



UE21CS352B - Object Oriented Analysis & Design using Java

Mini Project Report

“HOTEL MANAGEMENT SYSTEM USING JAVA ”

Submitted by:

Korey Varun	PES1UG21CS283
Krishna Sudarshan	PES1UG21CS289
Mahima	PES2UG21CS268

6th Semester 'E' Section

Prof. Bhargavi Mokashi

Asst. Professor
Dept. of CSE
PES University

January - May 2024

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY**

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PROBLEM STATEMENT:

The class diagram you provided shows a high-level overview of the classes involved in a hotel management system, but it doesn't include all the details needed to fully understand the system's functionality. Here's what can be gleaned from the class diagram:

- **Classes:** The class diagram includes the following classes:
 - **Controller:** This class seems to be the main class that interacts with other classes in the system. It has methods for handling restaurant operations, hotel operations, and admin operations.
 - **Customer:** This class likely represents a customer of the hotel. It has attributes for customer name, address, and phone number.
 - **Restaurant:** This class likely represents the hotel's restaurant. It has a method for displaying the menu.
 - **Admin:** This class likely represents an administrator of the system. It has attributes for username and password.
 - **Room:** This class likely represents a room in the hotel.
 - **Hotel:** This class likely represents the hotel itself. It has a method for displaying rooms.
- **Relationships:** The class diagram shows some of the relationships between the classes. For example, the Controller class has a relationship with the Customer, Restaurant, Admin, and Hotel classes. This suggests that the Controller class can interact with these other classes to perform operations.
- **Missing Information:** The class diagram doesn't show some important details, such as the methods of the Customer, Admin, Room, and Hotel classes. It also doesn't show how these classes interact with each other to perform specific tasks, such as making a reservation or checking out a guest.

Here's how you can improve the problem statement based on the class diagram:

- The system needs more clearly defined classes and their attributes to better represent a hotel management system.
- The relationships between the classes need to be fleshed out to show how they interact with each other to perform the hotel management system's functionalities.

Overall, the class diagram provides a starting point for understanding the hotel management system, but it needs more detail to fully capture the system's functionality.

DESCRIPTION:

Imagine a bustling hotel with guests coming and going, all managed seamlessly with a user-friendly system. This is where our Java Swing application with Model-View-Controller (MVC) architecture comes in, acting as the digital backbone for a robust hotel management system.

The system caters to three key user roles: admin, restaurant employee, and customers. Managers, the masterminds behind the scenes, wield the power to edit room and dish prices, ensuring smooth operations and profitability. They can also generate reports to gain valuable insights into the hotel's performance.

Receptionists, the frontline heroes, handle the crucial tasks of managing reservations. They can check room availability, facilitate online bookings for customers, and handle the check-in and check-out processes for arriving and departing guests.

Customers, eager to plan their getaways, can use the system to browse available rooms and their rates. The system displays clear information, allowing them to make informed choices and book their dream stay with ease.

Behind the scenes, the MVC architecture plays a crucial role in keeping the system organized and efficient. The Model layer acts as the data powerhouse, storing information about rooms, customers, reservations, and more. While a database connection isn't included in this Swing implementation, it's where the system would typically retrieve and store this vital information.

The View layer, built with Swing components, brings the system to life. Imagine user-friendly interfaces tailored to each user role. For managers, panels might display editable price lists and report generation options. Receptionists might have access to guest search bars, room assignment dropdowns, and check-in/out buttons. Customers could utilize calendars to select dates, dropdown menus to choose room types, and a prominent "Reserve" button to confirm their bookings. JPanels, JTextFields, JButtons, and JTables are just some of the Swing components that can be used to create these intuitive interfaces.

Finally, the Controller acts as the bridge between the View and the Model. When a manager clicks the "Update Price" button, the Controller intercepts this action, retrieves

the new price from the View, and communicates it to the Model for processing. Similarly, when a customer clicks "Reserve", the Controller checks availability with the Model, updates the database (not included here) if successful, and informs the View to display a confirmation message. This constant communication between the Controller, View, and Model ensures a smooth user experience and efficient system operation.

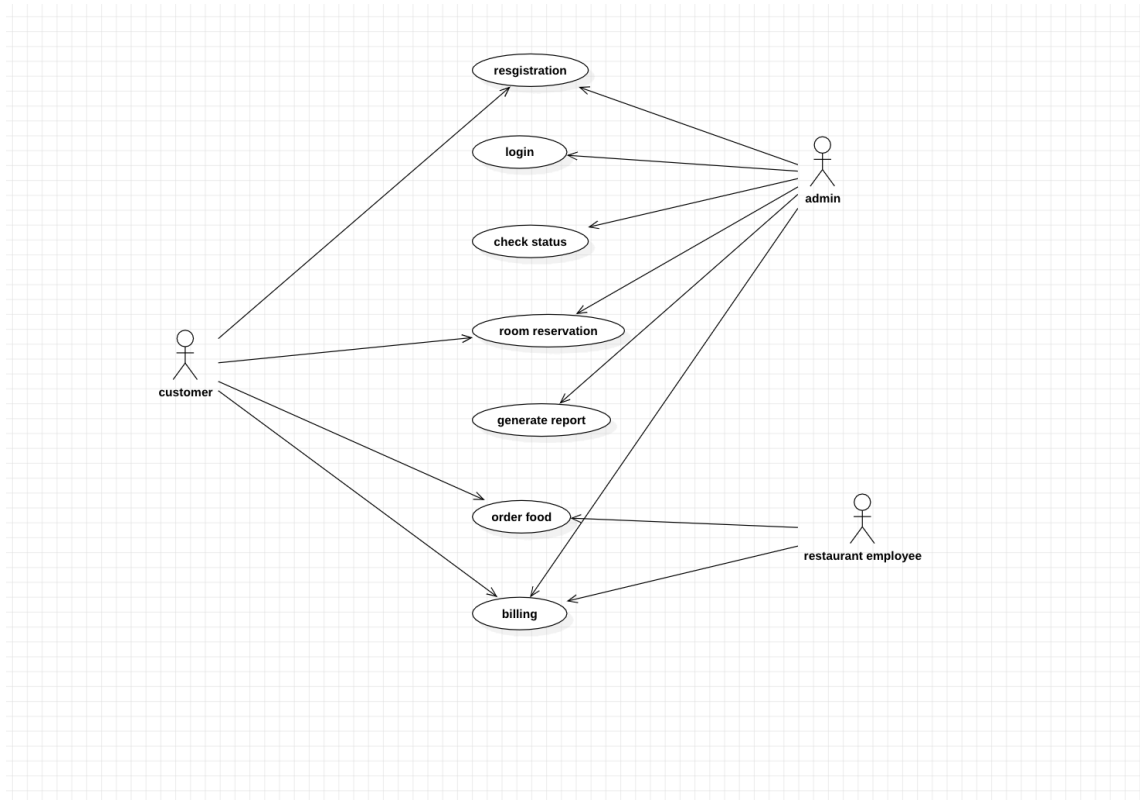
By leveraging the MVC architecture, this hotel management system boasts several advantages. Maintainability is enhanced as changes to the logic (Model), presentation (View), and user interaction (Controller) can be made independently. Code reusability is another benefit, as Views can be adapted for different controllers, and Models can be shared by multiple views. Finally, with each layer (Model, View, Controller) testable in isolation, the overall development process becomes more efficient and reliable.

