Database Project Report

Hotel Management System

(Phase 2)

Submitted by

Muskan Subnani (V487T392)

Shawn Ribaudo (J547A966)

Sushmitha Neerubai (C534C335)

Submitted to

Dr. Keenan Jackson

Department of Electrical Engineering and Computer Science

Spring Semester 2019

CONTENTS :

---

1. Introduction

1.1 Overview of the Project

The Hotel Management System, in all its uniqueness, is a comprehensive solution for efficient, quick and elegant management of a hotel and hospitality enterprise which includes a wide array of functional activities ranging from booking hotel rooms, booking hotel services in wide varieties hotel locations and choices. It's a robust system that accounts for taxes and seasonal changes of rates, therefore making it realistic and easy to implement in the real world.

2. Statement of purpose

- To handle the data of all hotels affiliated to a particular enterprise and means to setup records of an given hotel
- To manage and monitor all the customer transactions associated with every hotel, and to be able to retrieve the details of the desired transactions.
- To keep a constant track on the rooms as well as service available in each hotel,, their type, and their rate Necessary measures are to be taken as and when the above said status that is available hits the mark.

- To obtain useful statistical information about the hotel company (e.g.: The type of room that is popular among the customers, or which hotel location is most popular), which may provide a great impetus for the organizations to expand and improve their business.
- To provide customers with safe and easy methods of payments, that is storing such sensitive information with strong security.

- **Objective :**
  The main objective of the entire activity is to automate the process of day to day activities of Hotel like:

1. Room activities,
2. Admission of a New Customer,
3. Assign a room according to customer's demand,
4. Checkout of a computer and releasing the room
5. Finally compute the bill etc.
6. Advance online bookings.
7. Online Cancellation.
8. List of Regular customers through the email facility.
9. enables customers to update their details such as email address,password, address, date of birth etc.

- This project intends to introduce more user friendliness in the various activities such as record updation, maintenance, and searching.

- The searching of record has been made quite simple as all the details of the customer can be obtained by simply keying in the identification of that customer.

- Similarly, record maintenance and updation can also be accomplished by using the identification of the customer with all the details being automatically generated. These details are also being promptly automatically updated in the master file thus keeping the record absolutely up-to-date.

3. Description of the Entities

For this Hotel Management System. We have 10 entities and each entity hold different information. Below is the short description of each entity.

Hotel - The locations of the hotel and what type of the hotel.

Tax - Each hotel has different tax rate and type base on the location.

Service - Any kind of service that being charged to the customers including spa, room service, bar, ect.

Room - Information whether the room with the rate desired by the customer is available for the dates or not in that particular hotel.

Room Rate - Various of rate based on the schedule throughout the years and cost based on the room size.

Booking - Customers booking dates and number of guest in a room.

Customer - information about the customer's name, their contact information and where they live.

Billing - Create billings based on the service charges and room charge with tax. Also track billing to see if it is pay or not.

Payment - Payment information like the card details.

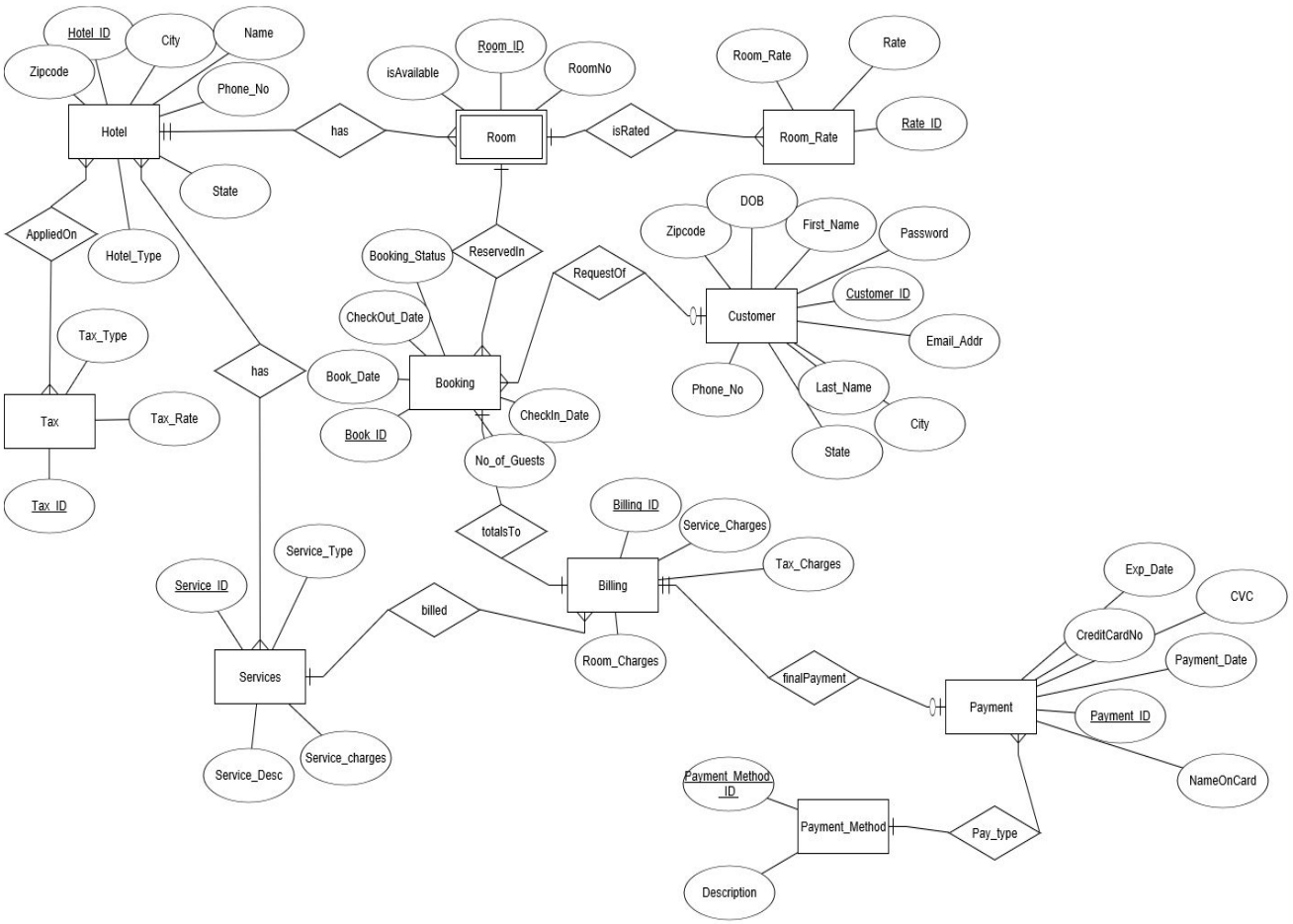
Payment Method - type of payment.

