Total cost: $36.1
Status: Delivered


============================
    Outstanding Payments
============================
Mon Apr 25 03:46:59 SGT 2022
Room
  - Weekdays: 5
  - Weekends: 2
  - Discount: 10.0%
  - Total cost: $649.36
Room Service
  - Discount: 0.0%
  - Total cost: $36.10
SubTotal : $685.46
GST      : $47.98
Service  : $68.55
Total    : $801.99
============================
```

Payment made



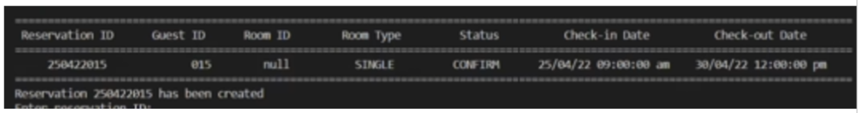
```
 Check In/Out options avaiable:
1) Check In
2) Check Out
3) Make Payment
4) Change assigned room
5) Return to MainUI
3
Enter reservation ID:
180422001
Payment by?
1) Cash
2) Credit/Debit Card
1
--Entries Saved--

Payment Completed
```

Check in walk in guest



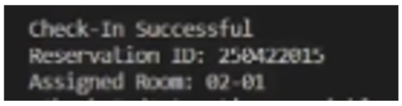
```
 Check In/Out options avaiable:
1) Check In
2) Check Out
3) Make Payment
4) Change assigned room
5) Return to MainUI
1
Have Reservations?(Y/N)
n
Please create Reservation first.
Are you a new Guest? (Y/N)
y
Please create Guest account first.
Enter GuestID (NRIC/Passport):
015
Checking whether guestID exists...
Creating new Guest......
Enter Guest Name:
Peter
Enter Address:
Woodlands
Enter Contact:
9123081
Enter Country:
Singapre
Enter Gender:
M
Enter Nationality:
Singaporean
```

Walk in guest on the day reservation



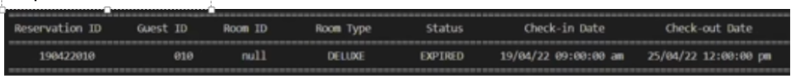
```
==================================================================================================================
 Reservation ID     Guest ID    Room ID      Room Type       Status          Check-in Date          Check-out Date
==================================================================================================================
    250422015          015        null         SINGLE        CONFIRM     25/04/22 09:00:00 am   30/04/22 12:00:00 pm
==================================================================================================================
Reservation 250422015 has been created
```

Walk in check in success



```
Check-In Successful
Reservation ID: 250422015
Assigned Room: 02-01
```

Expired reservation



```
==================================================================================================================
Reservation ID     Guest ID    Room ID      Room Type       Status          Check-in Date          Check-out Date
==================================================================================================================
   190422010          010        null         DELUXE        EXPIRED     19/04/22 09:00:00 am   25/04/22 12:00:00 pm
==================================================================================================================
```

## Reflection

- A consistent coding and style has to be discussed within the team to allow for better debugging experience. This can be further emphasised by following OODP principles. Abstraction allows for reusability of code.
- The vast number of attributes require an extensible way for updates to be implemented. By understanding hierarchy of class, upcasting and downcasting can be maintained more easily.
- Successful and error test cases should have been identified earlier to aid us in the coding process, ensuring that our program checks for said errors, and to correctly test out the functional requirements.