HOTEL MANAGEMNET SYSTEM

ITERATION 2

Date: 10/29/2019
Team Name: CS 6359.002 Team 3
Team Members:

1. Ajay Kishorkumar Danda
2. Bingjie Yan
3. Elijah Snow

FUNCTIONAL REQUIREMENTS OF SYSTEM

Functional Requirements:
We had 10 functional requirements initially and now we identified 2 more additional requirements where the Customer should be able to modify or delete its bookings.

Requirement ID	Functional Requirements
R1	The Customer should be able to Register.
R2	The Customer should be able to LOGIN.
R3	The Manager should be able to Register.
R4	The Manager should be able to LOGIN.
R5	The Manager should be allowed to Add Listings.
R6	The manager should be allowed to Modify Listings.
R7	The Manager should be allowed to Delete Listings.
R8	The Customer should be allowed to Book Room.
R9	The Customer should be allowed to Modify its Room Booking
R10	The Customer should be allowed to Delete a Room Booking.
R11	The Customer should be able to View the BILL.
R12	The Customer should be redirected to Payment gateway.

REQUIREMENT PRIORITY

Requirement ID	Priority	Functional Requirements
R1	5	The Customer should be able to Register.
R2	5	The Customer should be able to LOGIN.
R3	5	The Manager should be able to Register.
R4	5	The Manager should be able to LOGIN.
R5	4	The Manager should be allowed to Add Listings.
R6	3	The manager should be allowed to Modify Listings.

R7	3	The Manager should be allowed to Delete Listings.
R8	4	The Customer should be allowed to Book Room.
R9	3	The Customer should be allowed to Modify its Room Booking
R10	3	The Customer should be allowed to Delete a Room Booking.
R11	4	The Customer should be able to View the BILL.
R12	4	The Customer should be redirected to Payment gateway.

USE CASE REALIZATION AND TRACEABILITY MATRIX

The following are the use cases that are realized after refining the functional requirements:
Use Case 1: Registration
Use Case 2: Login
Use Case 3: Add Listing
Use Case 4: Delete Listing
Use Case 5: Modify Listing
Use Case 6: Book Room
Use Case 7: Delete Booking
Use Case 8: Modify Booking
Use Case 9: Bill Generation
Use Case 10: Payment Redirection

	Priority	TC1	TC2	TC3	TC4	TC5	TC6	TC7	TC8	TC9	TC10
R1	5	X									
R2	5		X								
R3	5	X									
R4	5		X								
R5	4			X							
R6	3					X					
R7	3				X						
R8	4						X				
R9	3								X		
R10	3							X			
R11	4									X	
R12	4										X
Total		10	10	4	3	3	4	3	3	4	4

NON-FUNCTIONAL REQUIREMENTS OF SYSTEM

There are no additional non-functional requirements identified in this iteration.

The following are some of the non-functional requirements of the system:

- Performance: The system will perform at its peak performance in most of the situations. The system will have fast throughput and utilization.
- Capacity: There is no limit to the number of customers the system can handle. However, the system will have only one manager for one hotel.
- Security: The system will have enough security so that no attacker can access the hotel and customer's private credentials.
- Availability: The system will be available to the customers for bookings.
- Maintainability: The system will be developed using all the important OOPs design patterns and hence will have good maintainability.

Use Cases

Use case 1: Registration: This use case covers the functionality of registration of Manager/ Customer. This use case was covered in the previous iteration but we refined the functioning of Registering a User either as Customer or Manager
Actors: Manager/ Customer

Manager/Customer	System
	1. Display the two registration options on terminal. Option 1 is Register as a Customer. Option 2 is Register as Manager
2. User enters the appropriate option.	3. Based on the given input, the system asks for the credentials of the User.
4. User enters all credentials like username, password, name, etc.	5. System validates all the credentials for their correctness and prompts if any missing.
6.User enters if any missing fields and submits.	7. The system will store this registration information in the Manager or Customer Database based on the option selected by the User in the Step 2. 8. The system sends a success feedback to the user.

Post condition: New user is added in the Customer/ Manager database.