4. Description of the Relationships

For this Hotel Management System. We have 9 relationships and each relationship hold different information. Read below to see the short description of each relationships.

- **Tax** *applied on* **hotel**.
- **Hotel** *has* **rooms**.
- **Room** *is rated* **room rate**.
- **Hotel** *has* **services**.
- **Booking** reserve in **room**.
- **Service** *billed* **billing**.
- **Billing** *final payment* **payment.**
- **Booking** *totals to* **billing.**
- **Customer** *request of* **booking.**
- **Payment** *pay type* **payment method.**

5. E/R Diagram


Hotel_ID
City
Name
Zipcode
Phone_No
Hotel
has
Room_ID
isAvailable
RoomNo
Room
isRated
Room_Rate
Rate
Rate_ID
State
AppliedOn
Hotel_Type
Booking_Status
ReservedIn
RequestOf
DOB
Zipcode
First_Name
Password
Customer_ID
Email_Addr
Customer
Phone_No
Last_Name
City
State
CheckOut_Date
Tax_Type
has
Book_Date
Booking
CheckIn_Date
Book_ID
No_of_Guests
Tax
Tax_Rate
Tax_ID
totalsTo
Billing_ID
Service_Charges
Tax_Charges
Billing
Service_Type
Service_ID
billed
Room_Charges
finalPayment
Exp_Date
CVC
CreditCardNo
Payment_Date
Payment
Payment_ID
NameOnCard
Services
Service_Desc
Service_charges
Payment_Method_ID
Payment_Method
Pay_type
Description

6. Relational Schemas

View Below to see the Schemas of entries and relationships.

Entities and Relationship Schemas

- **Hotel** (Hotel_ID, City, Phone_No, State, Hotel_type, Zip Code, Name)
- **Hotel Service** (HotelService_ID, *HotelId*, *ServiceID*)
- **HotelTax** (HotelTaxID, *Tax_ID*, *HotelID*)
- **Tax** (TaxID, Tax_Rate, Tax_Type)
- **Service** (Service_ID, Service_Type, Service_Desc, Service_Charges)
- **Room_Rate** (Rate_ID, Room_Rate,Type)
- **Room**(RoomID, is Available, *Hotel_ID, Rate_ID,* RoomNo)
- **Booking** (Book_ID, Booking_Status, CheckOut_Date, Book_Date, No_of_Guest, checkIn_Date, *Room_ID, Customer_ID*)
- **Customer** (Customer_ID, Zip Code, DOB, First_Name, Last_Name, Password, Email_Addr, City, State, Phone_No)
- **Billing** (Billing_ID, Service_Charges, Tax_Charges, Room_Charges, *Book_ID*, *Services_ID*)
- **Payment** (Payment_ID, Exp_Date, CreditCardNo, CVC, Payment_Date, NameOnCard, *Billing_ID*, *Payment_Method_ID*)
- **Payment_Method**(Payment_Method_Id, Description)

7. Sample Queries that this database can handle :

The database is created in such a way that it would be able to handle complex queries as well as simple ones. Below is a sample list of few queries that can be handled by the database created.

1. **To List all the hotel cities and number of hotels in each city .**

   It explains the number of hotels in each city.
2. **Get number of Reservation for a customer.**

   It gives all the reservations made by each customer.
3. **Update no of Guests.**

   If there is any change in the number of guests that the customer enter's this query allows them to update that.
4. **Average Number of Guests.**

   It shows the average of total guests in the hotel.
5. **Highest costing service.**

   It is used to show the most expensive service provided by each of them . It basically groups the hotel and their expensive service.
6. **Pay bill.**

   It asks the customer to pay the bills.
7. **Delete a customer.**

   If the customer changes his plan and plans to cancel the reservation then this query deletes the customer out of the database.
8. **Add a service in a certain hotel.**

   It adds the new services been introduced at the hotel at any point of time.
9. **Remove a service in a certain hotel.**

   It removes a particular service which the hotel does not provide anymore.
10. **Quit the program :**

    It closes the database.

8. SQL queries for relations on database and the input data provided.

8.1 Hotel:

```
create table Hotel (
    Hotel_ID int PRIMARY KEY not null,
    City varchar(255),
    Phone_No int,
    State varchar(255) ,
    Hotel_type varchar(255),
    Zipcode int,
    Name varchar(255)
);
```

```
MariaDB [dbuser20_database]> select * from Hotel;
+----------+----------+------------+-------+------------+---------+------------+
| Hotel_ID | City     | Phone_No   | State | Hotel_type | Zipcode | Name       |
+----------+----------+------------+-------+------------+---------+------------+
|        1 | Wichita  |  316565632 | KS    | Resort     |   67215 | Sun Shine  |
|        2 | Wichita  | 2147483647 | KS    | Motel      |   67215 | Moonlight  |
|        3 | New York | 2147483647 | NY    | Hotel      |   14792 | Paradise   |
+----------+----------+------------+-------+------------+---------+------------+
3 rows in set (0.00 sec)
```

8.2. Room:

```
create table Room(
        Room_ID int PRIMARY KEY not null AUTO_INCREMENT,
        isAvailable boolean,
        Hotel_ID int references Hotel_ID( Hotel_ID)
        on delete set null,
        Rate_ID int references Room_Rate(Rate_ID)
        on delete set null,
        RoomNO int
);
```

```
MariaDB [dbuser20_database]> select * from Room;
+---------+-------------+----------+--------+--------+
| Room_ID | isAvailable | Hotel_ID | RateID | RoomNo |
+---------+-------------+----------+--------+--------+
|       1 |           1 |        1 |      1 |    100 |
|       2 |           1 |        1 |      2 |    200 |
|       3 |           1 |        2 |      3 |    100 |
|       4 |           0 |        2 |      4 |    200 |
|       5 |           1 |        3 |      5 |    100 |
|       6 |           0 |        3 |      6 |    200 |
+---------+-------------+----------+--------+--------+
6 rows in set (0.00 sec)
```

8.3.Room Rate:

Create table Room_Rate
	Rate_ID int PRIMARY KEY not null
	Room_Rate float
	Type varchar(255)
);

```
MariaDB [dbuser20_database]> select * from Room_Rate;
+---------+-----------+--------+
| Rate_ID | Room_Rate | Type   |
+---------+-----------+--------+
|       1 |        79 | Single |
|       2 |       140 | Double |
|       3 |        40 | Single |
|       4 |        70 | Double |
|       5 |       110 | Single |
|       6 |       190 | Double |
+---------+-----------+--------+
6 rows in set (0.00 sec)
```

8.4Tax:

```
create table Tax (
        TaxID int PRIMARY KEY NOT NULL AUTO_INCREMENT,
        Tax_Rate float,
        Tax_Type varchar(255)
);
```

```
MariaDB [dbuser20_database]> select * from Tax;
+-------+----------+----------+
| TaxID | Tax_Rate | Tax_Type |
+-------+----------+----------+
|     1 |      5.5 | CityTax  |
|     2 |      7.2 | CityTax  |
+-------+----------+----------+
2 rows in set (0.01 sec)
```

8.5 Hotel tax:

```
Create table HotelTax (
        HotelTaxID int PRIMARY KEY Not Null AUTO_INCREMENT,
        Tax_ID int,
        HotelID int,
        foreign key ( Tax_ID) references Tax (TaxID)
        on delete set null,
        foreign key( HotelID) references Hotel (Hotel_ID)
        on delete set null
);
```

```
MariaDB [dbuser20_database]> select * from HotelTax;
+------------+--------+---------+
| HotelTaxID | Tax_ID | HotelID |
+------------+--------+---------+
|          1 |      1 |       1 |
|          2 |      1 |       2 |
|          3 |      2 |       3 |
+------------+--------+---------+
3 rows in set (0.00 sec)
```

8.6 Customer:

```
create table Customer(
        Customer_ID int PRIMARY KEY not null AUTO_INCREMENT,
        Zipcode int,
        DOB date,
        First_Name varchar(255),
        Last_Name varchar(255),
        Password varchar(255),
        Email_Addr varchar(255),
        City varchar(255),
        State varchar(255),
        Phone_No varchar(255)
);
```

```
MariaDB [dbuser20_database]> select * from Customer;
+-------------+---------+------------+------------+-----------+-----------+----------------+----------+-------+------------+
| Customer_ID | Zipcode | DOB        | First_Name | Last_Name | Password  | Email_Addr     | City     | State | Phone_No   |
+-------------+---------+------------+------------+-----------+-----------+----------------+----------+-------+------------+
|           1 |   67215 | 2000-06-15 | Cowboy     | Moon      | Flytomoon | moon@gmail.com | Wichita  | KS    | 3165296365 |
|           2 |   69705 | 0000-00-00 | Rabbit     | Killer    | Eatcarrot | Egg@gmail.com  | Houstan  | KS    | 3615269652 |
|           3 |   69955 | 1963-11-03 | Wolf       | Human     | Wavewolf  | Fang@gmail.com | New York | NY    | 3615125452 |
+-------------+---------+------------+------------+-----------+-----------+----------------+----------+-------+------------+
3 rows in set (0.00 sec)
```