In conclusion, this Java Swing application with MVC architecture offers a well-structured and user-friendly solution for hotel management. It empowers managers, streamlines workflows for receptionists, and provides a convenient booking platform for customers. This combination ensures a smooth-running hotel operation, leaving you free to focus on what truly matters - creating exceptional guest experiences.

ACTORS:

1. Admin
 2. Customer
 3. Restaurant Employee
-

USE CASE DIAGRAM:



USE CASES:

Use Case Specifications for Hotel Management System

1. Use Case Name: Register Customer

1.1. Actor: Customer

1.2. Preconditions:

- The customer is a new user of the system.

1.3. Description:

- The customer accesses the hotel's registration portal.
- The system prompts the customer to enter their details, including name, email address, phone number, and desired login credentials (username and password).
- The system validates the entered information (e.g., email format, password strength).
- Upon successful validation, the system creates a new customer account and sends a confirmation email to the customer.

1.4. Exceptions:

- If the entered information is invalid, the system displays error messages and prompts the customer to correct the information.
- If the username already exists, the system informs the customer and suggests alternative usernames.

1.5. Postconditions:

- The customer has a registered account in the system and can log in to access its features (e.g., room reservation).

2. Use Case Name: Login

2.1. Actor: Customer

2.2. Preconditions:

- The customer has a registered account in the system.

2.3. Description:

- The customer accesses the hotel's login page.
- The customer enters their username and password.
- The system validates the credentials against the stored user information.
- Upon successful login, the system grants the customer access to their personalized dashboard.

2.4. Exceptions:

- If the username or password is incorrect, the system displays an error message and allows the customer to retry.

- If the account is locked due to multiple failed login attempts, the system informs the customer and provides instructions on unlocking the account.

2.5. Postconditions:

- The customer is successfully logged into the system and can interact with features based on their user role (e.g., view room availability, make reservations).

3. Use Case Name: Search Rooms

3.1. Actor: Customer

3.2. Preconditions:

- The customer is logged into the system.

3.3. Description:

- The customer accesses the room search functionality.
- The customer enters search criteria, such as desired check-in and check-out dates, number of guests, and preferred room type (e.g., single, double, suite).
- The system searches the database for available rooms that meet the specified criteria.
- The system displays a list of available rooms along with their descriptions, amenities, and prices.

3.4. Exceptions:

- If no rooms are available based on the search criteria, the system informs the customer and suggests alternative dates or room types.
- If there is a system error during the search process, an error message is displayed, and the customer is advised to try again later.

3.5. Postconditions:

- The customer is presented with a list of rooms that match their preferences, allowing them to compare options and make an informed selection.

4. Use Case Name: Book Room

4.1. Actor: Customer

4.2. Preconditions:

- The customer is logged into the system.
- The customer has searched for available rooms and selected a preferred room.

4.3. Description:

- The customer selects the desired room and proceeds with the booking process.
- The customer enters additional information required for the booking, such as guest names and contact details.
- The system validates the entered information and checks for room availability on the selected dates.
- If the room is available, the system displays a confirmation screen with the booking details (room type, price, guest information).
- The customer chooses a payment method (e.g., online payment, pay at hotel) and completes the payment process (if applicable).
- Upon successful payment, the system confirms the booking and sends a booking confirmation email to the customer.

4.4. Exceptions:

- If the selected room becomes unavailable during the booking process, the system informs the customer and suggests alternative options.
- If the payment fails, the system displays an error message, and the booking is not confirmed.

4.5. Postconditions:

- The customer has a confirmed reservation for the chosen room on the specified dates.
- The booking details are stored in the system for future reference.

5. Use Case Name: Check-In

5.1. Actor: Receptionist

5.2. Preconditions:

- The receptionist has access to the hotel management system.
- A customer has a confirmed reservation for a room.

5.3. Description:

- The receptionist searches for the customer's reservation using the booking reference number or customer details. pen_spark

5.3. Description:

- The receptionist verifies the customer's identity (e.g., by checking ID) and collects any additional information required (e.g., number of guests arriving).
- The system displays the reservation details and allows the receptionist to select a room for the customer (if multiple rooms are associated with the reservation).
- The receptionist assigns a room key or provides access information to the customer.
- The system updates the reservation status to "Checked-In" and records the check-in date and time.

5.4. Exceptions:

- If the customer's reservation cannot be found, the receptionist may need to contact the manager for assistance.
- If the customer's payment has not been settled, the system may prompt the receptionist to collect payment before check-in.

5.5. Postconditions:

- The customer is successfully checked in and receives their room key or access information.
- The reservation status is updated to reflect the check-in.

6. Use Case Name: Check-Out

6.1. Actor: Receptionist

6.2. Preconditions:

- The receptionist has access to the hotel management system.

- A customer has a checked-in reservation.

6.3. Description:

- The receptionist searches for the customer's reservation using the room number or guest details.
- The system displays the reservation details, including any outstanding charges (e.g., minibar charges, room service).
- The receptionist reviews the charges with the customer and collects payment if necessary.
- The receptionist may process any applicable refunds (e.g., deposit on room key).
- The system updates the reservation status to "Checked-Out" and records the check-out date and time.

6.4. Exceptions:

- If the customer's reservation cannot be found, the receptionist may need to contact the manager for assistance.
- If the customer has damaged hotel property or has outstanding charges, the system may prompt the receptionist to handle these issues before check-out.

6.5. Postconditions:

- The customer is successfully checked out and returns their room key.
- The reservation status is updated to reflect the check-out.
- Any outstanding charges are settled, and refunds are processed (if applicable).

7. Use Case Name: Manage Room Service

7.1. Actor: Customer (can also be Restaurant Staff)

7.2. Preconditions:

- The customer (or Restaurant Staff) is logged into the system (if applicable).
- The customer has a checked-in reservation.

7.3. Description:

- The customer accesses the room service menu through the hotel management system or a dedicated room service application.
- The customer browses the menu and selects desired food and beverage items.
- The system allows the customer to specify any special requests or dietary restrictions.
- The customer confirms the order and selects a preferred delivery time.
- The system transmits the order electronically to the kitchen or restaurant staff.
- (Alternatively, the customer can call the restaurant directly to place an order).

7.4. Exceptions:

- If certain menu items are unavailable, the system informs the customer and suggests alternatives.
- If there is a system error during the order process, the customer may need to place the order by phone or contact the reception for assistance.

7.5. Postconditions:

- The customer's room service order is submitted to the kitchen or restaurant staff.
- The kitchen prepares the order and delivers it to the customer's room at the specified time.

8. Use Case Name: Manage User Accounts

8.1. Actor: Manager

8.2. Preconditions:

- The manager has access to the hotel management system with administrative privileges.

8.3. Description:

- The manager accesses the user account management section of the system.
- The system displays a list of existing user accounts (customers and staff).