Use case 5: Modify Listing: This use case covers the functionality of allowing a registered Manager to modify a hotel listing for the customers to book.
Pre-condition: Manager must be registered and logged in with the system. The listing that has to be modified must already exist in the listings of that hotel.
Actors: Manager

Manager	System
	1. Display the hotel listings for that manager.
2. Manager selects the option of modify listing in the system.	3. The system displays number of rooms available for all different types of rooms and their price.

	4. The system asks for which type of room is to be modified.
5. Manager enters the option of a particular room type that has to be modified.	6. The system asks for change in number of rooms to be made for that room type and its new price.
7. Manager increases or decreases the number of rooms for that room type and changes price if needed. 9. Manager logs out of the system.	8. The system makes the changes for that listing and sends a success message.

Post condition: A already existing hotel listing is modified by the manager of that hotel in the system.

Use case 7: Delete Booking: This use case covers the functionality of allowing a registered Customer to delete a hotel booking that already exists in the database.
Pre-condition: Customer must be registered and logged in with the system. The Customer should already have an existing room booking made.
Actors: Customer

Customer	System
	1. The system displays various operations that the Customer can perform when logged in. Options include Book room, Delete booking, Modify booking.
2. Customer selects the option of Delete Booking from the available options.	3. System displays all bookings made by customer. 4. System asks for which Booking has to be deleted by asking for booking_id.
5. Customer enters the booking_id for the booking that has to be deleted.	6. System checks whether the booking_id exists and promts error if no such ID found.
7. Customer reenters the booking_id If error occurred and submits.	8. System deletes the booking form the database and sends a success message to the customer.

Post condition: A specific room booking has been deleted in the database for a specific Customer.

Use case 8: Modify Booking: This use case covers the functionality of allowing a registered Customer to modify already existing booking.
Pre-condition: Customer must be registered and logged in with the system. The Customer should already have an existing room booking made.
Actors: Customer

Customer	System
	1. The system displays various operations that the Customer can perform when logged in. Options include Book room, Delete booking, Modify booking.

2. Customer selects the option of Modify Booking from the available options.	3. System displays all bookings made by customer. 4. System asks for which Booking has to be modified by asking for booking_id.
5. Customer enters the booking_id for the booking that has to be modified.	6. System asks for the new Check In and Check Out date that has to be modified for that booking.
7. Customer enters new check in and check out date for that particular booking and submits.	8. System updates the new check in and check out date for that booking. 9. System checks whether dates entered are valid (check_in date<=check_out date) 10. System prompts for error and asks for dates again.
11. Customer enters correct dates if error occurs.	12. System commits the changes to the database and sends a Success message to the Customer.

Post condition: A specific room booking has been modified in the database for a specific Customer.

Use case 9: Bill Generation: This use case covers the functionality of generating the bill amount for a registered Customer who has just booked a room with a hotel. Strategy Pattern will be applied in order to implement this use case which decides the total payable amount for the Customer after applying appropriate discount.
Pre-condition: Customer must be registered and logged in with the system. The Customer must make a room booking in order to proceed with bill generation.
Actors: Customer

Customer	System
1. The Customer confirms to Book the room and proceeds with bill generation.	2. The System gets the total payable amount for the booking and applies the Strategy Pattern for the same.
	3. System checks whether the Customer has already more than 5 bookings with the system. 4.If yes, then the FrequentCustomerStrategy is applied on the billAmount where the Customer gets a 10% discount. 5. If no, then the NormalCustomerStrategy is applied where there is no discount on the billAmount.
	6. Return the customer with the totalPayable value after applying strategy pattern.
7. Customer views bill and proceeds with Payment.	8. Displays success message of redirecting to payment gateway.

Post condition: A final totalPayable amount is calculated and displayed for the Customer to view.