8.7 Booking:

```
create table Booking(
        Book_ID int PRIMARY KEY not null AUTO_INCREMENT,
        Booking_Status boolean,
        CheckOut_Date date,
        Book_Date date,
        No_of_Guest int,
        checkIn_Date date,
        RoomID  int,
        CustomerID int,
        foreign key (RoomID) references Room(Room_ID)
        on delete set null
        on update cascade,
        foreign key (CustomerID) references Customer (Customer_ID)
        on delete set null
        on update cascade
);
```

```
MariaDB [dbuser20_database]> select * from Booking;
+---------+----------------+--------------+------------+-------------+-------------+--------+------------+---------+
| Book_ID | Booking_Status | CheckOut_Date | Book_Date  | No_of_Guest | checkIn_Date | RoomID | CustomerID | HotelID |
+---------+----------------+--------------+------------+-------------+-------------+--------+------------+---------+
|       1 |              1 | 2019-07-01    | 2019-04-01 |           3 | 2019-04-01   |      2 |          1 |       1 |
|       2 |              0 | 0000-00-00    | 2019-02-01 |           1 | 0000-00-00   |      3 |          2 |       2 |
|       3 |              1 | 0000-00-00    | 0000-00-00 |           1 | 2019-05-03   |      3 |          2 |       2 |
|       4 |              1 | 0000-00-00    | 2019-05-02 |           2 | 2019-06-03   |      6 |          3 |       3 |
+---------+----------------+--------------+------------+-------------+-------------+--------+------------+---------+
4 rows in set (0.00 sec)
```

8.8 Billing:

```
create table Billing(
    Billing_ID int PRIMARY KEY not null,
    Service_Charges float,
    Tax_Charges float,
    Room_Charges Float,
    BookID int,
    ServiceID int,
foreign key (BookID) references Booking (Book_ID),
foreign key (ServiceID) references Service(Service_ID)
);
```

```
MariaDB [dbuser20_database]> select * from Billing;
+------------+-----------------+-------------+--------------+--------+-----------+
| Billing_ID | Service_Charges | Tax_Charges | Room_Charges | BookID | ServiceID |
+------------+-----------------+-------------+--------------+--------+-----------+
|          1 |            7.19 |        7.59 |         NULL |      1 |         1 |
|          2 |              20 |        21.1 |         NULL |      1 |         2 |
|          3 |              20 |        21.1 |         NULL |      1 |         2 |
|          4 |            NULL |       443.1 |          420 |      1 |      NULL |
|          5 |            NULL |       126.6 |          120 |      3 |      NULL |
|          6 |            NULL |      407.36 |          380 |      4 |      NULL |
+------------+-----------------+-------------+--------------+--------+-----------+
6 rows in set (0.00 sec)
```

8.9 Payment:

```
Create table Payment(
        Payment_ID int PRIMARY KEY not null AUTO_INCREMENT,
        Exp_Date varchar(255),
        CreditCardNo int,
        CVC int,
        Payment_Date date,
        NameOnCard Varchar(255),
        BillingID int,
        Payment_MethodID int,
        foreign key (BillingID) references Billing(Billing_ID)
        on delete set null,
        foreign key (Payment_MethodID) references
        Payment_Method(Payment_Method_ID)
        on delete set null
);
```

```
MariaDB [dbuser20_database]> select * from Payment;
+------------+------------+-------------+------+--------------+-------------+-----------+------------------+
| Payment_ID | Exp_Date   | CreditCardNo | CVC  | Payment_Date | NameOnCard  | BillingID | Payment_MethodID |
+------------+------------+-------------+------+--------------+-------------+-----------+------------------+
|          1 | NULL       |         NULL | NULL | 2019-05-01   | NULL        |         1 |                1 |
|          2 | 2022-02-01 |   2147483647 |  987 | 2019-06-01   | Cowboy Moon |         2 |                2 |
|          3 | 2022-02-01 |   2147483647 |  987 | 2019-07-01   | Cowboy Moon |         3 |                2 |
|          4 | 2022-02-01 |   2147483647 |  987 | 2019-07-01   | Cowboy Moon |         4 |                2 |
+------------+------------+-------------+------+--------------+-------------+-----------+------------------+
4 rows in set (0.01 sec)
```

8.10 Payment method:

```
Create table Payment_Method(
        Payment_Method_Id int Primary key not null AUTO_INCREMENT,
        Description varchar(255)
);
```

```
MariaDB [dbuser20_database]> select * from Payment_Method;
+-------------------+-------------+
| Payment_Method_Id | Description |
+-------------------+-------------+
|                 1 | Cash        |
|                 2 | Credit Card |
|                 3 | Debit Card  |
+-------------------+-------------+
3 rows in set (0.01 sec)
```

8.11 Service:

```
create table Service(
        Service_ID int PRIMARY KEY not null AUTO_INCREMENT,
        Service_Type varchar(255),
        Service_Desc varchar(255),
        Service_Charges float,
);
```

```
MariaDB [dbuser20_database]> select * from Service;
+------------+--------------+-----------------------------+-----------------+
| Service_ID | Service_Type | Service_Desc                | Service_Charges |
+------------+--------------+-----------------------------+-----------------+
|          1 | Spa          | Amazing Spa                 |            7.19 |
|          2 | Food_Door    | Deliver meal to your room.  |              20 |
|          3 | Spa          | Enjoy your time             |            9.99 |
|          4 | Weight_Room  | Workout                     |               0 |
+------------+--------------+-----------------------------+-----------------+
4 rows in set (0.00 sec)
```

8.12 Hotel service:

```
create table HotelService(
        HotelService_ID int PRIMARY KEY not null AUTO_INCREMENT,
        HotelID int,
        ServiceID int,
        foreign key (HotelID) references Hotel(Hotel_ID)
        on delete set null,
        foreign key (ServiceID) references Service(Service_ID)
        on delete set null)
);
```

```
MariaDB [dbuser20_database]> select * from HotelService;
+-----------------+---------+-----------+
| HotelService_ID | HotelID | ServiceID |
+-----------------+---------+-----------+
|               1 |       1 |         1 |
|               2 |       1 |         2 |
|               3 |       2 |         1 |
|               4 |       2 |         2 |
|               5 |       3 |         3 |
|               6 |       3 |         4 |
+-----------------+---------+-----------+
6 rows in set (0.00 sec)
```

