1. Functionality specification

Users: customer, clerk and manager

1.1 Make Reservation

User: Customer

Input: check-in date, check-out date, room type, lastName, firstName, phone#, address

Output: A unique resID for reservation or "Sorry, no rooms available"

Basic Case:

The system will check if any room is available according to the given check-in date, check-out date, room type domain. If there available rooms, the system returns ‘OK’ and customer successfully make a reservation and get a resID. If there are no rooms available, then the system will tell the user there are no rooms available.

Exceptions:

If check-in date is after the check-out date, then transaction is aborted and "Sorry, no rooms available" is return.
If the check-in date or the check-out date is after the current date, then transaction is aborted and "Sorry, no rooms available" is return.

1.2 Modify Reservation

User: Customer

Input: The unique ID for reservation, new check-in date, new check-out date, new room type, lastName, phone#

Output: "OK" or "change not available"

Basic Case:

The system first check if lastName and phone# matched the record of the reservation ID. If nor, return “Wrong credential”.

The system will check if any room is available according to the given check-in date, check-out date, room type domain. If there available rooms, the system returns ‘OK’ and update the reservation according to the given resID. If there are no rooms available, then the system will tell the user "change not available".

Exceptions:

If check-in date is after the check-out date, then transaction is aborted and "change not available" is return.
If the check-in date or the check-out date is after the current date, then transaction is aborted and "change not available" is return.

1.3 Cancel Reservation

User: Customer

Input: A unique resID. lastName, phone#

Output: "OK" or "Invalid reservation ID"

Basic Case:

The system first check if lastName and phone# matched the record of the reservation ID. If nor, return “Wrong credential”.

The system will check if the reservation ID exist in the database. If it does, it will delete the reservation in the reservation table. If it does not exist, then the system will tell the user the input ID does not exist.

Exceptions:

The reservation ID is not in the database. the system will tell the user the input ID does not exist.

1.4 Check Available

User: Customer

Input: check-in date, check-out date, room type

Output: Total fee or "Sorry, no rooms available"

Basic Case:

The system will check if any room is available according to the given check-in date, check-out date, room type domain and return the total fee for the service. The customer can decide if they make the reservation with the estimated price. If there are no rooms available, then the system will tell the user there are no rooms available.

Exceptions:

If check-in date is after the check-out date, then transaction is aborted and "Sorry, no rooms available" is return.

If the check-in date or the check-out date is after the current date, then transaction is aborted and "Sorry, no rooms available" is return.

1.5 Check-in with reservation

User: Clerk

Input: resID

Output: Total fee or "Sorry, no rooms available"

Basic Case:

The clerk will check in the customer with a reservation. A room and a parking lot is assigned. The customer will pay the deposit.

Exceptions:

If resId does not exist, transaction is aborted and "Failed" is returned.

1.6 Check-in without reservation

User: Clerk

Input: lastName, firstName, phone#, address, RoomType

Output: checkinId, room#, lot#, deposit, checkinDate, or "Failed"

Basic Case:

The clerk will check in the customer without a reservation. A room and a parking lot is assigned. The customer will pay the deposit.

Exceptions:

If no room available, transaction is aborted and "Failed" is returned.

1.7 Check-out

User: Clerk

Input: room#

Output: checkoutId, fine, checkoutDate, roomFee, card#, expDate, or "Failed"

Basic Case:

The clerk will check out the customer. The customer will pay roomFee plus fine using the credit card. The deposit is returned to the customer. If the customer is a member, calculate the roomFee according to memberRate. If the customer is a member, add roomFee to his/her points.

Exceptions:

If the room with room# is not occupied, transaction is aborted and "Failed" is returned.
If credit card is not authorized, the customer is asked to pay cash.

1.8 AddMembership

User: Clerk

Input: lastName, firstName, phone#, address

Output: “OK” or "Failed"

Basic Case:
The clerk will grant the customer membership. Set points to 0.

Exceptions:

If the customer is already a member, transaction is aborted and "Failed" is returned.

1.9 RemoveMembership

User: Clerk

Input: lastName, firstName, phone#, address

Output: “OK” or "Failed"

Basic Case:
The clerk will remove the customer membership.