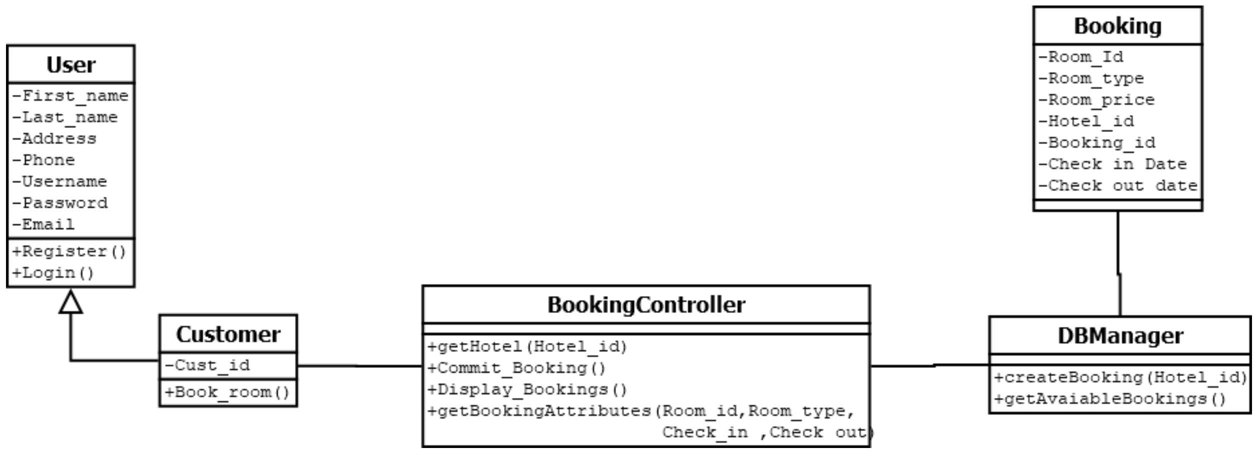
Use case 10: Payment Redirection: This use case covers the functionality of redirecting a registered Customer to PayPal after the Customer has decided to proceed with payment. The Payment Redirection will be using Adapter Pattern in which a PayPalAdapter implementing a PaymentAdapter that translates the makePayment request to the PayPalAdapter.
Pre-condition: Customer must be registered and logged in with the system. The Customer must confirm proceed with payment after bill generation.
Actors: Customer

Customer	System
1. The Customer confirms to proceed with payment.	2. The PaymentAdapter gets the entire object of the Booking made my customer along with the totalPayable amount.
	3. This Adapter further translates this object to the PayPalAdapter. 4. The PayPalAdapter will disintegrate the entire object and send only the totalPayableAmount to PayPal redirecting the Customer to PayPal for payment.
5. The Customer is redirected to the PayPal gateway.	

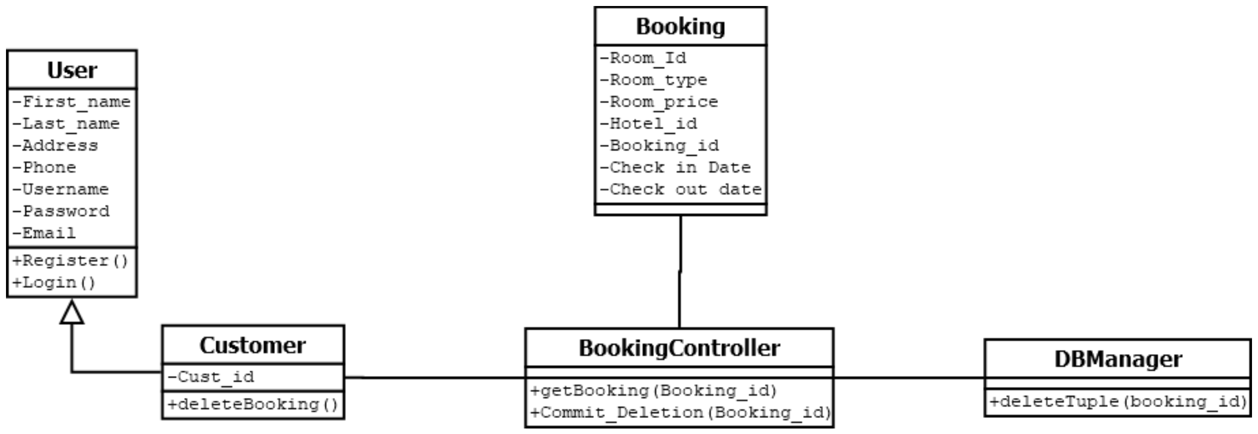
Post condition: The Customer is redirected to the PayPal Gateway.