9. Program in JAVA :

```
/ File: Menu.java
//
// This program illustrates the use of a menu, which would be the basis
// for constructing a larger program by adding more options, where each
// option is handled by a separate function.
//
import java.sql.*;
import java.util.Scanner;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;
import java.util.Date;
import java.time.LocalDate;


public class Menu
{
public static void main(String[] args)
{
int choice;
Connection conn = null;
try
{
// Step 1: connect to the database server using a connection string.
String host = "cslab-db.cs.wichita.edu";
int port = 3306;
String database = "dbuser20_database";
String user = "dbuser20";
String password = "tOnGHerkIJOv";
String url = String.format("jdbc:mariadb://%s:%s/%s?user=%s&password=%s",
host, port, database, user, password);
conn = DriverManager.getConnection(url);

// Step 2: Display the menu and get the user response.
choice = PrintMenuAndGetResponse( );
```

```
// Step 3: Respond to the menu choice.
switch (choice)
{
case 1: // A choice of 1 is to print all student names, IDs, and
// their majors.
ListAllHotels(conn);
Break;

case 2: // A choice of 1 is to print all student names, IDs, and
// their majors.
GetCustomerReservation(conn);
Break;

case 3: // A choice of 1 is to print all student names, IDs, and
// their majors.
UpdateCheckOutDate(conn);
Break;

case 4: // To quit the program.
GettingAverageOfGuest(conn);
Break;

case 5: // To quit the program.
HighestServiceCost(conn);
Break;

case 6: // To quit the program.
PayBill(conn);
Break;

case 7: // To delete a customer.
PrintAllCustomer(conn);
DeleteCustomer(conn);
PrintAllCustomer(conn);
Break;

case 8: //add Service
PrintAllHotel(conn);
InsertNewService(conn);
PrintAllService(conn);
```

```
Break;

case 9: //delete Service
PrintAllHotel(conn);
DeleteService(conn);
PrintAllService(conn);
Break;

case 10: // To quit the program.
System.out.println("Exiting Program");
break;
default: // Illegal choice for integers other than 1, 2 and 3.
System.out.println("Illegal choice");
Break;

}
}
catch(SQLException e)
{
e.printStackTrace();
}
finally
{

// Step 4: Disconnect from the database server.
try
{
if (conn != null)
conn.close();
}
catch(SQLException e)
{
e.printStackTrace();
}
}
}
// This function controls the user interaction with the menu.
public static int PrintMenuAndGetResponse( )
{
Scanner keyboard = new Scanner(System.in);
```

```
int response;
System.out.println("Choose from one of the following options:");
System.out.println(" 1. List all hotel cities and number of hotels in each city: ");//good
System.out.println(" 2.Get number of Reservation for a customer: ");//good
System.out.println(" 3. Update no of Guests: ");
System.out.println(" 4. Average Number of Guests: ");
System.out.println(" 5. Highest costing service: ");
System.out.println(" 6. Pay bill: ");
System.out.println(" 7. Delete a customer: ");
System.out.println(" 8. Add a service in a certain hotel: ");
System.out.println(" 9. Remove a service in a certain hotel: ");
System.out.println(" 10. Quit the program%n");
System.out.print("Your choice ==> ");
response = keyboard.nextInt();
// Leave a blank line before printing the output response.
System.out.println( );
return response;
}

// print all customer
public static void PrintAllCustomer(Connection conn) throws SQLException
{
  Statement stmt = conn.createStatement();
  String qry = "select Customer_ID, First_Name, Last_Name "
               +
               " from Customer";
  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-4s  %12s   %-12s%n", "ID", "First Name", "Last Name");

  // Then -- print the body of the output table.
  while (rs.next())
  {
    int sId = rs.getInt("Customer_ID");
    String    dFirst   = rs.getString("First_Name");
    String kLast = rs.getString("Last_Name");
```

```
    System.out.format("%-4s %12s  %-12s%n", sId, dFirst, kLast);
  }
  System.out.println();
  rs.close();
}
//delete one customer
public static void DeleteCustomer(Connection conn)throws SQLException
  {
    System.out.println("Choose a customerID you want to delete: ");

        Scanner keyboard = new Scanner(System.in);

    int choosenID = keyboard.nextInt();
      Statement stmt = conn.createStatement();
        String qry = "select * from Customer where  Customer_ID=" +
Integer.toString(choosenID);

        ResultSet rs = stmt.executeQuery(qry);
        if (rs.next() == false)
         {
           System.out.println("ResultSet in empty for the provided Customer_ID");
         }
         else

    {
      String qry1 = "DELETE FROM Customer"
                +
                " Where Customer_ID =" + Integer.toString(choosenID);
    ResultSet rs1 = stmt.executeQuery(qry1);
    System.out.println("Successful DELETE a customer");
    rs1.close();
  }
    rs.close();
  }


//print hotels
public static void PrintAllHotel(Connection conn) throws SQLException
 {
   Statement stmt = conn.createStatement();
```

```
  String qry = "select Hotel_ID, Name"
               +
               " from Hotel";
  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-4s  %12s%n", "Hotel_ID", "Name");

  // Then -- print the body of the output table.
  while (rs.next())
  {
    int sId = rs.getInt("Hotel_ID");
    String    sName   = rs.getString("Name");

    System.out.format("%-4s  %12s%n", sId, sName);
  }
  System.out.println();
  rs.close();
}

//print service
public static void PrintAllService(Connection conn) throws SQLException
{
  Statement stmt = conn.createStatement();
  String qry = "select *"
               +
               " from Service";
  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-12s  %12s   %-12s   %12s%n", "Service_ID", "Service_Type",
"Service_Desc","Service_Charges");

  // Then -- print the body of the output table.
  while (rs.next())
  {
```

```
    int sId = rs.getInt("Service_ID");
    String    sType   = rs.getString("Service_Type");
  String sDesc = rs.getString("Service_Desc");
    Float sCharge = rs.getFloat("Service_Charges");

    System.out.format("%-12s  %12s   %-12s   %12s%n", sId, sType, sDesc, sCharge);
