CIS016-1 – Principle of Programming

Hotel Online Customer Booking and Management System – Case Study

Date- May 5, 2023

Name - Yaman Maharjan

University Code - 2212387

Table of Contents

Introduction / Overview

In this project I was given the task of developing a hotel reservation system with the intention of offering an effective and user-friendly solution for reserving hotel rooms online. Things to notice while develop was to develop a platform that enables customers to easily look for available rooms, book reservations, and manage their reservations. I accomplished this using Java, a well-liked general-purpose programming language renowned for its durability and adaptability. I was able to develop a system that is modular, expandable, and simple to maintain thanks to Java's object-oriented programming (OOP) capabilities. For user authentication, database connectivity, and subscription management, I have employed Java's extensive collection of libraries and frameworks.

To make sure the system functions properly and reliably, I have conducted extensive testing and troubleshooting throughout the project. I can gladly state that we were able to create a hotel reservation system that satisfies the needs and expectations because of my dedication and hard work. My solution offers a seamless and simple user interface so that visitors can quickly look for available rooms, make reservations, and confidently manage their bookings. The system's effective reservation management features can aid hoteliers by streamlining processes and enhancing customer service.

Task Description

Luton Hotel provides a wide variety of options for lodging, including single, twin, and double rooms, all of which come equipped with a private bathroom, a telephone with an outside line, and a minibar. There are restaurants and bars in the hotel where visitors can have refreshments and add them to their accommodation bill. Additionally, room service is offered. The hotel receptionist can help customers order non-hotel services (like dry cleaning) and charge such orders to their rooms. The hotel would want to offer an online booking and booking check service for customers (individual or corporate).

Customers who aren't corporations and haven't registered yet must do so by entering their address, phone number, email address, and credit card information. Customers should be able to reserve a room by providing the check-in and check-out dates as well as if they need a double or single room. Additionally, customers ought to be able to view their reservations and modify or cancel them as necessary. A consumer can only do any of these functions after signing in, which necessitates registration. Additionally, your system must confirm that the desired accommodation (single, twin, or double room) is available during the intended stay.

Receptionists are in charge for matching each booking's request for a single, twin, or double room with the proper room. The receptionist should be able to view current bookings for each room to determine whether a room is available during the preferred period. It is essential to construct a database to keep records of consumer and corporate client information (name, address, email, phone number, etc.), reservations information (start and end dates, single, twin, or double room), and room details (detailing if a room is a single, twin, or double room). Each reservation must be given the precise single, twin, or double kind of room that is stated in the reservation. The reservation becomes activated after the visitor checks in. The guest checks out at the conclusion of their stay, and the reservation is complete. If they are a non-corporate customer, they must pay their entire bill at this moment, at which point it is settled. Corporate guests must check out for their reservation to be considered complete, but payment does not occur until the corporate client receives the monthly invoice. The system keeps records of bookings for six months before deleting those that are canceled or paid.

Project Plan / Schedule

Week NO	Tasks	Priority
Week 7	UI Design	LESS
Week 8-9	UML Design	MEDIUM
Week 10	Database Design	MEDIUM
Week 11-15	Coding and Testing	HIGH
Week 14-16	Report	HIGH

Tasks

Overview of Functional, Technical (Non-Functional Requirements) and Usability Requirements

Functional Requirements

Req. No	Requirement	Priority*
1	As customer must be able to register on the HBS	MUST
2	A customer must be able to login to the HBS	MUST
3	A customer must be able to logout of the HBS	COULD
4	Receptionist must be able login to the HBS	SHOULD
5	A receptionist must be able to logout of the HBS	COULD
6	A customer must be able to make booking	MUST
7	A customer must be able to view his/her bookings	MUST
8	A customer must be able to cancel a booking	MUST

9	Receptionist must be able to view all booking	SHOULD
10	Receptionist must be able to view all bookings	SHOULD
11	Receptionist must be able to cancel a booking	SHOULD
12	Receptionist must be able to confirm a booking	SHOULD

Non-functional Requirements

Req. No	Requirement	Priority*
1	The HBS should process input and return results within 10 seconds	
2	The HBS design should be sufficiently scalable and flexible to allow for further future enhancements	
3	The HBS users should not experience critical system failures. 99.99% ‘uptime’ should be achieved.	

Usability Requirements

Req. No	Requirement	Priority*
1	The HBS should incorporate a user-centric design	
2	The design should demonstrate evidence of a good understanding of interface design issues – for example, a consistent design for each form, layout of content, use of colour schemes and images, navigational methods, usability when viewed at various	

	screen resolutions and various monitor dimensions.	
3	All data entry forms should be short and easy to complete and there should be entry validation.	
4	The HBS short have clear and intuitive navigation	
5	The HBS should comply with WW3 Web Accessibility Standards (WCAG) Text easy to read and language and language style should be appropriate with absence of grammar / spelling errors	
	There should be a clear layout which remains consistent throughout the application. Style, layout, and content should be appropriate for the purpose of the application.	

*MOSCOW Notation:

M = MUST
S = SHOULD
C = COULD W = WON’T

Design:

UML Diagrams

- ### Use Case Diagram

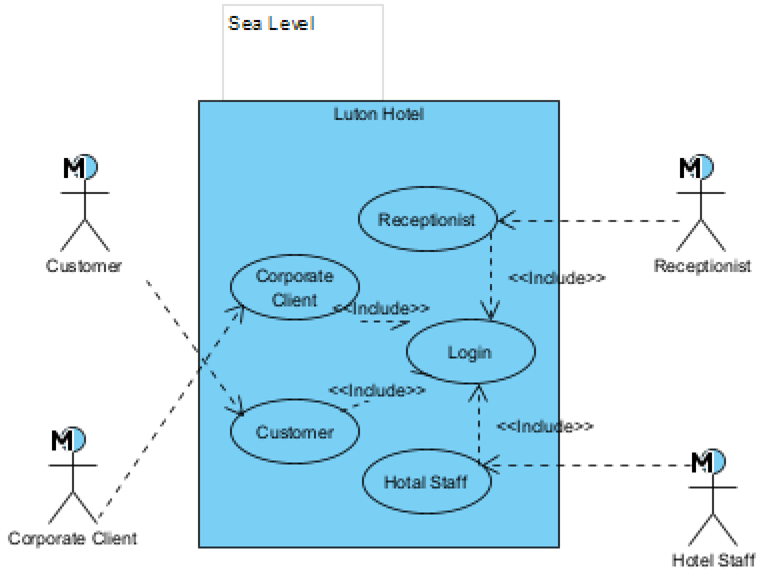


Figure 1: - Use Case Diagram (Sea Level)

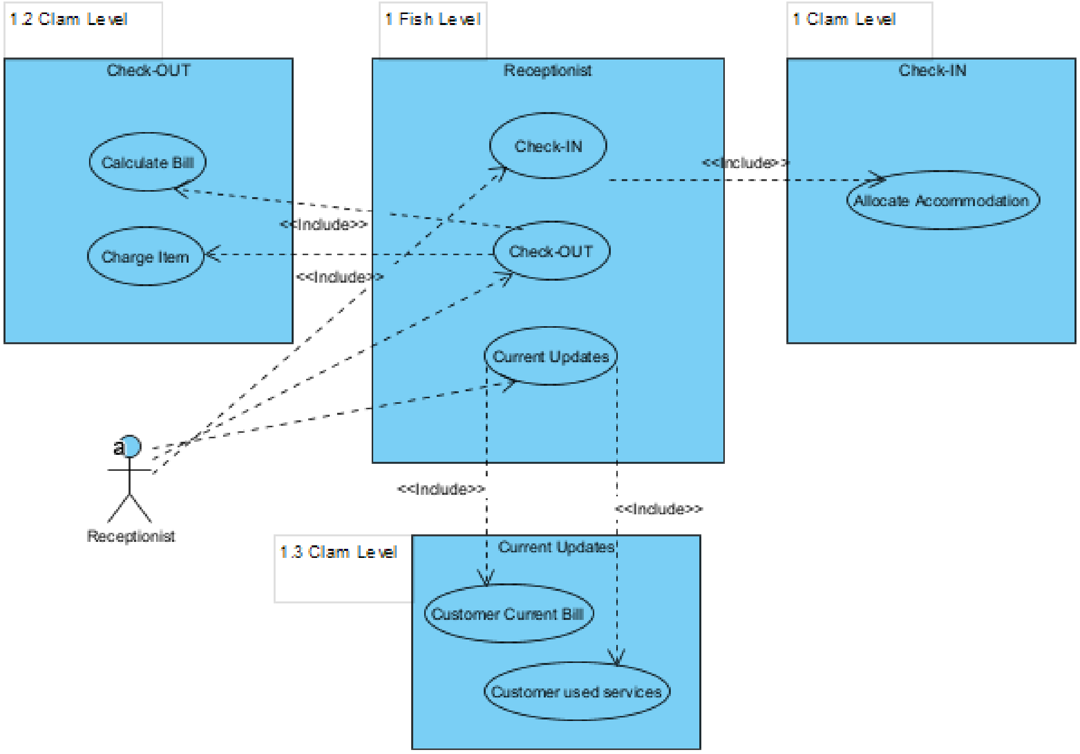


Figure 2:- Use Case Diagram (Receptionist)

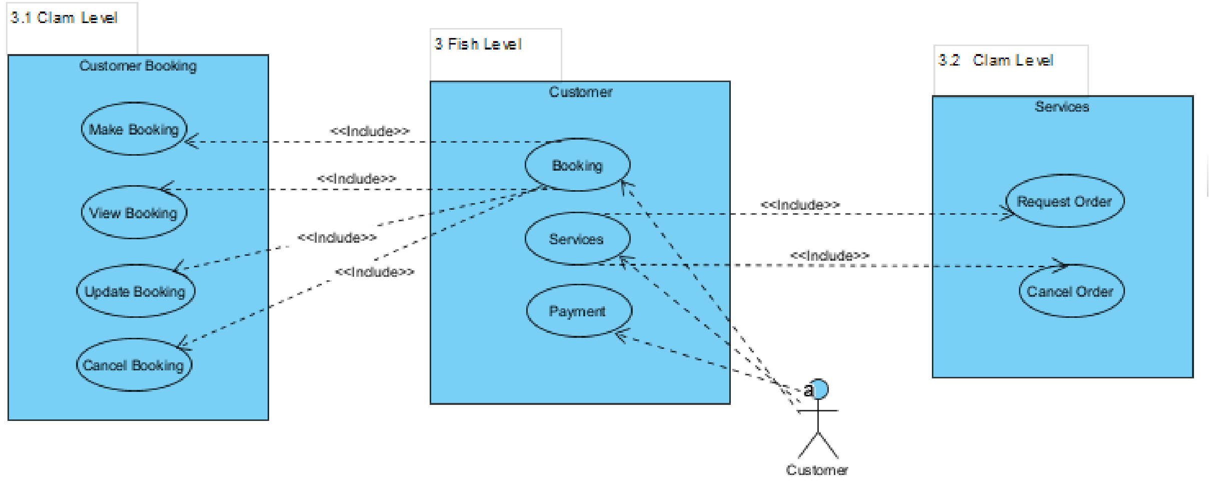


Figure 3:- Use Case Diagram (Customer)

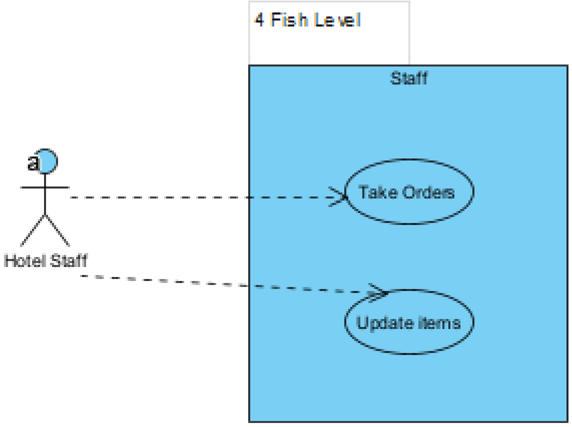


Figure 4:- Use Case Diagram (Hotel Staff)

- Use Case Specifications / Description

Check-IN

Each customer is checked in at the hotel when they arrive, and their room is then confirmed. All the customer information is confirmed, or if it was not given at the time of booking, it is included. Accommodations are assigned if they weren't already at the time of booking.

Allocate Accommodation

Every customer will receive a distinct room.

Check-OUT

Checking out is required for the reservation to be finalized before the guest leaves from the hotel. They get the complete room bill from the receptionist with each service they requested. The guest must verify and sign the bill if it is to be charged according to the customer or corporate user.

Calculate Bill

To generate total amount of a customer of what they have taken services as they were staying in the hotel including room, VAT, and extra services.

Charge Item

Charge all items or services that the guest has ordered.

Current Updates

Provide current information of the customer as of their stay include Customer Current Bill as well as their used service.

Customer Current Bill

Provide Current expenses of the customer on their request.

Customer used services

Provide information on their used services taken.

Booking

Booking includes overall everything that comes under booking services.

Make Booking

After a customer login into their account, they can request their desire room to stay from anywhere of their comfort. They can see the available rooms and request by providing check in and check out dates. Their room are confirmed only if they have provided their payment details or they can confirm it during the check in, until it considers as pending.

View Booking

Customer can view their requested booking to make any further requested as if they need it to be updated or to check if the information provided by them are correct or not.

Update Booking

To update the booking in case of wrong information are provided during the booking time. Once they check in into the hotel they can update the booking, so they must visit receptionist to increase or decrease their stay into the hotel.

Cancel Booking

Cancel if they change their mind or to change their booking room to match their requirements.

Pay Monthly Bill

Corporate company can pay their bill monthly which will only be generated at the end of every month.

Services

Customers or Receptionist can take or allocate the service which include all the facilities providing by the Hotel.

Request Order

Request their desire services which later will added on their bill during their check out.

Cancel Order

They even cancel if the order is not yet confirmed by the hotel staff members.

Payment

Non-corporate customers make payment for their check out from the hotel.

Take Orders

Hotel staff take the requested order form the customer or receptionist and provide it if that services are available.

Update items

Hotel Staff maintains the services in the hotel. They can add or remove the services which are currently available.

- ## Use Case Scenario

All the above use cases are the over functionality of the Hotel Booking System. The use cases are divided into 3 stage they are Sea Level, Fish Level and Clam Level. Sea Levels are the top view of the overall system. Fish level contain all the functions of a system. Clam level are the individual functionality of the system. All the clam level is included in Fish level whereas all sea level cases are associate in login function of overall system.

- ## Activity Diagram

Activity Diagram describe the overall flow of system on a certain level. Activity Diagram shows us actual figure on how system runs and communicate in between.

Below, there are Customers Activity Diagram and Receptionist Activity Diagram:

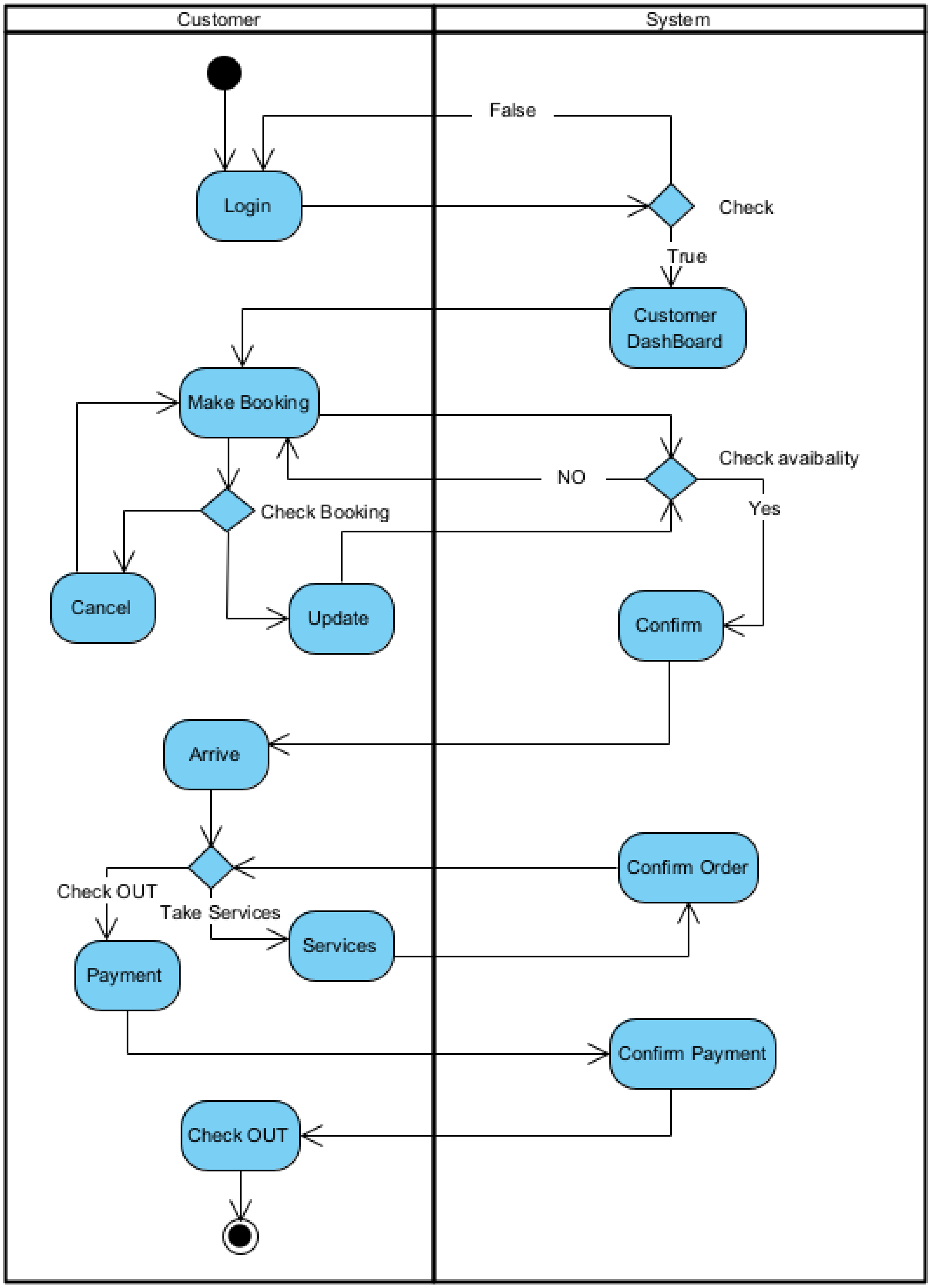


Figure 5:- Activity Diagram (Customer)

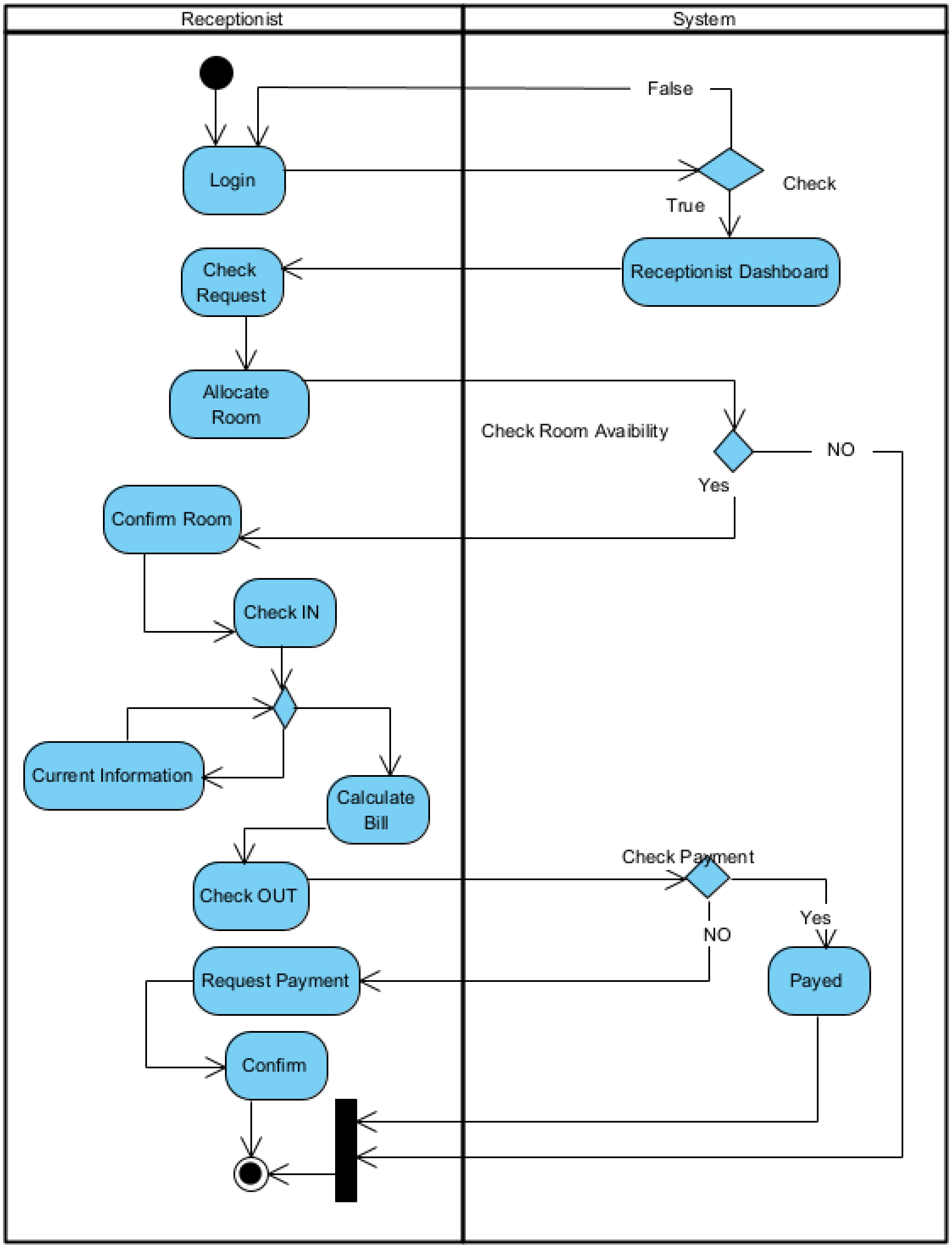


Figure 6:- Activity Diagram (Receptionist)

- ### Class Diagram

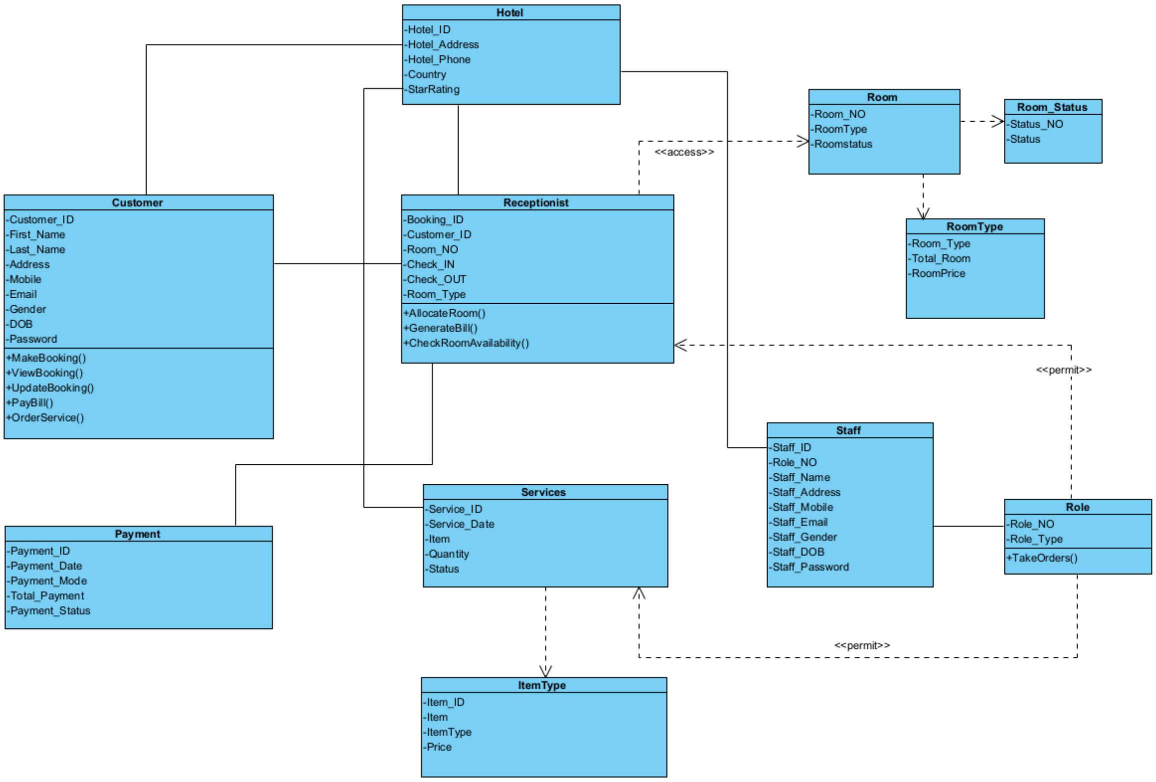


Figure 7:- Class Diagram

Database Design

- ### Entity Relationship Model

ERM is the part of database designing of an application system. This part shows on to the data and its entities are list on and help to segregate on specific manner. ERM diagram even help in relationship identification and relation between the entities.

- ## ERM Diagram

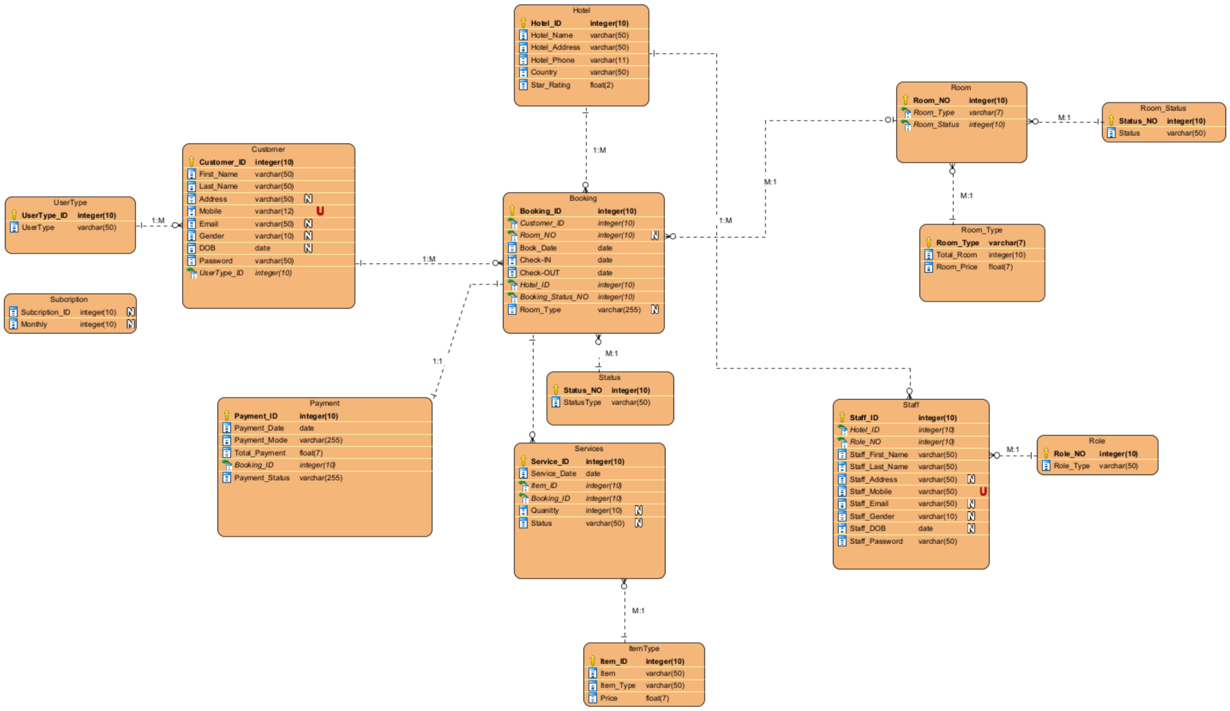


Figure 8:- ERM Diagram

- ## List of Entities
- **Hotel-** (Hotel_ID, Hotel_Name, Hotel_Address, Hotel_Phone, Country, Star_Rating)
- **Customer-** (Customer_ID,First_Name,Last_Name,Address,Email,Gender,DOB,Password,UserType_ID*)
- **UserType-** (UserType_ID,UserType)
- **Booking- (**Booking_ID,Room_NO*,Customer_ID*,Hotel_ID*,Status_NO*,Book_Date,Check_IN,Check_OUT, Room_Type)
- **Status-** (Status_NO,Status_Type)
- **Room-** (Room_NO,Room_Type*,Room_Status_NO*)
- **Room_Status-** (Room_Status_NO, Room_Status)
- **RoomType-** (RoomType, Total_Room, Room_Price)
- **Staff_ID- (**Staff_ID,Hotel_ID*,Role_NO*,Staff_First_Name,Staff_Last_Name,Staff_Address,Staff_Moblie,Staff_Email,Staff_Gender,Staff_DOB,Staff_Password)
- **Role-** (Role_NO,Role_Type)

- **Services-** (Service_ID,Booking_ID*,Item_ID*,Service_Date,Quantity,Status)
- **Item-** (Item_ID,Item,Item_Type,Item_Price)
- **Payment-** (Payment_ID,Booking_ID*,Payment_Date,Payment_Mode,Total_Payment,Payment_Status)

- ## Physical Database Design (including Data Dictionary)
  - ### *Skeleton Tables*

Hotel- (Hotel_ID, Hotel_Name, Hotel_Address, Hotel_Phone, Country, Star_Rating)

Customer- (Customer_ID,First_Name,Last_Name,Address,Email,Gender,DOB,Password,UserType_ID*)

UserType- (UserType_ID,UserType)

Booking- **(**Booking_ID,Room_NO*,Customer_ID*,Hotel_ID*,Status_NO*,Book_Date,Check_IN,Check_OUT,Room_Type)

Status- (Status_NO,Status_Type)

Room- (Room_NO,Room_Type*,Room_Status_NO*)

Room_Status- (Room_Status_NO, Room_Status)

RoomType- (RoomType, Total_Room, Room_Price)

Staff_ID- **(**Staff_ID,Hotel_ID*,Role_NO*,Staff_First_Name,Staff_Last_Name,Staff_Address,Staff_Moblie,Staff_Email,Staff_Gender,Staff_DOB,Staff_Password)

Role- (Role_NO,Role_Type)

Services- (Service_ID,Booking_ID*,Item_ID*,Service_Date,Quantity,Status)

Item- (Item_ID,Item,Item_Type,Item_Price)

Payment- (Payment_ID,Booking_ID*,Payment_Date,Payment_Mode,Total_Payment,Payment_Status)

- *Data Dictionary*

Customer							
Description: Customer details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Customer_ID (Primary)	int (10)	10	PK	No			Autoincremented Uniquely identifies every customer
First_Name	varchar	50		No			First Name of customer
Last_Name	varchar	50					Last Name of customer
Address	varchar	50		Yes			Address of customer
Mobile	varchar	15	Unique	No			Mobile of customer
Email	varchar	50		Yes		Must be email format containing an @ and a '.' Regex expression used	Email of customer
Gender	varchar	20		Yes			Gender of the customer
DOB	date			Yes			Date of birth of customer
Password	varchar	50		No		Should have at least 8 character and should have one capital letter and a number	Hotel password
UserType_ID* (Foreign Key)	int	10	FK				Type of user

Table 1:- Data Dictionary (Customer)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Customer_ID	NO
FOREIGN	BTREE	NO	UserType_ID	NO

UserType							
Description: User Type							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
UserType_ID (Primary)	integer	10	PK	No			Autoincremented Uniquely identifies every user type
User_Type	varchar	50		No			To store user type

Table 2:- Data Dictionary (UserType)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	UserType_ID	NO

Booking

Description: Booking details

Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Booking_ID (Primary)	int (10)	10	PK	No			Autoincremented Uniquely identifies every booking
Customer_ID* (Foreign Key)	int (10)	50	FK	No			ID of customer
Room_NO* (Foreign Key)	int (10)	50	FK	Yes			Room number of customer
Book_Date	date	50		No			Book date
Check_IN	date	15		No			Check in time of the customer
Check_OUT	date	50		Yes			Check out time of the customer
Booking_Status_NO* (Foreign Key)	int	10	FK	No			Status of the customer
Hotel_ID* (Foreign Key)	int	10	FK	No			Hotel id

Table 3:- Data Dictionary (Booking)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Booking_ID	NO
FOREIGN	BTREE	NO	Customer_ID	NO
FOREIGN	BTREE	NO	Room_NO	NO
FOREIGN	BTREE	NO	Booking_Status_NO	NO
FOREIGN	BTREE	NO	Hotel_ID	NO

Status

Description: Status details

Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Status_NO (Primary)	**int**	**10**	**PK**	**No**			Autoincremented Uniquely identifies every booking status
Status_Type	varchar	50		No			Status

Table 4:- Data Dictionary (Status)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Status_NO	NO

Hotel							
Description: Hotel details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Hotel_ID (Primary)	int (10) unsigned	10	PK	No			Autoincremented Uniquely identifies hotel
Hotel_name	varchar	50		No			Hotel Name
Hotel_Address	varchar	50					Hotel Address
Hotel_Phone	varchar	15		No			Hotel Phone Number
Country	varchar	15		No			Hotel Country
Star_Rating	float	2,1		No			Hotel Rating

Table 5:- Data Dictionary (Hotel)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Hotel_ID	NO

Payment							
Description: Payment details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Payment_ID (Primary)	int (10) unsigned	10	PK	No			Autoincremented Uniquely identifies every customer payment id
Booking_ID* (Foreign Key)	integer	10	FK	No			Customer ID
Services_ID* (Foreign Key)	integer	10	FK	Yes			Services
Date	Date			No			Payment Date
Payment_Mode	varchar	50		No			Payment method
Total_Payment	float	7		No			Total Payment
Status	varchar	50		No			Payment Status

Table 6:- Data Dictionary (Payment)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Payment_ID	NO
FOREIGN	BTREE	NO	Booking_ID	NO
FOREIGN	BTREE	NO	Services_NO	NO

Services							
Description: Services details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Service_ID (Primary)	int (10) unsigned	10	PK	No			Autoincremented Uniquely identifies every customer services
Booking_ID* (Foreign Key)	integer	10	FK				To store customer id
Date	Date			No			Services date
Item_ID* (Foreign Key)	Integer	10	FK	No			Services Type

Table 7:- Data Dictionary (Services)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Payment_ID	NO
FOREIGN	BTREE	NO	Booking_ID	NO
FOREIGN	BTREE	NO	Item_ID	NO

ItemType							
Description: ItemType details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Item_ID(Primary)	Integer	10	PK	No			Autoincremented Uniquely identifies every item
Item	Varchar	50		No			Item
Item_Type	varchar	50		No			Item Type
Price	float	7		No			To store item price

Table 8:- Data Dictionary (ItemType)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Item_ID	NO

Room							
Description: Room details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Room_NO (Primary)	integer	10	PK	No			Autoincremented Uniquely identifies every room
Room_Type* (Foreign Key)	varchar	50	FK	No			To store room type
Status_NO* (Foreign Key)	int	10	FK				Status of the room

Table 9:- Data Dictionary (Room)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Room_NO	NO
FOREIGN	BTREE	NO	Room_Type	NO
FOREIGN	BTREE	NO	Status_NO	NO

Room_Status							
Description: Room details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Status_NO (Primary)	integer	10	PK	No			Autoincremented Uniquely identifies every room
Room_Status	varchar	50		No			To store room type

Table 10:- Data Dictionary (Room Status)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Room_Status	NO

RoomType							
Description: RoomType details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Room_Type (Primary)	varchar	7	PK	No			Autoincremented Uniquely identifies every roomtype
Total_Room	integer	10		No			Total number of room
Price	float	7					Price of the room

Table 11:- Data Dictionary (Room Type)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	RoomType	NO

Staff							
Description: Staff details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Staff_ID (Primary)	int (10) unsigned	10	PK	No			Autoincremented Uniquely identifies every staff
Hotel_ID* (Foreign Key)	int	10	FK	No			Hotel Id
Role_NO* (Foreign Key)	integer	10	FK	No			Role of the Staff member
Staff_First_name	varchar	50		No			First name of staff
Staff_Last_name	varchar	50					Last name of staff
Staff_Address	varchar	50		No			Address of staff
Staff_Mobile	varchar	15		No			Mobile of staff
Staff_Email	varchar	50		No		Must be email format containing an @ and a ‘.’ Regex expression used	Email of staff
Staff_Gender	varchar	20		No			Gender of the staff
Staff_DOB	date			No			Date of birth of staff
Staff_Password	varchar	50		No			Hotel password