Class Diagrams

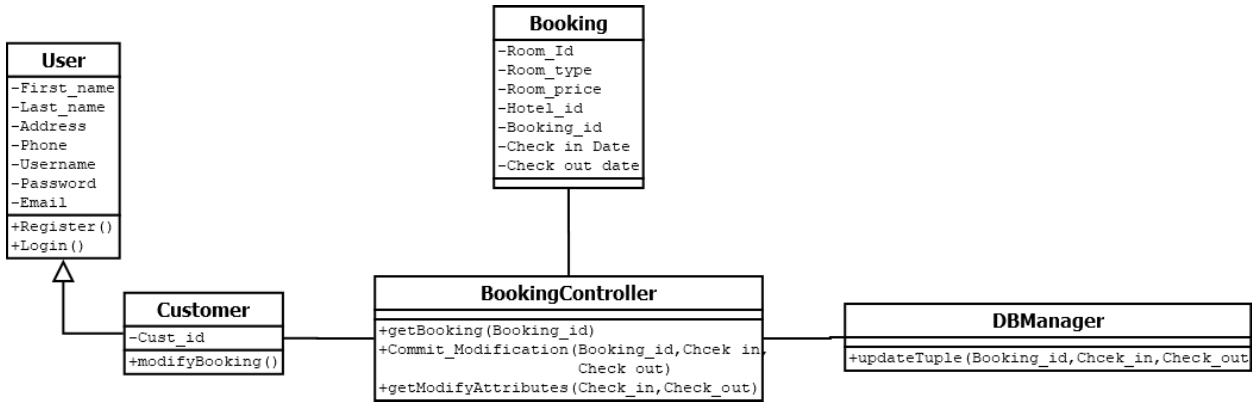
1. Class Diagram to Book Room. The Customer is the only user in this use case. This class diagram only focuses on the room booking part of the system and not any other functionality.



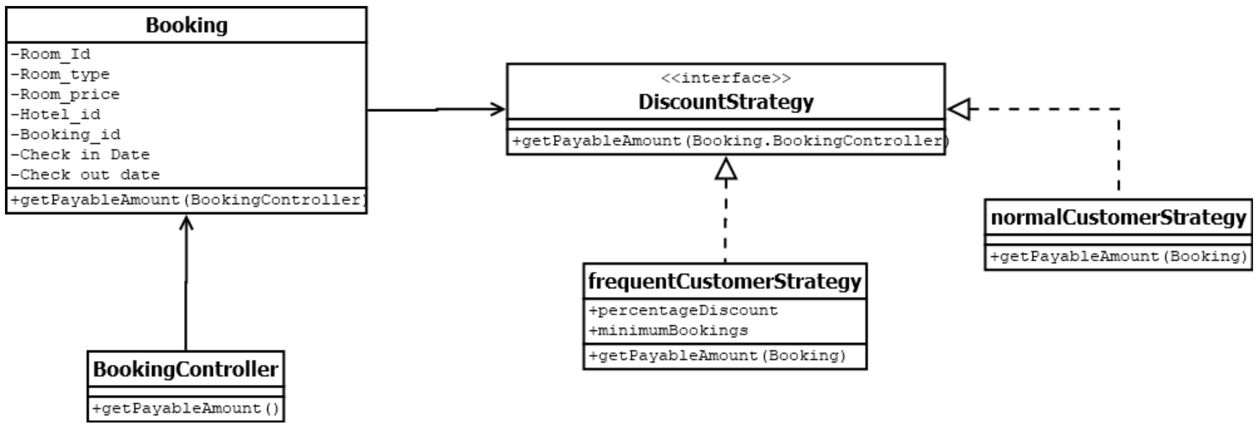
2. Class Diagram for Delete Booking. The Customer is the only user in this use case. This class diagram only focuses on the deletion of booking part of the system and not any other functionality.