  }
  System.out.println();
  rs.close();
 }

//adding service
 public static void InsertNewService(Connection conn)throws SQLException
 {
   Statement stmt = conn.createStatement();

   Scanner keyboard = new Scanner(System.in);
   String ServiceType,ServiceDesc;
   System.out.println("Which hotel will this new service apply? (enter one number)");
   String hotelIDNumber = keyboard.nextLine();

   System.out.println("Enter the service type: ");
   ServiceType = keyboard.nextLine();

   System.out.print("Enter the service description: ");

   ServiceDesc = keyboard.nextLine();
  ;
   System.out.println("Enter the service charge: ");
   Float ServiceCharges = keyboard.nextFloat();


   //Service insert
   String qry = "INSERT INTO Service(Service_Type, Service_Desc, Service_Charges)"
              +
              " VALUE('"+ ServiceType +"','"+ ServiceDesc +"','"+ ServiceCharges +")";

   ResultSet rs = stmt.executeQuery(qry);
   System.out.println( );
   rs.close();
```

```
  //HotelService insert
  //String qryHotelService = "INSERT INTO HotelService(HotelID, ServiceID)"
          //    +
            //  " VALUE("+ hotelIDNumber +","+ +")";
//  ResultSet rsHotelService = stmt.executeQuery(qryHotelService);
  //rsHotelService.close();

}

//delete one service
public static void DeleteService(Connection conn)throws SQLException
  {
    System.out.println("Choose a ServiceID you want to delete: ");
 Statement stmt = conn.createStatement();
       Scanner keyboard = new Scanner(System.in);

    int choosenServiceID = keyboard.nextInt();
    System.out.println("Which hotelID do you want this service to remove from? ");
    int choosenHotelID = keyboard.nextInt();

    //HotelService delete
    String qryHotelService = "DELETE FROM HotelService"
                +
                " Where ServiceID =" + Integer.toString(choosenServiceID)
                +
                " AND HotelID =" + Integer.toString(choosenHotelID);
    ResultSet rsHotelService = stmt.executeQuery(qryHotelService);
    rsHotelService.close();

    //Service delete

    String qry = "DELETE FROM Service"
                +
                " Where Service_ID =" + Integer.toString(choosenServiceID);
    ResultSet rs = stmt.executeQuery(qry);
    System.out.println("Successful DELETE a customer");
    rs.close();
```

```
   }

public static void ListAllHotels(Connection conn) throws SQLException
{
Statement stmt = conn.createStatement();
String qry = "select City , count( Hotel_ID) as NumofHotels"
+
" from Hotel "
+
"group by City ";
ResultSet rs = stmt.executeQuery(qry);
// Loop through the result set and print the output.
// First -- print the output column headings.
System.out.format("%n");
System.out.format("%-12s %-20s%n", "City", "NumofHotels");

// Then -- print the body of the output table.
while (rs.next())
{
String Cityname = rs.getString("City");
int NumberofHotels = rs.getInt("NumofHotels");
System.out.format("%-12s %-20s%n", Cityname, NumberofHotels);

}
System.out.println();
rs.close();
}


public static void GetCustomerReservation(Connection conn)throws SQLException
{
  Statement stmt = conn.createStatement();

  //Scanner keyboard = new Scanner(System.in);
  //String firstName;
  //String lastName;

  //System.out.println("Enter First Name: ");
  //firstName= keyboard.nextLine();
  //System.out.println("Enter Last Name: ");
```

```
  //lastName= keyboard.nextLine();

  String qry = "Select Customer_ID, First_Name, Last_Name, count(Book_ID) as BookIDS "
               +
               "from Hotel,Customer, Booking, Room"
               +
               " where Customer_ID=CustomerID"
               +
               " AND Room_ID=RoomID"
               +
               " AND Room.Hotel_ID=Hotel.Hotel_ID"
               +
                " Group by Customer_ID ;";

  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-12s  %4s  %4s %-20s%n", "CustomerID",
"First_Name","Last_Name", "NoOfBookings");

  // Then -- print the body of the output table.
  while (rs.next())
  {
    int id = rs.getInt("Customer_ID");
    String First_Name = rs.getString("First_Name");
    String Last_Name = rs.getString("Last_Name");
    int no = rs.getInt("BookIDS");
    System.out.format("%-12s  %12s  %8s %-20s%n",id, First_Name, Last_Name, no);
  }
  System.out.println( );
  rs.close();
}

public static void UpdateCheckOutDate(Connection conn) throws SQLException
{
Statement stmt = conn.createStatement();
Scanner keyboard = new Scanner(System.in);
int BookID;
```

```
System.out.println("Enter the bookingID: ");
BookID= keyboard.nextInt();
String qry = "select No_of_Guest"
+
" from Booking"
+
" where Book_ID="
+ BookID ;

// Loop through the result set and print the output.
// First -- print the output column headings.
System.out.println("original NoofGuests:");
ResultSet rs = stmt.executeQuery(qry);
if (rs.next())
{
int oldNoofGuests = rs.getInt("No_of_Guest");
System.out.format("%3d%n", oldNoofGuests);
}

int newNoofGuests;
System.out.println("Enter the NumberofGuests: ");
newNoofGuests= keyboard.nextInt();
String cmd = "update Booking set No_of_Guest=" + newNoofGuests + " where Book_ID="
+BookID ;
stmt.executeUpdate(cmd);
// Step 4: Show the changed MajorId of student number 1.
System.out.println("Here is the changed no of guests :");
rs = stmt.executeQuery(qry);
if (rs.next())
{
int noofguests = rs.getInt("No_of_Guest");
System.out.format("%3d%n", noofguests);
}
rs.close();

}

public static void GettingAverageOfGuest(Connection conn)throws SQLException
  {
    Statement stmt = conn.createStatement();
```

```
  System.out.println("The average number of guest per hotel are: ");


  String qry = "select Name, Avg(No_of_Guest) as Avg_No_of_Guest "
             +
             " from Booking, Hotel"
             +
             " Where HotelID = Hotel_ID"
             +
             " Group By HotelID";

  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-12s %-12s %n", "Hotel Name", "Average of Guest");

  // Then -- print the body of the output table.
  while (rs.next())
  {
    String sName = rs.getString("Name");