Table 12:- Data Dictionary (Staff)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Staff_ID	NO
FOREIGN	BTREE	NO	Hotel_ID	NO
FOREIGN	BTREE	NO	Role_NO	NO

Role							
Description: Role details							
Field Name	**Datatype**	**Length**	**Index**	**Null**	**Default**	**Validation rule**	**Description**
Role_NO (Primary)	integer	10	PK	No			Autoincremented Uniquely identifies role room
Role_Type	varchar	50		No			To store role type

Table 13:- Data Dictionary (Role)

Indexes

Keyname	Type	Unique	Column	Null
PRIMARY	BTREE	Yes	Role_NO	NO

User Interface Design

User Interface Design are the frontend layout of a system where user gets to interact with system. UI design help to easily visualize and take control over the running application. UI parts is where the user get to experience the application. Below there are UI Design of the Hotel Booking System:

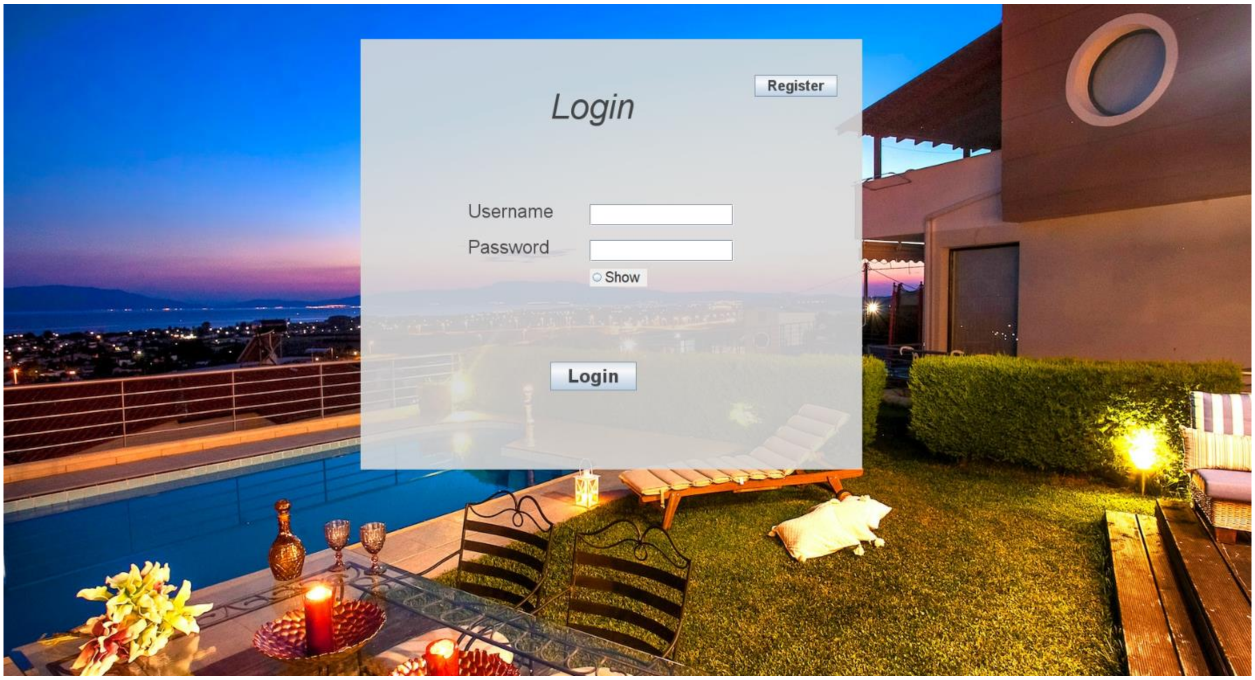
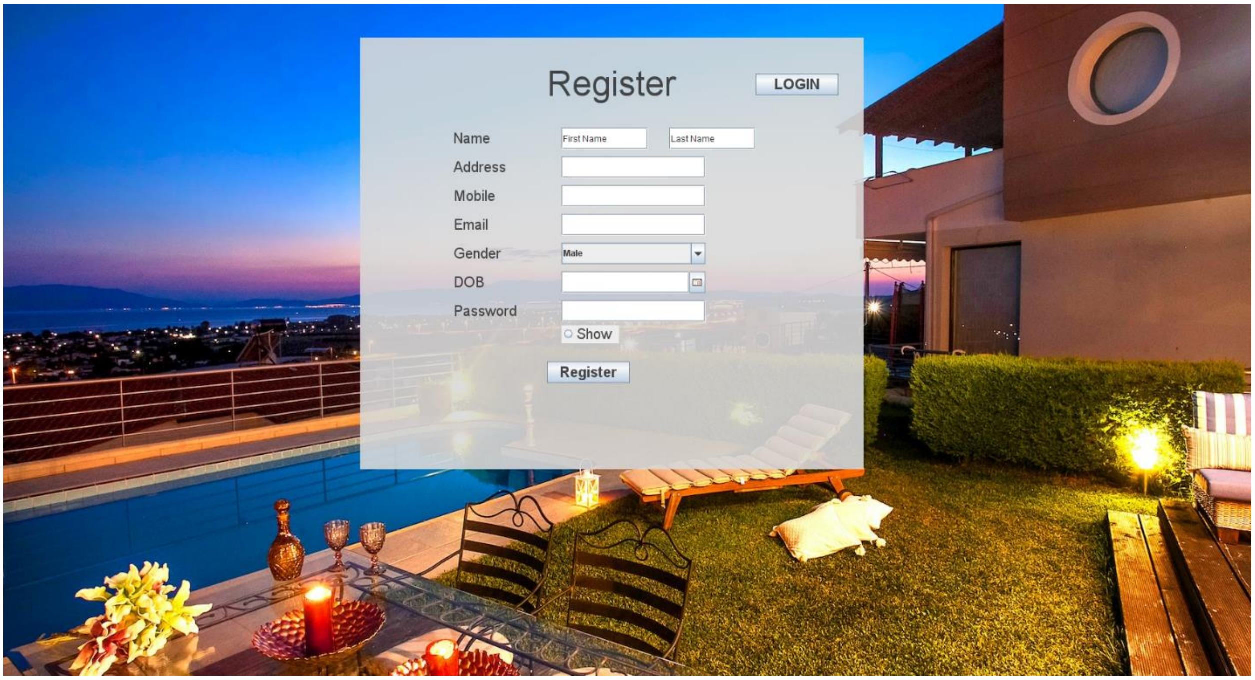


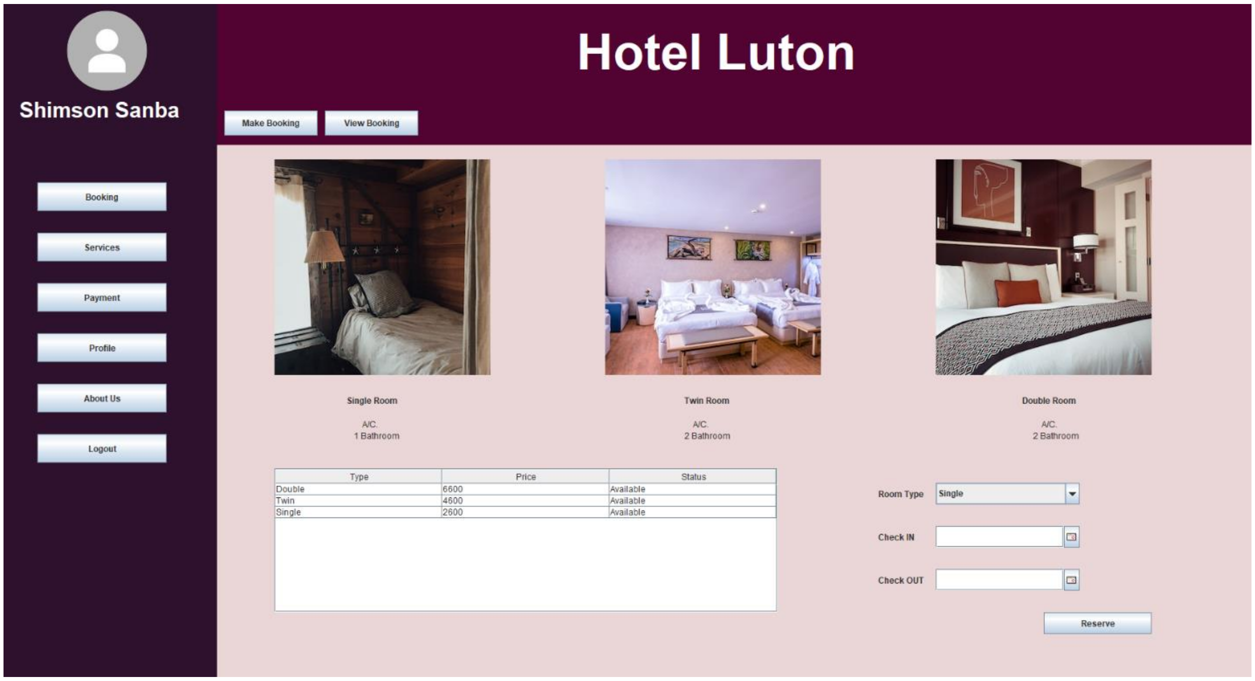
Figure 9:- UI (Login)



Register

Name: First Name | Last Name
Address:
Mobile:
Email:
Gender: Male
DOB:
Password:
○ Show

Figure 10:- UI (Register)



Hotel Luton

Shimson Sanba

Single Room
A/C.
1 Bathroom

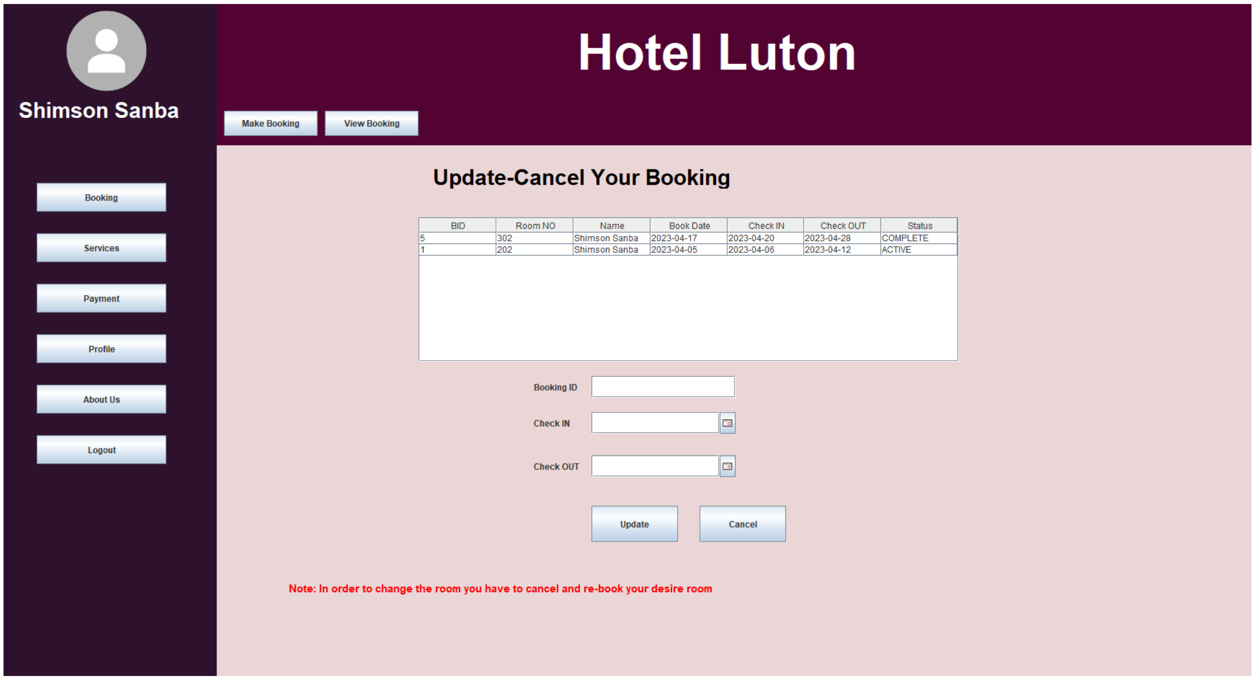
Twin Room
A/C.
2 Bathroom

Double Room
A/C.
2 Bathroom

Type	Price	Status
Double	6600	Available
Twin	4600	Available
Single	2600	Available

Room Type: Single
Check IN:
Check OUT:

Figure 11:- UI (Customer Make Booking)



Hotel Luton

Update-Cancel Your Booking

BID	Room NO	Name	Book Date	Check IN	Check OUT	Status
5	302	Shimson Sanba	2023-04-17	2023-04-20	2023-04-28	COMPLETE
1	202	Shimson Sanba	2023-04-05	2023-04-06	2023-04-12	ACTIVE

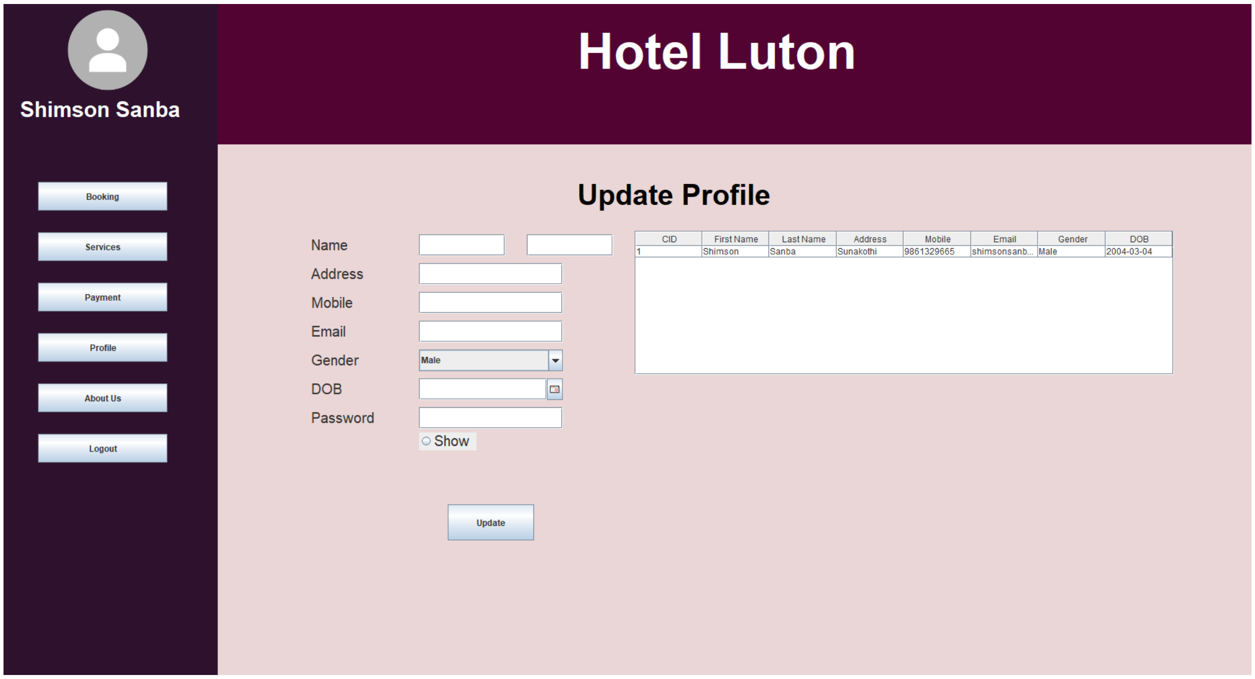
Booking ID

Check IN

Check OUT

Note: In order to change the room you have to cancel and re-book your desire room

Figure 12:- UI (Customer View Booking)



Hotel Luton

Update Profile

Name

Address

Mobile

Email

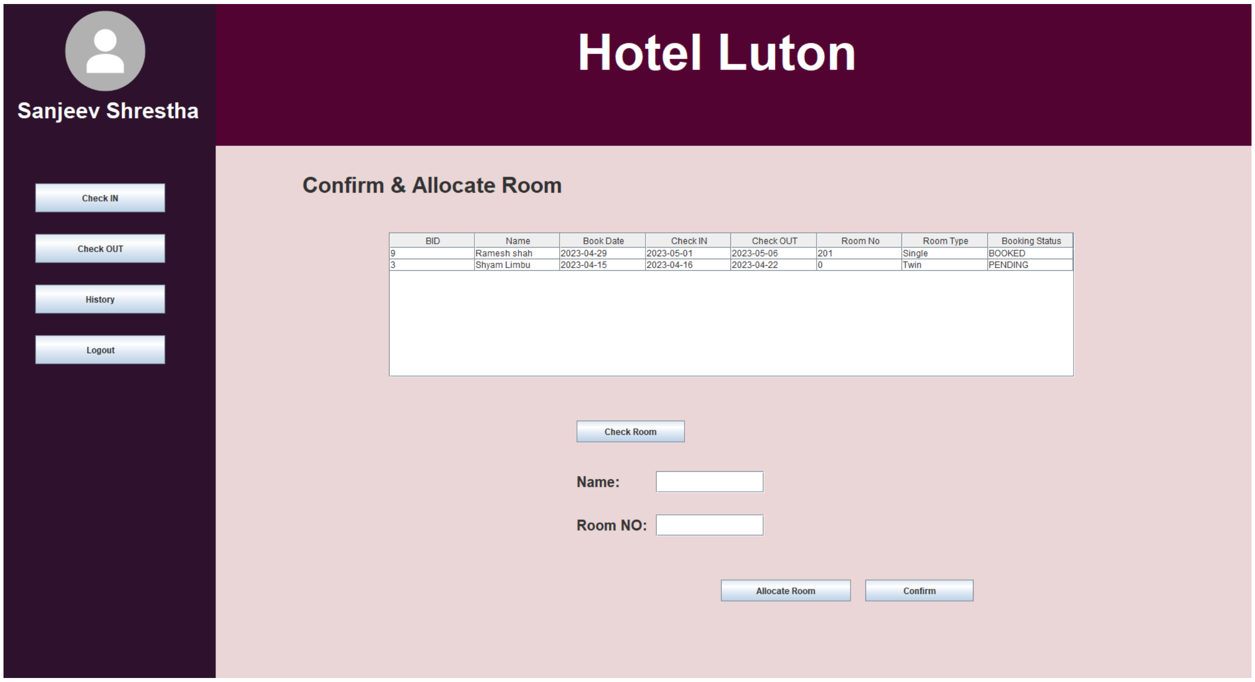
Gender: Male

DOB

Password

CID	First Name	Last Name	Address	Mobile	Email	Gender	DOB
1	Shimson	Sanba	Sunakothi	9861329665	shimsonsanb...	Male	2004-03-04

Figure 13:- UI (Customer Update Profile)



Hotel Luton

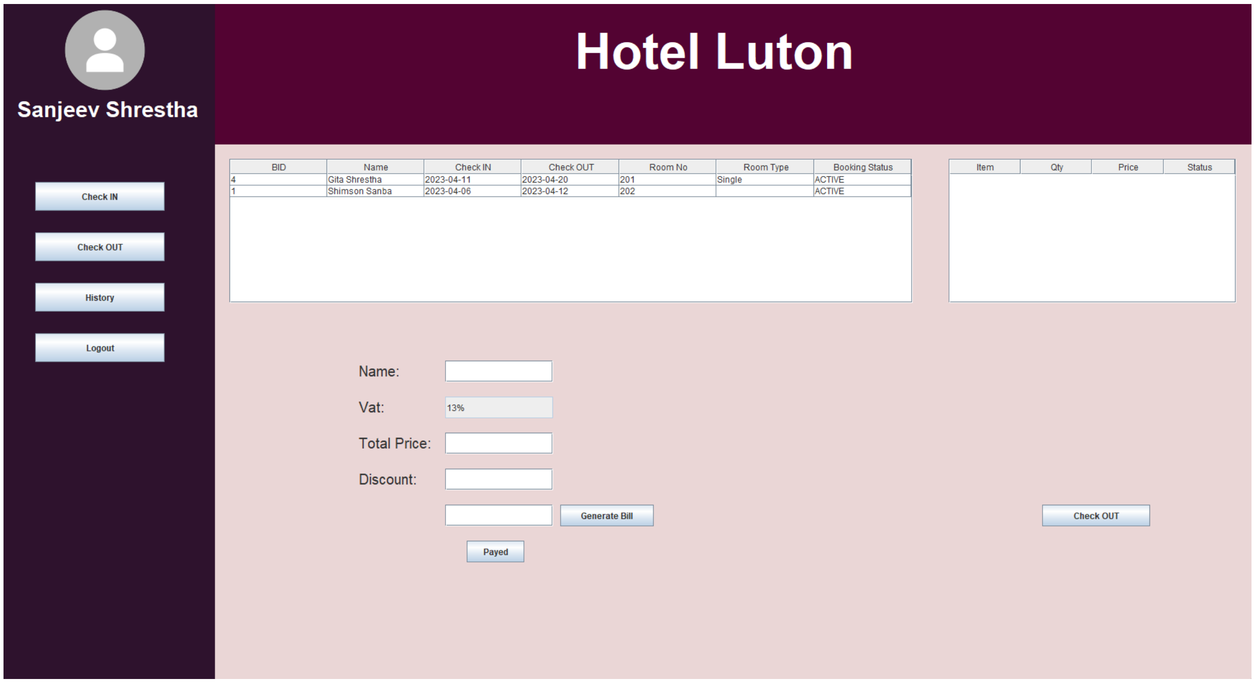
Confirm & Allocate Room

BID	Name	Book Date	Check IN	Check OUT	Room No	Room Type	Booking Status
9	Ramesh shah	2023-04-29	2023-05-01	2023-05-06	201	Single	BOOKED
3	Shyam Limbu	2023-04-15	2023-04-16	2023-04-22	0	Twin	PENDING

Name:

Room NO:

Figure 14:- UI (Receptionist Check IN)



Hotel Luton

BID	Name	Check IN	Check OUT	Room No	Room Type	Booking Status
4	Gita Shrestha	2023-04-11	2023-04-20	201	Single	ACTIVE
1	Shimson Sanba	2023-04-06	2023-04-12	202		ACTIVE

Item	Qty	Price	Status

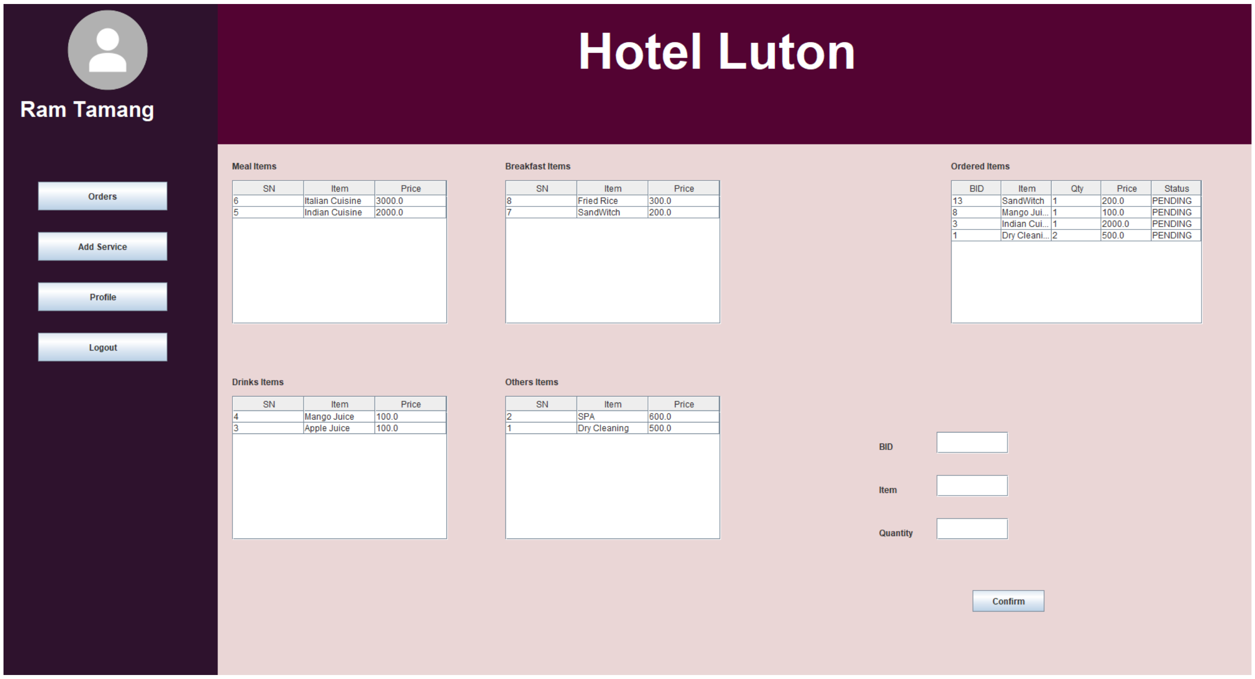
Name:

Vat: 13%

Total Price:

Discount:

Figure 15:- UI (Receptionist Check OUT)



Meal Items

SN	Item	Price
6	Italian Cuisine	3000.0
5	Indian Cuisine	2000.0

Breakfast Items

SN	Item	Price
8	Fried Rice	300.0
7	SandWitch	200.0

Ordered Items

BID	Item	Qty	Price	Status
13	SandWitch	1	200.0	PENDING
8	Mango Jui...	1	100.0	PENDING
3	Indian Cui...	1	2000.0	PENDING
1	Dry Cleani...	2	500.0	PENDING

Drinks Items

SN	Item	Price
4	Mango Juice	100.0
3	Apple Juice	100.0

Others Items

SN	Item	Price
2	SPA	600.0
1	Dry Cleaning	500.0

Figure 16:- UI (Staff Take Order)

Implementation

The project Hotel booking system was created using the java programming language. Java is one of the old and popular language to develop various kind of application. The development includes different stages which involves UML design to outline the system, functionality and its overall structure, database design to create schema, GUI design to create user interface solution, coding to implement the functionality of overall application and testing to ensure that the application provide all the essential working mechanism to maintain the application quality as well.

As the project is an individual work So, to manage the implementation process and the workload, I separate and allocate the specific time for a specific topic to be done during that period. Mostly I used agile methodology to track my progress, adding new functionality and to ensure that the project is on track.

Regarding the use of Integrated Development Environment (IDE), I have used Eclipse which is a popular choice for java developers due to its robust features and easy-to-use interface. I have used Java Swing to create GUI, which provide main component. As for the packages /libraries, I have used various libraries such as for date choosing, I have used JCalendar and for retrieving and sending data to database I have used MySQL-java-connector to establish dataflow during the application usage.

I have used client application to embed SQL rather than Client-Server (Sockets Programming architecture) because it was an easy method to implement MVC model and to get work with without an issue. So, the main reason or advantage of choosing this method is that it is easy to use and do not have to think extra for any other kind of process like socket connection etc.

During the coding process, there are many common problems that I have encountered such as bugs, unexpected behavior, design issue, and some technical error. To overcome those problems, I have used debugging features from IDE, review code to identity the flaw and fix the error after dedicated research from many blogs and tutorial site of related topic.

For the database management system, I have used MySQL which is known for its ease of use, scalability, and reliability. It is widely used and had a large community of developers and users where they can provide any kind of help regarding the SQL issue. No, I have not used University’s MySQL database because it was not always easy for me to establish connection from my home so for the database access, I have used WAMP.

To make connection with database I have used WAMP as said before which is quite efficient and more convenient then XAMPP that I used in last Semester. XAMPP used to give me very hard to so after some consultant with my teacher I am using WAMP which is working perfectly fine for me which is equivalents for my PC as well.

Regarding the development experience, Java is a powerful programming language that enables developers to create complex applications with ease. I feel like I have learned a new term and knowledge on coding as well as on software development. The project development has helped me to be more confident on application development, OOPs concepts and to learn furthermore deep in any kind of programming language. MCV model and OOPs concepts has helped me to maintain code and its reusability which is a lot easier than I used to thought in the beginning.

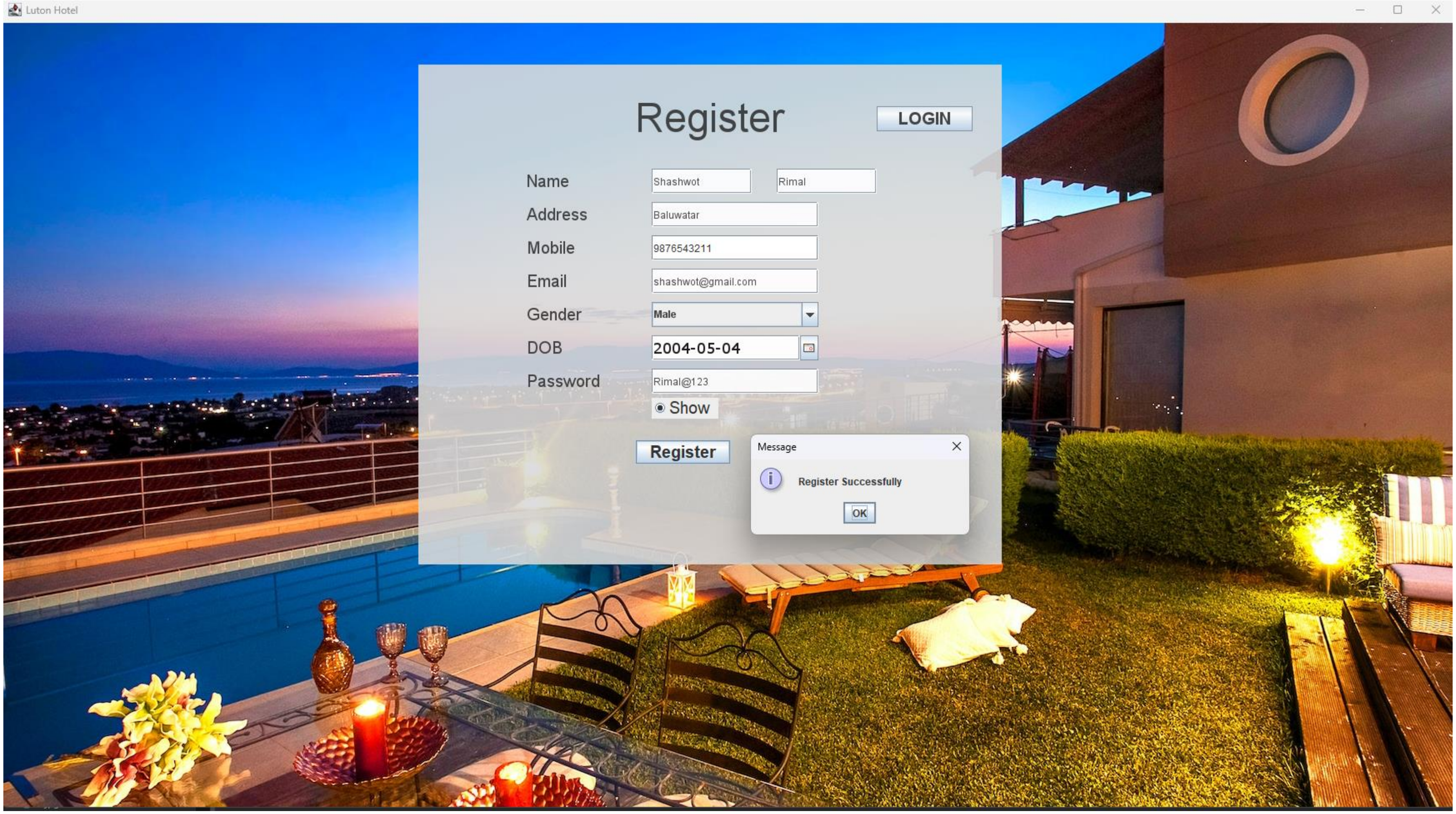
For the novel approaches to coding, I have used Model View Controller (MVC) method to meet the assignment requirements. In general, it has encouraged to be innovative and creative when developing the software. This method helps me to optimize my performance through efficient algorithms and from user-friendly interfaces.

In short, the hotel booking system was developed by using Java after following standard development stages such as UML design, database design, GUI, design, coding, and testing. Agile method, Eclipse IDE, MySQL, and WAMP are used for the development of the application after the well time management over the period.