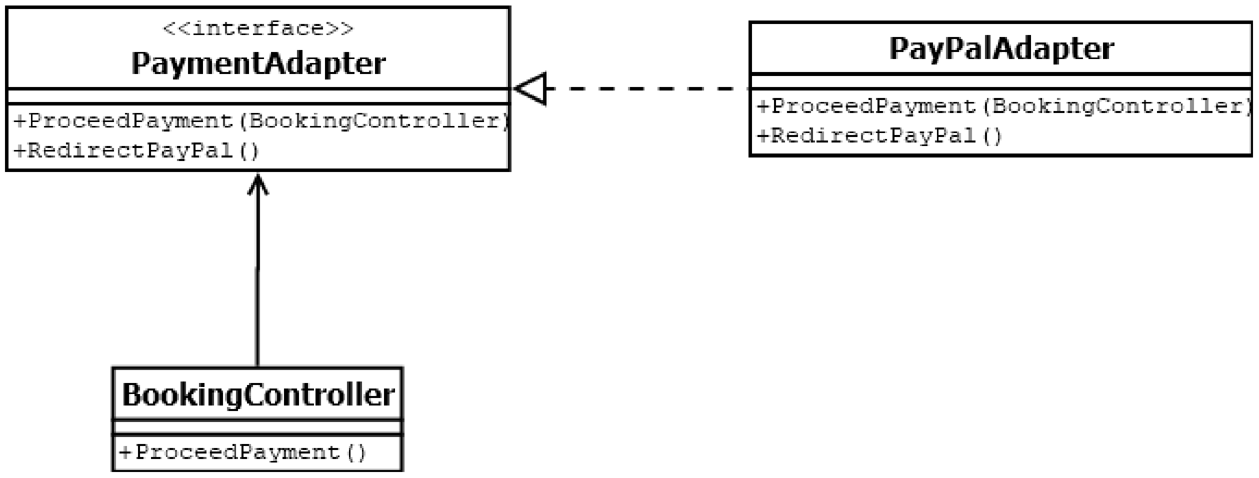
3. Class Diagram for Modify Booking. The Customer is the only user in this use case. This class diagram only focuses on the modification of Check in and Check out dates of existing booking and not any other functionality.