Exceptions:

If the customer is not a member, transaction is aborted and "Failed" is returned.

1.10 Change Room Type

User: Manager

Input: Room Number, New Room Type

Output: Room Type Change Successful/Unsuccessful

Basic Case:

For example, if the manager needs to change a room with 1 bed, to one with 3, this is the operation they'd use.

This operation allows the Manager to select a desired room type, and update the room's type.

Exceptions:

If the room is occupied, the manager can't change the room type.
If the Room Number doesn't exist, the operation is aborted.

1.11 ChangeRate

User: Manager

Input: Room Number, New Rate

Output: "Room rate was changed to 10$ a night"

Basic Case:

The manager wants to update the room rate because demand has gone up, so they increase the rate to their desired amount.

Exceptions:

If the Room Number doesn't exist, the operation is aborted
The rate must be greater than 0.

1.12 ChangeMemberRate

User: Manager

Input: Room Number, New Member Rate

Output: "Member Room rate was changed to 10$ a night"

Basic Case:

The manager decides to give members a nice new discount for the holidays, so they decrease the membership room rate to 10$.

Exceptions:

If the Room Number doesn't exist, the operation is aborted
The rate must be greater than 0.

1.13 AddRoom

User: Manager

Input: Room Type

Output: "Room number 10001 has been added to the hotel" or "New Room could not be added"

Basic Case:

The manager wants to expand the hotel, so they start construction on new rooms. Thus, new rooms need to be added.

Exception:

If the hotel has too many rooms, new room won't be added.

1.14 RemoveRoom

User: Manager

Input: Room Number

Output: "Room number 999 was removed from the hotel" or "Room number 999 could not be deleted, it is occupied" or "Room does not exist"

Basic Case:

There is a leak in the bathroom in room 999, and the easiest way to solve the problem is to remove the room completely.

Exception:

If the room is occupied, do not allow removal
If the selected room number does not exist, abort the operation

1.15 AddRoomType

User: Manager

Input: Room Type name, rate, member rate

Output: "New room type name has been added to the hotel" or "New Room type could not be added"

Basic Case:

Add a new room type.

1.16 ModifyRoomTypeName

User: Manager

Input: Old Room Type name, New Room Type Name

Output: “room type name has been modified” or “Room type name could not be modified”

Basic Case:

Modify a new room type name.

1.17 NumOfOccupants

User: Manager

Input: fromDate, toDate

Output: daily number of occupants between fromDate and toDate or “Error!”

Basic Case:

To help the manager understand the distribution of number of occupants.

Exceptions:

Stop if fromDate > toDate

2. Table schema

Customer(firstName: VARCHAR(20), lastName: VARCHAR(20), phoneNum: NUMBER(10), address: VARCHAR(50))
Primary key: firstName, lastName, phoneNum.
FD: (firstName, lastName, phoneNum) -> address

Member(firstName: VARCHAR(20), lastName: VARCHAR(20), phoneNum: NUMBER(10), points: INTEGER)
Primary key: firstName, lastName, phoneNum.
Foreigen key: (firstName, lastName, phoneNum) references **Customer**
FD: (firstName, lastName, phoneNum) -> points

Reservation(resId: NUMBER(10), fromDate: DATE, toDate: DATE, firstName: VARCHAR(20), lastName: VARCHAR(20), phoneNum: NUMBER(10), rtName: VARCHAR(20))
Primary key: resId
Foreign key:
(firstName, lastName, phoneNum) references **Customer**
(rtName) references **RoomType**
Constraint:
(firstName, lastName, phoneNum) NOT NULL
rtName NOT NULL
FD: resId -> (fromDate, toDate, firstName, lastName, phoneNum, rtName)