Testing

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
1-HBS	May 1, 2023	Customer Registration	1. Run the application 2. Click On Register button 3. Insert all the information 4. Click on the “Register” button.	1. The customer account should be created and display the “Register Successfully” message.	1. The customer account was created and displayed the “Register Successfully” message.	Test Pass

Table 14:- Testing (Registration)



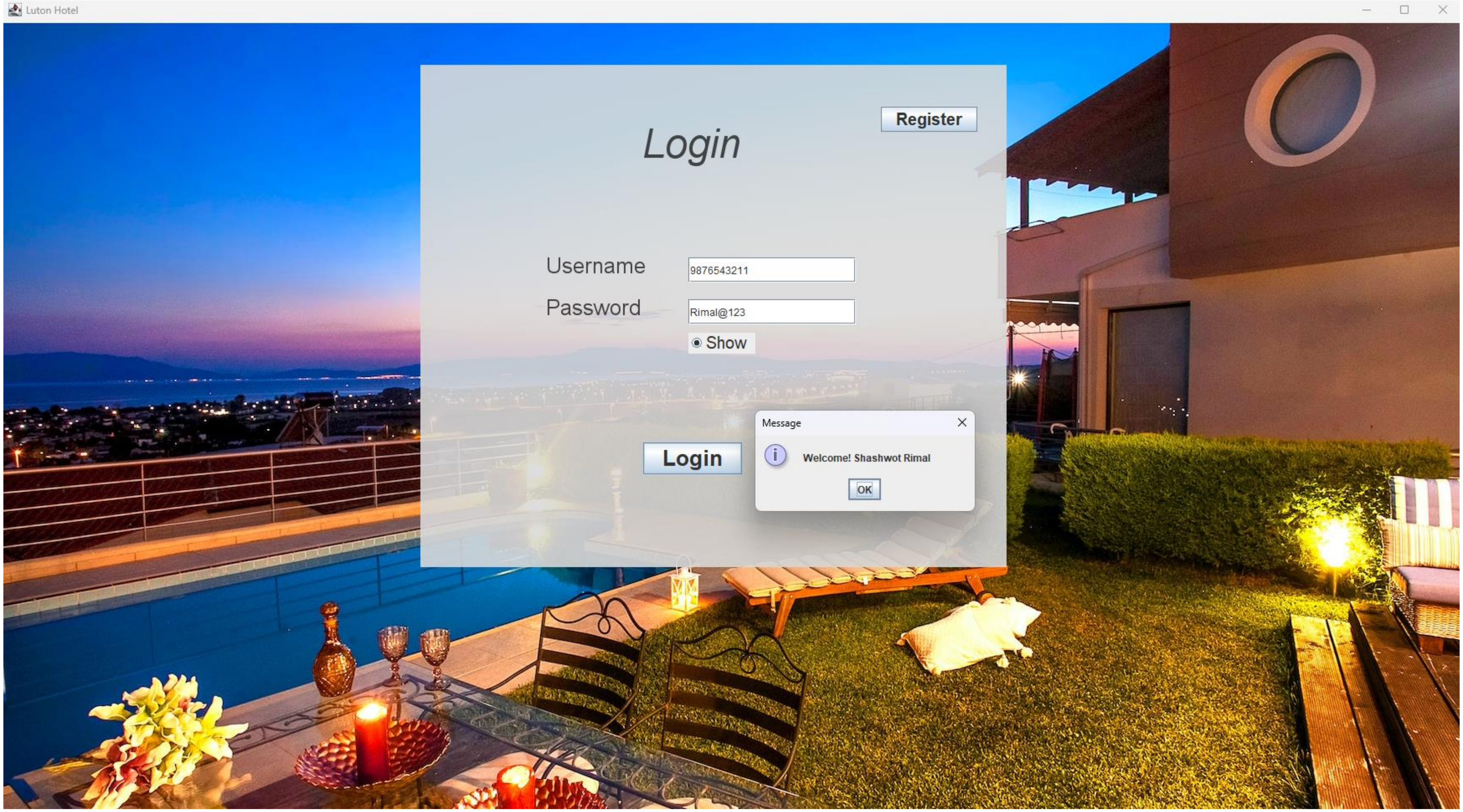
Name: Shashwot Rimal
Address: Baluwatar
Mobile: 9876543211
Email: shashwot@gmail.com
Gender: Male
DOB: 2004-05-04
Password: Rimal@123
● Show

Message
Register Successfully

Figure 17:- Testing (Register)

Test Case ID	**Test Date**	**Purpose of Test**	**Test Steps**	**Expected Results**	**Actual Results**	**Action**
2-HBS	May 1, 2023	Login	1. Run the application 2. Insert all the information 3. Click on the “Login” button.	1. The user should be Login into Dashboard after displaying “Welcome” Message	1. The user was able to Login into system after displaying “Welcome” message.	Test Pass

Table 15:- Testing (Login)



Username 9876543211

Password Rimal@123

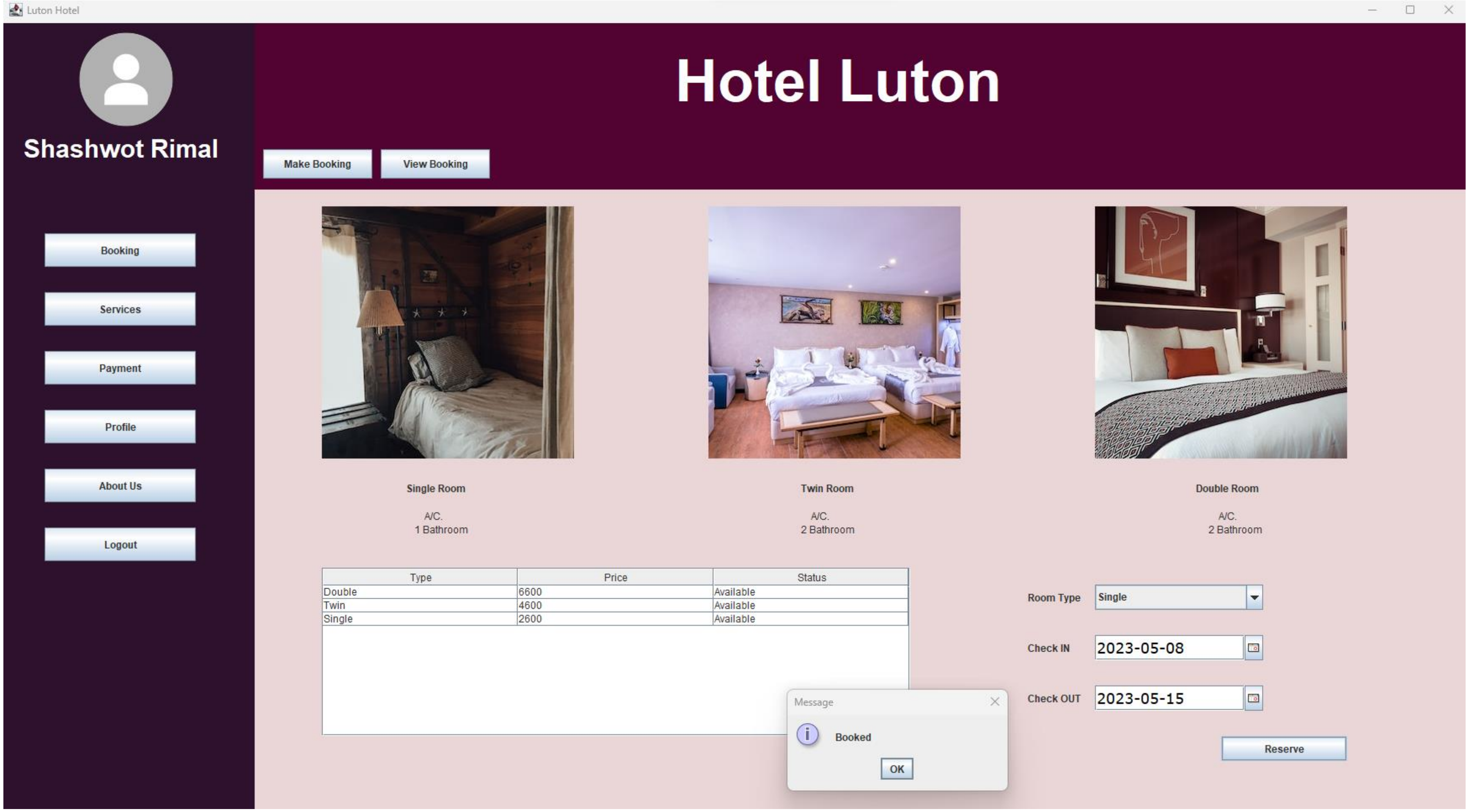
● Show

Welcome! Shashwot Rimal

Figure 18:- Testing (Login)

Test Case ID	**Test Date**	**Purpose of Test**	**Test Steps**	**Expected Results**	**Actual Results**	**Action**
3-HBS	May 1, 2023	Make a Booking	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Booking” button to make booking. 5. View Room price and necessary details. 6. Insert suitable check-in and check-out date and select room from drop down box. 7. Click on “Reserve” button to make reservation. 8. Display message “Booking Successful”	1. The user should be Login into Dashboard after displaying “Welcome” Message 2. User should be able to click on “Booking” button. 3. User should be able to view details and fill the information 4. User should be able to click on “Reserve” button and should see “Booking Successful” Message	1. Login successful 2. User can click on “Booking” 3. User can fill all the necessary form 4. User has successfully able to make their reservation 5. Message shown	Test Pass

Table 16:- Testing (Make Booking)



Type	Price	Status
Double	6600	Available
Twin	4600	Available
Single	2600	Available

Room Type: Single

Check IN: 2023-05-08

Check OUT: 2023-05-15

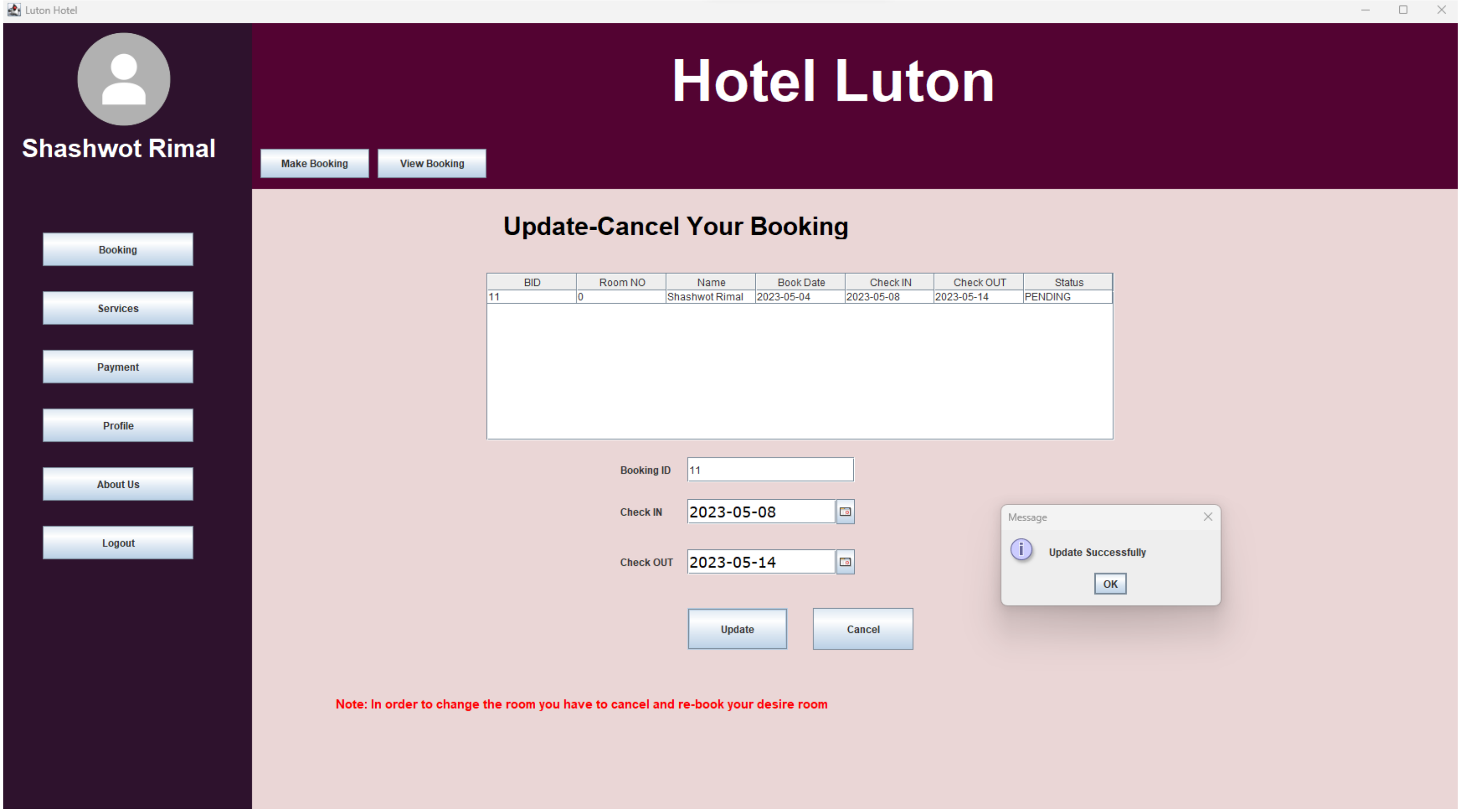
Message

Booked

Figure 19:- Testing (Make Booking)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
4-HBS	May 1, 2023	View Booking	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Booking” button to view booking. 5. View their booking status 6. Update check-in and check-out date OR cancel. 7. Click on “Update” OR “Cancel” button as per need 8. Display message.	1. The user should be Login into Dashboard after displaying “Welcome” Message 2. User should be able to click on “Booking” button. 3. User should be able to view details and fill the information 4. User should be able to click on “Update” OR “Cancel” button and should Message	1. Login successful 2. User can click on “Booking” 3. User can fill all the necessary form 4. User has successfully able to update their reservation 5. Message shown	Test Pass

Table 17:- Testing (View Booking)



Update-Cancel Your Booking

BID	Room NO	Name	Book Date	Check IN	Check OUT	Status
11	0	Shashwot Rimal	2023-05-04	2023-05-08	2023-05-14	PENDING

Booking ID 11

Check IN 2023-05-08

Check OUT 2023-05-14

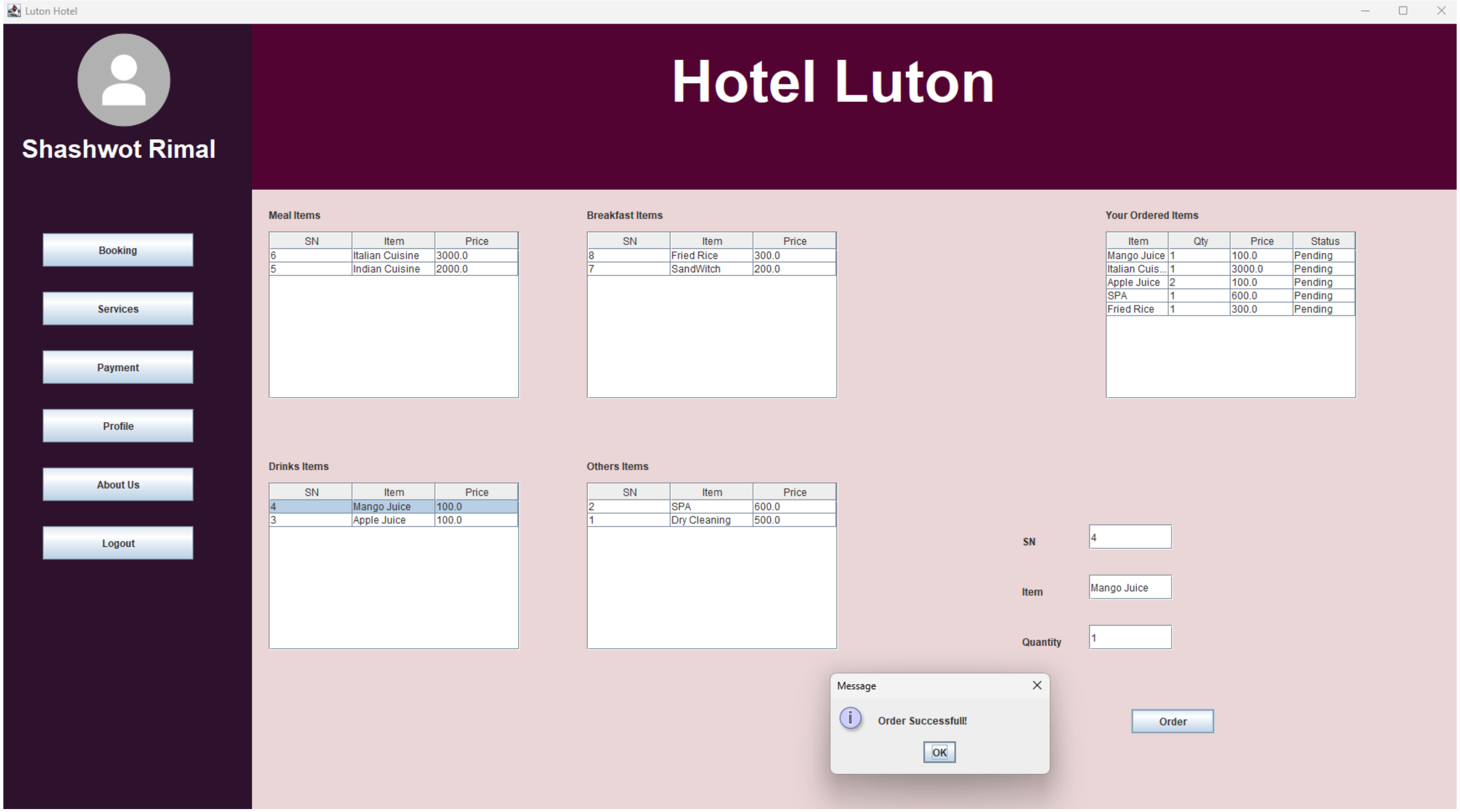
Update Successfully

Note: In order to change the room you have to cancel and re-book your desire room

Figure 20:- Testing (View Booking)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
5-HBS	May 1, 2023	Take Service	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Service” button to make order. 5. View Items 6. Order the items 7. Click on “Order” 8. Display message.	1. The user should be Login into Dashboard after displaying “Welcome” Message 2. User should be able to click on “Service” button. 3. User should be able to view details and fill the information after clicking on the table 4. User should be able to click on “Order” button and should Message	1. Login successful 2. User was able click on “Order” 3. User was able order all the necessary item 4. User successful to make order 5. Message shown	Test Pass

Table 18:- Testing (Take Service)



Meal Items

SN	Item	Price
6	Italian Cuisine	3000.0
5	Indian Cuisine	2000.0

Breakfast Items

SN	Item	Price
8	Fried Rice	300.0
7	SandWitch	200.0

Your Ordered Items

Item	Qty	Price	Status
Mango Juice	1	100.0	Pending
Italian Cuis...	1	3000.0	Pending
Apple Juice	2	100.0	Pending
SPA	1	600.0	Pending
Fried Rice	1	300.0	Pending

Drinks Items

SN	Item	Price
4	Mango Juice	100.0
3	Apple Juice	100.0

Others Items

SN	Item	Price
2	SPA	600.0
1	Dry Cleaning	500.0

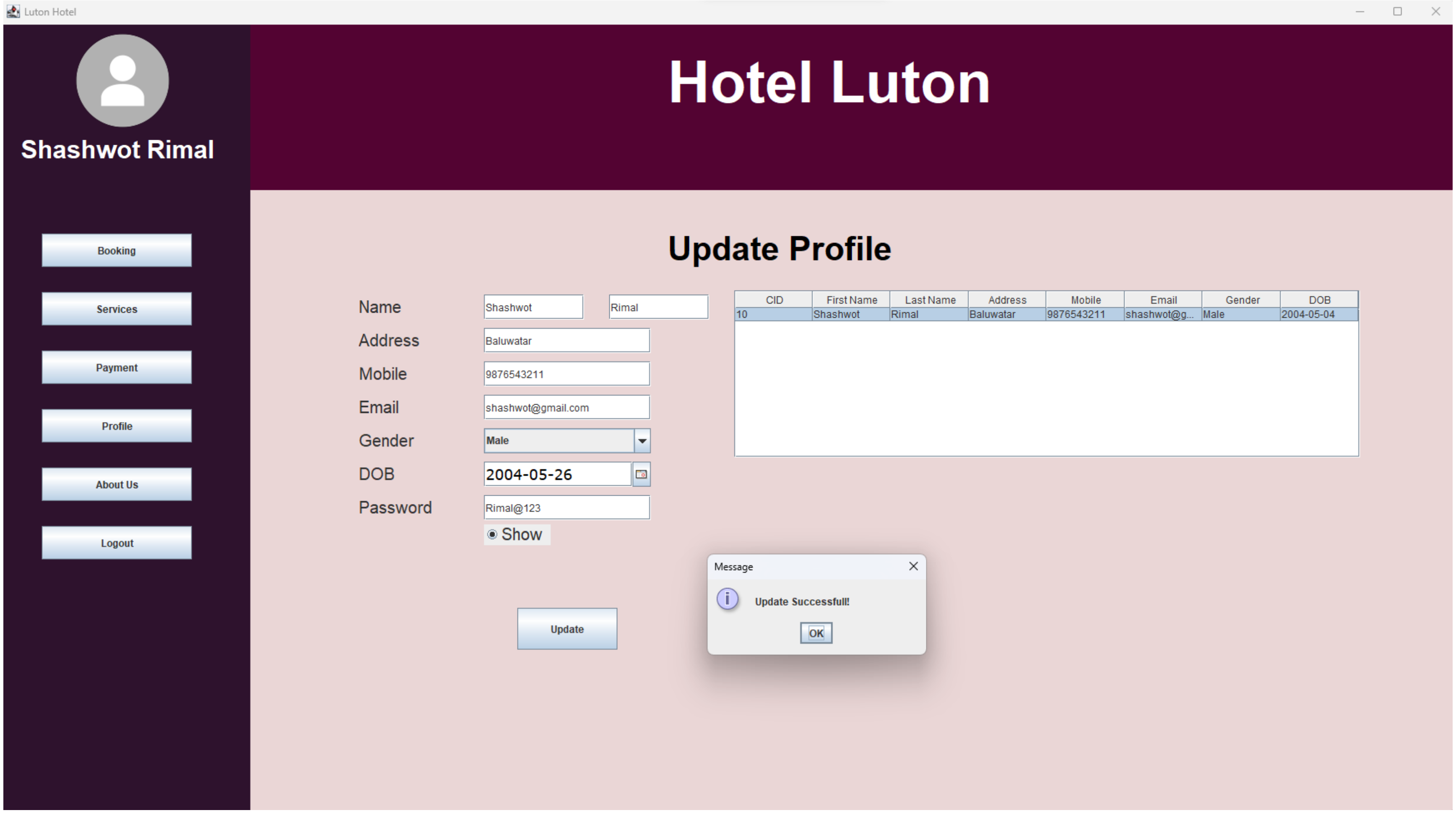
SN: 4
Item: Mango Juice
Quantity: 1

Message
Order Successfull!
OK

Figure 21:- Testing (Take Service)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
6-HBS	May 1, 2023	Update Profile	1. Run application 2. Insert all the information 3. Click on the "Login" button. 4. Click on "Profile" button to update. 5. Click on table data 6. Insert necessary details to be update 7. Click on "Update" 8. Display message.	1. The user should be Login into Dashboard after displaying "Welcome" Message 2. User should be able to click on "Profile" button. 3. User should be able to view details and fill the information after clicking on the table 4. User should be able to click on "Update" button and should See successful Message	1. Login successful 2. User was able click on "Profile" 3. User was able to update their profile accordingly 4. Message shown	Test Pass

Table 19:- Testing (Update Profile)



Hotel Luton

Shashwot Rimal

Update Profile

Name: Shashwot Rimal
Address: Baluwatar
Mobile: 9876543211
Email: shashwot@gmail.com
Gender: Male
DOB: 2004-05-26
Password: Rimal@123
● Show

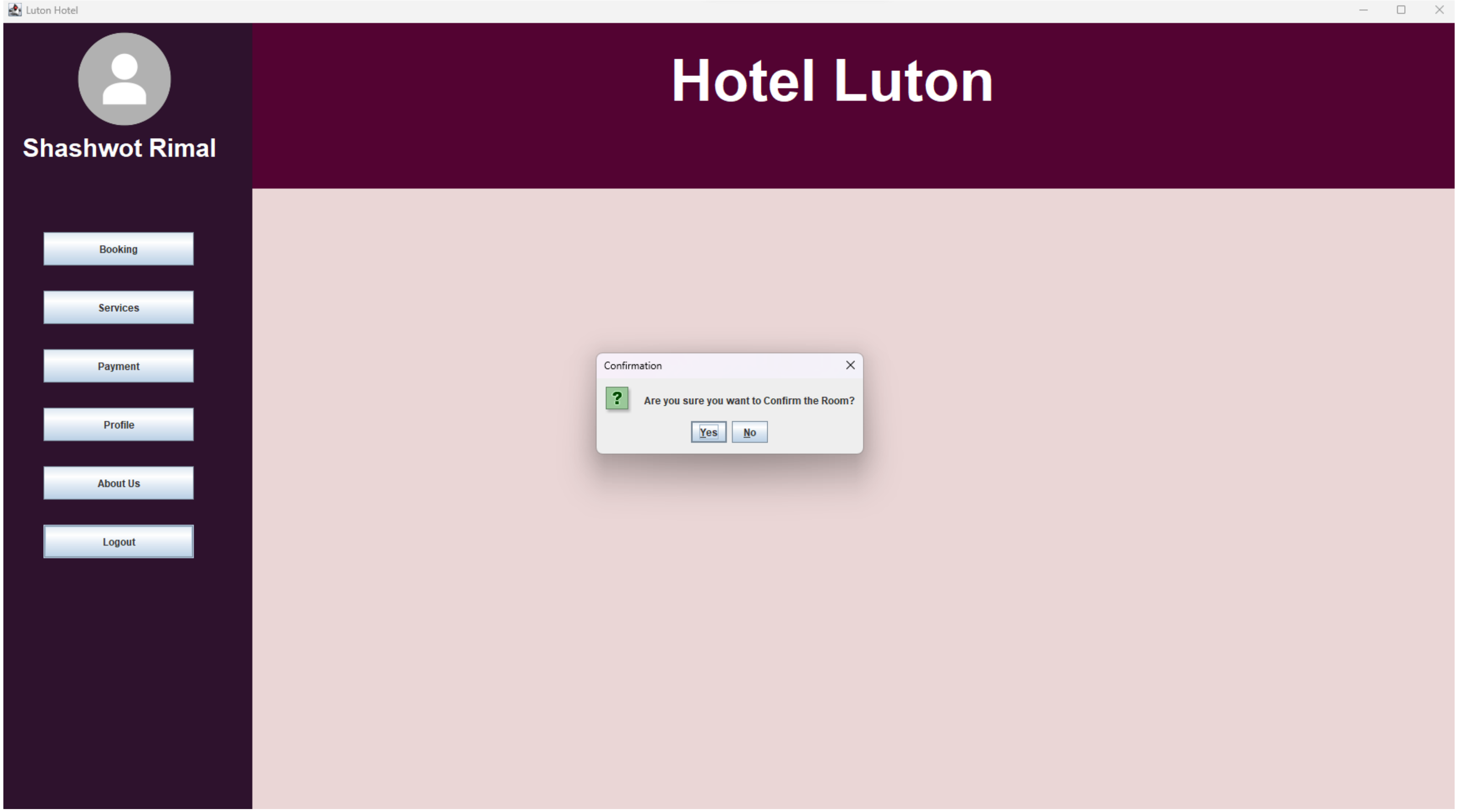
CID	First Name	Last Name	Address	Mobile	Email	Gender	DOB
10	Shashwot	Rimal	Baluwatar	9876543211	shashwot@g...	Male	2004-05-04

Message
Update Successfull!

Figure 22:- Testing (Update Profile)

Test Case ID	**Test Date**	**Purpose of Test**	**Test Steps**	**Expected Results**	**Actual Results**	**Action**
7-HBS	May 1, 2023	Logout	1. Run the application 2. Insert all the information 3. Click on the "Login" button. 4. Click on "Logout" button 5. Display Confirmation message.	1. The user should be Login into Dashboard after displaying "Welcome" Message 2. User should be able to click on "Logout" button 3. Confirmation message should pop up 4. Should be able to logout after pressing "Yes"	1. The user was able to Login into system after displaying "Welcome" message. 2. User was able to click on "Logout" button 3. Confirmation was popped up 4. Successfully logout after pressing "Yes"	Test Pass

Table 20:- Testing (Logout)

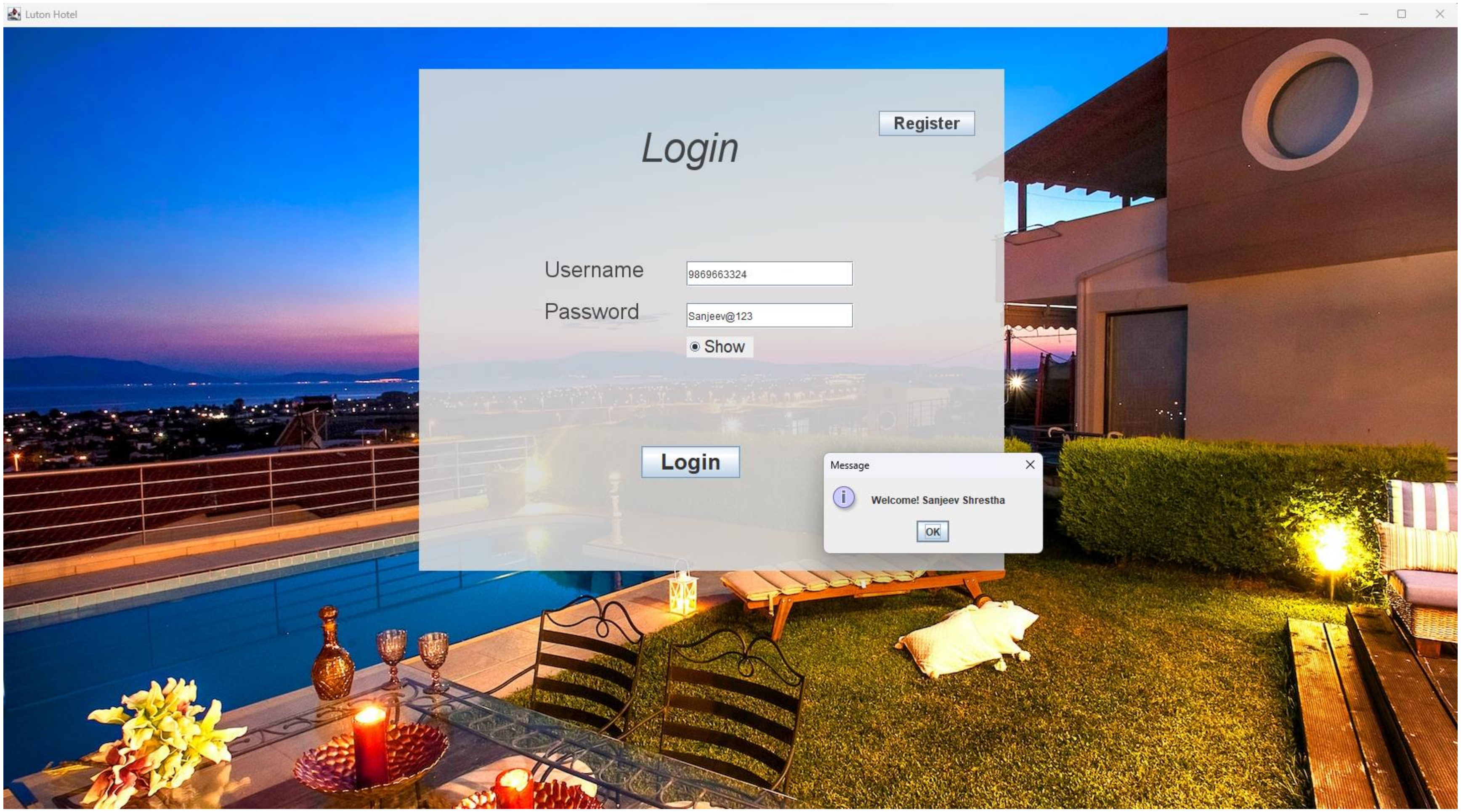


Are you sure you want to Confirm the Room?

Figure 23:- Testing (Logout)

Test Case ID	**Test Date**	**Purpose of Test**	**Test Steps**	**Expected Results**	**Actual Results**	**Action**
8-HBS	May 2, 2023	Receptionist Login	1. Run the application 2. Insert all the information 3. Click on the “Login” button.	1. The receptionist should be Login into Dashboard after displaying “Welcome” Message	1. The receptionist was able to Login into system after displaying “Welcome” message.	Test Pass

Table 21:- Testing (Receptionist Login)



Username 9869663324

Password Sanjeev@123

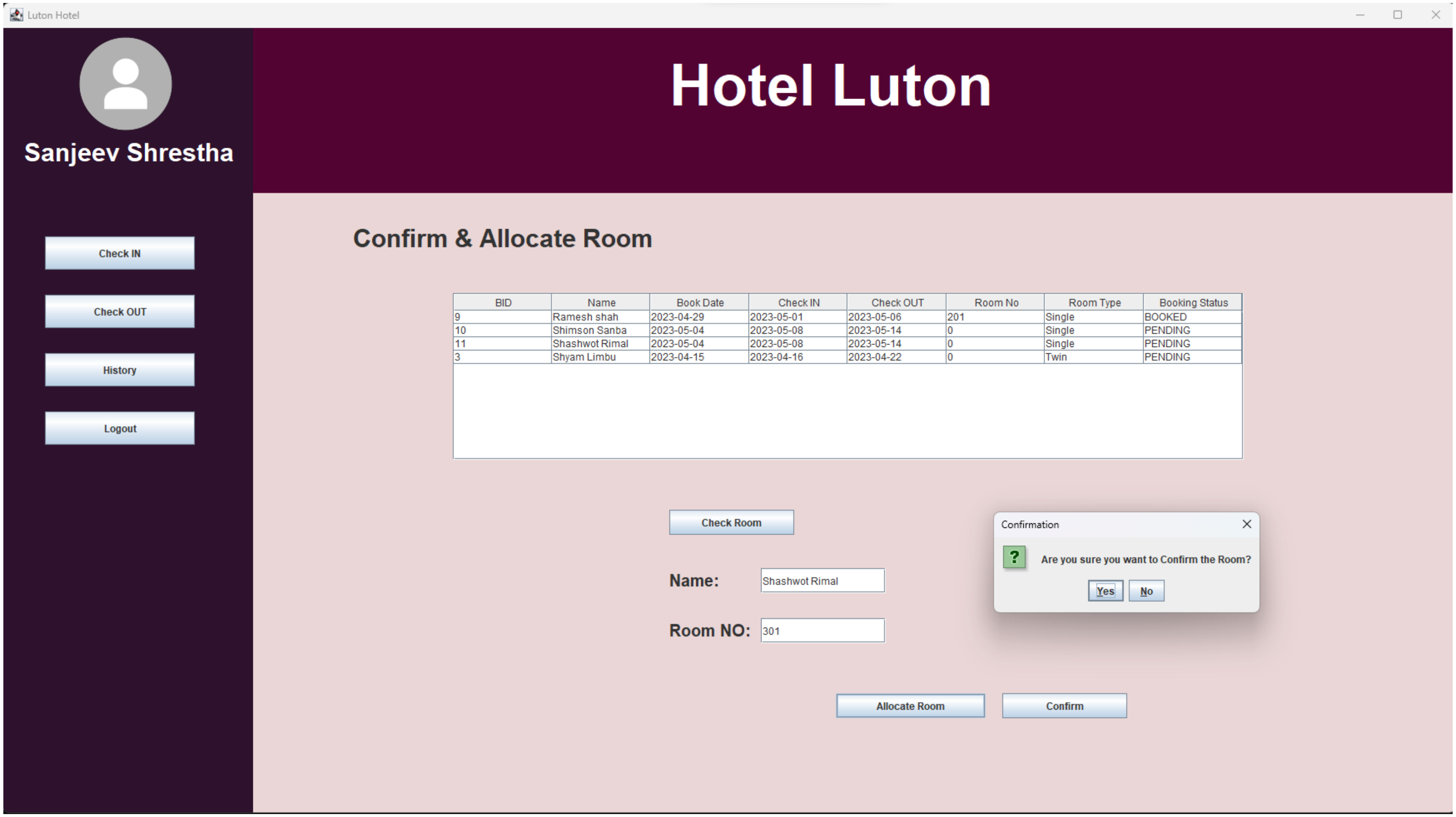
● Show

Welcome! Sanjeev Shrestha

Figure 24:- Testing (Receptionist Login)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
9-HBS	May 2, 2023	Allocate Room	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Check IN” button to view pending bookings. 5. View available room by clicking on “Check Room” button 6. Click on table row which room needs to be allocate 7. Insert available Room NO on text field 8. Click on “Allocate Room” 9. Display Confirmation message.	1. The receptionist should be Login into Dashboard after displaying “Welcome” Message 2. Receptionist should be able to click on “Check IN” button. 3. Receptionist should be able to click on “Check Room” button. 4. Receptionist should be able to view available room 5. Receptionist should be able insert room no in text field and push “Allocate Room” button 6. Display Confirmation Message	1. Login successful 2. Receptionist can click on “Check IN” 3. Receptionist can see all the necessary information 4. Receptionist has successfully able to insert and allocate proper room to the customer 5. Message shown and clicked “Yes”	Test Pass

Table 22:- Testing (Allocate Room)



Confirm & Allocate Room

BID	Name	Book Date	Check IN	Check OUT	Room No	Room Type	Booking Status
9	Ramesh shah	2023-04-29	2023-05-01	2023-05-06	201	Single	BOOKED
10	Shimson Sanba	2023-05-04	2023-05-08	2023-05-14	0	Single	PENDING
11	Shashwot Rimal	2023-05-04	2023-05-08	2023-05-14	0	Single	PENDING
3	Shyam Limbu	2023-04-15	2023-04-16	2023-04-22	0	Twin	PENDING

Name: Shashwot Rimal

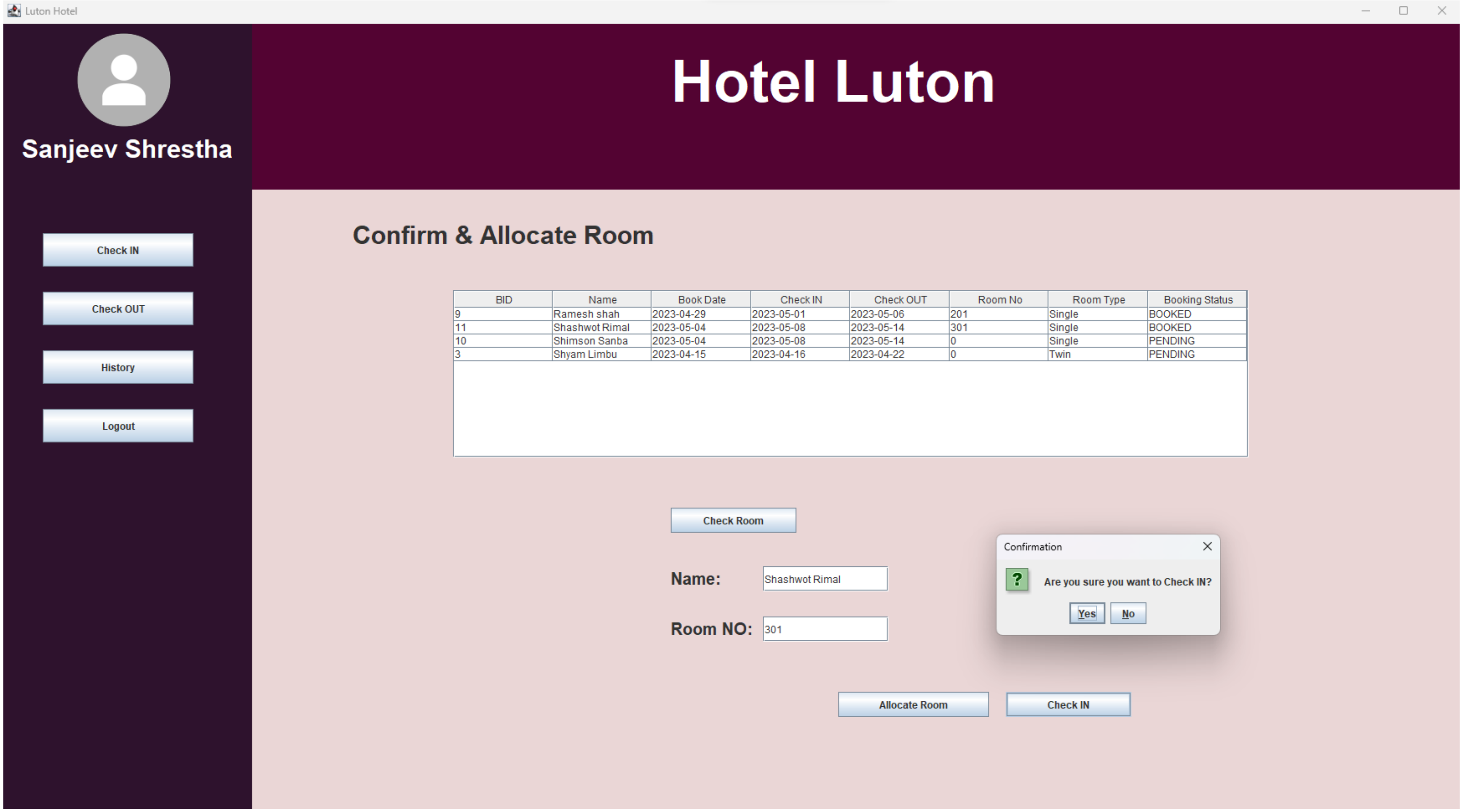
Room NO: 301

Are you sure you want to Confirm the Room?