4. Class Diagram for Bill Generation

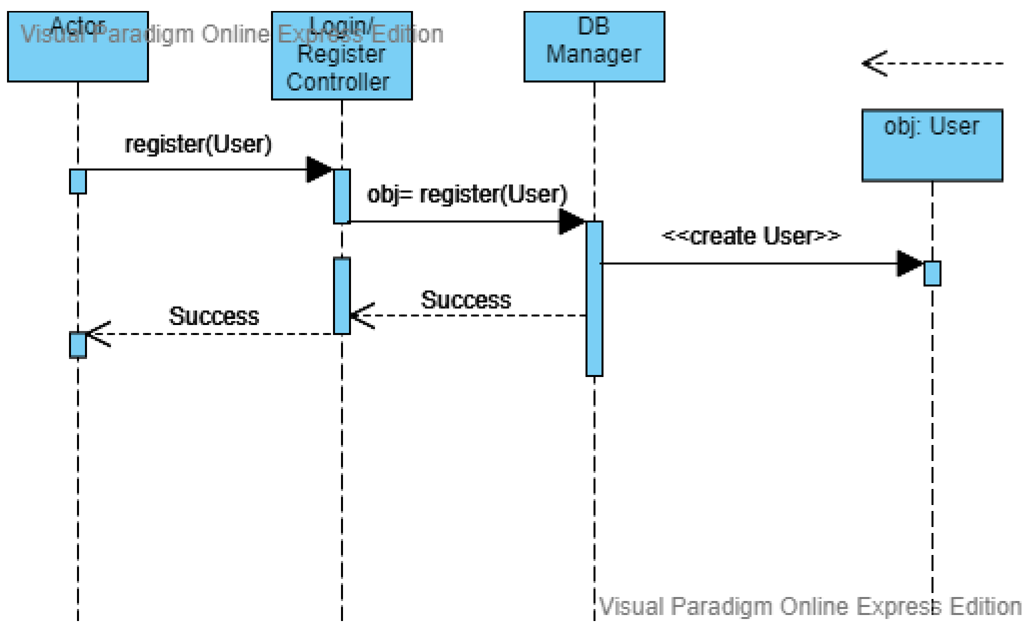


5. Class Diagram for Payment Redirection

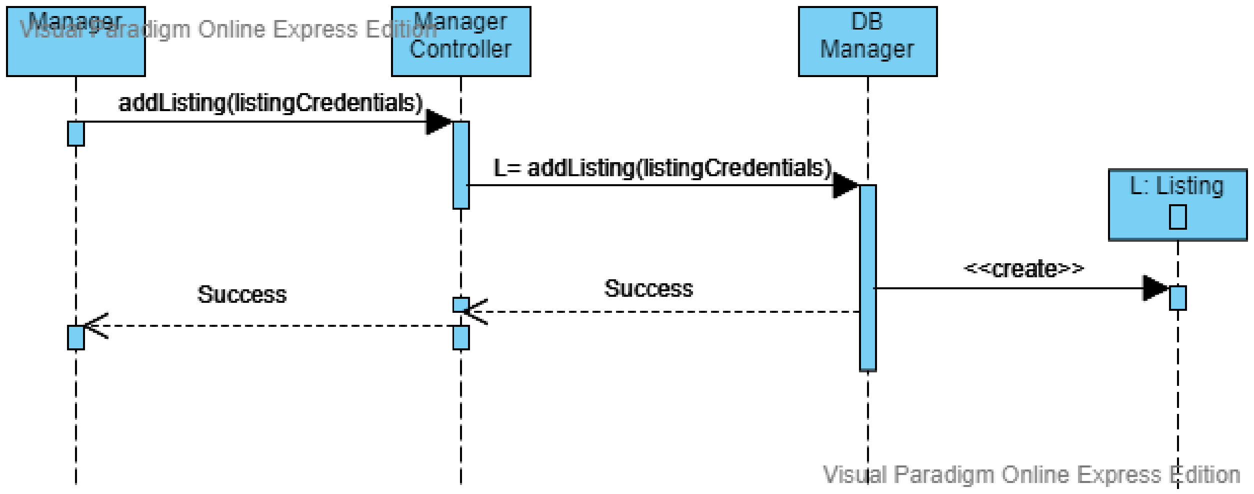


Sequence Diagrams

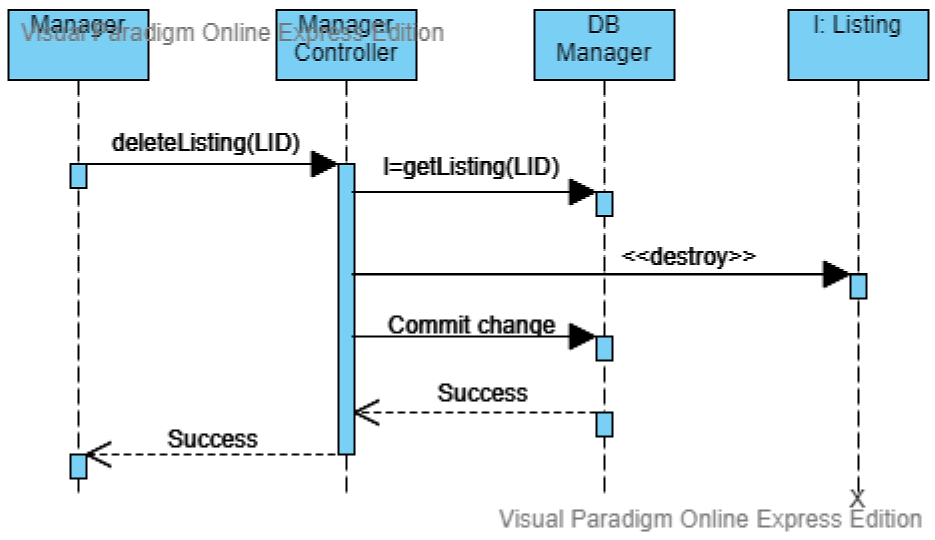
1. Refined Sequence Diagram for Registration of a new User with the system. The user can be any of the two actors. (Manager/Customer).