- The manager can view user details, edit account information, and reset passwords (if necessary).
- The manager can create new user accounts for staff members with assigned roles and permissions.
- The manager can deactivate or delete user accounts that are no longer in use.

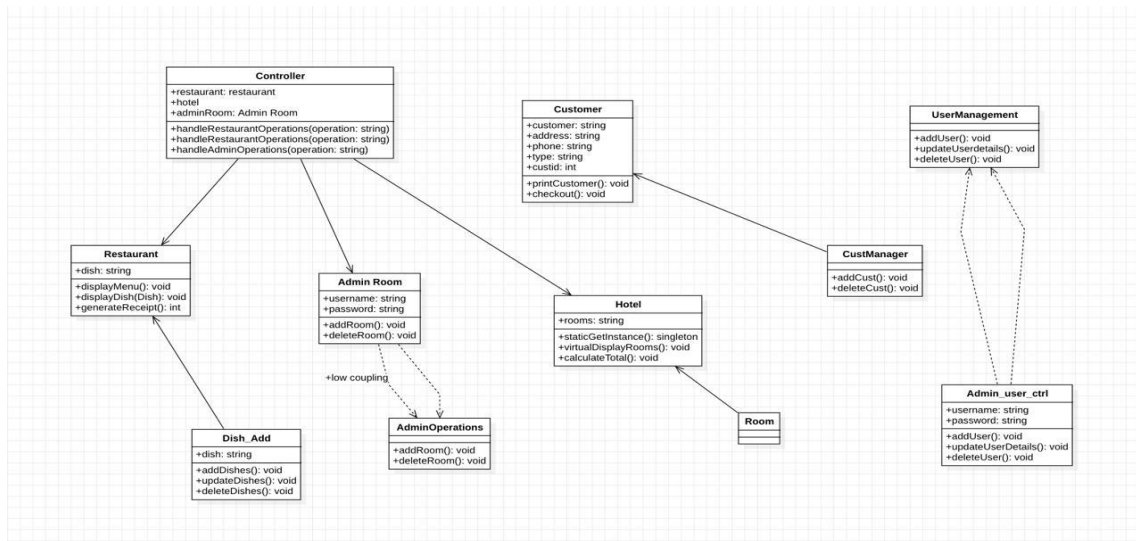
8.4. Exceptions:

- The manager cannot modify their own account without involving another administrator for security reasons.
- The system may enforce restrictions on deleting user accounts, especially for accounts with historical data associated with them.

8.5. Postconditions:

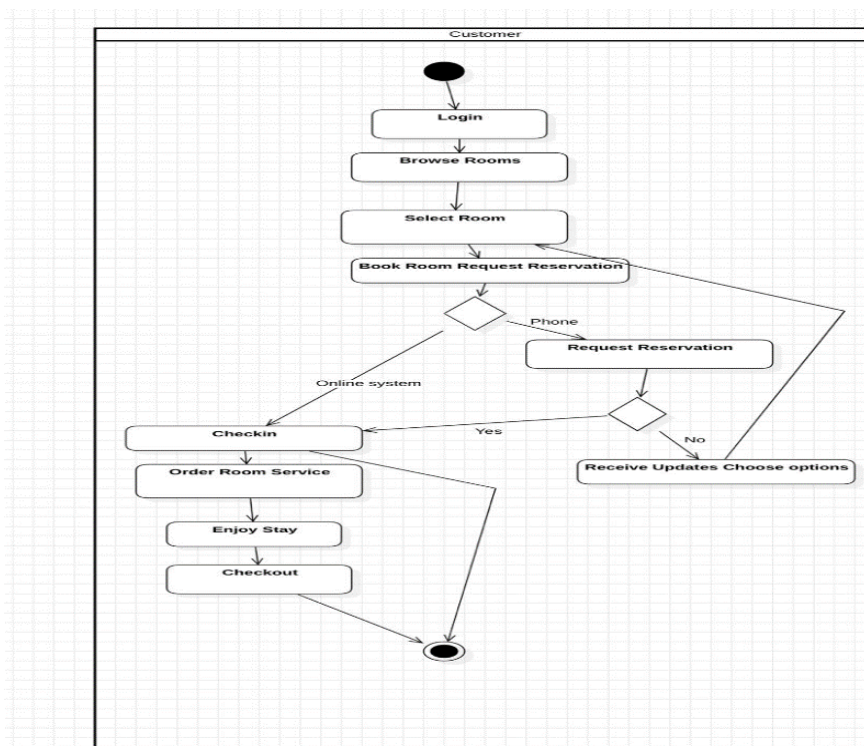
- User accounts are managed according to the hotel's needs, ensuring authorized access and maintaining system security.
-