Figure 25:- Testing (Allocate Room)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
10-HBS	May 2, 2023	Check IN	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Check IN” button to view pending/booked bookings. 5. Click on table row which booking needs to be check-in 6. Click on “Check IN” 7. Display Confirmation message.	1. The receptionist should be Login into Dashboard after displaying “Welcome” Message 2. Receptionist should be able to click on “Check IN” button. 3. Receptionist should be able to view bookings 4. Receptionist should be able to make Check IN 5. Display Confirmation Message	1. Login successful 2. Receptionist can click on “Check IN” 3. Receptionist can see all the necessary information 4. Receptionist has successfully able Check In the customer 5. Message shown and clicked “Yes”	Test Pass

Table 23:- Testing (Check IN)



Confirm & Allocate Room

BID	Name	Book Date	Check IN	Check OUT	Room No	Room Type	Booking Status
9	Ramesh shah	2023-04-29	2023-05-01	2023-05-06	201	Single	BOOKED
11	Shashwot Rimal	2023-05-04	2023-05-08	2023-05-14	301	Single	BOOKED
10	Shimson Sanba	2023-05-04	2023-05-08	2023-05-14	0	Single	PENDING
3	Shyam Limbu	2023-04-15	2023-04-16	2023-04-22	0	Twin	PENDING

Name: Shashwot Rimal

Room NO: 301

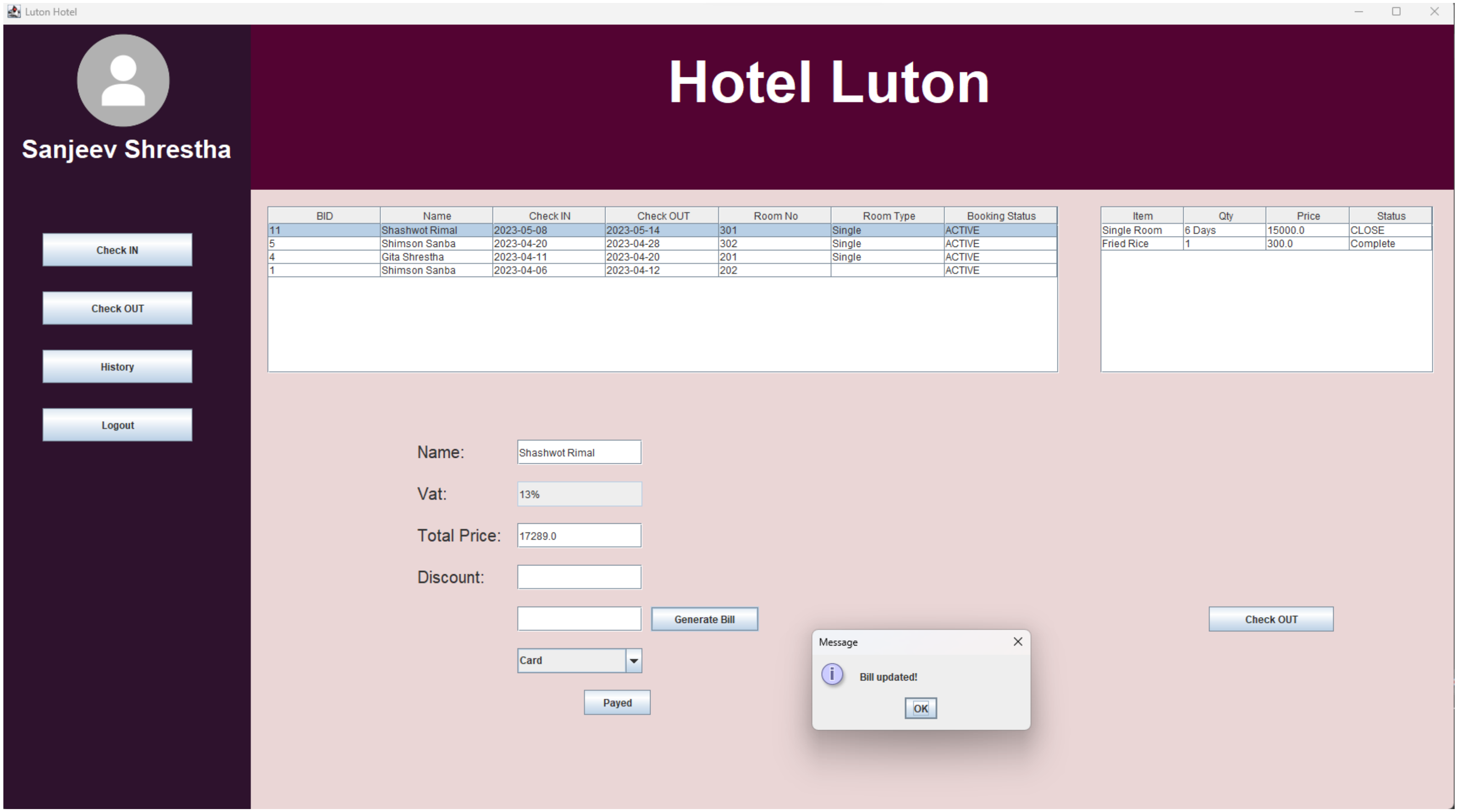
Confirmation

Are you sure you want to Check IN?

Figure 26:- Testing (Check IN)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
11-HBS	May 2, 2023	Generate Bill	1. Run application 2. Insert all the information 3. Click on the “Login” button. 4. Click on “Check OUT” button to view active bookings. 5. Click on table row which booking needs to generate bill 6. Click on “Check Generate Bill” 7. Display message.	1. The receptionist should be Login into Dashboard after displaying “Welcome” Message 2. Receptionist should be able to click on “Check OUT” button. 3. Receptionist should be able to view active bookings 4. Receptionist should be able to Generate Bill 5. Display Message	1. Login successful 2. Receptionist can click on “Check OUT” 3. Receptionist can see all the necessary information 4. Receptionist has successfully able Generate Bill 5. Message shown	Test Pass

Table 24:- Testing (Generate Bill)



BID	Name	Check IN	Check OUT	Room No	Room Type	Booking Status
11	Shashwot Rimal	2023-05-08	2023-05-14	301	Single	ACTIVE
5	Shimson Sanba	2023-04-20	2023-04-28	302	Single	ACTIVE
4	Gita Shrestha	2023-04-11	2023-04-20	201	Single	ACTIVE
1	Shimson Sanba	2023-04-06	2023-04-12	202		ACTIVE

Item	Qty	Price	Status
Single Room	6 Days	15000.0	CLOSE
Fried Rice	1	300.0	Complete

Name: Shashwot Rimal

Vat: 13%

Total Price: 17289.0

Discount:

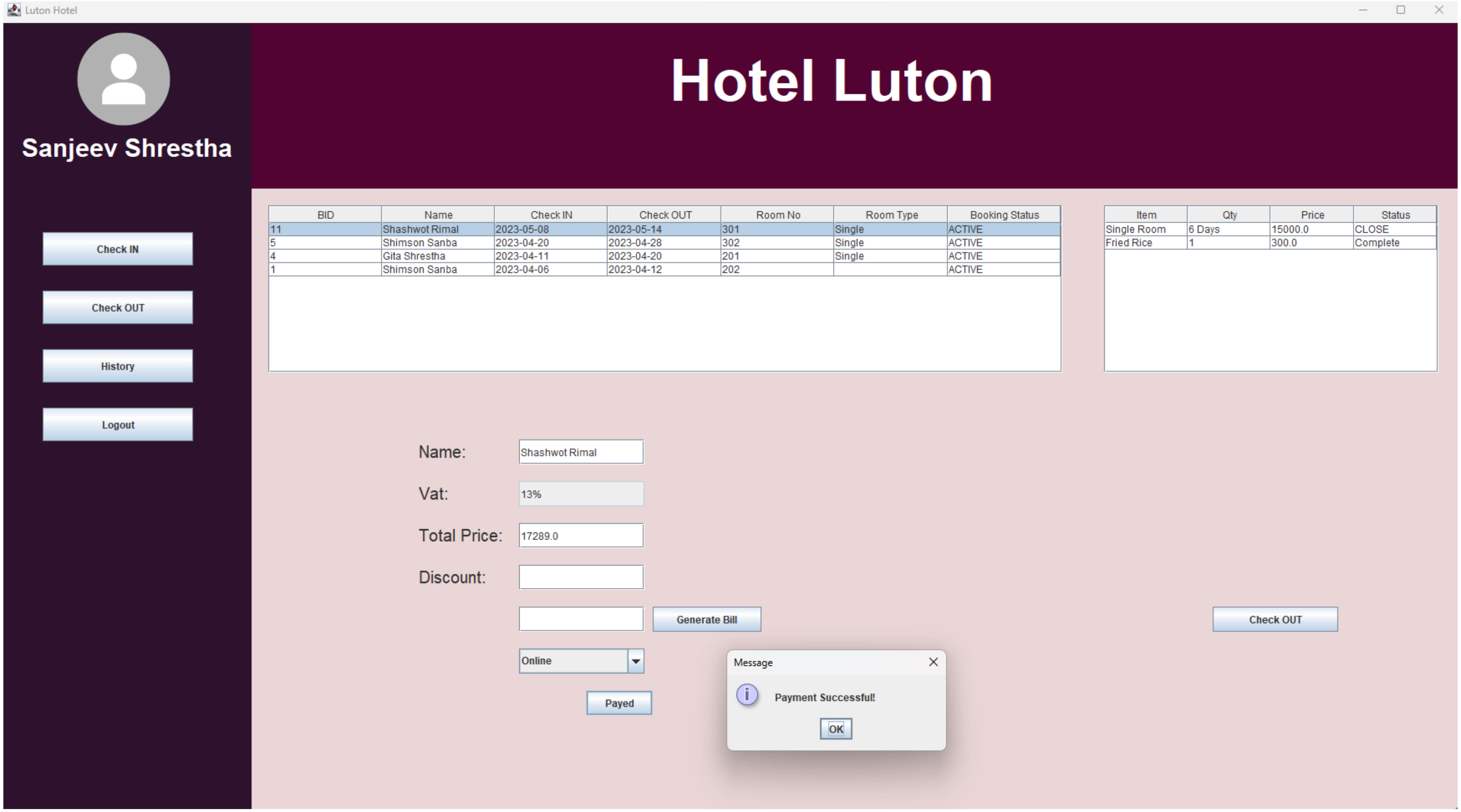
Card

Bill updated!

Figure 27:- Testing (Generate Bill)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
12-HBS	May 2, 2023	Payment	8. Run application 9. Insert all the information 10. Click on the “Login” button. 11. Click on “Check OUT” button to view active bookings. 12. Click on table row which booking needs to be pay 13. Click on “Payed” button 14. Display message.	6. The receptionist should be Login into Dashboard after displaying “Welcome” Message 7. Receptionist should be able to click on “Check OUT” button. 8. Receptionist should be able to view active bookings 9. Receptionist should be able to receive payment and proceed further by Clicking “Payed” button 10. Display Message	6. Login successful 7. Receptionist can click on “Check OUT” 8. Receptionist can see all the necessary information 9. Receptionist has successfully able take payment from the customer 10. Message shown	Test Pass

Table 25:- Testing (Payment)



BID	Name	Check IN	Check OUT	Room No	Room Type	Booking Status
11	Shashwot Rimal	2023-05-08	2023-05-14	301	Single	ACTIVE
5	Shimson Sanba	2023-04-20	2023-04-28	302	Single	ACTIVE
4	Gita Shrestha	2023-04-11	2023-04-20	201	Single	ACTIVE
1	Shimson Sanba	2023-04-06	2023-04-12	202		ACTIVE

Item	Qty	Price	Status
Single Room	6 Days	15000.0	CLOSE
Fried Rice	1	300.0	Complete

Name: Shashwot Rimal
Vat: 13%
Total Price: 17289.0
Discount:

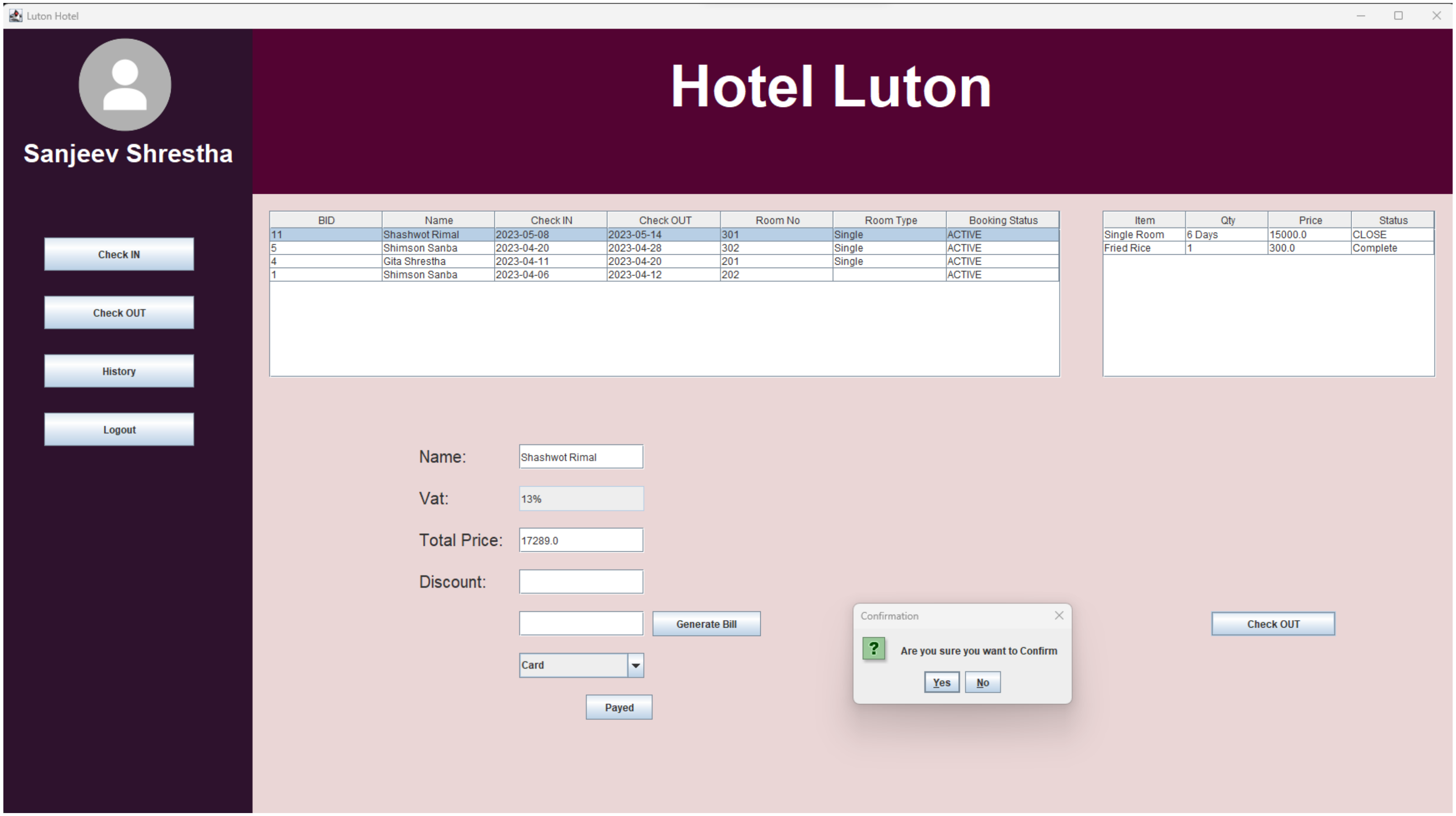
Online

Payment Successful!

Figure 28:- Testing (Payment)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
13-HBS	May 2, 2023	Check OUT	15. Run application 16. Insert all the information 17. Click on the “Login” button. 18. Click on “Check OUT” button to view active bookings. 19. Click on table row which booking needs to be check-out 20. Click on “Checkout” 21. Display Confirmation message.	11. The receptionist should be Login into Dashboard after displaying “Welcome” Message 12. Receptionist should be able to click on “Check OUT” button. 13. Receptionist should be able to view active bookings 14. Receptionist should be able to make Check OUT 15. Display Confirmation Message	11. Login successful 12. Receptionist can click on “Check OUT” 13. Receptionist can see all the necessary information 14. Receptionist has successfully able Check OUT the customer 15. Message shown and clicked “Yes”	Test Pass

Table 26:- Testing (Check OUT)



BID	Name	Check IN	Check OUT	Room No	Room Type	Booking Status
11	Shashwot Rimal	2023-05-08	2023-05-14	301	Single	ACTIVE
5	Shimson Sanba	2023-04-20	2023-04-28	302	Single	ACTIVE
4	Gita Shrestha	2023-04-11	2023-04-20	201	Single	ACTIVE
1	Shimson Sanba	2023-04-06	2023-04-12	202		ACTIVE

Item	Qty	Price	Status
Single Room	6 Days	15000.0	CLOSE
Fried Rice	1	300.0	Complete

Name: Shashwot Rimal

Vat: 13%

Total Price: 17289.0

Discount:

Card

Confirmation

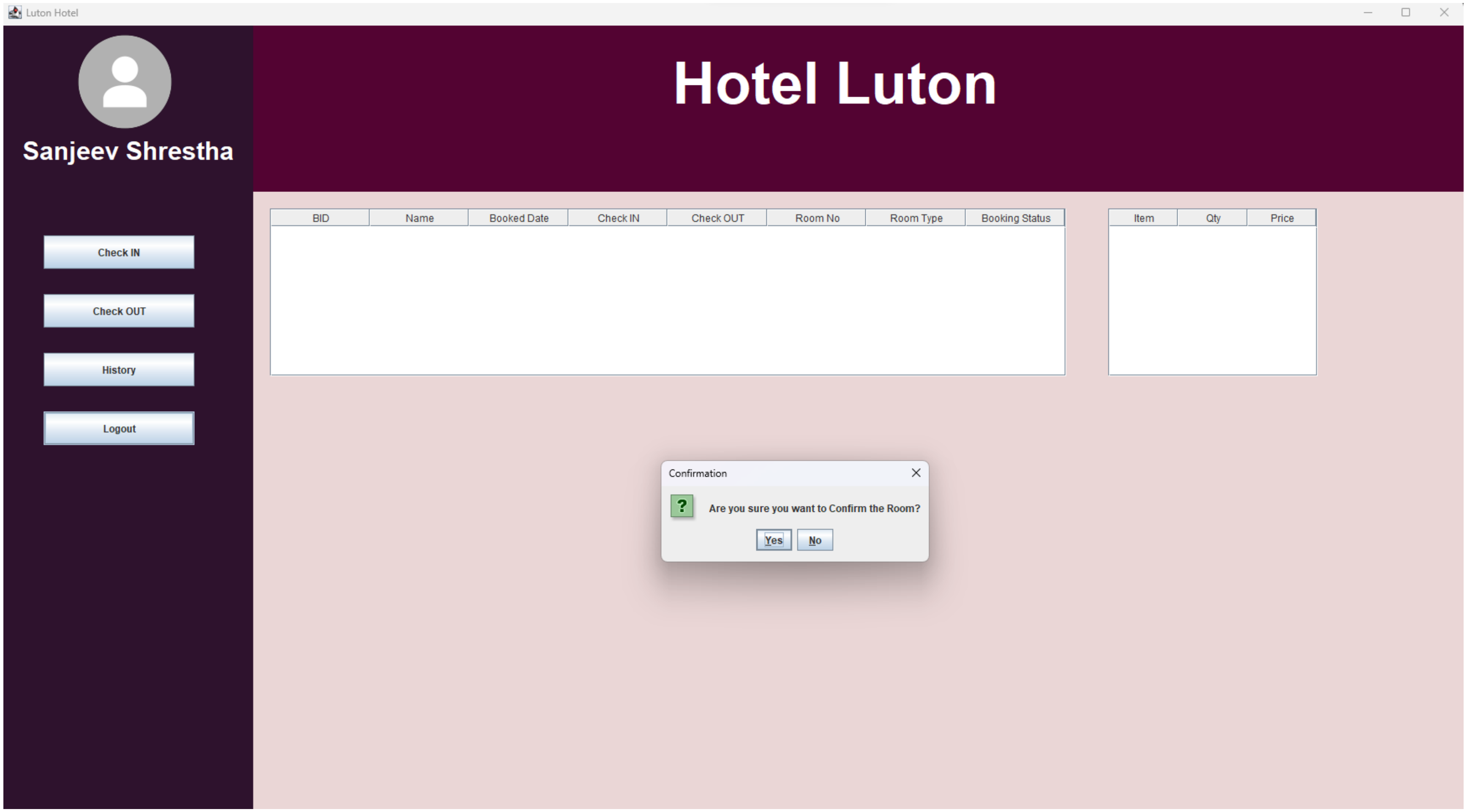
Are you sure you want to Confirm

Yes No

Figure 29:- Testing (Check OUT)

Test Case ID	Test Date	Purpose of Test	Test Steps	Expected Results	Actual Results	Action
14-HBS	May 2, 2023	Logout	6. Run the application 7. Insert all the information 8. Click on the “Login” button. 9. Click on “Logout” button 10. Display Confirmation message.	5. The receptionist should be Login into Dashboard after displaying “Welcome” Message 6. Receptionist should be able to click on “Logout” button 7. Confirmation message should pop up 8. Should be able to logout after pressing “Yes”	5. The receptionist was able to Login into system after displaying “Welcome” message. 6. Receptionist was able to click on “Logout” button 7. Confirmation was popped up 8. Successfully logout after pressing “Yes”	Test Pass

Table 27:- Testing (Receptionist Logout)



BID	Name	Booked Date	Check IN	Check OUT	Room No	Room Type	Booking Status

Item	Qty	Price

Are you sure you want to Confirm the Room?

Figure 30:- Testing (Receptionist Logout)

Discussion / Reflection / Critical Analysis

The hotel booking system project was a challenging task which required high range of programming knowledge and skills. Along the way I have encountered a lot of difficulties but after hard dedication I have overcome them and have produce a functional system that met the requirements of the assignment brief.

One of the most successful aspects of my project is that the use of Java Swing and libraries to create the user interface. This helps me to create a user-friendly interface which is easy to navigate. Additionally, the use of UML and database design help me to organize my plan and work effectively. However, some challenges regarding the amount of time required for development and testing gave me hard time to deal with. At the end week I found myself rushing, which impact on the quality of the project. Additionally, challenge to making payment system takes lots of research and debugging.

Looking back to where I have started, if I must do this project again, I will, manage my time on developing and testing to product quality on the software. I would also like to improve my exception handling part to make it more robust. Furthermore, I would like to add more features to improvise the quality of service.

As for this individual work, I have learned time management skill to perform productive work during this assignment. This project was a great opportunity to understand and learn programming languages concepts such as OOPs, database connection, as well as backend related tasks. This assignment has motivated me to learn more about practical application through various methods. I can assure that through proper planning and organization had given me great knowledge and skill on my overall quality as a software developer.

In the future, I would like to solve alike problem in different way. I will include advanced features like real-time updates, integrating external APIs and secure authentication and encryption.

Overall, this project was a valuable learning experience which has enhance my programming knowledge and skill and taught me valuable lessons on time management and project management.

Conclusion

I was tasked with creating a hotel reservation system for this project with the goal of providing an efficient and user-friendly method for making hotel reservations online. The development of a platform that enables clients to quickly search for available rooms, make reservations, and maintain their reservations. I achieved this using Java, a popular general-purpose programming language recognized for its sturdiness and adaptability. Java's object-oriented programming (OOP) features allowed me to create a system that is modular, scalable, and easy to maintain. I have used a variety of Java's libraries and frameworks for user authentication, database connectivity, and subscription management.

In conclusion, My Hotel Booking System is a comprehensive solution that aims to simplify the hotel booking process and provide a seamless experience for both guests and hotel owners. With our innovative use of Java and user-centric approach, we are confident that our solution will make a positive impact in the hospitality industry.

Appendix

VIEW (UI)