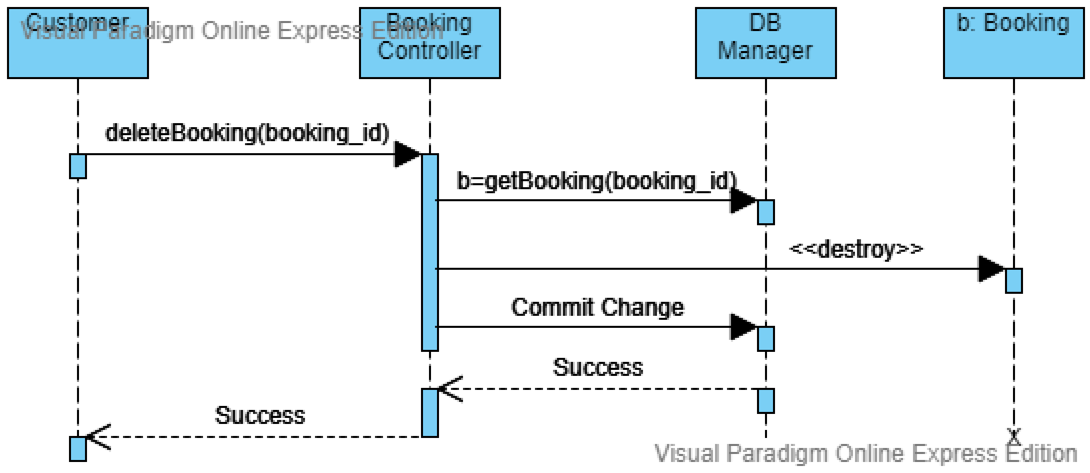
2. Sequence diagram for Add Listing. Here, there is only one actor Manager that has the rights to do so.



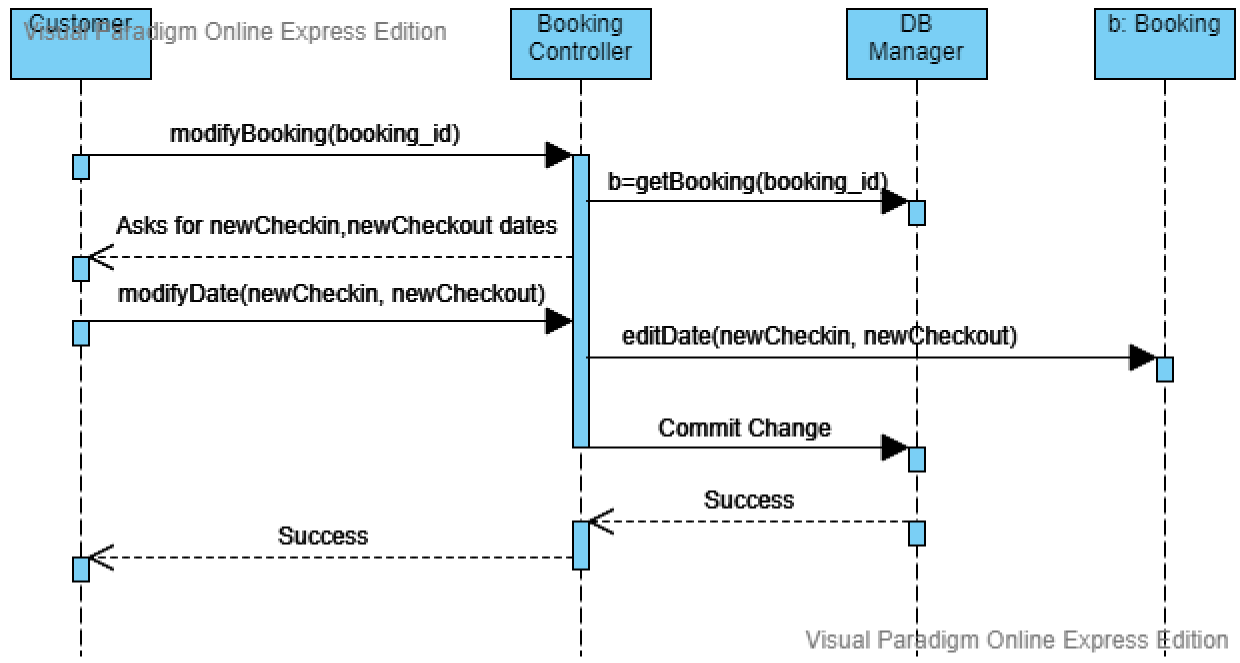
3. Sequence diagram for Delete Listing. Here, there is only one actor Manager that has the rights to do so.