CLASS DIAGRAM USING GRASP AND SOLID

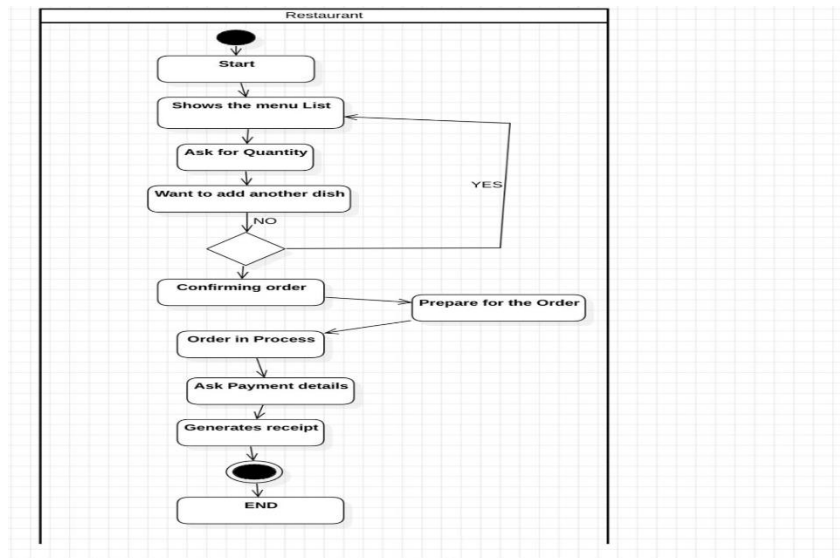


ACTIVITY DIAGRAMS

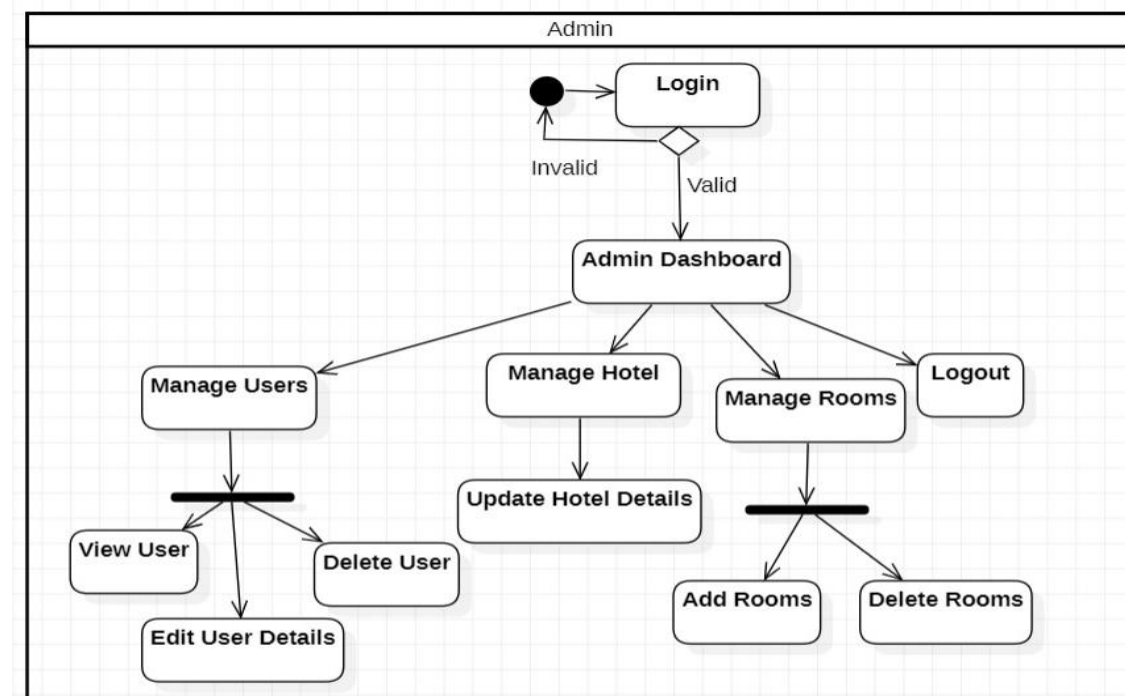
1. Customer



2. Restaurant

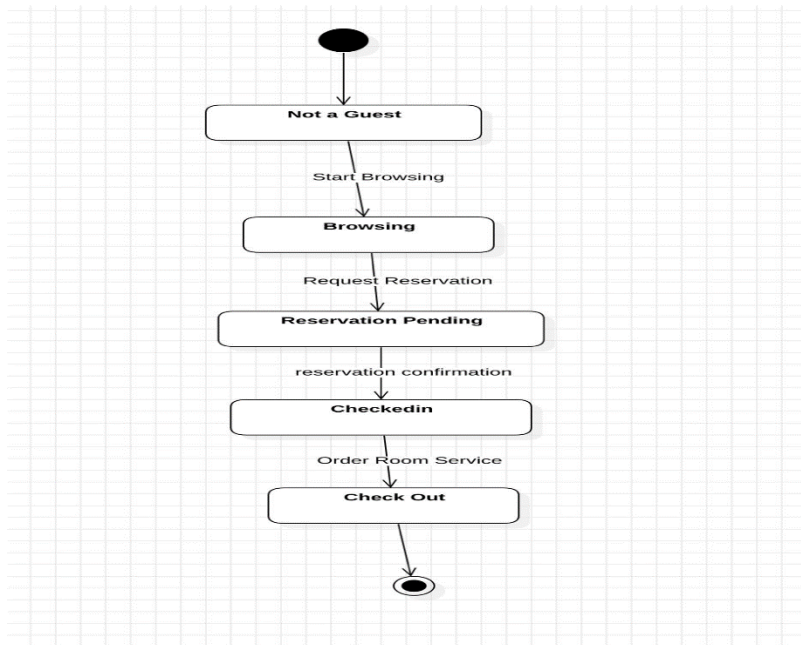


3. Admin

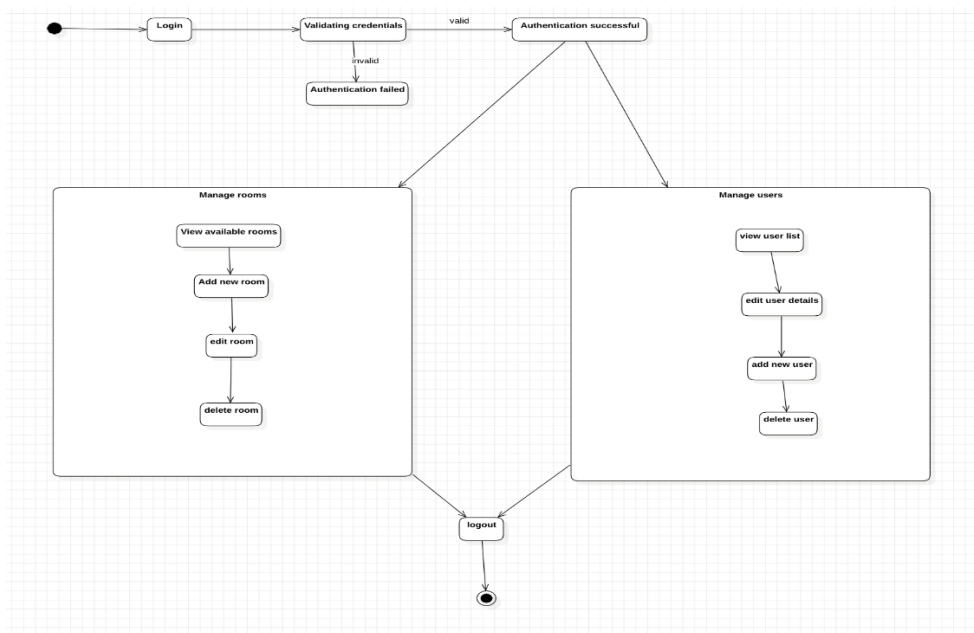


STATE DIAGRAMS

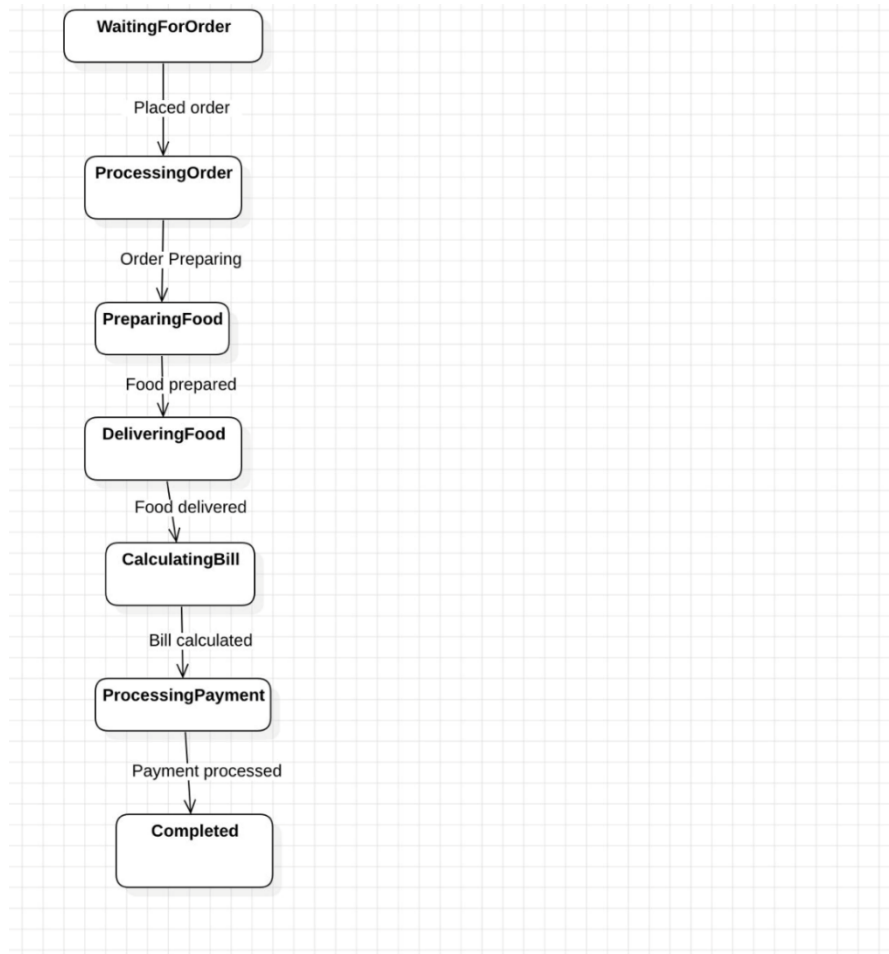