File name: Customer_Dashboard.java

```
package UI;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.text.SimpleDateFormat;
import java.time.LocalDate;
import java.util.Date;
import java.util.List;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;
import com.toedter.calendar.JDateChooser;
import DAO.ActiveBookingdao;
import DAO.BookingCRUDdao;
import DAO.CRUDdao;
import DAO.ItemJDBCdao;
import DAO.OrderJDBCdao;
import DAO.Paymentdao;
import Middleware.ActiveBooking_MW;
import Middleware.Booking_MW;
import Middleware.Customer_MW;

import Middleware.MealItem_MW;

import Middleware.Payment_MW;

import Middleware.Services_MW;

import Middleware.ViewBooking_MW;

import Middleware.ViewServices_MW;

public class Customer_Dashboard extends Dashboard_Frame implements ActionListener,MouseListener{

JLabel Name, Address, Mobile, Email, Gender, DOB, Password;

JTextField First_Name_txt, Last_Name_txt;

JTextField Address_txt, Mobile_txt, Email_txt;

JPasswordField Password_txt;

JDateChooser DOB_txt;

JButton Profile_Update_btn;

JComboBox<String> Gender_txt;

JRadioButton Radio_button;

JButton Update, Cancel,Order, Reserve_btn;

JTextField Booking_ID_txt,Total_price_txt;

JTextField Item, Quantity, SN;

JButton Booking_btn, Services_btn, Payment_btn, Profile_btn, About_Us_btn, Logout_btn,Done;

JButton Make_Booking, View_Booking;

JTable Booking_View_Table,View_Table1,Profile_View_Table,Make_Booking_View_Table,Item_Table,Meal_Item_Table,Drinks_Item_Table,Other_Services,Breakfast_Item_Table,Payment_View_Table;

JScrollPane scroll,itemscroll,Mealscroll,Drinkscroll,Otherscroll,Breakfastscroll,payment_scroll;
```

```
JLabel welcome, Booking_ID_lbl, Welcome_name ;

JDateChooser Check_IN, Check_OUT, Make_Check_IN, Make_Check_OUT;

JComboBox<String> Room_type, Payment_type;

DefaultTableModel
tableModel,Item_tableModel,Meal_Item_tableModel,Drinks_Item_tableModel,Other_
Services_tabelModel,Breakfast_Item_tableModel,Profile_tableModel;

DefaultTableModel tableModel2,paymenttableModel;

// Frame sharedFrame = Frame.getInstance();

static int ID, BID,pid;

static String NName;

public Customer_Dashboard(int id, String name) {

NName = name;

ID = id;

JLabel label = new JLabel(); //JLabel Creation

label.setIcon(new
ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\6.png")); //Sets the
image to be displayed as an icon

Dimension size = label.getPreferredSize(); //Gets the size of the image

label.setBounds(80, 0, size.height, size.width); //Sets the location of the
image

Left_panel.add(label); //Adds objects to the container

Welcome_name = new JLabel(NName);

Welcome_name.setFont(new Font("Arial", Font.CENTER_BASELINE, 30));

Welcome_name.setForeground(Color.WHITE);

Welcome_name.setBounds(25,130,250,40);

Left_panel.add(Welcome_name);

Booking_btn = new JButton("Booking");

Booking_btn.addActionListener(this);
Booking_btn.setBounds(50,250,180,40);

Left_panel.add(Booking_btn);

Services_btn = new JButton("Services");

Services_btn.setBounds(50,320,180,40);

Services_btn.addActionListener(this);

Left_panel.add(Services_btn);

Payment_btn = new JButton("Payment");

Payment_btn.setBounds(50,390,180,40);

Payment_btn.addActionListener(this);

Left_panel.add(Payment_btn);

Profile_btn = new JButton("Profile");

Profile_btn.addActionListener(this);

Profile_btn.setBounds(50,460,180,40);

Left_panel.add(Profile_btn);

// About_Us_btn = new JButton("About Us");

//// About_Us_btn.addActionListener(this);

// About_Us_btn.setBounds(50,530,180,40);

// Left_panel.add(About_Us_btn);

Logout_btn = new JButton("Logout");

Logout_btn.addActionListener(this);

Logout_btn.setBounds(50,530,180,40);

Left_panel.add(Logout_btn);

welcome = new JLabel("Hotel Luton");

welcome.setFont(new Font("Arial", Font.CENTER_BASELINE, 70));

welcome.setForeground(Color.WHITE);
welcome.setBounds(500,40,400,60);

Top_panel.add(welcome);

}

public void Booking() {

Make_Booking = new JButton("Make Booking");

Make_Booking.setBounds(10, 150, 130, 35);

Make_Booking.addActionListener(this);

Top_panel.add(Make_Booking);

View_Booking = new JButton("View Booking");

View_Booking.setBounds(150,150, 130, 35);

View_Booking.addActionListener(this);

Top_panel.add(View_Booking);

}

public void Table() {

JLabel Headline = new JLabel("Update-Cancel Your Booking");

Headline.setFont(new Font("Arial", Font.CENTER_BASELINE, 30));

Headline.setForeground(Color.BLACK);

Headline.setBounds(300, 30, 450, 30);

Center_panel.add(Headline);

JLabel Footline = new JLabel("Note: In order to change the room you have to cancel and re-book your desire room");

Footline.setFont(new Font("Arial", Font.CENTER_BASELINE, 15));

Footline.setForeground(Color.RED);

Footline.setBounds(100, 600, 650, 30);

Center_panel.add(Footline);
```

```
String[] Column = {"BID","Room NO","Name","Book Date","Check IN","Check OUT","Status"};

tableModel = new DefaultTableModel();

tableModel.setColumnIdentifiers(Column);

Booking_View_Table = new JTable();

Booking_View_Table.setModel(tableModel);

Booking_View_Table.addMouseListener(this);

Booking_View_Table.getTableHeader().setReorderingAllowed(false);

Booking_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Booking_View_Table.setFillsViewportHeight(true);

Booking_View_Table.setVisible(true);

scroll = new JScrollPane(Booking_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(280, 100, 750, 200);

Center_panel.add(scroll);

Update = new JButton("Update");

Update.setBounds(520,500,120,50);

Update.addActionListener(this);

// tableModel.fireTableDataChanged();

Center_panel.add(Update);

Cancel = new JButton("Cancel");

Cancel.setBounds(670,500,120,50);

Cancel.addActionListener(this);

Center_panel.add(Cancel);
```

```
Booking_ID_lbl = new JLabel("Booking ID");
Booking_ID_lbl.setBounds(440, 320, 80, 30);
Center_panel.add(Booking_ID_lbl);
Booking_ID_txt = new JTextField();
Booking_ID_txt.setBounds(520,320,200,30);
Center_panel.add(Booking_ID_txt);
JLabel Check_IN_lbl, Check_OUT_lbl;
Check_IN_lbl = new JLabel("Check IN");
Check_IN_lbl.setBounds(440, 370, 80, 30);
Center_panel.add(Check_IN_lbl);
Check_OUT_lbl = new JLabel("Check OUT");
Check_OUT_lbl.setBounds(440, 430, 80, 30);
Center_panel.add(Check_OUT_lbl);
Date date=new Date();
Check_IN = new JDateChooser();
Check_IN.setMinSelectableDate(date);
Check_IN.setDateFormatString("yyyy-MM-dd");
Check_IN.setFont(new Font("Verdana",Font.PLAIN,18));
Check_IN.setBounds(520,370,200,30);
Center_panel.add(Check_IN);
Date date2=new Date();
Check_OUT = new JDateChooser();
Check_OUT.setMinSelectableDate(date2);
Check_OUT.setDateFormatString("yyyy-MM-dd");
Check_OUT.setFont(new Font("Verdana",Font.PLAIN,18));
Check_OUT.setBounds(520,430,200,30);

Center_panel.add(Check_OUT);

}

public void MakeBooking() {

JLabel Room_type_lbl = new JLabel("Room Type");

Room_type_lbl.setBounds(920, 470, 80, 30);

Center_panel.add(Room_type_lbl);

String[] items = {"Single","Twin","Double"};

Room_type = new JComboBox<>(items);

Room_type.setSelectedItem("Green");

Room_type.setBounds(1000, 470, 200, 30);

Center_panel.add(Room_type);

JLabel Check_IN_lbl, Check_OUT_lbl;

Check_IN_lbl = new JLabel("Check IN");

Check_IN_lbl.setBounds(920, 530, 80, 30);

Center_panel.add(Check_IN_lbl);

Check_OUT_lbl = new JLabel("Check OUT");

Check_OUT_lbl.setBounds(920, 590, 80, 30);

Center_panel.add(Check_OUT_lbl);

Date date=new Date();

Make_Check_IN = new JDateChooser();

Make_Check_IN.setMinSelectableDate(date);

Make_Check_IN.setDateFormatString("yyyy-MM-dd");

Make_Check_IN.setFont(new Font("Verdana",Font.PLAIN,18));

Make_Check_IN.setBounds(1000,530,200,30);
```

```
Center_panel.add(Make_Check_IN);

Date date2=new Date();

Make_Check_OUT = new JDateChooser();

Make_Check_OUT.setMinSelectableDate(date2);

Make_Check_OUT.setDateFormatString("yyyy-MM-dd");

Make_Check_OUT.setFont(new Font("Verdana",Font.PLAIN,18));

Make_Check_OUT.setBounds(1000,590,200,30);

Center_panel.add(Make_Check_OUT);

Reserve_btn = new JButton("Reserve");

Reserve_btn.setBounds(1150, 650, 150, 30);

Reserve_btn.addActionListener(this);

Center_panel.add(Reserve_btn);

}

public void room() {

JLabel single_label = new JLabel(); //JLabel Creation

single_label.setIcon(new ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\Single.jpg")); //Sets the image to be displayed as an icon

Dimension size = single_label.getPreferredSize(); //Gets the size of the image

single_label.setBounds(80, 20, size.width, size.height); //Sets the location of the image

Center_panel.add(single_label);

JLabel Twin_label = new JLabel(); //JLabel Creation

Twin_label.setIcon(new ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\Twin.jpg")); //Sets the image to be displayed as an icon

Dimension size1 = Twin_label.getPreferredSize(); //Gets the size of the image
```

```
Twin_label.setBounds(540, 20, size1.width, size1.height); //Sets the location
of the image

Center_panel.add(Twin_label);

JLabel Double_label = new JLabel(); //JLabel Creation

Double_label.setIcon(new
ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\Double.jpg")); //Sets
the image to be displayed as an icon

Dimension size2 = Double_label.getPreferredSize(); //Gets the size of the
image

Double_label.setBounds(1000, 20, size2.width, size2.height); //Sets the
location of the image

Center_panel.add(Double_label);

JLabel Single_room_lbl = new JLabel("Single Room");

Single_room_lbl.setBounds(180, 340, 150, 30);

Center_panel.add(Single_room_lbl);

JLabel Twin_room_lbl = new JLabel("Twin Room");

Twin_room_lbl.setBounds(650, 340, 150, 30);

Center_panel.add(Twin_room_lbl);

JLabel Double_room_lbl = new JLabel("Double Room");

Double_room_lbl.setBounds(1120, 340, 150, 30);

Center_panel.add(Double_room_lbl);

JTextArea Single_room_btn = new JTextArea(" A/C. \n 1 Bathroom");

Single_room_btn.setBackground(new Color(234,214,214));

Single_room_btn.setBounds(175, 380, 90, 30);

Center_panel.add(Single_room_btn);

JTextArea Twin_room_btn = new JTextArea(" A/C. \n 2 Bathroom");

Twin_room_btn.setBackground(new Color(234,214,214));

Twin_room_btn.setBounds(635, 380, 90, 30);
```

```
Center_panel.add(Twin_room_btn);

JTextArea Double_room_btn = new JTextArea(" A/C. \n 2 Bathroom");

Double_room_btn.setBackground(new Color(234,214,214));

Double_room_btn.setBounds(1120, 380, 90, 30);

Center_panel.add(Double_room_btn);

String[] Columns = {"Type","Price","Status"};

tableModel2 = new DefaultTableModel();

tableModel2.setColumnIdentifiers(Columns);

Make_Booking_View_Table = new JTable();

Make_Booking_View_Table.setModel(tableModel2);

Make_Booking_View_Table.addMouseListener(this);

tableModel2.insertRow(0, new Object[] { "Single","2600","Available" });

tableModel2.insertRow(0, new Object[] { "Twin","4600","Available" });

tableModel2.insertRow(0, new Object[] { "Double","6600","Available" });

Make_Booking_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Make_Booking_View_Table.getTableHeader().setReorderingAllowed(false);

Make_Booking_View_Table.setFillsViewportHeight(true);

Make_Booking_View_Table.setVisible(true);

scroll = new JScrollPane(Make_Booking_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(80, 450, 700, 200);

Center_panel.add(scroll);

}
```

```
public void Services() {

JLabel YourOrders_lbl = new JLabel("Your Ordered Items");

YourOrders_lbl.setBounds(1020, 20, 150, 20);

Center_panel.add(YourOrders_lbl);

String[] Column2 = {"Item","Qty","Price","Status"};

Item_tableModel = new DefaultTableModel();

Item_tableModel.setColumnIdentifiers(Column2);

Item_Table = new JTable();

Item_Table.setModel(Item_tableModel);

Item_Table.getTableHeader().setReorderingAllowed(false);

Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Item_Table.setFillsViewportHeight(true);

Item_Table.setVisible(true);

itemscroll = new JScrollPane(Item_Table);

itemscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

itemscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

itemscroll.setBounds(1020, 50, 300, 200);

Center_panel.add(itemscroll);

JLabel Meal_lbl = new JLabel("Meal Items");

Meal_lbl.setBounds(20, 20, 80, 20);

Center_panel.add(Meal_lbl);

String[] Meal = {"SN","Item","Price"};

Meal_Item_tableModel = new DefaultTableModel();

Meal_Item_tableModel.setColumnIdentifiers(Meal);
```

```
Meal_Item_Table = new JTable();

Meal_Item_Table.setModel(Meal_Item_tableModel);

Meal_Item_Table.getTableHeader().setReorderingAllowed(false);

Meal_Item_Table.addMouseListener(this);

// Meal_Item_Table.set

Meal_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Meal_Item_Table.setFillsViewportHeight(true);

Meal_Item_Table.setVisible(true);

Mealscroll = new JScrollPane(Meal_Item_Table);

Mealscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Mealscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Mealscroll.setBounds(20, 50, 300, 200);

Center_panel.add(Mealscroll);

JLabel Breakfast_lbl = new JLabel("Breakfast Items");

Breakfast_lbl.setBounds(400, 20, 100, 20);

Center_panel.add(Breakfast_lbl);

String[] Breakfast = {"SN","Item","Price"};

Breakfast_Item_tableModel = new DefaultTableModel();

Breakfast_Item_tableModel.setColumnIdentifiers(Breakfast);

Breakfast_Item_Table = new JTable();

Breakfast_Item_Table.setModel(Breakfast_Item_tableModel);

Breakfast_Item_Table.getTableHeader().setReorderingAllowed(false);

Breakfast_Item_Table.addMouseListener(this);

Breakfast_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
Breakfast_Item_Table.setFillsViewportHeight(true);

Breakfast_Item_Table.setVisible(true);

Breakfastscroll = new JScrollPane(Breakfast_Item_Table);

Breakfastscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Breakfastscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Breakfastscroll.setBounds(400, 50, 300, 200);

Center_panel.add(Breakfastscroll);

JLabel Drinks_lbl = new JLabel("Drinks Items");

Drinks_lbl.setBounds(20, 320, 80, 20);

Center_panel.add(Drinks_lbl);

String[] Drinks = {"SN","Item","Price"};

Drinks_Item_tableModel = new DefaultTableModel();

Drinks_Item_tableModel.setColumnIdentifiers(Drinks);

Drinks_Item_Table = new JTable();

Drinks_Item_Table.setModel(Drinks_Item_tableModel);

Drinks_Item_Table.getTableHeader().setReorderingAllowed(false);

Drinks_Item_Table.addMouseListener(this);

Drinks_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Drinks_Item_Table.setFillsViewportHeight(true);

Drinks_Item_Table.setVisible(true);

Drinkscroll = new JScrollPane(Drinks_Item_Table);

Drinkscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Drinkscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
Drinkscroll.setBounds(20, 350, 300, 200);

Center_panel.add(Drinkscroll);

JLabel Other_lbl = new JLabel("Others Items");

Other_lbl.setBounds(400, 320, 80, 20);

Center_panel.add(Other_lbl);

String[] other = {"SN","Item","Price"};

Other_Services_tabelModel = new DefaultTableModel();

Other_Services_tabelModel.setColumnIdentifiers(other);

Other_Services = new JTable();

Other_Services.setModel( Other_Services_tabelModel);

Other_Services.getTableHeader().setReorderingAllowed(false);

Other_Services.addMouseListener(this);

Other_Services.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Other_Services.setFillsViewportHeight(true);

Other_Services.setVisible(true);

Otherscroll = new JScrollPane(Other_Services);

Otherscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Otherscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Otherscroll.setBounds(400, 350, 300, 200);

Center_panel.add(Otherscroll);

JLabel SN_lbl = new JLabel("SN");

SN_lbl.setBounds(920, 400, 80, 40);

Center_panel.add(SN_lbl);

JLabel Item_lbl = new JLabel("Item");
```

```
Item_lbl.setBounds(920, 460, 80, 40);

Center_panel.add(Item_lbl);

JLabel Quantity_lbl = new JLabel("Quantity");

Quantity_lbl.setBounds(920, 520, 80, 40);

Center_panel.add(Quantity_lbl);

SN = new JTextField();

SN.setBounds(1000, 400, 100, 30);

Center_panel.add(SN);

Item = new JTextField();

Item.setBounds(1000, 460, 100, 30);

Center_panel.add(Item);

Quantity = new JTextField();

Quantity.setBounds(1000, 520, 100, 30);

Center_panel.add(Quantity);

Order = new JButton("Order");

Order.setBounds(1050, 620, 100, 30);

Order.addActionListener(this);

Center_panel.add(Order);

}

public void payment() {

String[] Column = {"Payment ID","Booking ID","Date","Payment Mode","Total
Payment","Payment Status"};

paymenttableModel = new DefaultTableModel();

paymenttableModel.setColumnIdentifiers(Column);

Payment_View_Table = new JTable();
```

```
Payment_View_Table.setModel(paymenttableModel);

Payment_View_Table.getTableHeader().setReorderingAllowed(false);

Payment_View_Table.addMouseListener(this);

Payment_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Payment_View_Table.setFillsViewportHeight(true);

Payment_View_Table.setVisible(true);

payment_scroll = new JScrollPane(Payment_View_Table);

payment_scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

payment_scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

payment_scroll.setBounds(100, 80, 950, 200);

Center_panel.add(payment_scroll);

JLabel Total_price = new JLabel("Total Price:");

Total_price.setFont(new Font("Arial", Font.PLAIN, 20));

Total_price.setBounds(200, 400, 130, 30);

Center_panel.add(Total_price);

Total_price_txt = new JTextField();

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Total_price_txt.setBounds(320, 400, 150, 30);

Center_panel.add(Total_price_txt);

String[] items = {"Card","Cash","Online"};

Payment_type = new JComboBox<>(items);

Payment_type.setSelectedItem("Green");

Payment_type.setBounds(320, 450, 150, 30);

Center_panel.add(Payment_type);
```

```
Done = new JButton("Payed");

Done.addActionListener(this);

Done.setBounds(400, 500, 80, 30);

Center_panel.add(Done);

}

public void profile() {

JLabel Title = new JLabel("Update Profile");

Title.setFont(new Font("Arial", Font.CENTER_BASELINE, 40));

Title.setForeground(Color.BLACK);

Title.setBounds(500,40,400,60);

Center_panel.add(Title);

String[] Column = {"CID","First Name","Last
Name","Address","Mobile","Email","Gender","DOB"};

Profile_tableModel = new DefaultTableModel();

Profile_tableModel.setColumnIdentifiers(Column);

Profile_View_Table = new JTable();

Profile_View_Table.setModel(Profile_tableModel);

Profile_View_Table.getTableHeader().setReorderingAllowed(false);

Profile_View_Table.addMouseListener(this);

Profile_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Profile_View_Table.setFillsViewportHeight(true);

Profile_View_Table.setVisible(true);

scroll = new JScrollPane(Profile_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDE
D);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
scroll.setBounds(580, 120, 750, 200);

Center_panel.add(scroll);

Name = new JLabel("Name");

Name.setFont(new Font("Arial", Font.PLAIN, 20));

Name.setBounds(130,100,120,80);

Center_panel.add(Name);

Address = new JLabel("Address");

Address.setFont(new Font("Arial", Font.PLAIN, 20));

Address.setBounds(130,140,150,80);

Center_panel.add(Address);

Mobile = new JLabel("Mobile");

Mobile.setFont(new Font("Arial", Font.PLAIN, 20));

Mobile.setBounds(130,180,150,80);

Center_panel.add(Mobile);

Email = new JLabel("Email");

Email.setFont(new Font("Arial", Font.PLAIN, 20));

Email.setBounds(130,220,150,80);

Center_panel.add(Email);

Gender = new JLabel("Gender");

Gender.setFont(new Font("Arial", Font.PLAIN, 20));

Gender.setBounds(130,260,150,80);

Center_panel.add(Gender);

DOB = new JLabel("DOB");

DOB.setFont(new Font("Arial", Font.PLAIN, 20));

DOB.setBounds(130,300,150,80);
```

```
Center_panel.add(DOB);

Password = new JLabel("Password");

Password.setFont(new Font("Arial", Font.PLAIN, 20));

Password.setBounds(130,340,190,80);

Center_panel.add(Password);

First_Name_txt = new JTextField();

First_Name_txt.setBounds(280,125,120,30);

Center_panel.add(First_Name_txt);

Last_Name_txt = new JTextField();

Last_Name_txt.setBounds(430,125,120,30);

Center_panel.add(Last_Name_txt);

Address_txt = new JTextField();

Address_txt.setBounds(280,165,200,30);

Center_panel.add(Address_txt);

Mobile_txt = new JTextField();

Mobile_txt.setBounds(280,205,200,30);

Center_panel.add(Mobile_txt);

Email_txt = new JTextField();

Email_txt.setBounds(280,245,200,30);

Center_panel.add(Email_txt);

String[] items = {"Male","Female"};

Gender_txt = new JComboBox<>(items);

Gender_txt.setSelectedItem("Green");

Gender_txt.setBounds(280,285,200,30);

Center_panel.add(Gender_txt);
```

```
DOB_txt = new JDateChooser();

DOB_txt.setDateFormatString("yyyy-MM-dd");

DOB_txt.setFont(new Font("Verdana",Font.PLAIN,18));

DOB_txt.setBounds(280,325,200,30);

Center_panel.add(DOB_txt);

Password_txt = new JPasswordField();

Password_txt.setBounds(280,365,200,30);

Center_panel.add(Password_txt);

Radio_button = new JRadioButton("Show");

Radio_button.setBounds(280, 400, 80, 25);

Radio_button.setFont(new Font("Arial", Font.PLAIN, 20));

Center_panel.add(Radio_button);

Radio_button.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {

// show password chars

if (Radio_button.isSelected()) {

Password_txt.setEchoChar((char) 0);

}

// hide password chars

else {

Password_txt.setEchoChar('.');

}

}

});
```

```
Profile_Update_btn = new JButton("Update");

Profile_Update_btn.addActionListener(this);

Profile_Update_btn.setBounds(320,500,120,50);

Center_panel.add(Profile_Update_btn);

}

public void Logout() {

sharedFrame.switchToLogin();

sharedFrame.repaint();

sharedFrame.revalidate();

}

public void mouseClicked(MouseEvent e) {

if(e.getSource()==Make_Booking_View_Table) {

try {

// get the index of the selected row

int rows = Make_Booking_View_Table.getSelectedRow();

// get the table model

TableModel model1 = Make_Booking_View_Table.getModel();

// set the values of the text fields to the values from the selected row

String value = (String) model1.getValueAt(rows, 0);

Room_type.setSelectedItem(value);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Booking_View_Table) {
```

```
try {

// get the index of the selected row

int rows = Booking_View_Table.getSelectedRow();

// get the table model

TableModel model = Booking_View_Table.getModel();

int id = (int) model.getValueAt(rows, 0);

Booking_ID_txt.setText(Integer.toString(id));

// set the values of the text fields to the values from the selected row

// String check_in = model1.getValueAt(rows, 0).toString();

Date date = new SimpleDateFormat("yyyy-MM-
dd").parse((String)model.getValueAt(rows, 4));

Check_IN.setDate(date);

Date date2 = new SimpleDateFormat("yyyy-MM-
dd").parse((String)model.getValueAt(rows, 5));

Check_OUT.setDate(date2);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Meal_Item_Table) {

try {

// get the index of the selected row

int rows = Meal_Item_Table.getSelectedRow();

// get the table model

TableModel model = Meal_Item_Table.getModel();

// set the values of the text fields to the values from the selected row
```

```
int sn = (int) model.getValueAt(rows, 0);

SN.setText(Integer.toString(sn));

String item = (String) model.getValueAt(rows, 1);

Item.setText(item);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Breakfast_Item_Table) {

try {

// get the index of the selected row

int rows = Breakfast_Item_Table.getSelectedRow();

// get the table model

TableModel model = Breakfast_Item_Table.getModel();

// set the values of the text fields to the values from the selected row

int sn = (int) model.getValueAt(rows, 0);

SN.setText(Integer.toString(sn));

String item = (String) model.getValueAt(rows, 1);

Item.setText(item);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Drinks_Item_Table) {

try {
```

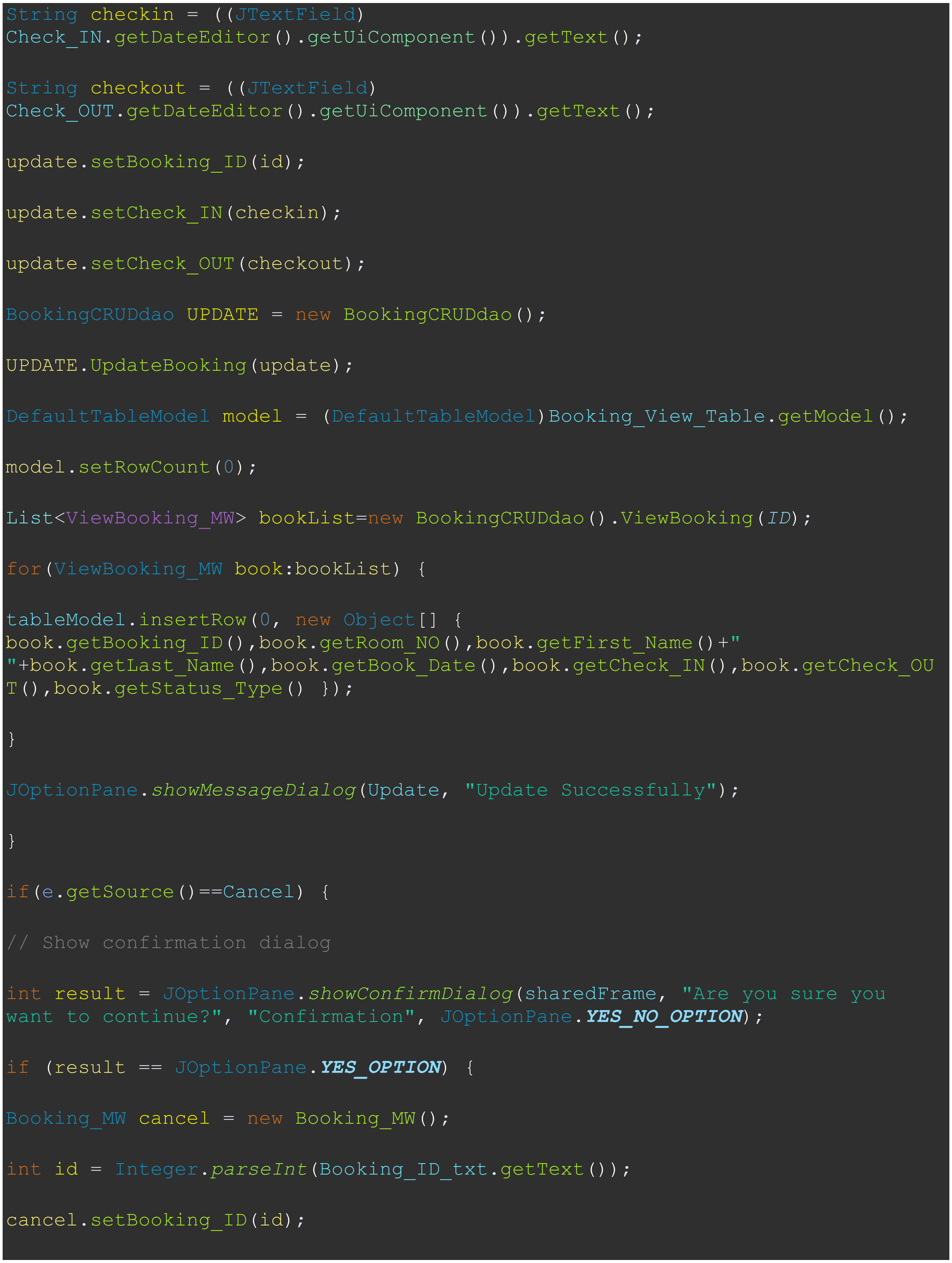
```
// get the index of the selected row

int rows = Drinks_Item_Table.getSelectedRow();

// get the table model

TableModel model = Drinks_Item_Table.getModel();

// set the values of the text fields to the values from the selected row

int sn = (int) model.getValueAt(rows, 0);

SN.setText(Integer.toString(sn));

String item = (String) model.getValueAt(rows, 1);

Item.setText(item);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Other_Services) {

try {

// get the index of the selected row

int rows = Other_Services.getSelectedRow();

// get the table model

TableModel model = Other_Services.getModel();

// set the values of the text fields to the values from the selected row

int sn = (int) model.getValueAt(rows, 0);

SN.setText(Integer.toString(sn));

String item = (String) model.getValueAt(rows, 1);

Item.setText(item);

} catch (Exception ex) {
```

```
System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Profile_View_Table) {

try {

// get the index of the selected row

int rows = Profile_View_Table.getSelectedRow();

// get the table model

TableModel model = Profile_View_Table.getModel();

// set the values of the text fields to the values from the selected row

String first_name = (String) model.getValueAt(rows, 1);

String last_name = (String) model.getValueAt(rows, 2);

String address = (String) model.getValueAt(rows, 3);

String mobile = (String) model.getValueAt(rows, 4);

String email = (String) model.getValueAt(rows, 5);

// String gender = (String) model.getValueAt(rows, 6);

Date dob = new SimpleDateFormat("yyyy-MM-dd").parse((String)
model.getValueAt(rows, 7));

First_Name_txt.setText(first_name);

Last_Name_txt.setText(last_name);

Address_txt.setText(address);

Mobile_txt.setText(mobile);

Email_txt.setText(email);

DOB_txt.setDate(dob);

} catch (Exception ex) {
```

```
System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Payment_View_Table) {

try {

// get the index of the selected row

int rows = Payment_View_Table.getSelectedRow();

// get the table model

TableModel model = Payment_View_Table.getModel();

// set the values of the text fields to the values from the selected row

int payid = (int) model.getValueAt(rows, 1);

pid=payid;

double price = (double) model.getValueAt(rows, 4);

Total_price_txt.setText(Double.toString(price));

}

catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

}

// override the other methods of the MouseListener interface

@Override

public void mousePressed(MouseEvent e) {

}

@Override
```

```
public void mouseReleased(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(e.getSource()==Booking_btn) {

Center_panel.removeAll();

Center_panel.repaint();

Top_panel.repaint();

Top_panel.revalidate();

Center_panel.revalidate();

Left_panel.repaint();

Left_panel.revalidate();

room();

MakeBooking();

Booking();

}

if(e.getSource()==View_Booking){

Center_panel.removeAll();
Center_panel.repaint();

Center_panel.revalidate();

Table();

}

if(e.getSource()==Make_Booking){

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

room();

MakeBooking();

}

if(e.getSource()==Profile_btn){

Top_panel.removeAll();

Top_panel.revalidate();

Top_panel.repaint();

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

profile();

Top_panel.add(welcome);

}

if(e.getSource()==Reserve_btn) {

Booking_MW book = new Booking_MW();

LocalDate currentDate = LocalDate.now();
```

```
String checkin = ((JTextField)
Make_Check_IN.getDateEditor().getUiComponent()).getText();

String checkout = ((JTextField)
Make_Check_OUT.getDateEditor().getUiComponent()).getText();

String room = (String) Room_type.getSelectedItem();

book.setCustomer_ID(ID);

book.setRoom_Type(room);

book.setBook_Date(currentDate.toString());

book.setCheck_IN(checkin);

book.setCheck_OUT(checkout);

book.setBooking_Status_NO(2);

book.setHotel_ID(1);

BookingCRUDdao booking = new BookingCRUDdao();

booking.MakeBooking(book);

JOptionPane.showMessageDialog(Reserve_btn, "Booked");

}

if(e.getSource()==View_Booking) {

List<ViewBooking_MW> bookList=new BookingCRUDdao().ViewBooking(ID);

for(ViewBooking_MW book:bookList) {

tableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getRoom_NO(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),book.getCheck_IN(),book.getCheck_OU
T(),book.getStatus_Type() });

}

}

if(e.getSource()==Update) {

Booking_MW update = new Booking_MW();

int id = Integer.parseInt(Booking_ID_txt.getText());
```



```
String checkin = ((JTextField)
Check_IN.getDateEditor().getUiComponent()).getText();

String checkout = ((JTextField)
Check_OUT.getDateEditor().getUiComponent()).getText();

update.setBooking_ID(id);

update.setCheck_IN(checkin);

update.setCheck_OUT(checkout);

BookingCRUDdao UPDATE = new BookingCRUDdao();

UPDATE.UpdateBooking(update);

DefaultTableModel model = (DefaultTableModel)Booking_View_Table.getModel();

model.setRowCount(0);

List<ViewBooking_MW> bookList=new BookingCRUDdao().ViewBooking(ID);

for(ViewBooking_MW book:bookList) {

tableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getRoom_NO(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),book.getCheck_IN(),book.getCheck_OU
T(),book.getStatus_Type() });

}

JOptionPane.showMessageDialog(Update, "Update Successfully");

}

if(e.getSource()==Cancel) {

// Show confirmation dialog

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to continue?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

Booking_MW cancel = new Booking_MW();

int id = Integer.parseInt(Booking_ID_txt.getText());

cancel.setBooking_ID(id);
BookingCRUDdao CANCEL = new BookingCRUDdao();

CANCEL.CancelBooking(cancel);

tableModel.fireTableDataChanged();

DefaultTableModel model = (DefaultTableModel)Booking_View_Table.getModel();

model.setRowCount(0);

List<ViewBooking_MW> bookList=new BookingCRUDdao().ViewBooking(ID);

for(ViewBooking_MW book:bookList) {

tableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getRoom_NO(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),book.getCheck_IN(),book.getCheck_OUT(),book.getStatus_Type() });

}

// JOptionPane.showConfirmDialog(Cancel, "Do you really want to calcel");

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

if(e.getSource()==Services_btn) {

Top_panel.removeAll();

Top_panel.revalidate();
Top_panel.repaint();

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

Top_panel.add(welcome);

Services();

ActiveBooking_MW active = new ActiveBooking_MW();

active.setCid(ID);

ActiveBookingdao act = new ActiveBookingdao();

act.ActiveBooking(active);

BID = active.getBid();

List<ViewServices_MW> itemList=new ItemJDBCdao().Viewitem(BID);

for(ViewServices_MW item:itemList) {

Item_tableModel.insertRow(0, new Object[] {
item.getItem(),item.getQuantity(),item.getPrice(),item.getService_Status()});

}

List<MealItem_MW> MealList=new ItemJDBCdao().Mealitem();

for(MealItem_MW Meal:MealList) {

Meal_Item_tableModel.insertRow(0, new Object[] {
Meal.getItem_ID(),Meal.getItem(),Meal.getPrice()});

}

List<MealItem_MW> BreakfastList=new ItemJDBCdao().Breakfastitem();

for(MealItem_MW breakfast:BreakfastList) {

Breakfast_Item_tableModel.insertRow(0, new Object[] {
breakfast.getItem_ID(),breakfast.getItem(),breakfast.getPrice()});

}
```

```
List<MealItem_MW> DrinksList=new ItemJDBCdao().Drinkitem();

for(MealItem_MW drink:DrinksList) {

Drinks_Item_tableModel.insertRow(0, new Object[] {
drink.getItem_ID(),drink.getItem(),drink.getPrice()});

}

List<MealItem_MW> OtherList=new ItemJDBCdao().Otheritem();

for(MealItem_MW other:OtherList) {

Other_Services_tabelModel.insertRow(0, new Object[] {
other.getItem_ID(),other.getItem(),other.getPrice()});

}

}

if(e.getSource()==Order) {

Services_MW service = new Services_MW();

service.setBooking_ID(BID);

int itemid = Integer.parseInt(SN.getText());

service.setItem_ID(itemid);

LocalDate currentDate = LocalDate.now();

service.setService_Date(currentDate.toString());

int quantity = Integer.parseInt(Quantity.getText());

service.setQuantity(quantity);

OrderJDBCdao order = new OrderJDBCdao();

order.MakeOrder(service);

DefaultTableModel model = (DefaultTableModel)Item_Table.getModel();

model.setRowCount(0);

List<ViewServices_MW> itemList=new ItemJDBCdao().Viewitem(BID);

for(ViewServices_MW item:itemList) {
```

```
Item_tableModel.insertRow(0, new Object[] {
item.getItem(),item.getQuantity(),item.getPrice(),item.getService_Status()});

}

JOptionPane.showMessageDialog(Order, "Order Successfull!");

}

if(e.getSource()==Payment_btn) {

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

payment();

Payment_MW pay = new Payment_MW();

ActiveBookingdao active = new ActiveBookingdao();

ActiveBooking_MW Activebooking = new ActiveBooking_MW();

Activebooking.setCid(ID);

active.ActiveBooking(Activebooking);

pay.setBooking_ID(Activebooking.getBid());

System.out.println(Activebooking.getBid());

// Paymentdao paying = new Paymentdao();

// paying.ViewPayment(pay);

List<Payment_MW> List=new Paymentdao().ViewPayment(pay);

for(Payment_MW paylist:List) {

paymenttableModel.insertRow(0, new Object[] {
paylist.getPayment_ID(),paylist.getBooking_ID(),paylist.getPayment_Date(),pay
list.getPayment_Mode(),paylist.getTotal_Payment(),paylist.getPayment_Status()
});

}

}
```

```
if(e.getSource()==Done) {

Payment_MW pay = new Payment_MW();

String type = (String) Payment_type.getSelectedItem();

pay.setPayment_Mode(type);

pay.setPayment_ID(pid);

// System.out.println(pid);

Paymentdao bill = new Paymentdao();

bill.UpdateBill(pay);

JOptionPane.showMessageDialog(Done, "Payment Successful!");

}

if(e.getSource()==Profile_btn) {

List<Customer_MW> List=new CRUDdao().Viewprofile(ID);

for(Customer_MW customer:List) {

Profile_tableModel.insertRow(0, new Object[] { customer.getCustomer_ID(),customer.getFirst_name(),customer.getLast_name(),customer.getAddress(),customer.getMobile(),customer.getEmail(),customer.getGender(),customer.getDOB(),customer.getPassword()});

}

}

if(e.getSource()==Profile_Update_btn) {

String first_name = First_Name_txt.getText();

String last_name = Last_Name_txt.getText();

String address = Address_txt.getText();

String mobile = Mobile_txt.getText();

String email = Email_txt.getText();

String gender = Gender_txt.getSelectedItem().toString();
String dob = ((JTextField)
DOB_txt.getDateEditor().getUiComponent()).getText();

@SuppressWarnings("deprecation")

String password = Password_txt.~~getText~~();

Customer_MW cust = new Customer_MW();

cust.setFirst_name(first_name);

cust.setLast_name(last_name);

cust.setAddress(address);

cust.setMobile(mobile);

cust.setEmail(email);

cust.setGender(gender);

cust.setDOB(dob);

cust.setPassword(password);

cust.setCustomer_ID(ID);

CRUDdao reg = new CRUDdao();

reg.CustomerUpdate(cust);

JOptionPane.showMessageDialog(Profile_Update_btn, "Update Successfull!");

DefaultTableModel model = (DefaultTableModel)Profile_View_Table.getModel();

model.setRowCount(0);

List<Customer_MW> List=new CRUDdao().Viewprofile(ID);

for(Customer_MW customer:List) {

Profile_tableModel.insertRow(0, new Object[] {
customer.getCustomer_ID(),customer.getFirst_name(),customer.getLast_name(),cu
stomer.getAddress(),customer.getMobile(),customer.getEmail(),customer.getGend
er(),customer.getDOB(),customer.getPassword()});

}

}
```

```
if(e.getSource()==Logout_btn) {

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Confirm the Room?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

Logout();

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

}

}
```

File name: Receptionist_Dashboard.java

```
package UI;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.temporal.ChronoUnit;

import java.util.Date;

import java.util.List;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;
import javax.swing.Timer;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;

import DAO.ActiveBookingdao;

import DAO.BookingCRUDdao;

import DAO.ItemJDBCdao;

import DAO.Paymentdao;

import DAO.RoomJDBCdao;

import Middleware.Amounts_MW;

import Middleware.Booking_MW;

import Middleware.Payment_MW;

import Middleware.ReceptionistCheckIN_MW;

import Middleware.ReceptionistCheckOUT_MW;

import Middleware.Room_MW;

import Middleware.ViewServices_MW;

public class Receptionist_Dashboard extends Dashboard_Frame implements
ActionListener, MouseListener {

JButton Check_IN_btn, Check_OUT_btn, History_btn,Logout_btn;

JButton Done;

JButton Conform_Check_OUT_btn;

JLabel welcome, Welcome_name;

JTable Check_IN_View_Table,
Check_OUT_View_Table,Check_OUT_Item_Table,Room_Check,
History_View_Table,History_Item_Table;

JScrollPane scroll,scroll2;

JButton Room_Available;
```

```
JComboBox<String> Payment_type;

JButton Confirm_btn, AllocateRoom_btn, generate_bill;

DefaultTableModel tableModel,checkOuttableModel,checkOut_item_tableModel,historytableModel;

JTextField Room_NO_txt, Name_txt;

JTextField Checkout_Name, Discount_txt, Vat_txt, Total_price_txt, Final_price;

Booking_MW Confirm = new Booking_MW();

Amounts_MW amount = new Amounts_MW();

// static

static int bid,pid;

// static

public Receptionist_Dashboard(String Name) {

JLabel label = new JLabel(); //JLabel Creation

label.setIcon(new ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\6.png")); //Sets the image to be displayed as an icon

Dimension size = label.getPreferredSize(); //Gets the size of the image

label.setBounds(80, 0, size.height, size.width); //Sets the location of the image

Left_panel.add(label); //Adds objects to the container

Welcome_name = new JLabel(Name);

Welcome_name.setFont(new Font("Arial", Font.CENTER_BASELINE, 30));

Welcome_name.setForeground(Color.WHITE);

Welcome_name.setBounds(25,130,280,40);

Left_panel.add(Welcome_name);

Check_IN_btn = new JButton("Check IN");

Check_IN_btn.addActionListener(this);
Check_IN_btn.setBounds(50,250,180,40);

Left_panel.add(Check_IN_btn);

Check_OUT_btn = new JButton("Check OUT");

Check_OUT_btn.addActionListener(this);

Check_OUT_btn.setBounds(50,320,180,40);

Left_panel.add(Check_OUT_btn);

History_btn = new JButton("History");

History_btn.setBounds(50,390,180,40);

History_btn.addActionListener(this);

Left_panel.add(History_btn);

// About_btn = new JButton("About");

// Profile_btn.addActionListener(this);

// Profile_btn.setBounds(50,460,180,40);

// Left_panel.add(Profile_btn);

Logout_btn = new JButton("Logout");

Logout_btn.addActionListener(this);

Logout_btn.setBounds(50,460,180,40);

Left_panel.add(Logout_btn);

welcome = new JLabel("Hotel Luton");

welcome.setFont(new Font("Arial", Font.CENTER_BASELINE, 70));

welcome.setForeground(Color.WHITE);

welcome.setBounds(500,40,400,60);

Top_panel.add(welcome);

}

public void Logout() {
```

```
sharedFrame.switchToLogin();

sharedFrame.repaint();

sharedFrame.revalidate();

}

public void Check_IN() {

JLabel Room_NO, Name, Heading;

// Heading = new JLabel("Check IN after conforming and allocating available room as per customer requested");

Heading = new JLabel("Confirm & Allocate Room");

Heading.setFont(new Font("Arial", Font.CENTER_BASELINE, 30));

Heading.setBounds(120, 40, 1000, 30);

Center_panel.add(Heading);

String[] Column = {"BID","Name","Book Date","Check IN","Check OUT","Room No","Room Type","Booking Status"};

tableModel = new DefaultTableModel();

tableModel.setColumnIdentifiers(Column);

Check_IN_View_Table = new JTable();

Check_IN_View_Table.setModel(tableModel);

Check_IN_View_Table.getTableHeader().setReorderingAllowed(false);

Check_IN_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Check_IN_View_Table.setFillsViewportHeight(true);

Check_IN_View_Table.setVisible(true);

Check_IN_View_Table.addMouseListener(this);

scroll = new JScrollPane(Check_IN_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(240, 120, 950, 200);

Center_panel.add(scroll);

Room_Available = new JButton("Check Room");

Room_Available.setBounds(500, 380, 150, 30);

Room_Available.addActionListener(this);

Center_panel.add(Room_Available);

Name = new JLabel("Name:");

Name.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));

Name.setBounds(500, 450, 150, 30);

Center_panel.add(Name);

Name_txt = new JTextField();

Name_txt.setBounds(610, 450, 150, 30);

Center_panel.add(Name_txt);

Room_NO = new JLabel("Room NO:");

Room_NO.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));

Room_NO.setBounds(500, 510, 150, 30);

Center_panel.add(Room_NO);

Room_NO_txt = new JTextField();

Room_NO_txt.setBounds(610, 510, 150, 30);

Center_panel.add(Room_NO_txt);

AllocateRoom_btn = new JButton("Allocate Room");

AllocateRoom_btn.setBounds(700,600,180,30);

AllocateRoom_btn.addActionListener(this);

Center_panel.add(AllocateRoom_btn);
```

```
Confirm_btn = new JButton("Check IN");

Confirm_btn.setBounds(900,600,150,30);

Confirm_btn.addActionListener(this);

Center_panel.add(Confirm_btn);

timer.start();

}

public void Check_OUT() {

JLabel Name, Discount, Vat, Total_price ;

String[] Column = {"BID","Name","Check IN","Check OUT","Room No","Room
Type","Booking Status"};

checkOuttableModel = new DefaultTableModel();

checkOuttableModel.setColumnIdentifiers(Column);

Check_OUT_View_Table = new JTable();

Check_OUT_View_Table.setModel(checkOuttableModel);

Check_OUT_View_Table.getTableHeader().setReorderingAllowed(false);

Check_OUT_View_Table.addMouseListener(this);

Check_OUT_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Check_OUT_View_Table.setFillsViewportHeight(true);

Check_OUT_View_Table.setVisible(true);

scroll = new JScrollPane(Check_OUT_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDE
D);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(20, 20, 950, 200);

Center_panel.add(scroll);

String[] Column2 = {"Item","Qty","Price","Status"};
```

```
checkOut_item_tableModel = new DefaultTableModel();

checkOut_item_tableModel.setColumnIdentifiers(Column2);

Check_OUT_Item_Table = new JTable();

Check_OUT_Item_Table.setModel(checkOut_item_tableModel);

Check_OUT_Item_Table.getTableHeader().setReorderingAllowed(false);

Check_OUT_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Check_OUT_Item_Table.setFillsViewportHeight(true);

Check_OUT_Item_Table.setVisible(true);

scroll2 = new JScrollPane(Check_OUT_Item_Table);

scroll2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEED
ED);

scroll2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll2.setBounds(1020, 20, 400, 200);

Center_panel.add(scroll2);

Name = new JLabel("Name:");

Name.setFont(new Font("Arial", Font.PLAIN, 20));

Name.setBounds(200, 300, 80, 30);

Center_panel.add(Name);

Vat = new JLabel("Vat:");

Vat.setFont(new Font("Arial", Font.PLAIN, 20));

Vat.setBounds(200, 350, 80, 30);

Center_panel.add(Vat);

Total_price = new JLabel("Total Price:");

Total_price.setFont(new Font("Arial", Font.PLAIN, 20));

Total_price.setBounds(200, 400, 130, 30);
```

```
Center_panel.add(Total_price);

Discount = new JLabel("Discount:");

Discount.setFont(new Font("Arial", Font.PLAIN, 20));

Discount.setBounds(200, 450, 100, 30);

Center_panel.add(Discount);

Checkout_Name = new JTextField();

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Checkout_Name.setBounds(320, 300, 150, 30);

Center_panel.add(Checkout_Name);

Vat_txt = new JTextField("13%");

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Vat_txt.setEditable(false);

Vat_txt.setBounds(320, 350, 150, 30);

Center_panel.add(Vat_txt);

Total_price_txt = new JTextField();

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Total_price_txt.setBounds(320, 400, 150, 30);

Center_panel.add(Total_price_txt);

Discount_txt = new JTextField();

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Discount_txt.setBounds(320, 450, 150, 30);

Center_panel.add(Discount_txt);

Final_price = new JTextField();

// Name_txt.setFont(new Font("Arial", Font.PLAIN, 20));

Final_price.setBounds(320, 500, 150, 30);
```

```
Center_panel.add(Final_price);

String[] items = {"Card","Cash","Online"};

Payment_type = new JComboBox<>(items);

Payment_type.setSelectedItem("Green");

Payment_type.setBounds(320, 550, 150, 30);

Center_panel.add(Payment_type);

generate_bill = new JButton("Generate Bill");

generate_bill.addActionListener(this);

generate_bill.setBounds(480, 500, 130, 30);

Center_panel.add(generate_bill);

Done = new JButton("Payed");

Done.addActionListener(this);

Done.setBounds(400, 600, 80, 30);

Center_panel.add(Done);

Conform_Check_OUT_btn = new JButton("Check OUT");

Conform_Check_OUT_btn.addActionListener(this);

Conform_Check_OUT_btn.setBounds(1150,500,150,30);

Center_panel.add(Conform_Check_OUT_btn);

}

public void History() {

String[] Column = {"BID","Name","Booked Date","Check IN","Check OUT","Room
No","Room Type","Booking Status"};

// DefaultTableModel tableModel;

historytableModel = new DefaultTableModel();

historytableModel.setColumnIdentifiers(Column);
```

```
History_View_Table = new JTable();

History_View_Table.setModel(historytableModel);

History_View_Table.getTableHeader().setReorderingAllowed(false);

History_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

History_View_Table.setFillsViewportHeight(true);

History_View_Table.setVisible(true);

scroll = new JScrollPane(History_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(20, 20, 950, 200);

Center_panel.add(scroll);

String[] Column2 = {"Item","Qty","Price"};

DefaultTableModel tableModel2;

tableModel2 = new DefaultTableModel();

tableModel2.setColumnIdentifiers(Column2);

History_Item_Table = new JTable();

History_Item_Table.setModel(tableModel2);

History_Item_Table.getTableHeader().setReorderingAllowed(false);

History_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

History_Item_Table.setFillsViewportHeight(true);

History_Item_Table.setVisible(true);

scroll2 = new JScrollPane(History_Item_Table);

scroll2.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll2.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
```

```java
scroll2.setBounds(1020, 20, 250, 200);

Center_panel.add(scroll2);

}

public void Room_Check(JFrame frame) {

frame = new JFrame();

frame.setTitle("Luton Hotel");

frame.setSize(400,200);

// getContentPane().setBackground(Color.LIGHT_GRAY);

frame.setLayout(null);

frame.setVisible(true);

frame.setPreferredSize(new Dimension(400, 300));

// Pack the components in the JFrame

frame.pack();

// Get the screen size

Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();

// Calculate the coordinates to center the JFrame

int x = (screenSize.width - frame.getWidth()) / 2;

int y = (screenSize.height - frame.getHeight()) / 2;

// Set the location of the JFrame to the center of the screen

frame.setLocation(x, y);

// frame.setDefaultCloseOperation(frame.EXIT_ON_CLOSE);

String[] Column = {"Room No","Room Type","Room Status"};

DefaultTableModel tableModel;

tableModel = new DefaultTableModel();

tableModel.setColumnIdentifiers(Column);
```