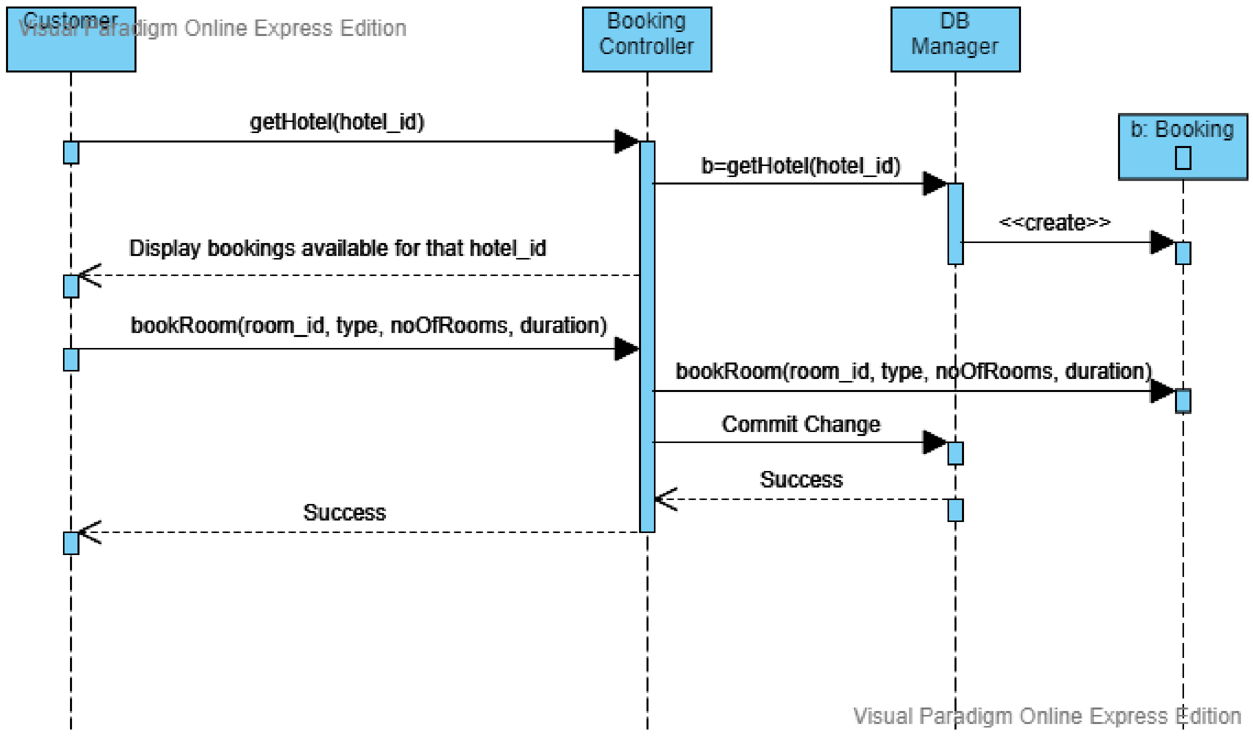
4. Sequence diagram for Delete Booking. Here, there is only one actor Customer that has the rights to do so.



5. Sequence diagram for Modify Booking. Here, there is only one actor Customer that has the rights to do so.


Customer
Visual Paradigm Online Express Edition
Booking Controller
DB Manager
b: Booking
modifyBooking(booking_id)
b=getBooking(booking_id)
Asks for newCheckin,newCheckout dates
modifyDate(newCheckin, newCheckout)
editDate(newCheckin, newCheckout)
Commit Change
Success
Success
Visual Paradigm Online Express Edition

6. Sequence diagram to Book Room. Here, there is only one actor Customer that has the rights to do so.


Customer
Visual Paradigm Online Express Edition
Booking Controller
DB Manager
b: Booking
getHotel(hotel_id)
b=getHotel(hotel_id)
<<create>>
Display bookings available for that hotel_id
bookRoom(room_id, type, noOfRooms, duration)
bookRoom(room_id, type, noOfRooms, duration)
Commit Change
Success
Success
Visual Paradigm Online Express Edition

Code is available on GitHub: https://github.com/UTDClassroom/CS6359002-Course_Project-team3