1.Customer



2.Admin



3.Restaurant



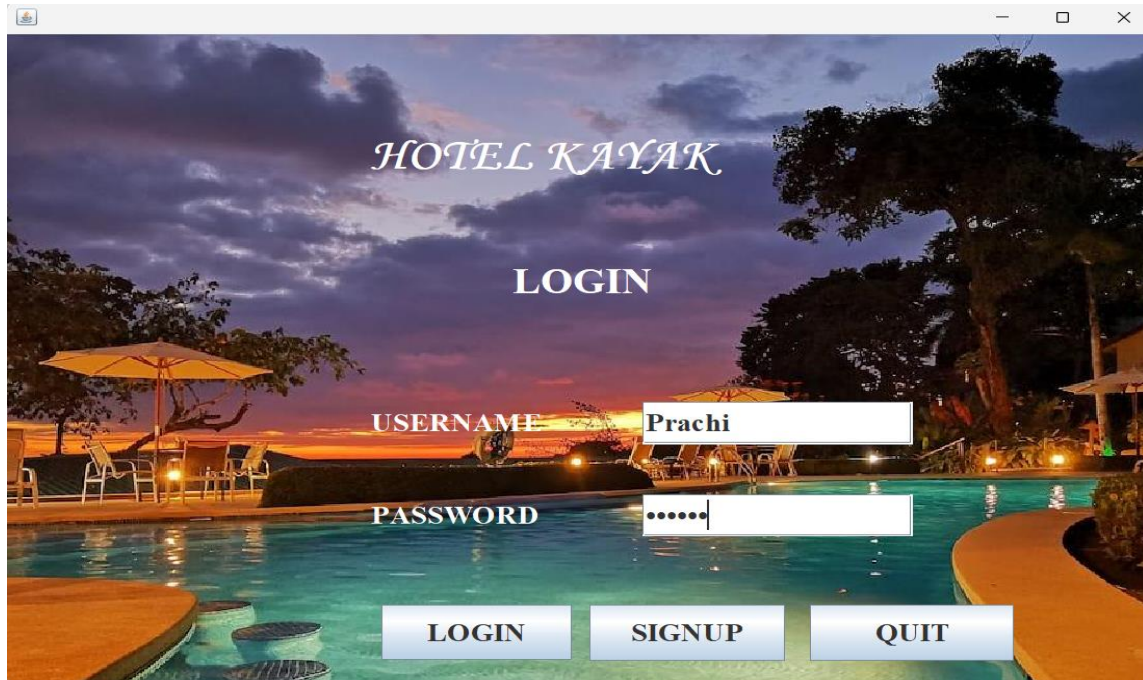
Github link with codebase:

https://github.com/KrishnaSudarshan002/Hotel_Management_Java

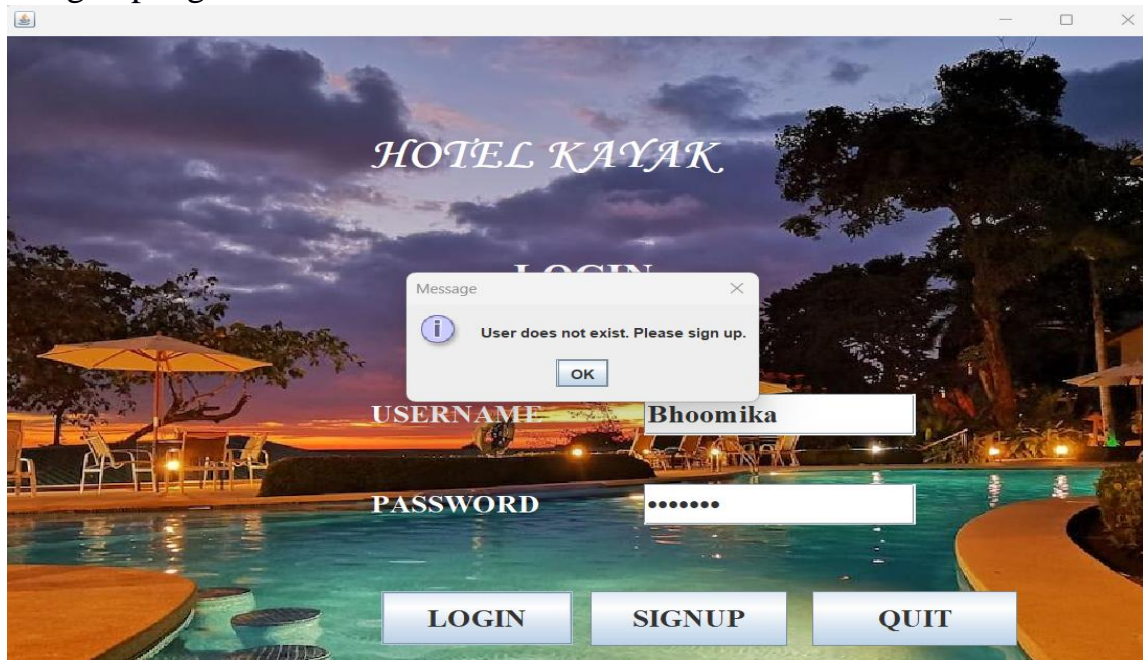
Screenshots with input values populated and output shown:

User side:

1.LoginPage:

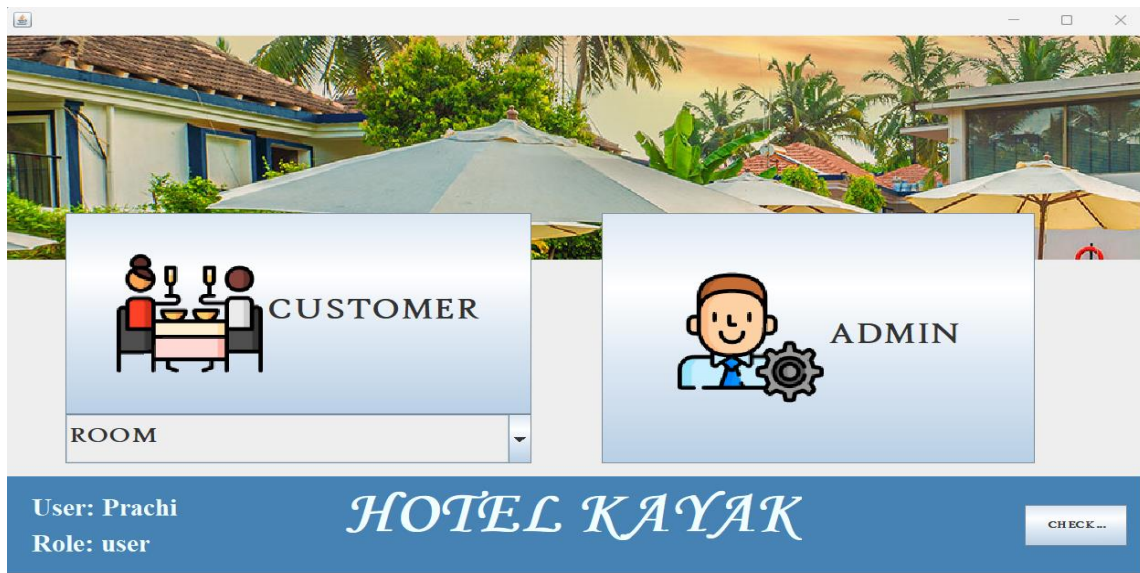


2.SignUpPage:



A screenshot of a web application window for the sign-up page. It features three input fields: "Name:" with the text "Bhoomika", "Email:" with the text "bhoomika@gmail.com", and "Password:" with seven dots. Below these fields is a "Sign Up" button.

Room customer:



The login screen features a background image of a hotel building with palm trees. It has two main login buttons: 'CUSTOMER' with an icon of people at a table, and 'ADMIN' with an icon of a person and a gear. Below the 'CUSTOMER' button is a dropdown menu labeled 'ROOM'. At the bottom, a blue banner displays the user's name 'Prachi', role 'user', the hotel name 'HOTEL KAYAK' in a stylized font, and a 'CHECK ...' button.

CUSTOMER

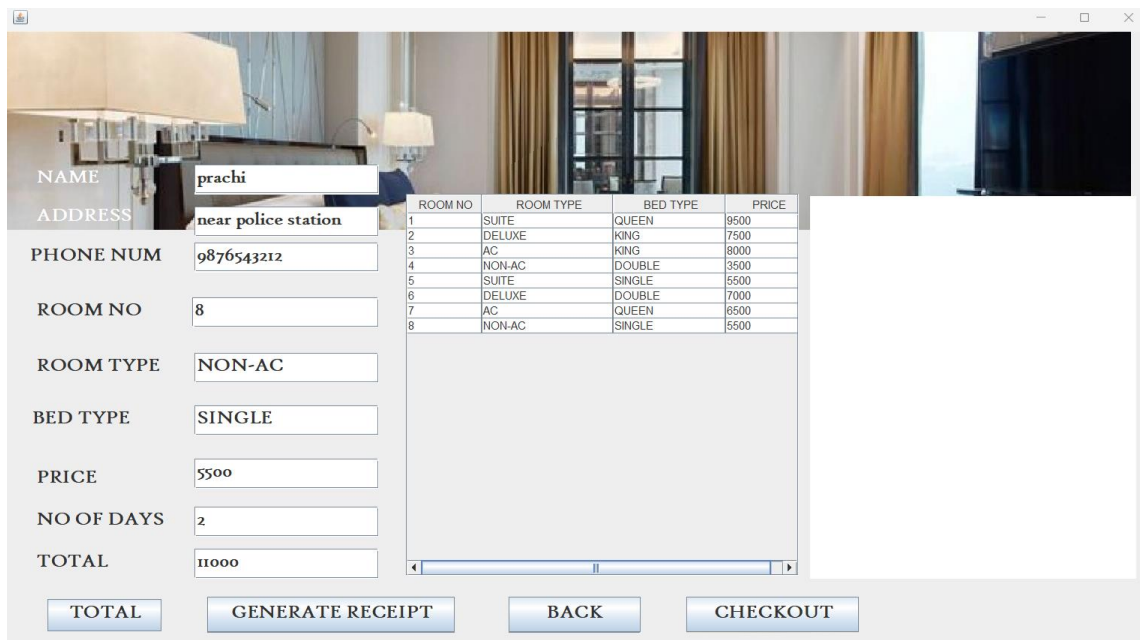
ADMIN

ROOM

User: Prachi
Role: user

HOTEL KAYAK

CHECK ...



The booking form is set against a background image of a hotel room. It includes input fields for personal details and booking specifics. A table lists available room types and their prices. At the bottom, there are buttons for 'TOTAL', 'GENERATE RECEIPT', 'BACK', and 'CHECKOUT'.

NAME: prachi

ADDRESS: near police station

PHONE NUM: 9876543212

ROOM NO: 8

ROOM TYPE: NON-AC

BED TYPE: SINGLE

PRICE: 5500

NO OF DAYS: 2

TOTAL: 11000

ROOM NO	ROOM TYPE	BED TYPE	PRICE
1	SUITE	QUEEN	9500
2	DELUXE	KING	7500
3	AC	KING	8000
4	NON-AC	DOUBLE	3500
5	SUITE	SINGLE	5500
6	DELUXE	DOUBLE	7000
7	AC	QUEEN	6500
8	NON-AC	SINGLE	5500

TOTAL GENERATE RECEIPT BACK CHECKOUT

NAME: prachi
ADDRESS: near police station
PHONE NUM: 9876543212
ROOM NO: 8
ROOM TYPE: NON-AC
BED TYPE: SINGLE
PRICE: 5500
NO OF DAYS: 2
TOTAL: 11000

ROOM NO	ROOM TYPE	BED TYPE	PRICE
	SUITE	QUEEN	9500
	DELUXE	KING	7500
	AC	KING	8000
	NON-AC	DOUBLE	3500
	SUITE	SINGLE	5500
	DELUXE		
	AC		
	NON-AC		

YOUR BILL RECEIPT

TIME : 24/04/13 19:39:40
NAME : prachi
ADDRESS : near police station
PHONE NUM : 9876543212
ROOM TYPE : NON-AC
BED TYPE : SINGLE
TOTAL AMOUNT : 11000

TOTAL GENERATE RECEIPT BACK CHECKOUT

NAME: prachi
ADDRESS: near police station
PHONE NUM: 9876543212
ROOM NO: 8
ROOM TYPE: NON-AC
BED TYPE: SINGLE
PRICE: 5500
NO OF DAYS: 2
TOTAL: 11000

ROOM NO	ROOM TYPE	BED TYPE	PRICE
	SUITE	QUEEN	9500
	DELUXE	KING	7500
	AC	KING	8000
	NON-AC	DOUBLE	3500
	SUITE	SINGLE	5500
	DELUXE		
	AC		
	NON-AC		

YOUR BILL RECEIPT

TIME : 24/04/13 19:39:40
NAME : prachi
ADDRESS : near police station
PHONE NUM : 9876543212
ROOM TYPE : NON-AC
BED TYPE : SINGLE
TOTAL AMOUNT : 11000

TOTAL GENERATE RECEIPT BACK CHECKOUT

Restaurant customer :



The login interface features a background image of a restaurant patio with a large umbrella. It includes two login buttons: 'CUSTOMER' with an icon of two people at a table, and 'ADMIN' with an icon of a person and a gear. A dropdown menu labeled 'RESTAURANT' is positioned below the 'CUSTOMER' button. At the bottom, a blue banner displays the user information 'User: Omisha' and 'Role: user', the restaurant name 'HOTEL KAYAK' in a stylized font, and a 'CHECK ...' button.