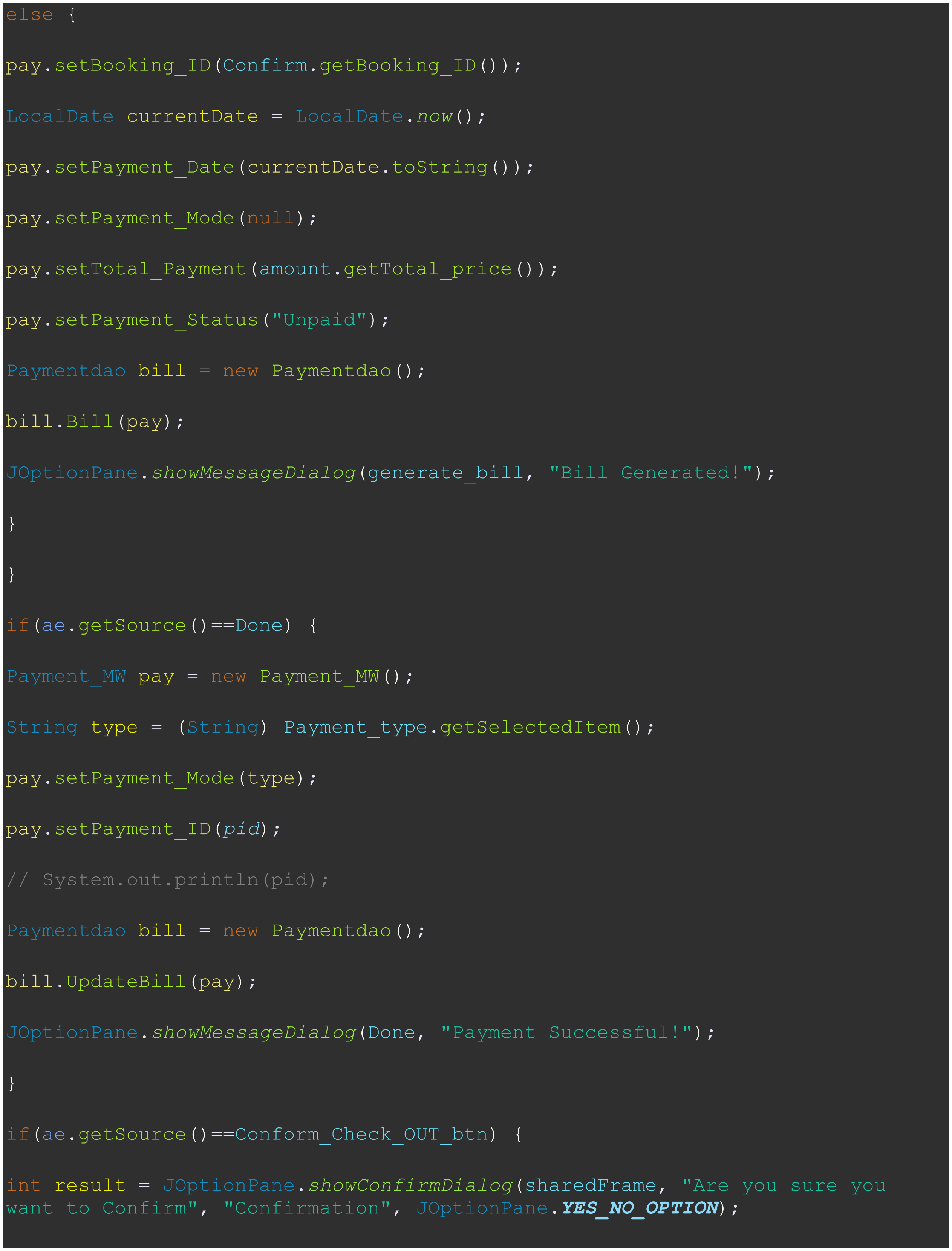
```
Room_Check = new JTable();

Room_Check.setModel(tableModel);

Room_Check.getTableHeader().setReorderingAllowed(false);

Room_Check.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Room_Check.setFillsViewportHeight(true);

Room_Check.setVisible(true);

scroll = new JScrollPane(Room_Check);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(0, 0, 400, 300);

frame.add(scroll);

List<Room_MW> roomList=new RoomJDBCdao().ViewAvailableBooking();

for(Room_MW room:roomList) {

tableModel.insertRow(0, new Object[] {
room.getRoom_NO(),room.getRoom_Type(),room.getRoom_Status()});

}

}

Timer timer = new Timer(2000, new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

DefaultTableModel model = (DefaultTableModel)Check_IN_View_Table.getModel();

model.setRowCount(0);

List<ReceptionistCheckIN_MW> bookList=new
BookingCRUDdao().ViewPendingBooking();

for(ReceptionistCheckIN_MW book:bookList) {
```

```
tableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),book.getCheck_IN(),book.getCheck_OU
T(),book.getRoom_NO(),book.getRoom_Type(),book.getStatus_Type() });

}

}

}

);

// public static void main(String[] args) {

// new Receptionist_Dashboard();

// }

@Override

public void actionPerformed(ActionEvent ae) {

// TODO Auto-generated method stub

if(ae.getSource()==Check_IN_btn) {

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

Check_IN();

List<ReceptionistCheckIN_MW> bookList=new
BookingCRUDdao().ViewPendingBooking();

for(ReceptionistCheckIN_MW book:bookList) {

tableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),book.getCheck_IN(),book.getCheck_OU
T(),book.getRoom_NO(),book.getRoom_Type(),book.getStatus_Type() });

}

}

if(ae.getSource()==Check_OUT_btn) {
```

```
Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

Check_OUT();

List<ReceptionistCheckOUT_MW> bookList=new
BookingCRUDdao().ViewActiveBooking();

for(ReceptionistCheckOUT_MW book:bookList) {

checkOuttableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getCheck_IN(),book.getCheck_OUT(),book.getRoom_NO(
),book.getRoom_Type(),book.getStatus_Type() });

}

}

if(ae.getSource()==History_btn) {

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

History();

List<ReceptionistCheckOUT_MW> bookList=new BookingCRUDdao().ViewAllBooking();

for(ReceptionistCheckOUT_MW book:bookList) {

historytableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getBook_Date(),
book.getCheck_IN(),book.getCheck_OUT(),book.getRoom_NO(),book.getRoom_Type(),
book.getStatus_Type() });

}

}

if(ae.getSource()==Room_Available) {

Room_Check(sharedFrame);
```

```
}

if(ae.getSource()==AllocateRoom_btn) {

int roomno = Integer.parseInt(Room_NO_txt.getText());

Confirm.setRoom_NO(roomno);

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Confirm the Room?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

BookingCRUDdao checkin = new BookingCRUDdao();

checkin.AllocateRoom(Confirm);

RoomJDBCdao room = new RoomJDBCdao();

room.ClosedRoomStatus(Confirm);

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

if(ae.getSource()==Confirm_btn) {

// Booking_MW Confirm = new Booking_MW();

int roomno = Integer.parseInt(Room_NO_txt.getText());

Confirm.setRoom_NO(roomno);
```

```
int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Check IN?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

BookingCRUDdao checkin = new BookingCRUDdao();

checkin.ConfirmBooking(Confirm);

RoomJDBCdao room = new RoomJDBCdao();

room.ClosedRoomStatus(Confirm);

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

if(ae.getSource()==Logout_btn) {

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Confirm the Room?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

Logout();

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");
// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

if(ae.getSource()==generate_bill) {

ActiveBookingdao check = new ActiveBookingdao();

Payment_MW pay = new Payment_MW();

pay.setBooking_ID(Confirm.getBooking_ID());

// pay.setBooking_ID(bid);

boolean result = check.CheckBill(pay);

// Payment_MW pay = new Payment_MW();

if(result==true) {

check.ActiveBill(pay);

pid = pay.getPayment_ID();

pay.setPayment_ID(pid);

LocalDate currentDate = LocalDate.now();

pay.setPayment_Date(currentDate.toString());

pay.setTotal_Payment(amount.getTotal_price());

Paymentdao bill = new Paymentdao();

bill.Updateoldbill(pay);

System.out.println(amount.getTotal_price());

JOptionPane.showMessageDialog(generate_bill, "Bill updated!");

}
```



```
else {

pay.setBooking_ID(Confirm.getBooking_ID());

LocalDate currentDate = LocalDate.now();

pay.setPayment_Date(currentDate.toString());

pay.setPayment_Mode(null);

pay.setTotal_Payment(amount.getTotal_price());

pay.setPayment_Status("Unpaid");

Paymentdao bill = new Paymentdao();

bill.Bill(pay);

JOptionPane.showMessageDialog(generate_bill, "Bill Generated!");

}

}

if(ae.getSource()==Done) {

Payment_MW pay = new Payment_MW();

String type = (String) Payment_type.getSelectedItem();

pay.setPayment_Mode(type);

pay.setPayment_ID(pid);

// System.out.println(pid);

Paymentdao bill = new Paymentdao();

bill.UpdateBill(pay);

JOptionPane.showMessageDialog(Done, "Payment Successful!");

}

if(ae.getSource()==Conform_Check_OUT_btn) {

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Confirm", "Confirmation", JOptionPane.YES_NO_OPTION);
```

```
if (result == JOptionPane.YES_OPTION) {

BookingCRUDdao checkout = new BookingCRUDdao();

checkout.Ckeck_OUTBooking(Confirm);

RoomJDBCdao open = new RoomJDBCdao();

open.OpenRoomStatus(Confirm);

DefaultTableModel model = (DefaultTableModel)Check_OUT_View_Table.getModel();

model.setRowCount(0);

List<ReceptionistCheckOUT_MW> bookList=new
BookingCRUDdao().ViewActiveBooking();

for(ReceptionistCheckOUT_MW book:bookList) {

checkOuttableModel.insertRow(0, new Object[] {
book.getBooking_ID(),book.getFirst_Name()+"
"+book.getLast_Name(),book.getCheck_IN(),book.getCheck_OUT(),book.getRoom_NO(
),book.getRoom_Type(),book.getStatus_Type() });

}

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

}

public void mouseClicked(MouseEvent e) {
```

```
if(e.getSource()==Check_IN_View_Table) {

try {

// get the index of the selected row

int rows = Check_IN_View_Table.getSelectedRow();

// get the table model

TableModel model1 = Check_IN_View_Table.getModel();

// set the values of the text fields to the values from the selected row

String value = (String) model1.getValueAt(rows, 1);

Name_txt.setText(value);

int room = (int) model1.getValueAt(rows, 5);

Room_NO_txt.setText(Integer.toString(room));

int bid = (int) model1.getValueAt(rows, 0);

Confirm.setBooking_ID(bid);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Check_OUT_View_Table) {

try {

// get the index of the selected row

int rows = Check_OUT_View_Table.getSelectedRow();

// get the table model

TableModel model = Check_OUT_View_Table.getModel();

// set the values of the text fields to the values from the selected row

int bid = (int) model.getValueAt(rows, 0);
```

```
Confirm.setBooking_ID(bid);

int roomno = (int) model.getValueAt(rows, 4);

Confirm.setRoom_NO(roomno);

String name = (String) model.getValueAt(rows, 1);

Checkout_Name.setText(name);

Date date = new SimpleDateFormat("yyyy-MM-
dd").parse((String)model.getValueAt(rows, 2));

Date date2 = new SimpleDateFormat("yyyy-MM-
dd").parse((String)model.getValueAt(rows, 3));

String dateString = new SimpleDateFormat("yyyy-MM-dd").format(date);

String dateString2 = new SimpleDateFormat("yyyy-MM-dd").format(date2);

// Calculate the number of stay days

LocalDate startdate = LocalDate.parse(dateString);

LocalDate enddate = LocalDate.parse(dateString2);

Long Stay = ChronoUnit.DAYS.between(startdate, enddate);

DefaultTableModel clearmodel =
(DefaultTableModel)Check_OUT_Item_Table.getModel();

clearmodel.setRowCount(0);

List<ViewServices_MW> itemList=new ItemJDBCdao().Viewitem(bid);

for(ViewServices_MW item:itemList) {

checkOut_item_tableModel.insertRow(0, new Object[] {
item.getItem(),item.getQuantity(),item.getQuantity()*item.getPrice(),item.get
Service_Status()});

}

List<Room_MW> List=new RoomJDBCdao().ViewRoomPrice(roomno);

for(Room_MW room:List) {

float price;
checkOut_item_tableModel.insertRow(0, new Object[] { room.getRoom_Type()+'
'+"Room",Stay+" "+"Days",Stay*room.getRoom_price(),room.getRoom_Status()});

price = Stay*room.getRoom_price();

amount.setPrice(price);

}

List<ViewServices_MW> total=new ItemJDBCdao().Viewitem(bid);

double totalprice = 0.0;

double vatprice;

for(ViewServices_MW item:total) {

// checkOut_item_tableModel.insertRow(0, new Object[] {
item.getItem(),item.getQuantity(),item.getPrice(),item.getService_Status()});

item.getPrice();

totalprice +=item.getQuantity()*item.getPrice();

amount.setTotal_price(totalprice);

// System.out.println(totalprice);

// totalprice = totalprice+price;

}

double total_price = amount.getTotal_price();

float price = amount.getPrice();

total_price = total_price+price;

vatprice = total_price*0.13;

total_price = total_price+vatprice;

//

amount.setTotal_price(total_price);

Total_price_txt.setText(Double.toString(total_price));

} catch (Exception ex) {
```

```
System.out.println("Error" + ex.getMessage());

}

}

}

// override the other methods of the MouseListener interface

@Override

public void mousePressed(MouseEvent e) {

}

@Override

public void mouseReleased(MouseEvent e) {

}

@Override

public void mouseEntered(MouseEvent e) {

}

@Override

public void mouseExited(MouseEvent e) {

}

}
```

File name: Staff_Dashboard.java

```
package UI;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseEvent;

import java.awt.event.MouseListener;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPasswordField;

import javax.swing.JRadioButton;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextField;

import javax.swing.table.DefaultTableModel;

import javax.swing.table.TableModel;
import com.toedter.calendar.JDateChooser;

import DAO.CRUDdao;

import DAO.ItemJDBCdao;

import DAO.OrderJDBCdao;

import Middleware.MealItem_MW;

import Middleware.Services_MW;

import Middleware.Staff_MW;

import Middleware.ViewServices_MW;

public class Staff_Dashboard extends Dashboard_Frame implements
ActionListener,MouseListener {

JLabel Name, Address, Mobile, Email, Gender, DOB, Password;

JTextField First_Name_txt, Last_Name_txt;

JTextField Address_txt, Mobile_txt, Email_txt;

JPasswordField Password_txt;

JDateChooser DOB_txt;

JButton Profile_Update_btn;

JComboBox<String> Gender_txt;

JRadioButton Radio_button;

JButton Orders_btn, Add_service_btn, Profile_btn, Logout_btn;

JLabel welcome, Welcome_name;

JTable
Item_Table,Meal_Item_Table,Drinks_Item_Table,Other_Services,Breakfast_Item_Ta
ble, Profile_View_Table;

DefaultTableModel
Item_tableModel,Meal_Item_tableModel,Drinks_Item_tableModel,Other_Services_ta
belModel,Breakfast_Item_tableModel,Profile_tableModel;

JScrollPane
itemscroll,Mealscroll,Drinkscroll,Otherscroll,Breakfastscroll,scroll;
```

```
JTextField Item, Quantity, SID;

JButton Order;

static int ID;

public Staff_Dashboard(int id, String Name) {

JLabel label = new JLabel(); //JLabel Creation

label.setIcon(new
ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\6.png")); //Sets the
image to be displayed as an icon

Dimension size = label.getPreferredSize(); //Gets the size of the image

label.setBounds(80, 0, size.height, size.width); //Sets the location of the
image

Left_panel.add(label); //Adds objects to the container

Welcome_name = new JLabel(Name);

Welcome_name.setFont(new Font("Arial", Font.CENTER_BASELINE, 30));

Welcome_name.setForeground(Color.WHITE);

Welcome_name.setBounds(25,130,280,40);

Left_panel.add(Welcome_name);

ID = id;

Orders_btn = new JButton("Orders");

Orders_btn.addActionListener(this);

Orders_btn.setBounds(50,250,180,40);

Left_panel.add(Orders_btn);

Add_service_btn = new JButton("Add Service");

Add_service_btn.addActionListener(this);

Add_service_btn.setBounds(50,320,180,40);

Left_panel.add(Add_service_btn);
```

```
Profile_btn = new JButton("Profile");

Profile_btn.addActionListener(this);

Profile_btn.setBounds(50,390,180,40);

Left_panel.add(Profile_btn);

welcome = new JLabel("Hotel Luton");

welcome.setFont(new Font("Arial", Font.CENTER_BASELINE, 70));

welcome.setForeground(Color.WHITE);

welcome.setBounds(500,40,400,60);

Logout_btn = new JButton("Logout");

Logout_btn.addActionListener(this);

Logout_btn.setBounds(50,460,180,40);

Left_panel.add(Logout_btn);

Top_panel.add(welcome);

}

public void Logout() {

sharedFrame.switchToLogin();

sharedFrame.repaint();

sharedFrame.revalidate();

}

public void Service() {

JLabel YourOrders_lbl = new JLabel("Ordered Items");

YourOrders_lbl.setBounds(1020, 20, 150, 20);

Center_panel.add(YourOrders_lbl);

String[] Column2 = {"BID","Item","Qty","Price","Status"};

Item_tableModel = new DefaultTableModel();
```

```
Item_tableModel.setColumnIdentifiers(Column2);

Item_Table = new JTable();

Item_Table.setModel(Item_tableModel);

Item_Table.getTableHeader().setReorderingAllowed(false);

Item_Table.addMouseListener(this);

Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Item_Table.setFillsViewportHeight(true);

Item_Table.setVisible(true);

itemscroll = new JScrollPane(Item_Table);

itemscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

itemscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

itemscroll.setBounds(1020, 50, 350, 200);

Center_panel.add(itemscroll);

JLabel Meal_lbl = new JLabel("Meal Items");

Meal_lbl.setBounds(20, 20, 80, 20);

Center_panel.add(Meal_lbl);

String[] Meal = {"SN","Item","Price"};

Meal_Item_tableModel = new DefaultTableModel();

Meal_Item_tableModel.setColumnIdentifiers(Meal);

Meal_Item_Table = new JTable();

Meal_Item_Table.setModel(Meal_Item_tableModel);

Meal_Item_Table.getTableHeader().setReorderingAllowed(false);

Meal_Item_Table.addMouseListener(this);

Meal_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);
Meal_Item_Table.setFillsViewportHeight(true);

Meal_Item_Table.setVisible(true);

Mealscroll = new JScrollPane(Meal_Item_Table);

Mealscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Mealscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Mealscroll.setBounds(20, 50, 300, 200);

Center_panel.add(Mealscroll);

JLabel Breakfast_lbl = new JLabel("Breakfast Items");

Breakfast_lbl.setBounds(400, 20, 100, 20);

Center_panel.add(Breakfast_lbl);

String[] Breakfast = {"SN","Item","Price"};

Breakfast_Item_tableModel = new DefaultTableModel();

Breakfast_Item_tableModel.setColumnIdentifiers(Breakfast);

Breakfast_Item_Table = new JTable();

Breakfast_Item_Table.setModel(Breakfast_Item_tableModel);

Breakfast_Item_Table.getTableHeader().setReorderingAllowed(false);

Breakfast_Item_Table.addMouseListener(this);

Breakfast_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Breakfast_Item_Table.setFillsViewportHeight(true);

Breakfast_Item_Table.setVisible(true);

Breakfastscroll = new JScrollPane(Breakfast_Item_Table);

Breakfastscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Breakfastscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);
Breakfastscroll.setBounds(400, 50, 300, 200);

Center_panel.add(Breakfastscroll);

JLabel Drinks_lbl = new JLabel("Drinks Items");

Drinks_lbl.setBounds(20, 320, 80, 20);

Center_panel.add(Drinks_lbl);

String[] Drinks = {"SN","Item","Price"};

Drinks_Item_tableModel = new DefaultTableModel();

Drinks_Item_tableModel.setColumnIdentifiers(Drinks);

Drinks_Item_Table = new JTable();

Drinks_Item_Table.setModel(Drinks_Item_tableModel);

Drinks_Item_Table.getTableHeader().setReorderingAllowed(false);

Drinks_Item_Table.addMouseListener(this);

Drinks_Item_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Drinks_Item_Table.setFillsViewportHeight(true);

Drinks_Item_Table.setVisible(true);

Drinkscroll = new JScrollPane(Drinks_Item_Table);

Drinkscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Drinkscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Drinkscroll.setBounds(20, 350, 300, 200);

Center_panel.add(Drinkscroll);

JLabel Other_lbl = new JLabel("Others Items");

Other_lbl.setBounds(400, 320, 80, 20);

Center_panel.add(Other_lbl);

String[] other = {"SN","Item","Price"};
```

```
Other_Services_tabelModel = new DefaultTableModel();

Other_Services_tabelModel.setColumnIdentifiers(other);

Other_Services = new JTable();

Other_Services.setModel( Other_Services_tabelModel);

Other_Services.getTableHeader().setReorderingAllowed(false);

Other_Services.addMouseListener(this);

Other_Services.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Other_Services.setFillsViewportHeight(true);

Other_Services.setVisible(true);

Otherscroll = new JScrollPane(Other_Services);

Otherscroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

Otherscroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

Otherscroll.setBounds(400, 350, 300, 200);

Center_panel.add(Otherscroll);

JLabel SN_lbl = new JLabel("BID");

SN_lbl.setBounds(920, 400, 80, 40);

Center_panel.add(SN_lbl);

JLabel Item_lbl = new JLabel("Item");

Item_lbl.setBounds(920, 460, 80, 40);

Center_panel.add(Item_lbl);

JLabel Quantity_lbl = new JLabel("Quantity");

Quantity_lbl.setBounds(920, 520, 80, 40);

Center_panel.add(Quantity_lbl);

SID = new JTextField();
```

```
SID.setBounds(1000, 400, 100, 30);

Center_panel.add(SID);

Item = new JTextField();

Item.setBounds(1000, 460, 100, 30);

Center_panel.add(Item);

Quantity = new JTextField();

Quantity.setBounds(1000, 520, 100, 30);

Center_panel.add(Quantity);

Order = new JButton("Confirm");

Order.setBounds(1050, 620, 100, 30);

Order.addActionListener(this);

Center_panel.add(Order);

}

public void profile() {

String[] Column = {"CID","First Name","Last
Name","Address","Mobile","Email","Gender","DOB"};

Profile_tableModel = new DefaultTableModel();

Profile_tableModel.setColumnIdentifiers(Column);

Profile_View_Table = new JTable();

Profile_View_Table.setModel(Profile_tableModel);

Profile_View_Table.getTableHeader().setReorderingAllowed(false);

Profile_View_Table.addMouseListener(this);

Profile_View_Table.setAutoResizeMode(JTable.AUTO_RESIZE_ALL_COLUMNS);

Profile_View_Table.setFillsViewportHeight(true);

Profile_View_Table.setVisible(true);
```

```
scroll = new JScrollPane(Profile_View_Table);

scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED);

scroll.setBounds(580, 120, 750, 200);

Center_panel.add(scroll);

Name = new JLabel("Name");

Name.setFont(new Font("Arial", Font.PLAIN, 20));

Name.setBounds(130,100,120,80);

Center_panel.add(Name);

Address = new JLabel("Address");

Address.setFont(new Font("Arial", Font.PLAIN, 20));

Address.setBounds(130,140,150,80);

Center_panel.add(Address);

Mobile = new JLabel("Mobile");

Mobile.setFont(new Font("Arial", Font.PLAIN, 20));

Mobile.setBounds(130,180,150,80);

Center_panel.add(Mobile);

Email = new JLabel("Email");

Email.setFont(new Font("Arial", Font.PLAIN, 20));

Email.setBounds(130,220,150,80);

Center_panel.add(Email);

Gender = new JLabel("Gender");

Gender.setFont(new Font("Arial", Font.PLAIN, 20));

Gender.setBounds(130,260,150,80);
```

```
Center_panel.add(Gender);
DOB = new JLabel("DOB");
DOB.setFont(new Font("Arial", Font.PLAIN, 20));
DOB.setBounds(130,300,150,80);
Center_panel.add(DOB);
Password = new JLabel("Password");
Password.setFont(new Font("Arial", Font.PLAIN, 20));
Password.setBounds(130,340,190,80);
Center_panel.add(Password);
First_Name_txt = new JTextField();
First_Name_txt.setBounds(280,125,120,30);
Center_panel.add(First_Name_txt);
Last_Name_txt = new JTextField();
Last_Name_txt.setBounds(430,125,120,30);
Center_panel.add(Last_Name_txt);
Address_txt = new JTextField();
Address_txt.setBounds(280,165,200,30);
Center_panel.add(Address_txt);
Mobile_txt = new JTextField();
Mobile_txt.setBounds(280,205,200,30);
Center_panel.add(Mobile_txt);
Email_txt = new JTextField();
Email_txt.setBounds(280,245,200,30);
Center_panel.add(Email_txt);
String[] items = {"Male","Female"};
```

```
Gender_txt = new JComboBox<>(items);
Gender_txt.setSelectedItem("Green");
Gender_txt.setBounds(280,285,200,30);
Center_panel.add(Gender_txt);
DOB_txt = new JDateChooser();
DOB_txt.setDateFormatString("yyyy-MM-dd");
DOB_txt.setFont(new Font("Verdana",Font.PLAIN,18));
DOB_txt.setBounds(280,325,200,30);
Center_panel.add(DOB_txt);
Password_txt = new JPasswordField();
Password_txt.setBounds(280,365,200,30);
Center_panel.add(Password_txt);
Radio_button = new JRadioButton("Show");
Radio_button.setBounds(280, 400, 80, 25);
Radio_button.setFont(new Font("Arial", Font.PLAIN, 20));
Center_panel.add(Radio_button);
Radio_button.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent evt) {
// show password chars
if (Radio_button.isSelected()) {
Password_txt.setEchoChar((char) 0);
}
// hide password chars
else {
```

```
Password_txt.setEchoChar('.');

}

}

});

Profile_Update_btn = new JButton("Update");

Profile_Update_btn.addActionListener(this);

Profile_Update_btn.setBounds(320,500,120,50);

Center_panel.add(Profile_Update_btn);

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(e.getSource()==Orders_btn) {

Top_panel.removeAll();

Top_panel.revalidate();

Top_panel.repaint();

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

Top_panel.add(welcome);

Service();

List<ViewServices_MW> itemList=new ItemJDBCdao().ViewAllitem();

for(ViewServices_MW item:itemList) {

Item_tableModel.insertRow(0, new Object[] {
item.getService_ID(),item.getItem(),item.getQuantity(),item.getPrice(),item.getService_Status()});
```

```
}

List<MealItem_MW> MealList=new ItemJDBCdao().Mealitem();

for(MealItem_MW Meal:MealList) {

Meal_Item_tableModel.insertRow(0, new Object[] {
Meal.getItem_ID(),Meal.getItem(),Meal.getPrice()});

}

List<MealItem_MW> BreakfastList=new ItemJDBCdao().Breakfastitem();

for(MealItem_MW breakfast:BreakfastList) {

Breakfast_Item_tableModel.insertRow(0, new Object[] {
breakfast.getItem_ID(),breakfast.getItem(),breakfast.getPrice()});

}

List<MealItem_MW> DrinksList=new ItemJDBCdao().Drinkitem();

for(MealItem_MW drink:DrinksList) {

Drinks_Item_tableModel.insertRow(0, new Object[] {
drink.getItem_ID(),drink.getItem(),drink.getPrice()});

}

List<MealItem_MW> OtherList=new ItemJDBCdao().Otheritem();

for(MealItem_MW other:OtherList) {

Other_Services_tabelModel.insertRow(0, new Object[] {
other.getItem_ID(),other.getItem(),other.getPrice()});

}

}

if(e.getSource()==Order) {

Services_MW service = new Services_MW();

int sid = Integer.parseInt(SID.getText());

service.setService_ID(sid);

OrderJDBCdao confirm = new OrderJDBCdao();
```

```
confirm.ConfirmOrder(service);

DefaultTableModel model = (DefaultTableModel)Item_Table.getModel();

model.setRowCount(0);

List<ViewServices_MW> itemList=new ItemJDBCdao().ViewAllitem();

for(ViewServices_MW item:itemList) {

Item_tableModel.insertRow(0, new Object[] {
item.getService_ID(),item.getItem(),item.getQuantity(),item.getPrice(),item.g
etService_Status()});

}

}

if(e.getSource()==Logout_btn) {

int result = JOptionPane.showConfirmDialog(sharedFrame, "Are you sure you
want to Confirm the Room?", "Confirmation", JOptionPane.YES_NO_OPTION);

if (result == JOptionPane.YES_OPTION) {

Logout();

System.out.println("User clicked Yes");

// Perform action for Yes option

} else if (result == JOptionPane.NO_OPTION) {

System.out.println("User clicked No");

// Perform action for No option

} else if (result == JOptionPane.CANCEL_OPTION) {

System.out.println("User clicked Cancel");

// Perform action for Cancel option

}

}

if(e.getSource()==Profile_btn) {
```

```
Top_panel.removeAll();

Top_panel.revalidate();

Top_panel.repaint();

Center_panel.removeAll();

Center_panel.repaint();

Center_panel.revalidate();

profile();

Top_panel.add(welcome);

List<Staff_MW> List=new CRUDdao().StaffViewprofile(ID);

for(Staff_MW staff:List) {

Profile_tableModel.insertRow(0, new Object[] {
staff.getStaff_ID(),staff.getStaff_First_name(),staff.getStaff_Last_name(),staff.getStaff_Address(),staff.getStaff_Mobile(),staff.getStaff_Email(),staff.getStaff_Gender(),staff.getStaff_DOB(),staff.getStaff_Password()});

}

}

if(e.getSource()==Profile_Update_btn) {

String first_name = First_Name_txt.getText();

String last_name = Last_Name_txt.getText();

String address = Address_txt.getText();

String mobile = Mobile_txt.getText();

String email = Email_txt.getText();

String gender = Gender_txt.getSelectedItem().toString();

String dob = ((JTextField)
DOB_txt.getDateEditor().getUiComponent()).getText();

@SuppressWarnings("deprecation")

String password = Password_txt.~~getText~~();
```

```
Staff_MW cust = new Staff_MW();

cust.setStaff_First_name(first_name);

cust.setStaff_Last_name(last_name);

cust.setStaff_Address(address);

cust.setStaff_Mobile(mobile);

cust.setStaff_Email(email);

cust.setStaff_Gender(gender);

cust.setStaff_DOB(dob);

cust.setStaff_Password(password);

cust.setStaff_ID(ID);

CRUDdao upd = new CRUDdao();

upd.StaffUpdate(cust);

JOptionPane.showMessageDialog(Profile_Update_btn, "Update Successfull!");

DefaultTableModel model = (DefaultTableModel)Profile_View_Table.getModel();

model.setRowCount(0);

List<Staff_MW> List=new CRUDdao().StaffViewprofile(ID);

for(Staff_MW staff:List) {

Profile_tableModel.insertRow(0, new Object[] {staff.getStaff_ID(),staff.getStaff_First_name(),staff.getStaff_Last_name(),staff.getStaff_Address(),staff.getStaff_Mobile(),staff.getStaff_Email(),staff.getStaff_Gender(),staff.getStaff_DOB(),staff.getStaff_Password()});

}

}

}

// public static void main(String[] args) {

// new Staff_Dashboard();
// }

@Override

public void mouseClicked(MouseEvent e) {

// TODO Auto-generated method stub

if(e.getSource()==Item_Table) {

try {

// get the index of the selected row

int rows = Item_Table.getSelectedRow();

// get the table model

TableModel model = Item_Table.getModel();

// set the values of the text fields to the values from the selected row

int sid = (int) model.getValueAt(rows, 0);

SID.setText(Integer.toString(sid));

String item = (String) model.getValueAt(rows, 1);

Item.setText(item);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

if(e.getSource()==Profile_View_Table) {

try {

// get the index of the selected row

int rows = Profile_View_Table.getSelectedRow();

// get the table model

TableModel model = Profile_View_Table.getModel();
// set the values of the text fields to the values from the selected row

String first_name = (String) model.getValueAt(rows, 1);

String last_name = (String) model.getValueAt(rows, 2);

String address = (String) model.getValueAt(rows, 3);

String mobile = (String) model.getValueAt(rows, 4);

String email = (String) model.getValueAt(rows, 5);

// String gender = (String) model.getValueAt(rows, 6);

Date dob = new SimpleDateFormat("yyyy-MM-dd").parse((String)
model.getValueAt(rows, 7));

First_Name_txt.setText(first_name);

Last_Name_txt.setText(last_name);

Address_txt.setText(address);

Mobile_txt.setText(mobile);

Email_txt.setText(email);

DOB_txt.setDate(dob);

} catch (Exception ex) {

System.out.println("Error" + ex.getMessage());

}

}

}

}
```

File name: Frame.java

```
package UI;

import javax.swing.JFrame;

public class Frame extends JFrame{

/**

*

*/

private static final long serialVersionUID = 1L;

@SuppressWarnings("unused")

private Register register;

@SuppressWarnings("unused")

private Customer_Dashboard customer_dashboard;

@SuppressWarnings("unused")

private Receptionist_Dashboard receptionist_dashboard;

@SuppressWarnings("unused")

private Staff_Dashboard staff_dashboard;

@SuppressWarnings("unused")

private Login login;

// JFrame Window_main;

private static Frame instance = null;

private Frame(){

setTitle("Luton Hotel");

setSize(1760,980);

setLayout(null);
setVisible(true);

setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}

public static Frame getInstance() {

if (instance == null) {

instance = new Frame();

}

return instance;

}

public static void main(String[] args) {

Frame mainWindow = Frame.getInstance();

mainWindow.setVisible(true);

}

public void switchToRegister() {

getContentPane().removeAll();

// getContentPane().add(register);

// pack();

register = new Register();

}

public void switchToCustomerdashboard(int id, String name) {

getContentPane().removeAll();

// getContentPane().add(register);

// pack();

customer_dashboard = new Customer_Dashboard(id, name);

}
```

```
public void switchToReceptionistdashboard(int id, String name) {

getContentPane().removeAll();

// getContentPane().add(register);

// pack();

receptionist_dashboard = new Receptionist_Dashboard(name);

}

public void switchToStaffdashboard(int id, String name) {

getContentPane().removeAll();

// getContentPane().add(register);

// pack();

staff_dashboard = new Staff_Dashboard(id, name);

}

public void switchToLogin() {

getContentPane().removeAll();

// getContentPane().add(register);

// pack();

login = new Login();

}

}
```

File name: Dashboard_Frame.java

```
package UI;

import java.awt.Color;

import java.awt.event.MouseEvent;

import javax.swing.JPanel;

public class Dashboard_Frame {

protected JPanel Left_panel, Top_panel, Center_panel;

Frame sharedFrame = Frame.getInstance();

public Dashboard_Frame() {

Left_panel = new JPanel();

Left_panel.setSize(300,980);

Left_panel.setBounds(0,0,300,980);

Left_panel.setBackground(new Color(46,18,45)); // set the panel background
color to red

Left_panel.setVisible(true);

Left_panel.setLayout(null);

sharedFrame.add(Left_panel);

Top_panel = new JPanel();

Top_panel.setBounds(300,0,1460,198);

Top_panel.setBackground(new Color(83,3,50)); // set the panel background
color to red

Top_panel.setVisible(true);

Top_panel.setLayout(null);

sharedFrame.add(Top_panel);

Center_panel = new JPanel();
Center_panel.setBounds(300,198,1460,782);

Center_panel.setBackground(new Color(234,214,214)); // set the panel
background color to red

Center_panel.setVisible(true);

Center_panel.setLayout(null);

sharedFrame.add(Center_panel);

}

public static void main(String[] args) {

new Dashboard_Frame();

}

public void mousePressed(MouseEvent e) {

// TODO Auto-generated method stub

}

public void mouseReleased(MouseEvent e) {

// TODO Auto-generated method stub

}

public void mouseExited(MouseEvent e) {

// TODO Auto-generated method stub

}

public void mouseEntered(MouseEvent e) {

// TODO Auto-generated method stub

}

}
```

File name: Login.java

```
package UI;

import java.awt.AlphaComposite;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.*;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JRadioButton;

import javax.swing.JTextField;

import DAO.Logindao;

import Middleware.Login_MW;

public class Login implements ActionListener{

private JLabel Login_lbl, Username_lbl, Password_lbl;

private JTextField Username_txt;

private JPasswordField Password_txt;

private JButton Login_btn, Register_btn;
private JRadioButton radioButton;

private JPanel panel;

Frame sharedFrame = Frame.getInstance();

public static void main(String[] args) {

new Login();

}

public Login() {

panel = new JPanel(){

/**

*

*/

private static final long serialVersionUID = 1L;

@Override

protected void paintComponent(Graphics g) {

Graphics2D g2 = (Graphics2D) g;

g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.9f));
// set the opacity to 90%

g2.setColor(getBackground());

g2.fillRect(0, 0, getWidth(), getHeight());

super.paintComponent(g);

}

};

// add components to panel here

panel.setLayout(null);

panel.setBounds(500, 50, 700, 600);
panel.setBackground(new Color(217, 217, 217)); // set the panel background
color to red

panel.setOpaque(false); // set the panel opacity to transparent

sharedFrame.getContentPane().add(panel);

Login_lbl = new JLabel("Login");

Username_lbl = new JLabel("Username");

Password_lbl = new JLabel("Password");

Login_lbl.setFont(new Font("Arial", Font.ITALIC, 48));

Username_lbl.setFont(new Font("Arial", Font.PLAIN, 26));

Password_lbl.setFont(new Font("Arial", Font.PLAIN, 26));

Login_lbl.setOpaque(false);

Username_lbl.setOpaque(false);

Username_lbl.setBackground( new Color(217, 217, 217, 50) );

Login_lbl.setBounds(265,55,180,80);

Username_lbl.setBounds(150,200,120,80);

Password_lbl.setBounds(150,250,120,80);

panel.add(Login_lbl);

panel.add(Username_lbl);

panel.add(Password_lbl);

Username_txt = new JTextField();

Password_txt = new JPasswordField();

Username_txt.setBounds(320,230,200,30);

Password_txt.setBounds(320,280,200,30);

panel.add(Username_txt);

panel.add(Password_txt);
```

```
Login_btn = new JButton("Login");
Login_btn.setFont(new Font("Arial", Font.CENTER_BASELINE, 26));
Login_btn.setBounds(265,450,120,40);
panel.add(Login_btn);
radioButton = new JRadioButton("Show");
radioButton.setBounds(320, 320, 80, 25);
radioButton.setFont(new Font("Arial", Font.PLAIN, 20));
panel.add(radioButton);
radioButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent evt) {
// show password chars
if (radioButton.isSelected()) {
Password_txt.setEchoChar((char) 0);
}
// hide password chars
else {
Password_txt.setEchoChar('.');
}
}
});
Register_btn = new JButton("Register");
Register_btn.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));
Register_btn.setBounds(550,50,115,30);
Register_btn.addActionListener(new ActionListener() {
```

```
public void actionPerformed(ActionEvent e) {

sharedFrame.dispose();

sharedFrame.switchToRegister();

}

});

panel.add(Register_btn);

sharedFrame.add(panel);

sharedFrame.setVisible(true);

JLabel label = new JLabel(); //JLabel Creation

label.setIcon(new ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\4.jpg")); //Sets the image to be displayed as an icon

Dimension size = label.getPreferredSize(); //Gets the size of the image

label.setBounds(0, 0, size.width, size.height); //Sets the location of the image

sharedFrame.add(label); //Adds objects to the container

sharedFrame.setVisible(true); // Exhibit the frame

sharedFrame.getContentPane().setBackground(Color.LIGHT_GRAY);

Login_btn.addActionListener(this);

}

@Override

public void actionPerformed(ActionEvent e) {

// TODO Auto-generated method stub

if(e.getSource()==Login_btn) {

String Username = Username_txt.getText();

@SuppressWarnings("deprecation")
```

```
String Password = Password_txt.~~getText~~();

Logindao log = new Logindao();

// CheckAuth log = new CheckAuth();

Login_MW lg = new Login_MW();

lg.setMobile(Username);

lg.setPassword(Password);

// boolean result = log.CustomerAuth(lg);

boolean result = log.checkUserAuth(lg);

boolean empresult = log.checkStaffAuth(lg);

// System.out.println(lg.getNewPassword());

if(lg.getNewPassword().equals(Password) && result == true) {

String name = lg.getName();

JOptionPane.showMessageDialog(Login_btn, "Welcome!"+' '+name);

int id = lg.getId();

System.out.println(id);

sharedFrame.switchToCustomerdashboard(id, name);

sharedFrame.repaint();

sharedFrame.revalidate();

}

else if(lg.getNewPassword().equals(Password) && empresult == true) {

String name = lg.getName();

JOptionPane.showMessageDialog(Login_btn, "Welcome!"+' '+name);

int id = lg.getId();

int role = lg.getRole();

// String name = lg.getName();
// System.out.println(id);

if(role==2) {

sharedFrame.switchToReceptionistdashboard(id,name);

sharedFrame.repaint();

sharedFrame.revalidate();

}

else if(role==3) {

sharedFrame.switchToStaffdashboard(id, name);

sharedFrame.repaint();

sharedFrame.revalidate();

}

}

else {

JOptionPane.showMessageDialog(Login_btn, "Error to Login!");

}

}

}

}
```

File name: Register.java

```
package UI;

import java.awt.AlphaComposite;

import java.awt.Color;

import java.awt.Container;

import java.awt.Dimension;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import javax.swing.ImageIcon;

import javax.swing.JButton;

import javax.swing.JComboBox;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JRadioButton;

import javax.swing.JTextField;

import com.toedter.calendar.JDateChooser;
import DAO.CRUDdao;

import DAO.registervalidation;

import Middleware.Customer_MW;

public class Register implements ActionListener{

JPanel panel;

Frame sharedFrame = Frame.getInstance();

private JLabel register, Name, Address, Mobile, Email, Gender, DOB, Password;

private JTextField First_Name_txt, Last_Name_txt;

private JTextField Address_txt, Mobile_txt, Email_txt;

private JButton Register_btn,Login_btn;

private JDateChooser DOB_txt;

private JRadioButton radioButton;

private JPasswordField Password_txt;

private JComboBox<String> Gender_txt;

public static void main(String[] args) {

new Register();

}

public Register() {

panel = new JPanel(){

/**

*

*/

private static final long serialVersionUID = -4045321404491759938L;

@Override

protected void paintComponent(Graphics g) {
```

```
Graphics2D g2 = (Graphics2D) g;

g2.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 0.9f));
// set the opacity to 60%

g2.setColor(getBackground());

g2.fillRect(0, 0, getWidth(), getHeight());

super.paintComponent(g);

}

};

panel.setLayout(null);

panel.setBounds(500, 50, 700, 600);

panel.setBackground(new Color(217, 217, 217)); // set the panel background
color to red

panel.setOpaque(false); // set the panel opacity to transparent

sharedFrame.getContentPane().add(panel);

register = new JLabel("Register");

register.setFont(new Font("Arial", Font.PLAIN, 48));

register.setBounds(260,25,180,80);

panel.add(register);

Name = new JLabel("Name");

Name.setFont(new Font("Arial", Font.PLAIN, 20));

Name.setBounds(130,100,120,80);

panel.add(Name);

Address = new JLabel("Address");

Address.setFont(new Font("Arial", Font.PLAIN, 20));

Address.setBounds(130,140,150,80);

panel.add(Address);
```

```
Mobile = new JLabel("Mobile");

Mobile.setFont(new Font("Arial", Font.PLAIN, 20));

Mobile.setBounds(130,180,150,80);

panel.add(Mobile);

Email = new JLabel("Email");

Email.setFont(new Font("Arial", Font.PLAIN, 20));

Email.setBounds(130,220,150,80);

panel.add(Email);

Gender = new JLabel("Gender");

Gender.setFont(new Font("Arial", Font.PLAIN, 20));

Gender.setBounds(130,260,150,80);

panel.add(Gender);

DOB = new JLabel("DOB");

DOB.setFont(new Font("Arial", Font.PLAIN, 20));

DOB.setBounds(130,300,150,80);

panel.add(DOB);

Password = new JLabel("Password");

Password.setFont(new Font("Arial", Font.PLAIN, 20));

Password.setBounds(130,340,190,80);

panel.add(Password);

First_Name_txt = new JTextField("First Name");

First_Name_txt.addMouseListener(new MouseAdapter(){

@Override

public void mouseClicked(MouseEvent e){

First_Name_txt.setText("");
}

});

First_Name_txt.setBounds(280,125,120,30);

panel.add(First_Name_txt);

Last_Name_txt = new JTextField("Last Name");

Last_Name_txt.addMouseListener(new MouseAdapter(){

@Override

public void mouseClicked(MouseEvent e){

Last_Name_txt.setText("");

}

});

Last_Name_txt.setBounds(430,125,120,30);

panel.add(Last_Name_txt);

Address_txt = new JTextField();

Address_txt.setBounds(280,165,200,30);

panel.add(Address_txt);

Mobile_txt = new JTextField();

Mobile_txt.setBounds(280,205,200,30);

panel.add(Mobile_txt);

Email_txt = new JTextField();

Email_txt.setBounds(280,245,200,30);

panel.add(Email_txt);

// Gender_txt = new JTextField();

// Gender_txt.setBounds(280,285,200,30);

// panel.add(Gender_txt);
```

```
String[] items = {"Male","Female"};

Gender_txt = new JComboBox<>(items);

Gender_txt.setSelectedItem("Green");

Gender_txt.setBounds(280,285,200,30);

panel.add(Gender_txt);

DOB_txt = new JDateChooser();

DOB_txt.setDateFormatString("yyyy-MM-dd");

DOB_txt.setFont(new Font("Verdana",Font.PLAIN,18));

DOB_txt.setBounds(280,325,200,30);

panel.add(DOB_txt);

Password_txt = new JPasswordField();

Password_txt.setBounds(280,365,200,30);

panel.add(Password_txt);

Register_btn = new JButton("Register");

Register_btn.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));

Register_btn.addActionListener(this);

Register_btn.setBounds(260,450,115,30);

panel.add(Register_btn);

radioButton = new JRadioButton("Show");

radioButton.setBounds(280, 400, 80, 25);

radioButton.setFont(new Font("Arial", Font.PLAIN, 20));

panel.add(radioButton);

radioButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent evt) {
```

```
// show password chars
if (radioButton.isSelected()) {
Password_txt.setEchoChar((char) 0);
}
// hide password chars
else {
Password_txt.setEchoChar('.');
}
}
});
Login_btn = new JButton("LOGIN");
Login_btn.setFont(new Font("Arial", Font.CENTER_BASELINE, 20));
Login_btn.setBounds(550,50,115,30);
panel.add(Login_btn);
Login_btn.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
sharedFrame.dispose();
sharedFrame.switchToLogin();
}
});
Container c = sharedFrame.getContentPane(); //Gets the content layer
JLabel label = new JLabel(); //JLabel Creation
label.setIcon(new ImageIcon("D:\\Project\\Eclipse\\Assignment\\src\\UI\\4.jpg")); //Sets the image to be displayed as an icon
Dimension size = label.getPreferredSize(); //Gets the size of the image
```

```
label.setBounds(0, 0, size.width, size.height); //Sets the location of the
image

c.add(label); //Adds objects to the container

sharedFrame.setVisible(true); // Exhibit the frame

sharedFrame.getContentPane().setBackground(Color.LIGHT_GRAY);

}

public void actionPerformed(ActionEvent e) {

if(e.getSource()==Register_btn) {

String first_name = First_Name_txt.getText();

String last_name = Last_Name_txt.getText();

String address = Address_txt.getText();

String mobile = Mobile_txt.getText();

String email = Email_txt.getText();

String gender = Gender_txt.getSelectedItem().toString();

String dob = ((JTextField)
DOB_txt.getDateEditor().getUiComponent()).getText();

@SuppressWarnings("deprecation")

String password = Password_txt.~~getText~~();

// boolean validFirstname = true;

// boolean validEmail = false;

//

// String logMessage =" ";

registervalidation val = new registervalidation();

boolean resultFName = val.First_Name(first_name);

if (resultFName == true) {

boolean Gender1 = val.Gender(gender);
```

```
if (Gender1 == true) {

boolean result = val.Mobile(mobile);

if (result == true) {

boolean emailresult = val.Email(email);

if (emailresult == true) {

boolean resultpassword=val.Password(password);

if(resultpassword==true) {

Customer_MW cust = new Customer_MW();

cust.setFirst_name(first_name);

cust.setLast_name(last_name);

cust.setAddress(address);

cust.setMobile(mobile);

cust.setEmail(email);

cust.setGender(gender);

cust.setDOB(dob);

cust.setPassword(password);

JOptionPane.showMessageDialog(Register_btn, "Register Successfully");

CRUDdao reg = new CRUDdao();

reg.RegisterCustomer(cust);

}

else {

JOptionPane.showMessageDialog(null, "Please enter proper password!");

}

}

else {
```

```
JOptionPane.showMessageDialog(null, "Invalid Email");

}

}

else {

JOptionPane.showMessageDialog(null, "Invalid Mobile Number");

}

}

else {

JOptionPane.showMessageDialog(null, "Invalid Gender");

}

}

else {

JOptionPane.showMessageDialog(null, "Invalid Name");

}

}

}

}
```

Model

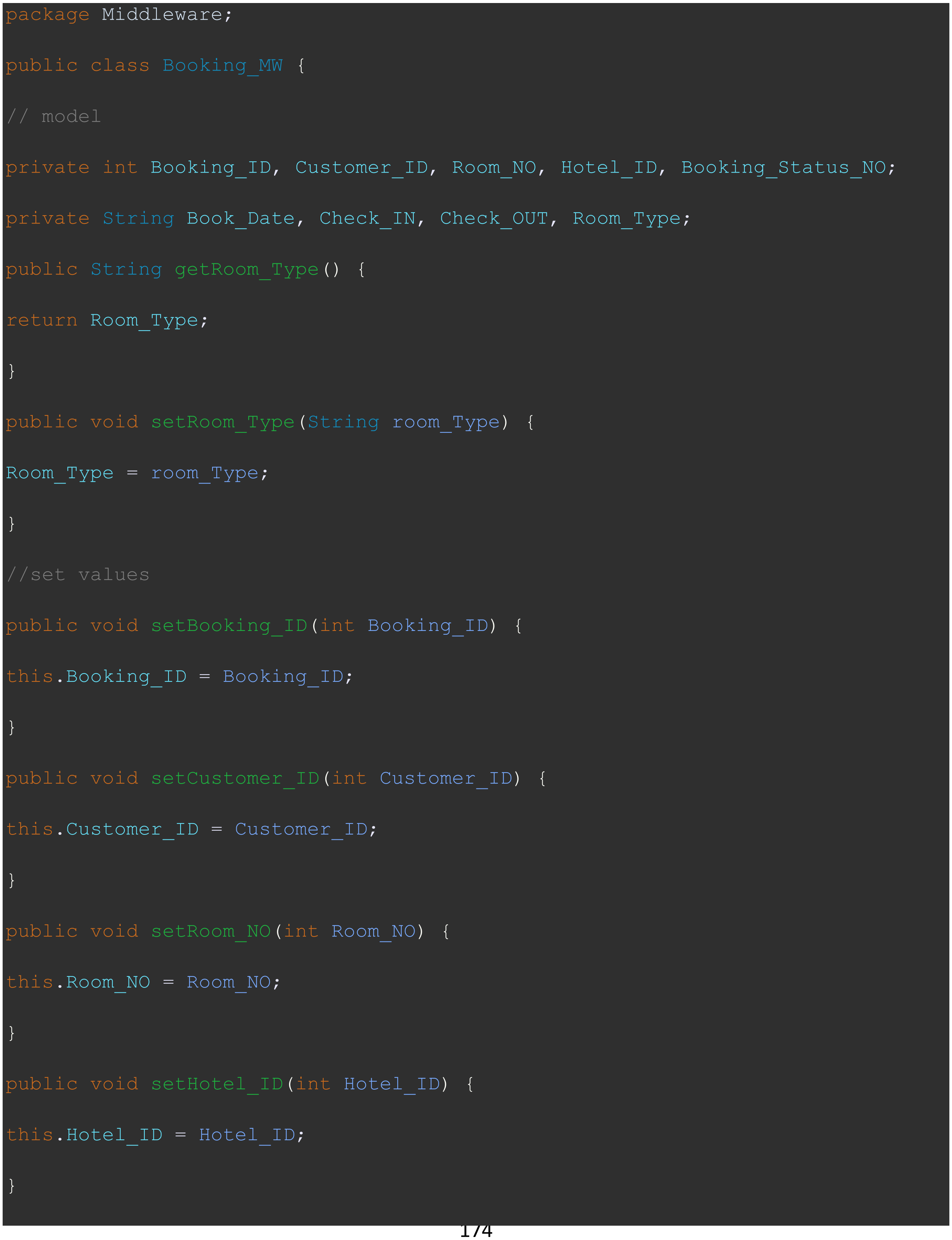
File name: ActiveBooking_MW.java

```
package Middleware;

public class ActiveBooking_MW {

int bid;

private int cid;

public int getCid() {

return cid;

}

public void setCid(int cid) {

this.cid = cid;

}

public int getBid() {

return bid;

}

public void setBid(int bid) {

this.bid = bid;

}

}
```

File name: Amount_MW.java

```
package Middleware;

public class Amounts_MW {

float price;

double total_price;

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

public double getTotal_price() {

return total_price;

}

public void setTotal_price(double total_price) {

this.total_price = total_price;

}

}
```

File name: Booking_MW.java



```
package Middleware;

public class Booking_MW {

// model

private int Booking_ID, Customer_ID, Room_NO, Hotel_ID, Booking_Status_NO;

private String Book_Date, Check_IN, Check_OUT, Room_Type;

public String getRoom_Type() {

return Room_Type;

}

public void setRoom_Type(String room_Type) {

Room_Type = room_Type;

}

//set values

public void setBooking_ID(int Booking_ID) {

this.Booking_ID = Booking_ID;

}

public void setCustomer_ID(int Customer_ID) {

this.Customer_ID = Customer_ID;

}

public void setRoom_NO(int Room_NO) {

this.Room_NO = Room_NO;

}

public void setHotel_ID(int Hotel_ID) {

this.Hotel_ID = Hotel_ID;

}
```

```
public void setBook_Date(String Book_Date) {

this.Book_Date = Book_Date;

}

public void setCheck_IN(String Check_IN) {

this.Check_IN = Check_IN;

}

public void setCheck_OUT(String Check_OUT) {

this.Check_OUT = Check_OUT;

}

public void setBooking_Status_NO(int Booking_Status_NO) {

this.Booking_Status_NO = Booking_Status_NO;

}

//get valuesis

public int getBooking_ID() {

return Booking_ID;

}

public int getCustomer_ID() {

return Customer_ID;

}

public int getRoom_NO() {

return Room_NO;

}

public int getHotel_ID() {

return Hotel_ID;

}
```

```
public String getBook_Date() {

return Book_Date;

}

public String getCheck_IN() {

return Check_IN;

}

public String getCheck_OUT() {

return Check_OUT;

}

public int getBooking_Status_NO() {

return Booking_Status_NO;

}

}
```

File name: Booking_Status.java

```
package Middleware;

public class Booking_Status {

private int Booking_Stauts_NO;

private String Booking_StatusType;

public int getBooking_Stauts_NO() {

return Booking_Stauts_NO;

}

public void setBooking_Stauts_NO(int booking_Stauts_NO) {

Booking_Stauts_NO = booking_Stauts_NO;

}

public String getBooking_StatusType() {

return Booking_StatusType;

}

public void setBooking_StatusType(String booking_StatusType) {

Booking_StatusType = booking_StatusType;

}

}
```

Filename: Customer_MW.java

```
package Middleware;

public class Customer_MW {

//model

private int Customer_ID, UserType_ID=1;

private String First_name, Last_name, Address, Mobile, Email, Gender, DOB,
Password;

//set values

public void setCustomer_ID(int Customer_ID) {

this.Customer_ID = Customer_ID;

}

public void setFirst_name(String First_name) {

this.First_name = First_name;

}

public void setLast_name(String Last_name) {

this.Last_name = Last_name;

}

public void setAddress(String Address) {

this.Address = Address;

}

public void setMobile(String Mobile) {

this.Mobile = Mobile;

}

public void setEmail(String Email) {

this.Email = Email;
```

```
}

public void setGender(String Gender) {

this.Gender = Gender;

}

public void setDOB(String DOB) {

this.DOB = DOB;

}

public void setPassword(String Password) {

this.Password = Password;

}

public void setUserType_ID(int UserType_ID) {

this.UserType_ID = UserType_ID;

}

//get values

public int getCustomer_ID() {

return Customer_ID;

}

public String getFirst_name() {

return First_name;

}

public String getLast_name() {

return Last_name;

}

public String getAddress() {

return Address;
```

```
}

public String getMobile() {

return Mobile;

}

public String getEmail() {

return Email;

}

public String getGender() {

return Gender;

}

public String getDOB() {

return DOB;

}

public String getPassword() {

return Password;

}

public int getUserType_ID() {

return UserType_ID;

}

}
```

Filename: ItemType.java

```
package Middleware;

public class ItemType {

private int Item_ID;

private String Item, Item_Type;

private float Price;

public int getItem_ID() {

return Item_ID;

}

public void setItem_ID(int item_ID) {

Item_ID = item_ID;

}

public String getItem() {

return Item;

}

public void setItem(String item) {

Item = item;

}

public String getItem_Type() {

return Item_Type;

}

public void setItem_Type(String item_Type) {

Item_Type = item_Type;

}
public float getPrice() {

return Price;

}

public void setPrice(float price) {

Price = price;

}

}
```

Filename: Login_MW.java

```
package Middleware;

public class Login_MW {

private int Id, Role;

private String Mobile, Password, Name, newPassword="a";

public String getNewPassword() {

return newPassword;

}

public void setNewPassword(String newPassword) {

this.newPassword = newPassword;

}

public String getName() {

return Name;

}

public void setName(String name) {

Name = name;

}

public int getRole() {

return Role;

}

public void setRole(int role) {

Role = role;

}

public int getId() {

return Id;
}

public void setId(int id) {

Id = id;

}

public String getMobile() {

return Mobile;

}

public void setMobile(String mobile) {

Mobile = mobile;

}

public String getPassword() {

return Password;

}

public void setPassword(String password) {

Password = password;

}

}
```

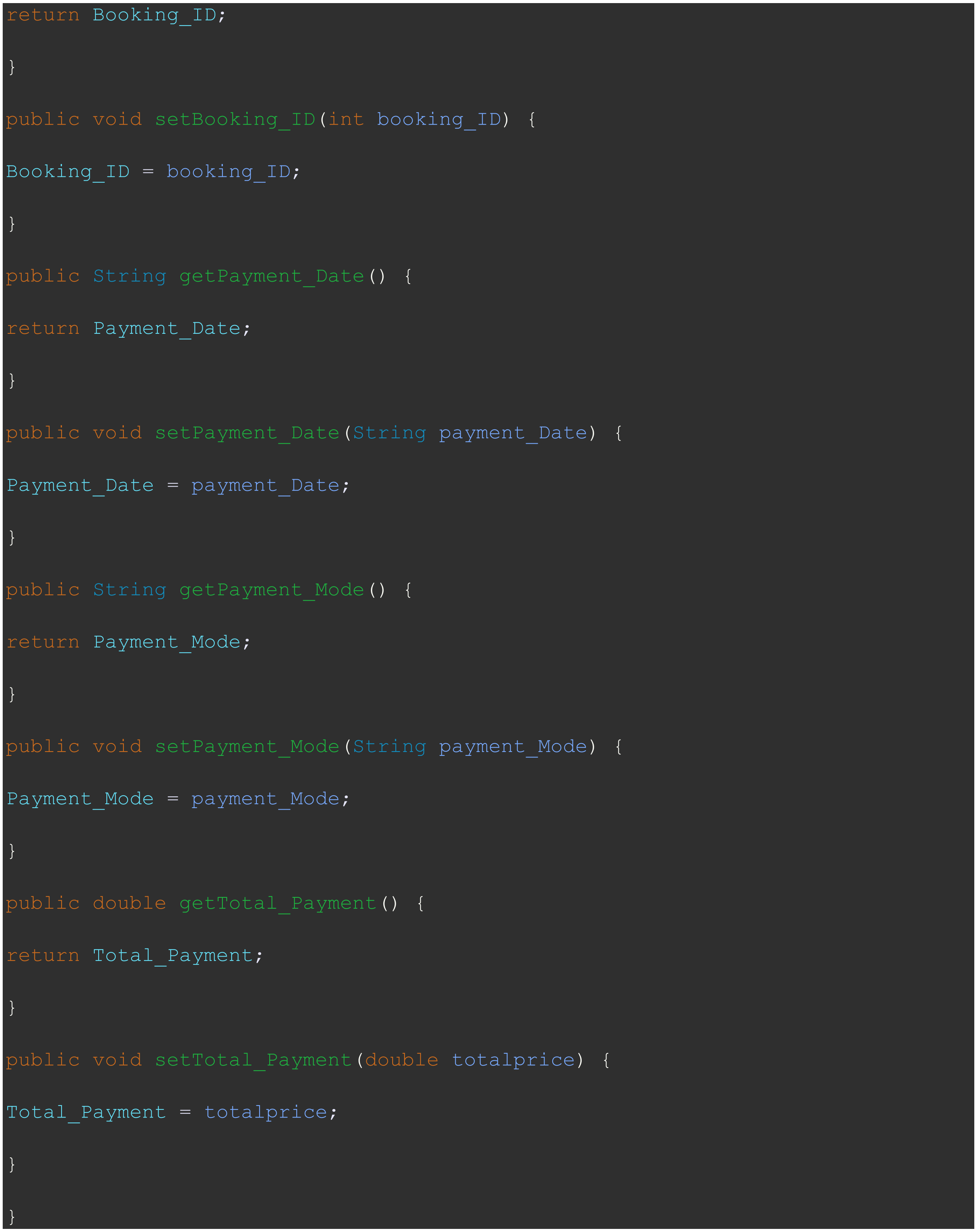
Filename: MealItem_MW.java

```
package Middleware;

public class MealItem_MW {

private int Item_ID;

private String Item, Item_Type;

private float Price;

public float getPrice() {

return Price;

}

public void setPrice(float price) {

Price = price;

}

public int getItem_ID() {

return Item_ID;

}

public void setItem_ID(int item_ID) {

Item_ID = item_ID;

}

public String getItem() {

return Item;

}

public void setItem(String item) {

Item = item;

}

public String getItem_Type() {
```

```
return Item_Type;

}

public void setItem_Type(String item_Type) {

Item_Type = item_Type;

}

}
```

Filename: Payment_MW.java

```
package Middleware;

public class Payment_MW {

private int Payment_ID, Booking_ID;

private String Payment_Date, Payment_Mode,Payment_Status, Name;

public String getName() {

return Name;

}

public void setName(String name) {

Name = name;

}

public String getPayment_Status() {

return Payment_Status;

}

public void setPayment_Status(String payment_Status) {

Payment_Status = payment_Status;

}

private double Total_Payment;

public int getPayment_ID() {

return Payment_ID;

}

public void setPayment_ID(int payment_ID) {

Payment_ID = payment_ID;

}

public int getBooking_ID() {
```



```
return Booking_ID;

}

public void setBooking_ID(int booking_ID) {

Booking_ID = booking_ID;

}

public String getPayment_Date() {

return Payment_Date;

}

public void setPayment_Date(String payment_Date) {

Payment_Date = payment_Date;

}

public String getPayment_Mode() {

return Payment_Mode;

}

public void setPayment_Mode(String payment_Mode) {

Payment_Mode = payment_Mode;

}

public double getTotal_Payment() {

return Total_Payment;

}

public void setTotal_Payment(double totalprice) {

Total_Payment = totalprice;

}

}
```

Filename: Receptionist_CheckIN_MW.java

```
package Middleware;

public class ReceptionistCheckIN_MW {

private int Booking_ID, Customer_ID, Room_NO, Hotel_ID, Booking_Status_NO;

private String Book_Date, Check_IN, Check_OUT, Room_Type, First_Name,
Last_Name,Status_Type;

public int getBooking_ID() {

return Booking_ID;

}

public void setBooking_ID(int booking_ID) {

Booking_ID = booking_ID;

}

public int getCustomer_ID() {

return Customer_ID;

}

public void setCustomer_ID(int customer_ID) {

Customer_ID = customer_ID;

}

public int getRoom_NO() {

return Room_NO;

}

public void setRoom_NO(int room_NO) {

Room_NO = room_NO;

}

public int getHotel_ID() {
```

```
return Hotel_ID;

}

public void setHotel_ID(int hotel_ID) {

Hotel_ID = hotel_ID;

}

public int getBooking_Status_NO() {

return Booking_Status_NO;

}

public void setBooking_Status_NO(int booking_Status_NO) {

Booking_Status_NO = booking_Status_NO;

}

public String getBook_Date() {

return Book_Date;

}

public void setBook_Date(String book_Date) {

Book_Date = book_Date;

}

public String getCheck_IN() {

return Check_IN;

}

public void setCheck_IN(String check_IN) {

Check_IN = check_IN;

}

public String getCheck_OUT() {

return Check_OUT;
```

```
}

public void setCheck_OUT(String check_OUT) {

Check_OUT = check_OUT;

}

public String getRoom_Type() {

return Room_Type;

}

public void setRoom_Type(String room_Type) {

Room_Type = room_Type;

}

public String getFirst_Name() {

return First_Name;

}

public void setFirst_Name(String first_Name) {

First_Name = first_Name;

}

public String getLast_Name() {

return Last_Name;

}

public void setLast_Name(String last_Name) {

Last_Name = last_Name;

}

public String getStatus_Type() {

return Status_Type;

}
```

```
public void setStatus_Type(String status_Type) {

Status_Type = status_Type;

}

}
```

Filename: Receptionist_CheckOUT_MW.java

```
package Middleware;

public class ReceptionistCheckOUT_MW {

private int Booking_ID, Customer_ID, Room_NO, Hotel_ID, Booking_Status_NO;

private String Book_Date, Check_IN, Check_OUT, Room_Type, First_Name,
Last_Name,Status_Type;

public int getBooking_ID() {

return Booking_ID;

}

public void setBooking_ID(int booking_ID) {

Booking_ID = booking_ID;

}

public int getCustomer_ID() {

return Customer_ID;

}

public void setCustomer_ID(int customer_ID) {

Customer_ID = customer_ID;

}

public int getRoom_NO() {

return Room_NO;

}

public void setRoom_NO(int room_NO) {

Room_NO = room_NO;

}

public int getHotel_ID() {
```

```
return Hotel_ID;

}

public void setHotel_ID(int hotel_ID) {

Hotel_ID = hotel_ID;

}

public int getBooking_Status_NO() {

return Booking_Status_NO;

}

public void setBooking_Status_NO(int booking_Status_NO) {

Booking_Status_NO = booking_Status_NO;

}

public String getBook_Date() {

return Book_Date;

}

public void setBook_Date(String book_Date) {

Book_Date = book_Date;

}

public String getCheck_IN() {

return Check_IN;

}

public void setCheck_IN(String check_IN) {

Check_IN = check_IN;

}

public String getCheck_OUT() {

return Check_OUT;
```

```
}

public void setCheck_OUT(String check_OUT) {

Check_OUT = check_OUT;

}

public String getRoom_Type() {

return Room_Type;

}

public void setRoom_Type(String room_Type) {

Room_Type = room_Type;

}

public String getFirst_Name() {

return First_Name;

}

public void setFirst_Name(String first_Name) {

First_Name = first_Name;

}

public String getLast_Name() {

return Last_Name;

}

public void setLast_Name(String last_Name) {

Last_Name = last_Name;

}

public String getStatus_Type() {

return Status_Type;

}
```

```
public void setStatus_Type(String status_Type) {

Status_Type = status_Type;

}

}
```

CIS016-1 – Principles of Programming 2022-2023
Assignment 2 – Hotel Online Customer Booking and Management System
University ID: 2212387 | Full Name: Yaman Maharjan

Filename: Room_MW.java

```
package Middleware;

public class Room_MW {

private int Room_NO;

private String Room_Type, Room_Status;

private float room_price;

public float getRoom_price() {

return room_price;

}

public void setRoom_price(float room_price) {

this.room_price = room_price;

}

public int getRoom_NO() {

return Room_NO;

}

public void setRoom_NO(int room_NO) {

Room_NO = room_NO;

}

public String getRoom_Type() {

return Room_Type;

}

public void setRoom_Type(String room_type) {

Room_Type = room_type;

}

public String getRoom_Status() {
```

```
return Room_Status;

}

public void setRoom_Status(String room_Status) {

Room_Status = room_Status;

}

}
```

Filename: Services_MW.java

```
package Middleware;
public class Services_MW {
private int Service_ID, Item_ID, Booking_ID, Quantity=1;
private String Service_Status;
public String getService_Status() {
return Service_Status;
}
public void setService_Status(String service_Status) {
Service_Status = service_Status;
}
public int getQuantity() {
return Quantity;
}
public void setQuantity(int quantity) {
Quantity = quantity;
}
private String Service_Date;
public int getService_ID() {
return Service_ID;
}
public void setService_ID(int service_ID) {
Service_ID = service_ID;
}
public int getItem_ID() {
```

```
return Item_ID;

}

public void setItem_ID(int item_ID) {

Item_ID = item_ID;

}

public int getBooking_ID() {

return Booking_ID;

}

public void setBooking_ID(int booking_ID) {

Booking_ID = booking_ID;

}

public String getService_Date() {

return Service_Date;

}

public void setService_Date(String service_Date) {

Service_Date = service_Date;

}

}
```