    int dAverage = rs.getInt("Avg_No_of_Guest");

    System.out.format("%-12s %-12s%n", sName, dAverage);
  }
  System.out.println( );
  rs.close();
}

public static void HighestServiceCost(Connection conn)throws SQLException
  {
    Statement stmt = conn.createStatement();


    System.out.println("The most expensive service is: ");
```

```
  String qry = "select Name, Max(Service_Charges) as Service_Charges"
               +
               " from Service, Hotel, HotelService"
               +
               " where Service_ID = ServiceID "
               +
               " and Hotel_ID = HotelID"

               +" Group by Name";

  ResultSet rs = stmt.executeQuery(qry);

  // Loop through the result set and print the output.
  // First -- print the output column headings.
  System.out.format("%n");
  System.out.format("%-12s %-6s%n", "HotelName", "Service_Charges");

  // Then -- print the body of the output table.
  while (rs.next())
  {
    String sName = rs.getString("Name");

    String fCharge = rs.getString("Service_Charges");

    System.out.format("%-12s %-6s%n", sName, fCharge);
  }
  System.out.println( );
  rs.close();
}

public static void PayBill(Connection conn) throws SQLException
{
Statement stmt = conn.createStatement();

Scanner keyboard = new Scanner(System.in);
int BillID;
System.out.println("Enter the BillId: ");
String BillingId= keyboard.nextLine();
BillID= Integer.parseInt(BillingId);
```

```
String qry = "select * from Billing" ;

ResultSet rs = stmt.executeQuery(qry);
if (rs.next() == false)
 {
   System.out.println("ResultSet in empty for the provided billing ID");
 }
else
  {
     System.out.print("Please Enter Payment type  1. CreditCard 2. DebitCard 3. Cash");

     String optionline= keyboard.nextLine();

    int option= Integer.parseInt(optionline);

     if( option==3)
     {
       //String qry2= "select max(Payment_ID) as maxID from Payment" ;
       //ResultSet rs1 = stmt.executeQuery(qry2);
       //int id = rs1.getInt("maxID");


        String cmd = "Insert into Payment (Exp_Date, CreditCardNo, CVC, Payment_Date,
NameOnCard, BillingID, Payment_MethodID) Values (NULL,NULL,
NULL,'2019-05-17',NULL,"+ BillID + ", 1)";
        stmt.executeUpdate(cmd);

       System.out.println("Thank you for your payment ");
       String qry1 = "select Payment_ID, Exp_Date,CreditCardNo,CVC, Payment_Date,
NameOnCard from Payment";

       ResultSet rs2 = stmt.executeQuery(qry1);
       // Loop through the result set and print the output.
       // First -- print the output column headings.
       System.out.format("%n");
       System.out.format("%-12s %4s %4s %4s %4s %-4s%n",
"Payment_ID","Exp_Date","CreditCard","CVC","Payment_Date", "NameOnCard");
       // Then -- print the body of the output table.
       while (rs2.next())
```

```
          {
            int ID = rs2.getInt("Payment_ID");
            String ExpDate = rs2.getString("Exp_Date");
            int CreditCard= rs2.getInt("CreditCardNo");
            int Cvc= rs2.getInt("CVC");
            Date Payment_Date= rs2.getDate("Payment_Date");
            String NameonCard= rs2.getString("NameOnCard");
            System.out.format("%-12s %4s %4s %4s %4s %-4s%n", ID, ExpDate, CreditCard,
Cvc, Payment_Date, NameonCard);
          }
          System.out.println();

          rs2.close();
         }
         else if (option==2 ||option==1)
         {
           System.out.println("Enter Name on CardName");
           String NameOnCardUser;
           NameOnCardUser= keyboard.nextLine();

           System.out.println("Enter CardNo");
           String Card;
           Card= keyboard.nextLine();

           int CardNo= Integer.parseInt(Card);
           System.out.println("Enter ExpDate");
           String ExpDatefromUser;
           ExpDatefromUser= keyboard.nextLine();
           System.out.println("Enter CVC");
           int CvcfromUser;
           CvcfromUser= keyboard.nextInt();
           String cmd = "Insert into Payment (Exp_Date, CreditCardNo, CVC, Payment_Date,
NameOnCard, BillingID, Payment_MethodID) Values (" +ExpDatefromUser+ "," + CardNo +
"," + CvcfromUser+ ", '2019-05-17',"+NameOnCardUser+ "," + BillID + ", 2)";
           stmt.executeUpdate(cmd);

          System.out.println("Thank you for your payment ");
          String qry1 = "select Payment_ID, Exp_Date,CreditCardNo,CVC, Payment_Date,
NameOnCard from Payment";
```

```
        ResultSet rs2 = stmt.executeQuery(qry1);
        //Loop through the result set and print the output.
        // First -- print the output column headings.
        System.out.format("%n");
        System.out.format("%-12s %4s %4s %4s %4s %-4s%n",
"Payment_ID","Exp_Date","CreditCard","CVC","Payment_Date", "NameOnCard");
        // Then -- print the body of the output table.
        while (rs2.next())
        {
          int ID = rs2.getInt("Payment_ID");
          Date ExpDate = rs2.getDate("Exp_Date");
          int CreditCard= rs2.getInt("CreditCardNo");
          int Cvc= rs2.getInt("CVC");
          Date Payment_Date= rs2.getDate("Payment_Date");
          String NameonCard= rs2.getString("NameOnCard");
          System.out.format("%-12s %4s %4s %4s %4s %-4s%n", ID, ExpDate, CreditCard,
Cvc, Payment_Date, NameonCard);
        }
        System.out.println();

        rs2.close();
       }


       else
       {
          System.out.println("Wrong Payment Option");

       }
      }

  rs.close();

  }
 }
```

Contributions :

Shawn Ribaudo:

- Code
- Enter data to table
- Doing E/R
- Brainstorming E/R
- Set up a group server for communication (discord)
- Come up with a FAKE record for database and connect to other tables.
- Doing SQL for Queries
- Research Hotel
- Doing report

Susmitha Neerubai:

- Enter data to table
- Doing E/R
- Brainstorming E/R
- Doing the report and make sure everything is set up.
- Doing SQL for queries
- Understand how hotel database work in general
- Screenshot all the result
- Enter the fake record for database

Muskan Subnani:

- Code
- Set up a table
- Doing E/R
- Brainstorming E/R
- Doing SQL for queries
- Look up hotel idea.
- Debugging the code