CUSTOMER

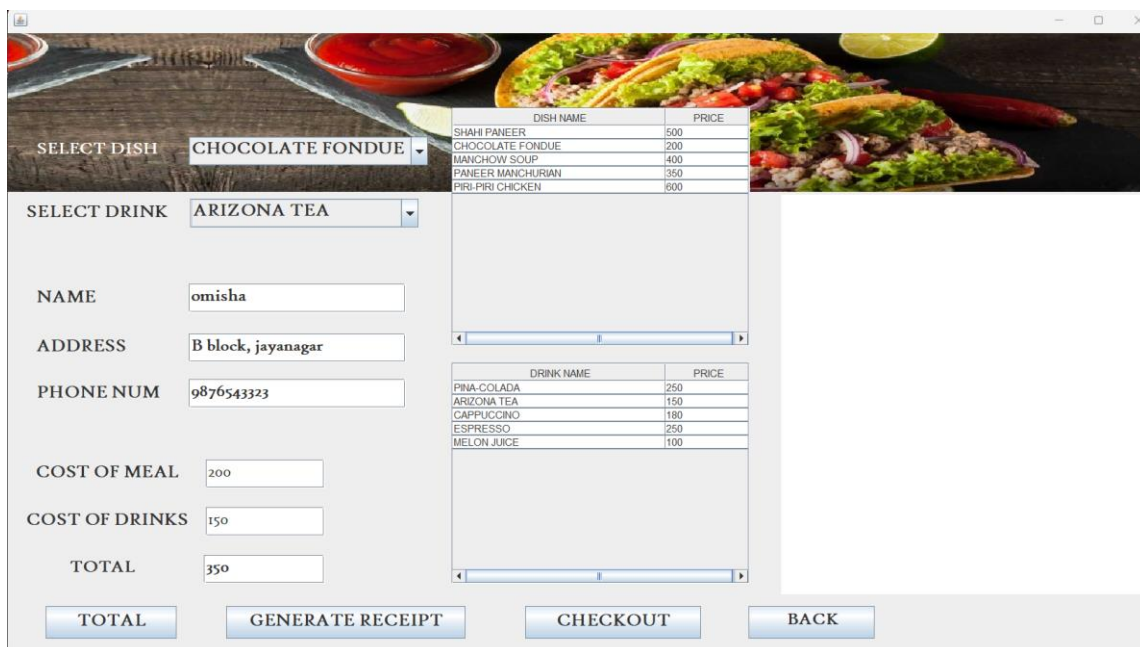
ADMIN

RESTAURANT

User: Omisha
Role: user

HOTEL KAYAK

CHECK ...



The order interface has a background image of food. It includes a 'SELECT DISH' dropdown menu currently set to 'CHOCOLATE FONDUE'. Below it is a 'SELECT DRINK' dropdown menu set to 'ARIZONA TEA'. There are input fields for 'NAME' (omisha), 'ADDRESS' (B block, jayanagar), and 'PHONE NUM' (9876543323). To the right, there are two tables: one for dishes and one for drinks. Below these tables are input fields for 'COST OF MEAL' (200), 'COST OF DRINKS' (150), and 'TOTAL' (350). At the bottom, there are four buttons: 'TOTAL', 'GENERATE RECEIPT', 'CHECKOUT', and 'BACK'.

SELECT DISH CHOCOLATE FONDUE

SELECT DRINK ARIZONA TEA

NAME omisha

ADDRESS B block, jayanagar

PHONE NUM 9876543323

DISH NAME	PRICE
SHAHI PANEER	500
CHOCOLATE FONDUE	200
MANCHOW SOUP	400
PANEER MANCHURIAN	350
PIPS-PIPS CHICKEN	600


DRINK NAME	PRICE
PINA-COLADA	250
ARIZONA TEA	150
CAPPUCCINO	180
ESPRESSO	250
MELON JUICE	100

COST OF MEAL 200

COST OF DRINKS 150

TOTAL 350

TOTAL GENERATE RECEIPT CHECKOUT BACK



SELECT DISH

CHOCOLATE FONDUE

SELECT DRINK

ARIZONA TEA

NAME

omisha

ADDRESS

B block, jayanagar

PHONE NUM

9876543323

COST OF MEAL

200

COST OF DRINKS

150

TOTAL

350

TOTAL

GENERATE RECEIPT

CHECKOUT


BACK

DISH NAME	PRICE
SHAWI PANEER	500
CHOCOLATE FONDUE	200
MANCHOW SOUP	400
PANEER MANCHURIAN	350
PIRI-PIRI CHICKEN	600

DRINK NAME	PRICE
PINA COLADA	250
ARIZONA TEA	150
CAPPUCCINO	180
ESPRESSO	250
MELON JUICE	100

YOUR BILL RECEIPT

TIME : 24/04/13 19:58:56
NAME : omisha
ADDRESS : B block, jayanagar
PHONE NUM : 9876543323
ORDERED DISH : CHOCOLATE FONDUE
ORDERED DRINK : ARIZONA TEA
TOTAL BILL : 350



SELECT DISH

CHOCOLATE FONDUE

SELECT DRINK

ARIZONA TEA

NAME

omisha

ADDRESS

B block, jayanagar

PHONE NUM

9876543323

COST OF MEAL

200

COST OF DRINKS

150

TOTAL

350

TOTAL

GENERATE RECEIPT

CHECKOUT

BACK

DISH NAME	PRICE
SHAWI PANEER	500
CHOCOLATE FONDUE	200
MANCHOW SOUP	400
PANEER MANCHURIAN	350
PIRI-PIRI CHICKEN	600

DRINK NAME	PRICE
PINA COLADA	250
ARIZONA TEA	150
CAPPUCCINO	180
ESPRESSO	250
MELON JUICE	100

YOUR BILL RECEIPT

TIME : 24/04/13 19:58:56
NAME : omisha
ADDRESS : B block, jayanagar
PHONE NUM : 9876543323
ORDERED DISH : CHOCOLATE FONDUE
ORDERED DRINK : ARIZONA TEA
TOTAL BILL : 350