Filename: Staff_MW.java

```
package Middleware;

public class Staff_MW {

private int Staff_ID, Hotel_ID, Role_NO;

private String Staff_First_name, Staff_Last_name, Staff_Address,
Staff_Mobile, Staff_Email, Staff_Gender, Staff_DOB, Staff_Password;

public int getStaff_ID() {

return Staff_ID;

}

public void setStaff_ID(int staff_ID) {

Staff_ID = staff_ID;

}

public int getHotel_ID() {

return Hotel_ID;

}

public void setHotel_ID(int hotel_ID) {

Hotel_ID = hotel_ID;

}

public int getRole_NO() {

return Role_NO;

}

public void setRole_NO(int role_NO) {

Role_NO = role_NO;

}

public String getStaff_First_name() {
```

```
return Staff_First_name;

}

public void setStaff_First_name(String staff_First_name) {

Staff_First_name = staff_First_name;

}

public String getStaff_Last_name() {

return Staff_Last_name;

}

public void setStaff_Last_name(String staff_Last_name) {

Staff_Last_name = staff_Last_name;

}

public String getStaff_Address() {

return Staff_Address;

}

public void setStaff_Address(String staff_Address) {

Staff_Address = staff_Address;

}

public String getStaff_Mobile() {

return Staff_Mobile;

}

public void setStaff_Mobile(String staff_Mobile) {

Staff_Mobile = staff_Mobile;

}

public String getStaff_Email() {

return Staff_Email;
```

```
}

public void setStaff_Email(String staff_Email) {

Staff_Email = staff_Email;

}

public String getStaff_Gender() {

return Staff_Gender;

}

public void setStaff_Gender(String staff_Gender) {

Staff_Gender = staff_Gender;

}

public String getStaff_DOB() {

return Staff_DOB;

}

public void setStaff_DOB(String staff_DOB) {

Staff_DOB = staff_DOB;

}

public String getStaff_Password() {

return Staff_Password;

}

public void setStaff_Password(String staff_Password) {

Staff_Password = staff_Password;

}

}
```

Filename: ViewBooking_MW.java

```
package Middleware;

public class ViewBooking_MW {

private int Booking_ID, Customer_ID, Room_NO, Hotel_ID, Booking_Status_NO;

private String Book_Date, Check_IN, Check_OUT, Room_Type, First_Name,
Last_Name,Status_Type;

public String getStatus_Type() {

return Status_Type;

}

public void setStatus_Type(String status_Type) {

Status_Type = status_Type;

}

public int getBooking_ID() {

return Booking_ID;

}

public void setBooking_ID(int booking_ID) {

Booking_ID = booking_ID;

}

public int getCustomer_ID() {

return Customer_ID;

}

public void setCustomer_ID(int customer_ID) {

Customer_ID = customer_ID;

}

public int getRoom_NO() {
```

```
return Room_NO;

}

public void setRoom_NO(int room_NO) {

Room_NO = room_NO;

}

public int getHotel_ID() {

return Hotel_ID;

}

public void setHotel_ID(int hotel_ID) {

Hotel_ID = hotel_ID;

}

public int getBooking_Status_NO() {

return Booking_Status_NO;

}

public void setBooking_Status_NO(int booking_Status_NO) {

Booking_Status_NO = booking_Status_NO;

}

public String getBook_Date() {

return Book_Date;

}

public void setBook_Date(String book_Date) {

Book_Date = book_Date;

}

public String getCheck_IN() {

return Check_IN;
```

```
}

public void setCheck_IN(String check_IN) {

Check_IN = check_IN;

}

public String getCheck_OUT() {

return Check_OUT;

}

public void setCheck_OUT(String check_OUT) {

Check_OUT = check_OUT;

}

public String getRoom_Type() {

return Room_Type;

}

public void setRoom_Type(String room_Type) {

Room_Type = room_Type;

}

public String getFirst_Name() {

return First_Name;

}

public void setFirst_Name(String first_Name) {

First_Name = first_Name;

}

public String getLast_Name() {

return Last_Name;

}
```

```
public void setLast_Name(String last_Name) {

Last_Name = last_Name;

}

}
```

Filename: ViewServices_MW.java

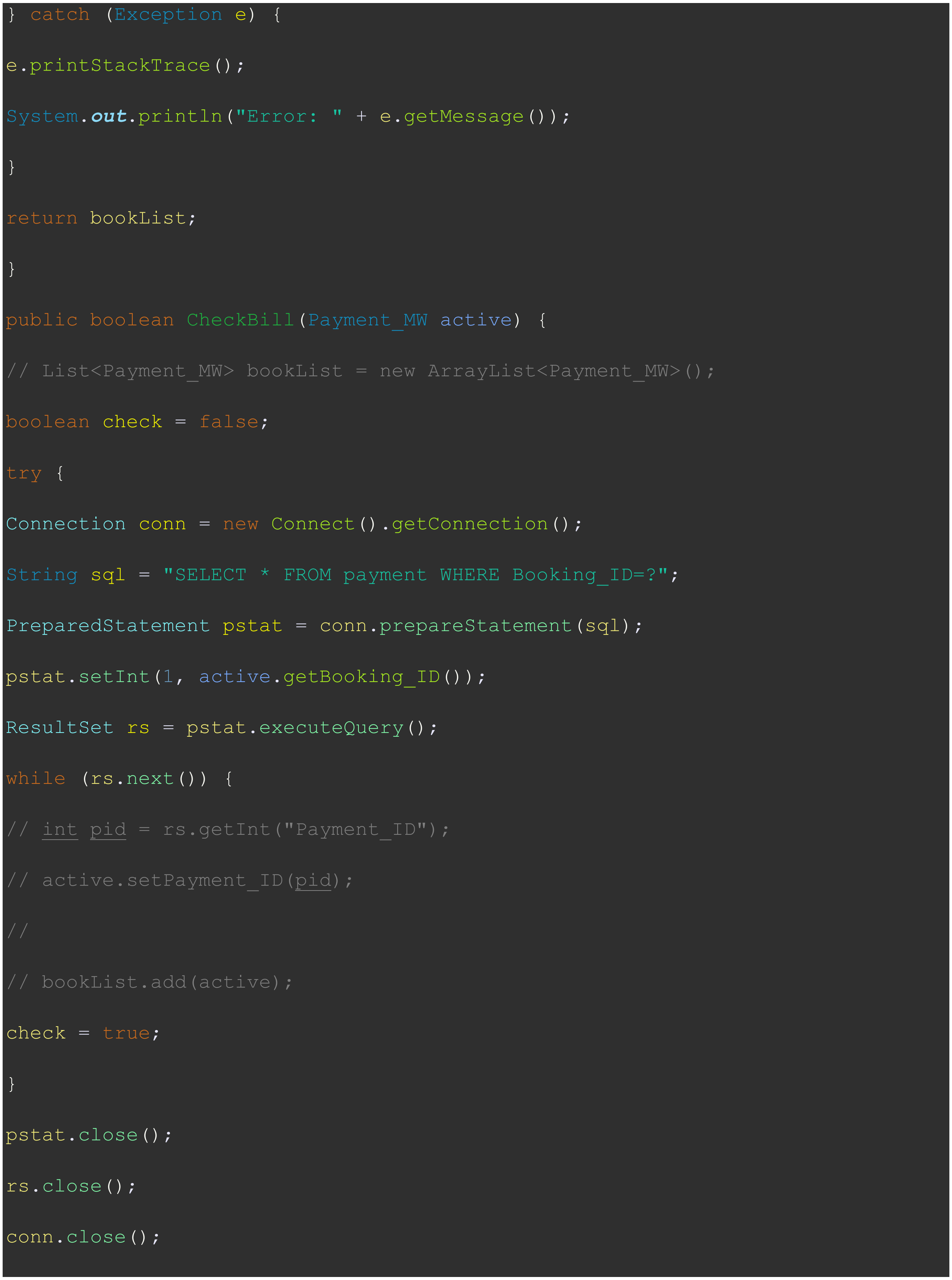
```
package Middleware;

public class ViewServices_MW {

private int Service_ID;

public int getService_ID() {

return Service_ID;

}

public void setService_ID(int service_ID) {

Service_ID = service_ID;

}

private String Item, Service_Status;

public String getService_Status() {

return Service_Status;

}

public void setService_Status(String service_Status) {

Service_Status = service_Status;

}

private int Quantity, Item_ID, BID;

public int getBID() {

return BID;

}

public void setBID(int bID) {

BID = bID;

}

private float Price;
public int getItem_ID() {

return Item_ID;

}

public void setItem_ID(int item_ID) {

Item_ID = item_ID;

}

public String getItem() {

return Item;

}

public void setItem(String item) {

Item = item;

}

public int getQuantity() {

return Quantity;

}

public void setQuantity(int quantity) {

Quantity = quantity;

}

public float getPrice() {

return Price;

}

public void setPrice(float price) {

Price = price;

}

}
```

Controller

Filename: ActiveBookingdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import Middleware.ActiveBooking_MW;

import Middleware.Payment_MW;

public class ActiveBookingdao {

public List<ActiveBooking_MW> ActiveBooking(ActiveBooking_MW active) {

List<ActiveBooking_MW> bookList = new ArrayList<ActiveBooking_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM booking WHERE Customer_ID=? AND
Booking_Status_NO=1";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, active.getCid());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

int bid = rs.getInt("Booking_ID");

active.setBid(bid);

bookList.add(active);

}
```

```
pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public List<Payment_MW> ActiveBill(Payment_MW active) {

List<Payment_MW> bookList = new ArrayList<Payment_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM payment WHERE Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, active.getBooking_ID());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

int pid = rs.getInt("Payment_ID");

active.setPayment_ID(pid);

bookList.add(active);

}

pstat.close();

rs.close();

conn.close();
```



```
} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public boolean CheckBill(Payment_MW active) {

// List<Payment_MW> bookList = new ArrayList<Payment_MW>();

boolean check = false;

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM payment WHERE Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, active.getBooking_ID());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

// int pid = rs.getInt("Payment_ID");

// active.setPayment_ID(pid);

//

// bookList.add(active);

check = true;

}

pstat.close();

rs.close();

conn.close();
```

```
} catch (Exception e) {

check = false;

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return check;

}

}
```

Filename: BookingCRUDdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import Middleware.Booking_MW;

import Middleware.ReceptionistCheckIN_MW;

import Middleware.ReceptionistCheckOUT_MW;

import Middleware.ViewBooking_MW;

public class BookingCRUDdao {

public void MakeBooking(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "INSERT INTO booking (Booking_ID, Customer_ID, Room_NO,
Book_date, Check_IN, Check_OUT, Booking_Status_NO, Hotel_ID, Room_Type)
Values(?,?,?,?,?,?,?,?,?)";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1,booking.getBooking_ID());

pstat.setInt(2, booking.getCustomer_ID());

pstat.setInt(3, booking.getRoom_NO());

pstat.setString(4, booking.getBook_Date());

pstat.setString(5, booking.getCheck_IN());

pstat.setString(6, booking.getCheck_OUT());

pstat.setInt(7, booking.getBooking_Status_NO());
pstat.setInt(8, booking.getHotel_ID());
pstat.setString(9, booking.getRoom_Type());
pstat.executeUpdate();
pstat.close();
conn.close();
}
catch(Exception e){
System.out.println("Error: " + e.getMessage());
}
}
public void UpdateBooking(Booking_MW booking) {
try {
Connection conn = new Connect().getConnection();
String sql = "Update booking set Check_IN=?, Check_OUT=? WHERE Booking_ID=?";
PreparedStatement pstat = conn.prepareStatement(sql);
pstat.setString(1, booking.getCheck_IN());
pstat.setString(2, booking.getCheck_OUT());
pstat.setInt(3, booking.getBooking_ID());
pstat.executeUpdate();
pstat.close();
conn.close();
}
catch(Exception e){
System.out.println("Error: " + e.getMessage());
}
```

```
}
public void CancelBooking(Booking_MW booking) {
try {
Connection conn = new Connect().getConnection();
String sql = "Update booking set Booking_Status_NO=4 WHERE Booking_ID=?";
PreparedStatement pstat = conn.prepareStatement(sql);
// pstat.setString(1, booking.getCheck_IN());
// pstat.setString(2, booking.getCheck_OUT());
pstat.setInt(1, booking.getBooking_ID());
pstat.executeUpdate();
pstat.close();
conn.close();
}
catch(Exception e){
System.out.println("Error: " + e.getMessage());
}
}
public List<ViewBooking_MW> ViewBooking(int id) {
List<ViewBooking_MW> bookList = new ArrayList<ViewBooking_MW>();
try {
Connection conn = new Connect().getConnection();
String sql = "SELECT b.*, c.First_Name, c.Last_Name, s.Status_Type FROM booking b LEFT JOIN customer c ON b.Customer_ID = c.Customer_ID LEFT JOIN bookingstatus s ON b.Booking_Status_NO=s.Status_NO WHERE b.Customer_ID=?";
PreparedStatement pstat = conn.prepareStatement(sql);
pstat.setInt(1, id);
```

```
ResultSet rs = pstat.executeQuery();
while (rs.next()) {
ViewBooking_MW booking = new ViewBooking_MW();
int bid = rs.getInt("Booking_ID");
int cid = rs.getInt("Customer_ID");
int room_no = rs.getInt("Room_NO");
String book_date = rs.getString("Book_date");
String check_in = rs.getString("Check_IN");
String check_out = rs.getString("Check_OUT");
// int booking_status_no = rs.getInt("Booking_Status_NO");
String booking_status = rs.getString("Status_Type");
String first_name = rs.getString("First_Name");
String last_name = rs.getString("Last_Name");
booking.setBooking_ID(bid);
booking.setCustomer_ID(cid);
booking.setRoom_NO(room_no);
booking.setBook_Date(book_date);
booking.setCheck_IN(check_in);
booking.setCheck_OUT(check_out);
booking.setStatus_Type(booking_status);
booking.setFirst_Name(first_name);
booking.setLast_Name(last_name);
bookList.add(booking);
}
pstat.close();
```

```
rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public List<ReceptionistCheckIN_MW> ViewPendingBooking() {

List<ReceptionistCheckIN_MW> bookList = new
ArrayList<ReceptionistCheckIN_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT b.*, c.First_Name, c.Last_Name, s.Status_Type FROM
booking b LEFT JOIN customer c ON b.Customer_ID = c.Customer_ID LEFT JOIN
bookingstatus s ON b.Booking_Status_NO=s.Status_NO WHERE s.Status_Type IN
(?,?)";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "Pending");

pstat.setString(2, "Booked");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

ReceptionistCheckIN_MW booking = new ReceptionistCheckIN_MW();

int bid = rs.getInt("Booking_ID");

int cid = rs.getInt("Customer_ID");

int room_no = rs.getInt("Room_NO");

String book_date = rs.getString("Book_date");
String check_in = rs.getString("Check_IN");
String check_out = rs.getString("Check_OUT");
String room_type = rs.getString("Room_Type");
String booking_status = rs.getString("Status_Type");
String first_name = rs.getString("First_Name");
String last_name = rs.getString("Last_Name");
booking.setBooking_ID(bid);
booking.setCustomer_ID(cid);
booking.setRoom_NO(room_no);
booking.setBook_Date(book_date);
booking.setCheck_IN(check_in);
booking.setCheck_OUT(check_out);
booking.setRoom_Type(room_type);
booking.setStatus_Type(booking_status);
booking.setFirst_Name(first_name);
booking.setLast_Name(last_name);
bookList.add(booking);
}
pstat.close();
rs.close();
conn.close();
} catch (Exception e) {
e.printStackTrace();
System.out.println("Error: " + e.getMessage());
}
```

```
return bookList;

}

public void AllocateRoom(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update booking set Room_NO=?, Booking_Status_NO=? WHERE Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, booking.getRoom_NO());

pstat.setInt(2, 5);

pstat.setInt(3, booking.getBooking_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public void ConfirmBooking(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update booking set Room_NO=?, Booking_Status_NO=? WHERE Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, booking.getRoom_NO());
pstat.setInt(2, 1);

pstat.setInt(3, booking.getBooking_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public void Ckeck_OUTBooking(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update booking set Booking_Status_NO=? WHERE Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, 3);

pstat.setInt(2, booking.getBooking_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}
```

```
public List<ReceptionistCheckOUT_MW> ViewActiveBooking() {

List<ReceptionistCheckOUT_MW> bookList = new
ArrayList<ReceptionistCheckOUT_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT b.*, c.First_Name, c.Last_Name, s.Status_Type FROM
booking b LEFT JOIN customer c ON b.Customer_ID = c.Customer_ID LEFT JOIN
bookingstatus s ON b.Booking_Status_NO=s.Status_NO WHERE s.Status_Type=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "Active");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

ReceptionistCheckOUT_MW booking = new ReceptionistCheckOUT_MW();

int bid = rs.getInt("Booking_ID");

int cid = rs.getInt("Customer_ID");

int room_no = rs.getInt("Room_NO");

String book_date = rs.getString("Book_date");

String check_in = rs.getString("Check_IN");

String check_out = rs.getString("Check_OUT");

String room_type = rs.getString("Room_Type");

String booking_status = rs.getString("Status_Type");

String first_name = rs.getString("First_Name");

String last_name = rs.getString("Last_Name");

booking.setBooking_ID(bid);

booking.setCustomer_ID(cid);

booking.setRoom_NO(room_no);
booking.setBook_Date(book_date);

booking.setCheck_IN(check_in);

booking.setCheck_OUT(check_out);

booking.setRoom_Type(room_type);

booking.setStatus_Type(booking_status);

booking.setFirst_Name(first_name);

booking.setLast_Name(last_name);

bookList.add(booking);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public List<ReceptionistCheckOUT_MW> ViewAllBooking() {

List<ReceptionistCheckOUT_MW> bookList = new
ArrayList<ReceptionistCheckOUT_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT b.*, c.First_Name, c.Last_Name, s.Status_Type FROM
booking b LEFT JOIN customer c ON b.Customer_ID = c.Customer_ID LEFT JOIN
bookingstatus s ON b.Booking_Status_NO=s.Status_NO WHERE s.Status_Type IN
(?,?)";
```

```
PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "Cancel");

pstat.setString(2, "COMPLETE");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

ReceptionistCheckOUT_MW booking = new ReceptionistCheckOUT_MW();

int bid = rs.getInt("Booking_ID");

int cid = rs.getInt("Customer_ID");

int room_no = rs.getInt("Room_NO");

String book_date = rs.getString("Book_date");

String check_in = rs.getString("Check_IN");

String check_out = rs.getString("Check_OUT");

String room_type = rs.getString("Room_Type");

String booking_status = rs.getString("Status_Type");

String first_name = rs.getString("First_Name");

String last_name = rs.getString("Last_Name");

booking.setBooking_ID(bid);

booking.setCustomer_ID(cid);

booking.setRoom_NO(room_no);

booking.setBook_Date(book_date);

booking.setCheck_IN(check_in);

booking.setCheck_OUT(check_out);

booking.setRoom_Type(room_type);

booking.setStatus_Type(booking_status);

booking.setFirst_Name(first_name);
booking.setLast_Name(last_name);

bookList.add(booking);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

}
```

Filename: Connect.java

```
package DAO;

import java.sql.Connection;

import java.sql.DriverManager;

public class Connect {

String dbURL = "jdbc:mysql://localhost:3306/hotelbookingsystem";

String username = "root";

String password = "";

Connection conn = null;

public Connection getConnection() {

try {

conn = DriverManager.getConnection(dbURL,username,password);

}

catch(Exception e) {

System.out.println("Error: "+e.getMessage());

}

return conn;

}

}
```

Filename: CRUDdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import Middleware.Customer_MW;

import Middleware.Staff_MW;

public class CRUDdao {

public void RegisterCustomer(Customer_MW customer) {

try {

Connection conn = new Connect().getConnection();

String sql = "INSERT INTO Customer (Customer_ID,First_name, Last_name,
Address, Mobile, Email, Gender, DOB, Password,UserType_ID)
Values(?,?,?,?,?,?,?,?,?,?)";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1,customer.getCustomer_ID());

pstat.setString(2, customer.getFirst_name());

pstat.setString(3, customer.getLast_name());

pstat.setString(4, customer.getAddress());

pstat.setString(5, customer.getMobile());

pstat.setString(6, customer.getEmail());

pstat.setString(7, customer.getGender());

pstat.setString(8, customer.getDOB());

pstat.setString(9, customer.getPassword());
pstat.setInt(10, customer.getUserType_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public void CustomerUpdate(Customer_MW customer) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update customer set First_name=?, Last_name=?, Address=?,
Mobile=?, Email=?, Gender=?, DOB=?, Password=? WHERE Customer_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, customer.getFirst_name());

pstat.setString(2, customer.getLast_name());

pstat.setString(3, customer.getAddress());

pstat.setString(4, customer.getMobile());

pstat.setString(5, customer.getEmail());

pstat.setString(6, customer.getGender());

pstat.setString(7, customer.getDOB());

pstat.setString(8, customer.getPassword());

pstat.setInt(9, customer.getCustomer_ID());

pstat.executeUpdate();
pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public List<Customer_MW> Viewprofile(int id) {

List<Customer_MW> bookList = new ArrayList<Customer_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM customer WHERE Customer_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, id);

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

Customer_MW booking = new Customer_MW();

int cid = rs.getInt("Customer_ID");

String first_name = rs.getString("First_Name");

String last_name = rs.getString("Last_Name");

String address = rs.getString("Address");

String mobile = rs.getString("Mobile");

String email = rs.getString("Email");

String gender = rs.getString("Gender");

String dob = rs.getString("DOB");
String password = rs.getString("Password");

booking.setCustomer_ID(cid);

booking.setFirst_name(first_name);

booking.setLast_name(last_name);

booking.setAddress(address);

booking.setMobile(mobile);

booking.setEmail(email);

booking.setGender(gender);

booking.setDOB(dob);

booking.setPassword(password);

bookList.add(booking);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public List<Staff_MW> StaffViewprofile(int id) {

List<Staff_MW> bookList = new ArrayList<Staff_MW>();

try {

Connection conn = new Connect().getConnection();
```

```
String sql = "SELECT * FROM staff WHERE Staff_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, id);

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

Staff_MW booking = new Staff_MW();

int cid = rs.getInt("Staff_ID");

String first_name = rs.getString("Staff_First_Name");

String last_name = rs.getString("Staff_Last_Name");

String address = rs.getString("Staff_Address");

String mobile = rs.getString("Staff_Mobile");

String email = rs.getString("Staff_Email");

String gender = rs.getString("Staff_Gender");

String dob = rs.getString("Staff_DOB");

String password = rs.getString("Staff_Password");

booking.setStaff_ID(id);

booking.setStaff_First_name(first_name);

booking.setStaff_Last_name(last_name);

booking.setStaff_Address(address);

booking.setStaff_Mobile(mobile);

booking.setStaff_Email(email);

booking.setStaff_Gender(gender);

booking.setStaff_DOB(dob);

booking.setStaff_Password(password);

bookList.add(booking);
}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public void StaffUpdate(Staff_MW staff) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update Staff set Staff_First_name=?, Staff_Last_name=?,
Staff_Address=?, Staff_Mobile=?, Staff_Email=?, Staff_Gender=?, Staff_DOB=?,
Staff_Password=? WHERE Staff_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, staff.getStaff_First_name());

pstat.setString(2, staff.getStaff_Last_name());

pstat.setString(3, staff.getStaff_Address());

pstat.setString(4, staff.getStaff_Mobile());

pstat.setString(5, staff.getStaff_Email());

pstat.setString(6, staff.getStaff_Gender());

pstat.setString(7, staff.getStaff_DOB());

pstat.setString(8, staff.getStaff_Password());

pstat.setInt(9, staff.getStaff_ID());
pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

}
```

Filename: ItemJDBCdao.java

```
package DAO;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import Middleware.MealItem_MW;
import Middleware.ViewServices_MW;
public class ItemJDBCdao {
public List<ViewServices_MW> Viewitem(int id) {
List<ViewServices_MW> itemList = new ArrayList<ViewServices_MW>();
try {
Connection conn = new Connect().getConnection();
String sql = "SELECT s.*, i.Item, i.Price FROM services s LEFT JOIN itemtype i ON s.Item_ID = i.Item_ID WHERE s.Booking_ID=?";
PreparedStatement pstat = conn.prepareStatement(sql);
pstat.setInt(1, id);
ResultSet rs = pstat.executeQuery();
while (rs.next()) {
ViewServices_MW item = new ViewServices_MW();
String itemname = rs.getString("Item");
String status = rs.getString("Status");
int quantity = rs.getInt("Quantity");
float price = rs.getFloat("Price");
```

```
item.setItem(itemname);

item.setQuantity(quantity);

item.setPrice(price);

item.setService_Status(status);

itemList.add(item);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return itemList;

}

public List<MealItem_MW> Mealitem() {

List<MealItem_MW> MealList = new ArrayList<MealItem_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT Item, Price, Item_ID FROM itemtype WHERE Item_Type = ?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "Meals");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

MealItem_MW meal = new MealItem_MW();
```

```
String itemname = rs.getString("Item");

float price = rs.getFloat("Price");

int item_id = rs.getInt("Item_ID");

meal.setItem(itemname);

meal.setPrice(price);

meal.setItem_ID(item_id);

MealList.add(meal);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return MealList;

}

public List<MealItem_MW> Breakfastitem() {

List<MealItem_MW> MealList = new ArrayList<MealItem_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT Item, Price, Item_ID FROM itemtype WHERE Item_Type = ?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "Breakfast");

ResultSet rs = pstat.executeQuery();
```

```
while (rs.next()) {

MealItem_MW meal = new MealItem_MW();

String itemname = rs.getString("Item");

float price = rs.getFloat("Price");

int item_id = rs.getInt("Item_ID");

meal.setItem(itemname);

meal.setPrice(price);

meal.setItem_ID(item_id);

MealList.add(meal);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return MealList;

}

public List<MealItem_MW> Drinkitem() {

List<MealItem_MW> MealList = new ArrayList<MealItem_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT Item, Price, Item_ID FROM itemtype WHERE Item_Type = ?";

PreparedStatement pstat = conn.prepareStatement(sql);
```

```
pstat.setString(1, "Drinks");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

MealItem_MW meal = new MealItem_MW();

String itemname = rs.getString("Item");

float price = rs.getFloat("Price");

int item_id = rs.getInt("Item_ID");

meal.setItem(itemname);

meal.setPrice(price);

meal.setItem_ID(item_id);

MealList.add(meal);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return MealList;

}

public List<MealItem_MW> Otheritem() {

List<MealItem_MW> MealList = new ArrayList<MealItem_MW>();

try {

Connection conn = new Connect().getConnection();
```

```
String sql = "SELECT Item, Price, Item_ID FROM itemtype WHERE Item_Type = ?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "amenities");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

MealItem_MW meal = new MealItem_MW();

String itemname = rs.getString("Item");

float price = rs.getFloat("Price");

int item_id = rs.getInt("Item_ID");

meal.setItem(itemname);

meal.setPrice(price);

meal.setItem_ID(item_id);

MealList.add(meal);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return MealList;

}

public List<ViewServices_MW> ViewAllitem() {

List<ViewServices_MW> itemList = new ArrayList<ViewServices_MW>();
```

```
try {

Connection conn = new Connect().getConnection();

String sql = "SELECT s.*, i.Item, i.Price FROM services s LEFT JOIN itemtype
i ON s.Item_ID = i.Item_ID WHERE Status=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, "PENDING");

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

ViewServices_MW item = new ViewServices_MW();

String itemname = rs.getString("Item");

int quantity = rs.getInt("Quantity");

int sid = rs.getInt("Service_ID");

float price = rs.getFloat("Price");

String status = rs.getString("Status");

item.setItem(itemname);

item.setQuantity(quantity);

item.setPrice(price);

item.setService_Status(status);

item.setService_ID(sid);

itemList.add(item);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {
```

```
e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return itemList;

}

}
```

Filename: Logindao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import Middleware.Login_MW;

public class Logindao {

public void Login() {

}

public boolean checkUserAuth(Login_MW userAuth) {

boolean autherize = false;

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM customer where Mobile=? and Password=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, userAuth.getMobile());

pstat.setString(2, userAuth.getPassword());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

autherize = true;

int id = rs.getInt("Customer_ID");

String Name = rs.getString("First_Name");

String LastName = rs.getString("Last_Name");

String password = rs.getString("Password");

userAuth.setId(id);
userAuth.setNewPassword(password);

userAuth.setName(Name+" "+LastName);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

autherize = false;

}

return autherize;

}

public boolean checkStaffAuth(Login_MW userAuth) {

boolean autherize = false;

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT * FROM staff where Staff_Mobile=? and Staff_Password
LIKE BINARY ?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, userAuth.getMobile());

pstat.setString(2, userAuth.getPassword());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

autherize = true;
```

```
int id = rs.getInt("Staff_ID");
int role = rs.getInt("Role_NO");
// Login_MW lg = new Login_MW();
userAuth.setId(id);
userAuth.setRole(role);
String Name = rs.getString("Staff_First_Name");
String LastName = rs.getString("Staff_Last_Name");
String password = rs.getString("Staff_Password");
userAuth.setName(Name+" "+LastName);
userAuth.setNewPassword(password);
// System.out.println(userAuth.getId());
// System.out.println(Name);
}
pstat.close();
rs.close();
conn.close();
} catch (Exception e) {
e.printStackTrace();
System.out.println("Error: " + e.getMessage());
autherize = false;
}
return autherize;
}
}
```

Filename: OrderJDBCdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import Middleware.Services_MW;

public class OrderJDBCdao {

public void MakeOrder(Services_MW service) {

try {

Connection conn = new Connect().getConnection();

String sql = "INSERT INTO services (Service_ID, Booking_ID, Date, Item_ID,
Quantity,Status) Values(?,?,?,?,?,?)";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1,service.getService_ID());

pstat.setInt(2, service.getBooking_ID());

pstat.setString(3, service.getService_Date());

pstat.setInt(4, service.getItem_ID());

pstat.setInt(5, service.getQuantity());

pstat.setString(6, "Pending");

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}
}

public void ConfirmOrder(Services_MW service) {

try {

Connection conn = new Connect().getConnection();

String sql = "UPDATE services set Status=? WHERE Service_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1,"Complete");

pstat.setInt(2, service.getService_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

}
```

Filename: Paymentdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import Middleware.Payment_MW;

public class Paymentdao {

public void Bill(Payment_MW payment) {

try {

Connection conn = new Connect().getConnection();

String sql = "INSERT INTO payment (Payment_ID ,Booking_ID, Date, Payment_Mode, Total_Payment, Payment_Status) Values(?,?,?,?,?,?)";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, payment.getPayment_ID());

pstat.setInt(2, payment.getBooking_ID());

pstat.setString(3, payment.getPayment_Date());

pstat.setString(4, payment.getPayment_Mode());

pstat.setDouble(5, payment.getTotal_Payment());

pstat.setString(6, payment.getPayment_Status());

pstat.executeUpdate();

pstat.close();

conn.close();

}
```

```
catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public void UpdateBill(Payment_MW payment) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update payment set Payment_Mode=?, Payment_Status=? WHERE
Payment_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, payment.getPayment_Mode());

pstat.setString(2, "Paid");

pstat.setInt(3, payment.getPayment_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public void Updateoldbill(Payment_MW payment) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update payment set Date=?, Total_Payment=? WHERE Payment_ID=?";
```

```
PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setString(1, payment.getPayment_Date());

pstat.setDouble(2, payment.getTotal_Payment());

pstat.setInt(3, payment.getPayment_ID());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

public List<Payment_MW> ViewPayment(Payment_MW payment) {

List<Payment_MW> bookList = new ArrayList<Payment_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT p.* FROM payment p LEFT JOIN booking b ON p.Booking_ID = b.Booking_ID WHERE p.Booking_ID=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, payment.getBooking_ID());

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

// ReceptionistCheckIN_MW booking = new ReceptionistCheckIN_MW();

int pid = rs.getInt("Payment_ID");

int bid = rs.getInt("Booking_ID");
String date = rs.getString("Date");

String mode = rs.getString("Payment_Mode");

double price = rs.getDouble("Total_Payment");

String status = rs.getString("Payment_Status");

payment.setPayment_ID(pid);

payment.setBooking_ID(bid);

payment.setPayment_Date(date);

payment.setPayment_Mode(mode);

payment.setTotal_Payment(price);

payment.setPayment_Status(status);

bookList.add(payment);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

}
```

Filename: RoomJDBCdao.java

```
package DAO;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.util.ArrayList;

import java.util.List;

import Middleware.Booking_MW;

import Middleware.Room_MW;

public class RoomJDBCdao {

public List<Room_MW> ViewAvailableBooking() {

List<Room_MW> bookList = new ArrayList<Room_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT r.*, s.Room_Status FROM room r LEFT JOIN room_status s ON r.Status=s.Status_NO";

PreparedStatement pstat = conn.prepareStatement(sql);

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

Room_MW booking = new Room_MW();

int room_no = rs.getInt("Room_NO");

String room_type = rs.getString("Room_Type");

String room_status = rs.getString("Room_Status");

booking.setRoom_NO(room_no);

booking.setRoom_Type(room_type);
booking.setRoom_Status(room_status);

bookList.add(booking);

}

pstat.close();

rs.close();

conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public void ClosedRoomStatus(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update room set Status=? WHERE Room_NO=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, 3);

pstat.setInt(2, booking.getRoom_NO());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());
```

```
}

}

public List<Room_MW> ViewRoomPrice(int roomno) {

List<Room_MW> bookList = new ArrayList<Room_MW>();

try {

Connection conn = new Connect().getConnection();

String sql = "SELECT r.*, s.Price, b.Room_Status FROM room r LEFT JOIN
roomtype s ON r.Room_Type=s.Room_Type LEFT JOIN room_status b ON
r.Status=b.Status_NO WHERE Room_NO=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, roomno);

ResultSet rs = pstat.executeQuery();

while (rs.next()) {

Room_MW booking = new Room_MW();

int room_no = rs.getInt("Room_NO");

String room_type = rs.getString("Room_Type");

String room_status = rs.getString("Room_Status");

float room_price = rs.getFloat("Price");

booking.setRoom_NO(room_no);

booking.setRoom_Type(room_type);

booking.setRoom_Status(room_status);

booking.setRoom_price(room_price);

bookList.add(booking);

}

pstat.close();

rs.close();
```

```
conn.close();

} catch (Exception e) {

e.printStackTrace();

System.out.println("Error: " + e.getMessage());

}

return bookList;

}

public void OpenRoomStatus(Booking_MW booking) {

try {

Connection conn = new Connect().getConnection();

String sql = "Update room set Status=? WHERE Room_NO=?";

PreparedStatement pstat = conn.prepareStatement(sql);

pstat.setInt(1, 1);

pstat.setInt(2, booking.getRoom_NO());

pstat.executeUpdate();

pstat.close();

conn.close();

}

catch(Exception e){

System.out.println("Error: " + e.getMessage());

}

}

}
```

Filename: registervalidation.java

```
package DAO;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class registervalidation {

public boolean First_Name(String firstname) {

boolean result = false;

String regex = "^[A-Z][a-z]{2,10}$";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(firstname);

result = m.matches();

return result;

}

public boolean Last_Name(String lastname) {

boolean result = false;

String regex = "^[A-Z][a-z]{2,10}$";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(lastname);

result = m.matches();

return result;

}

public boolean Gender(String gender) {

boolean result = false;

if (gender.equals("Male")||gender.equals("Female")||gender.equals("Other")) {

result = true;
} else {

result = false;

}

return result;

}

public boolean Mobile(String mobile) {

boolean result = false;

String regex = "^[9]{1}[678]{1}[0-9]{8}$";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(mobile);

result = m.matches();

return result;

}

public boolean Email(String st) {

boolean result = false;

String regex = "^[a-z]{1}[a-z0-9_.]{5,20}[@]{1}[a-z]{5,10}[.]{1}[com]{2,3}";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(st);

result = m.matches();

return result;

}

public boolean UserName(String st) {

boolean result = false;

String regex = "^[A-Z]{1}[a-z]{2,10}[0-9\\_\\.]{1,20}";

Pattern p = Pattern.compile(regex);
```

```
Matcher m = p.matcher(st);

result = m.matches();

return result;

}

public boolean Password(String st) {

boolean result = false;

String regex = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)[a-zA-Z\\d]{8,}$";

Pattern p = Pattern.compile(regex);

Matcher m = p.matcher(st);

result = m.matches();

return result;

}

}
```