Checkin(checkinId: NUMBER(10), deposit: NUMBER(6), resId: NUMBER(10), checkinDate: DATE, firstName: VARCHAR(20), lastName: VARCHAR(20), phoneNum: NUMBER(10), roomNum: NUMBER(4), lotNum: NUMBER(4))
Primary key: checkinId
Foreign key:
(resId) references **Reservation**
(firstName, lastName, phoneNum) references **Customer**
(roomNum) references **Room**
(lotNum) references **ParkingLot**
Constraint:
(firstName, lastName, phoneNum) NOT NULL
roomNum NOT NULL
lotNum NOT NULL
deposit >= 0 NOT NULL
FD: checkinId -> (deposit, resId, checkinDate, firstName, lastName, phoneNum, roomNum, lotNum)

Checkout(checkoutId: NUMBER(10), checkoutDate: DATE, cardNum: NUMBER(16), expDate: NUMBER(4), roomFee: NUMBER(6), fine: NUMBER(6), checkinId NUMBER(10))
Primary key: checkoutId
Foreign key: (checkinId) references **Checkin**
Constrant:
checkinId NOT NULL
roomFee is calculated based on checkinDate, checkoutDate, rate and memberRate.
FD: checkoutId -> (checkoutDate, cardNum, expDate, roomFee, fine, checkinId)

RoomType(rtName: VARCHAR(20), rate: NUMBER(20), memberRate: NUMBER(10), numOfRooms: INTEGER(50))
Primary key: rtName
FD: rtName -> (rate, memberRate, numOfRooms)

RoomOf(roomNum:NUMBER(4), occupied:INTEGER, rtName: VARCHAR(20))
Primary key: roomNum
Foreigen key: (rtName) references to **RoomType**
Constraint:
(rtName) NOT NULL
(occupied) NOT NULL , 0 means empty room, 1 means occupied room
FD: roomNum -> (occupied, rtName)

ParkingLot(lotNum: NUMBER(4), occupied:INTEGER)
Primary key: lotNum
Constraint:
(occupied) NOT NULL , 0 means empty lot, 1 means occupied lot
FD: lotNum -> occupied

3. Normalization

All the tables are in BCNF.

4. SQL DDL

```
CREATE TABLE Customer(
        firstName VARCHAR(20),
        lastName VARCHAR(20),
        phoneNum NUMBER(10),
        address VARCHAR(50),
        PRIMARY KEY (firstName, lastName, phoneNum)
        );

CREATE TABLE Member(
        firstName VARCHAR(20),
        lastName VARCHAR(20),
        phoneNum NUMBER(10),
        points: INTEGER,
        PRIMARY KEY (firstName, lastName, phoneNum),
        FOREIGN KEY (firstName, lastName, phoneNum) REFERENCES
Customer
        );


CREATE TABLE Reservation(
        resId NUMBER(10),
```

fromDate DATE NOT NULL,
toDate DATE NOT NULL,
firstName VARCHAR(20) NOT NULL,
lastName VARCHAR(20) NOT NULL,
phoneNum NUMBER(10) NOT NULL,
rtName: VARCHAR(20) NOT NULL,
PRIMARY KEY (resId),
FOREIGN KEY (firstName, lastName, phoneNum) REFERENCES Customer,
FOREIGN KEY (rtName) REFERENCES RoomType
);

CREATE TABLE Checkin(
checkinId NUMBER(10),
deposit NUMBER(6) NOT NULL,
resId NUMBER(10),
checkinDate DATE NOT NULL,
firstName VARCHAR(20) NOT NULL,
lastName VARCHAR(20) NOT NULL,
phoneNum NUMBER(10) NOT NULL,
roomNum: NUMBER(4) NOT NULL,
lotNum: NUMBER(4) NOT NULL,
PRIMARY KEY (checkinId),
FOREIGN KEY (firstName, lastName, phoneNum) REFERENCES Customer,
FOREIGN KEY (resId) REFERENCES Reservation,
FOREIGN KEY (roomNum) REFERENCES Room,
FOREIGN KEY (lotNum) REFERENCES ParkingLot
);

CREATE TABLE Checkout(
checkoutId NUMBER(10) PRIMARY KEY,
checkoutDate DATE NOT NULL,
cardNum NUMBER(16),
expDate NUMBER(4),
roomFee NUMBER(6),
fine NUMBER(6),
checkinId NUMBER(10) NOT NULL,
Foreign key: (checkinId) references **Checkin**
);

CREATE TABLE RoomType(
rtName VARCHAR(20),
rate NUMBER(20),

```
    memberRate NUMBER(10),
    numOfRooms INTEGER(50)
    );

CREATE TABLE ParkingLot(
    lotNum NUMBER(4),
    occupied INTEGER,
    );


CREATE TABLE RoomOf(
    roomNum NUMBER(4),
    occupied INTEGER NOT NULL,
    rtName: VARCHAR(20) NOT NULL,
    FOREIGN KEY(rtName) REFERENCES RoomType
    );
```

5. Sample instances

Customer

firstName	lastName	phoneNum	address
John	Li	6042049382	1234 E 1st, Vancouver, BC, Canada
Jim	Wang	6049284938	345 Main St, New York, NY, USA
Jerry	Snow	2837483728	2211 Sunset Blvd, Los Angeles, CA, USA
Jack	Will	5837284723	2321 Market St, San Francisco, CA, USA
Jenny	Henry	9847382772	NULL

Member

firstName	lastName	phoneNum	points
John	Li	6042049382	1022
Jim	Wang	6049284938	0
Jerry	Snow	2837483728	210
Jon	Wolf	2948237483	10298
James	Pat	8374827212	684

Reservation

resId	fromDate	toDate	firstName	lastName	phoneNum	rtName
2015052301	01-06-2015	06-06-2015	Jon	Wolf	2948237483	single room
2015052302	02-06-2015	03-06-2015	Jim	Wang	6049284938	single room
2015052303	02-06-2015	03-06-2015	Jerry	Snow	2837483728	superior single room
2015052601	02-06-2015	03-06-2015	Jerry	Snow	2837483728	double room
2015052602	10-06-2015	25-06-2015	Jenny	Henry	9847382772	superior single room

Checkin

checkinId	deposit	resId	checkinDate	firstName	lastName	phoneNum	roomNum	lotNum
2015060101	100	2015052301	01-06-2015	Jon	Wolf	2948237483	0001	0001
2015060201	100	2015052302	02-06-2015	Jim	Wang	6049284938	0010	0002
2015060202	150	2015052303	02-06-2015	Jerry	Snow	2837483728	0032	0003
20150	200	20150	02-06-	Jerry	Snow	28374	0029	0004

60203		52601	2015			83728		
2015061001	100	2015052602	10-06-2015	Jenny	Henry	9847382772	0046	0005

ParkingLot

lotNum	occupied
0001	0
0002	1
0003	0
0004	0
0005	1

RoomType

rtName	rate	memberRate	numOfRooms
single room	150	120	28
double room	190	160	31
superior single room	200	170	17
superior double room	250	220	9
family suit	260	230	7

Room

roomNum	occupied	rtName
0001	1	single room
0002	1	superior double room
0003	0	double room
0004	1	single room
0005	0	family suit

checkoutId	checkoutDate	cardNum	expDate	roomFee	fine	checkinId
2015052901	20150601	1111	20170601	850	0	2015060101
2015052902	20150601	2222	20170601	750	0	2015060201
2015052903	20150601	3333	20170601	650	0	2015060202
2015052904	20150601	4444	20170601	550	0	2015060203
2015052905	20150601	5555	20170601	150	0	2015061001

checkinId	deposit	resId	checkinDate	firstName	lastName	phoneNum	roomNum	lotNum
2015060101	100	2015052301	01-06-2015	Jon	Wolf	2948237483	0001	0001
2015060201	100	2015052302	02-06-2015	Jim	Wang	6049284938	0010	0002
2015060202	150	2015052303	02-06-2015	Jerry	Snow	2837483728	0032	0003
2015060203	200	2015052601	02-06-2015	Jerry	Snow	2837483728	0029	0004
2015061001	100	2015052602	10-06-2015	Jenny	Henry	9847382772	0046	0005

6. Platform to use

The CS Ugrad Oracle installation and JDBC.

7. Data in the final application

Fake data.

8. Division of labour

GUI: Erik, Larry
Backend: Bo, Guang