Message

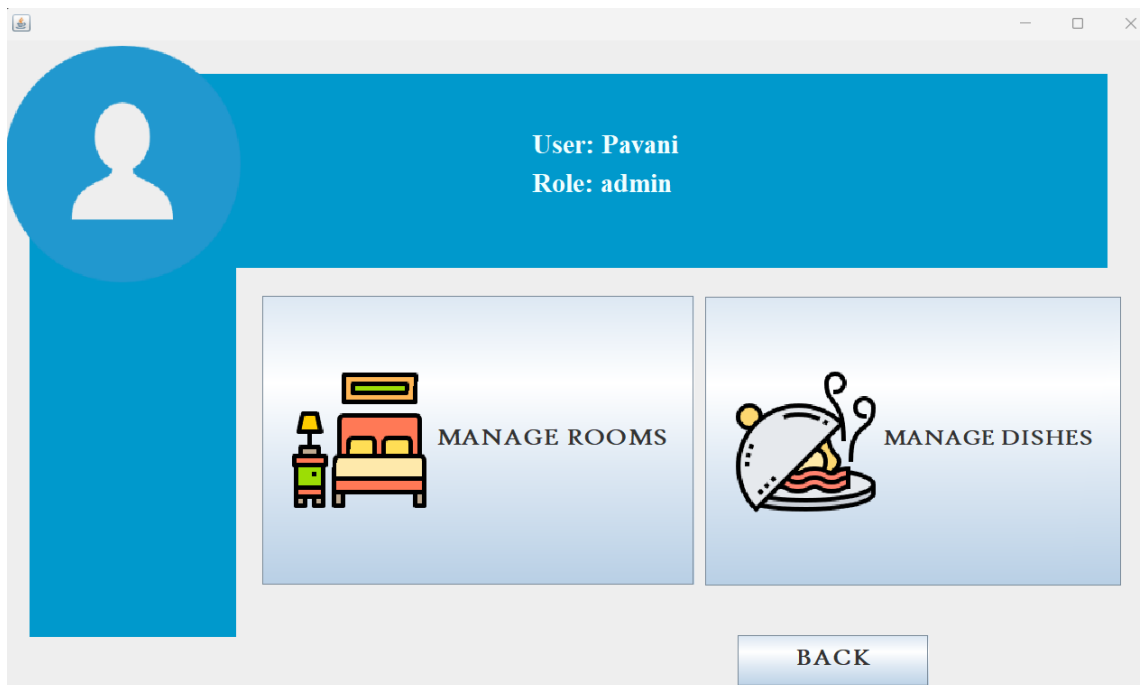
Checked out Successfully




OK

Admin side:



MANAGE ROOMS:





ROOM TYPE

AC

BED TYPE

SINGLE

PRICE

6000

ROOM NO	ROOM TYPE	BED TYPE	PRICE
1	SUITE	QUEEN	9500
2	DELUXE	KING	7500
3	AC	KING	8000
4	NON-AC	DOUBLE	3500
5	SUITE	SINGLE	5500
6	DELUXE	DOUBLE	7000
7	AC	QUEEN	6500
8	NON-AC	SINGLE	5500

+

ADD ROOMS

×


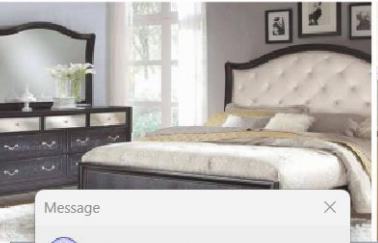

DELETE ROOMS

✓

UPDATE ROOMS

←

BACK



ROOM TYPE

AC

BED TYPE

SINGLE

PRICE

6000

Message

i

New Room Added

OK

R	TYPE	PRICE	
1		9500	
2		7500	
3		8000	
4	NON-AC	DOUBLE	3500
5	SUITE	SINGLE	5500
6	DELUXE	DOUBLE	7000
7	AC	QUEEN	6500
8	NON-AC	SINGLE	5500

+

ADD ROOMS

×


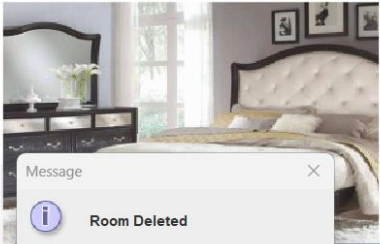

DELETE ROOMS

✓

UPDATE ROOMS

←

BACK

ROOM TYPE
BED TYPE
PRICE

Message

Room Deleted

OK

	ROOM TYPE	PRICE
2	DELUXE	9500
3	AC	7500
4	NON-AC	8000
5	SUITE	3500
6	DELUXE	5500
7	DOUBLE	7000
8	AC	6500
9	NON-AC	5500
10	SINGLE	6000

+ ADD ROOMS
✗ DELETE ROOMS
✓ UPDATE ROOMS
← BACK

MANAGE DISHES:





CUSTOMER






ADMIN

ROOM

User: Nidhii
Role: admin

HOTEL KAYAK

CHECK...



Message

New Dish Added

OK

DISH NAME

PINK SAUCE PASTA

DISH PRICE

350

DISH TYPE

MEAL




	DISH TYPE	PRICE
1	MEAL	500
2	MEAL	200
3	MEAL	400
4	MEAL	350
5	MEAL	600
6	DRINK	250
7	DRINK	150
8	DRINK	180
9	DRINK	250
10	DRINK	100

+ ADD DISH

✖ DELETE DISH

↻ UPDATE DISH

⬅ BACK



Message

Dish Deleted

OK

DISH NAME

MELON JUICE

DISH PRICE

350

DISH TYPE

DRINK




DISH NO	DISH NAME	DISH TYPE	PRICE
1	CONDUE	Meal	150
2	UP	MEAL	200
3	HURIAN	MEAL	400
4	KEN	MEAL	350
5		MEAL	600
6		DRINK	250
7		DRINK	150
8	CAPPUCCINO	DRINK	180
9	MANCHOW SOUP	Meal	400
10	MELON JUICE	DRINK	100
11	PINK SAUCE PASTA	MEAL	350

+ ADD DISH

✖ DELETE DISH

↻ UPDATE DISH

⬅ BACK







DISH NAME


DISH PRICE


DISH TYPE

DISH NO	DISH NAME	DISH TYPE	PRICE
1	CHOLE	Meal	150
2	CHOCOLATE FONDUE	MEAL	200
3	MANCHOW SOUP	MEAL	400
4	PANEER MANCHURIAN	MEAL	350
5	PIRI-PIRI CHICKEN	MEAL	600
6	PINA-COLADA	DRINK	250
7	ARIZONA TEA	DRINK	150
8	CAPPUCCINO	DRINK	180
9	MANCHOW SOUP	Meal	400
11	PINK SAUCE PASTA	MEAL	350


 ADD DISH

 DELETE DISH

 UPDATE DISH

 BACK

After updating dish :






DISH NAME


DISH PRICE


DISH TYPE

DISH NO	DISH NAME	DISH TYPE	PRICE
1	CHOLE	Meal	150
2	CHOCOLATE FONDUE	MEAL	200
3	MANCHOW SOUP	MEAL	400
4	PANEER MANCHURIAN	MEAL	350
5	PIRI-PIRI CHICKEN	MEAL	600
6	PINA-COLADA	DRINK	250
7	ARIZONA TEA	DRINK	150
8	CAPPUCCINO	DRINK	180
9	MANCHOW SOUP	Meal	400
11	MELON JUICE	DRINK	50

 ADD DISH

 DELETE DISH

 UPDATE DISH

 BACK

Error page if user not exist:

