Software Engineering 1
Hotel Reservation System

Ahmed Elkholy, Otabek Erkinov, Tatevik Sargsyan, Osman Can Savran

24.01.2023

Abstract

This project aims to create a simple hotel booking system, making it easy for customers to reserve rooms and for staff to manage reservations. The system will offer a straightforward way to check room availability, book stays, and handle payments, all in one place.

Contents

1 Introduction

This project is about making a new hotel reservation app that is easy to use and helpful for both customers and hotels. We want to improve how people book hotels and make hotel management simpler. The app will connect customers with many different hotels, making the booking process easy and smooth. It will also allow customers and hotel managers to talk directly, making the service more personal. In the app, there are three main users: the Customer, the Hotel Manager, and the Admin. Customers can easily find hotels, book rooms, and chat with hotel managers. Hotel Managers can use the app to manage their hotel details, like room availability, and talk to customers. Admins can add or remove hotel managers to the system and use reports to make smart decisions for improvement.

2 User Stories

As a...	I want to...	So that...	MoSCoW criteria
Admin	add Hotel Managers	hotel managers can maintain and update their hotel details	M
Admin	remove Hotel Managers	I ensure only legitimate hotel managers have access	M
Admin	generate reports	I analyze platform performance and usage	S
Hotel Manager	modify user reviews	I modify user reviews	W
Hotel Manager	add a new hotel	customers can view and book it	M
Hotel Manager	remove a new hotel	I can ensure that outdated or incorrect listings are not visible to customers	M
Hotel Manager	edit hotel details	I provide accurate and up-to-date information	M
Hotel Manager	respond to customer reviews	engage with our guests and address any concerns	S
Hotel Manager	add discounts	our hotel attracts more customers	C
Hotel Manager	remove discounts	customers do not outdated prices	C
Hotel Manager	chat with customers	I can promptly assist our customers	C
Customer	search for hotels	I can make an informed decision and choose where to stay	M
Customer	make a booking	I ensure I have a place to stay during my trip	M
Customer	modify my booking	I can make necessary changes to my reservation without having to cancel and re-book.	C
Customer	cancel my booking	I can inform the hotel that I no longer need the reservation	S
Customer	make a payment	I pay for the room and confirm my reservation	M
Customer	write a review	I provide feedback and help other travelers	S
Customer	chat with the Hotel Manager	I can ask specific questions about my booking or room preferences	S

3 Acceptance Criteria and non-functional requirements

3.1 Manage Hotel Managers on the Website

Acceptance Criteria

- Can Admin add a Hotel Manager to the platform? Yes.
- Can Admin remove a Hotel Manager from the platform? Yes.
- Can a Hotel Manager add themselves to the platform without admin intervention? No.

Non-functional Requirement

- Can Admin perform these actions 24/7? Yes.
- Is there a limit to the number of Hotel Managers an Admin can add? No.

3.2 Manage Hotel Listings

Acceptance Criteria

- Can Hotel Manager add a new hotel listing? Yes.
- Can Hotel Manager remove or edit an existing hotel listing? Yes.
- Are there checks to ensure information accuracy by the Hotel Manager? Yes.
- Can a Customer modify hotel listings? No.

Non-functional Requirement

- Do updates to hotel listings reflect in real-time? Yes.
- Is there an admin approval needed to add hotel discounts? No.
- Can Hotel Managers manage listings 24/7? Yes.

3.3 Hotel Booking, Reviews, and Chats

Acceptance Criteria

- Can Customer search and find hotel listings? Yes.
- Can Customer make, modify, and cancel a booking? Yes.
- Can Customer write a review? Yes.
- Can Customer chat with the Hotel Manager? Yes.
- Can a Hotel Manager respond to a customer's review? Yes.
- Can a Hotel Manager initiate a chat with any Customer? No.

Non-functional Requirement

- Can a Customer chat with the Hotel Manager 24/7? No.
- Are Customer reviews visible to other Customers? Yes.

3.4 Payment

Acceptance Criteria

- Can Customer make a payment for their booking? Yes.
- Is there a verification process after payment? Yes.
- Can Hotel Manager issue a payment? No.

Non-functional Requirement

- Is the transaction security provided by the platform? No.
- Can Customers make payments 24/7? Yes.

4 Use Cases

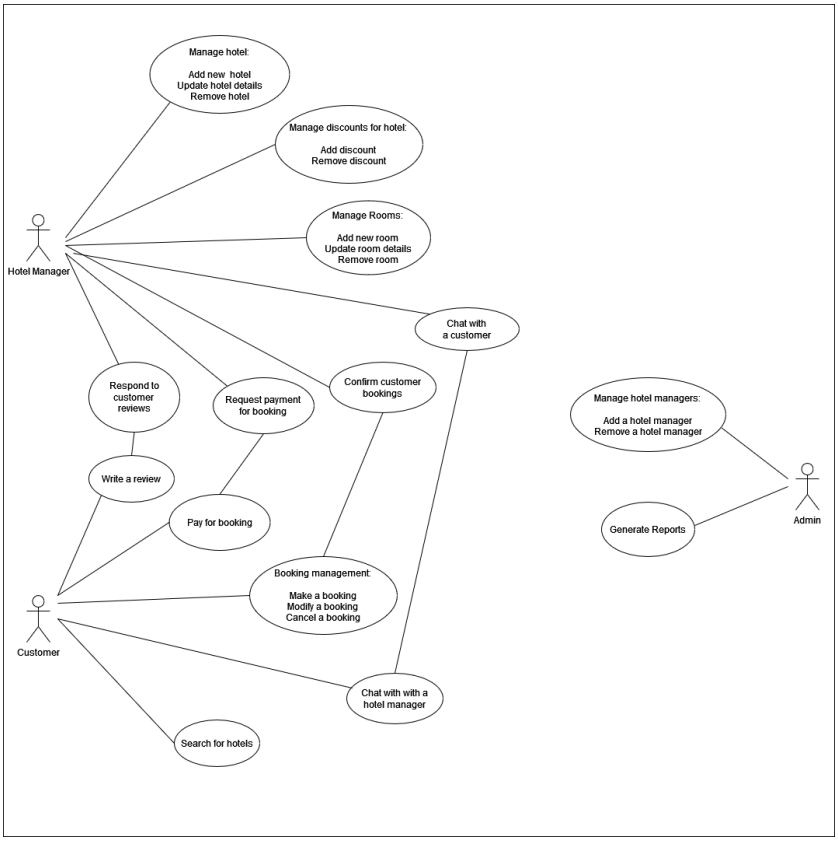

Figure 1: Use case diagram

This use case diagram showcases the primary interactions within a hotel management system across three main roles: Hotel Manager, Customer, and Admin. The Hotel Manager oversees hotel operations, manages rooms, and interacts with customers. Customers can search for hotels, manage bookings, and communicate with hotel managers. Meanwhile, the Admin supervises hotel manager accounts and generates system reports.

5 Class Diagrams

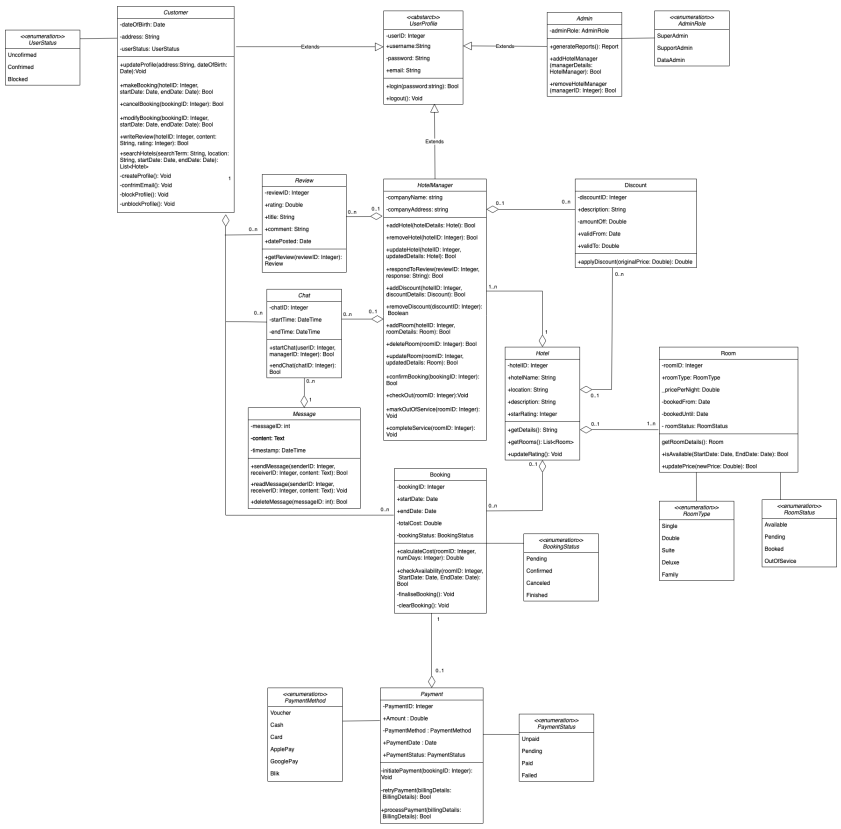

Figure 2: Class diagram

- UserProfile: A foundational class representing the general user profile in the system. It contains fundamental attributes like user ID, username, email, and password. The class provides basic authentication methods for logging in and logging out.
- Customer: Represents a typical user or customer in the system who seeks to book hotels. Apart from general profile attributes, a Customer has personal details like date of birth and address. The methods allow customers to manage their bookings, update their profiles, and interact with hotel listings.
- HotelManager: Represents a user responsible for managing hotel details and interactions with customers. A HotelManager might be associated with a company and has methods that allow them to manage hotel listings, rooms, discounts, and respond to reviews.

- Admin: This class symbolizes a system administrator with specific roles, such as a SuperAdmin or SupportAdmin. An Admin can perform high-level tasks like generating reports or managing hotel managers within the platform.
- Chat: Represents an interaction or conversation between users, possibly a customer and a manager. Each chat session has a unique ID, start time, and end time. The methods enable starting and ending a chat.
- Message: This class is associated with the Chat class and represents individual messages sent during a conversation. Each message has an ID, content, and timestamp. The methods facilitate sending and managing messages.
- Review: Represents feedback or reviews given by customers about hotels. Each review contains a rating, title, comment, and the date it was posted. The class provides a method to retrieve the review's details.
- Booking: Symbolizes a reservation made by a customer for a hotel room. The booking contains details like start and end dates, total cost, and a unique ID. Methods allow checking room availability and calculating the cost of the stay.
- Payment: Represents financial transactions within the system, usually associated with bookings. Payments have details like the amount, method used, and date of the transaction. The primary method processes these payments.
- Room: Symbolizes individual rooms within a hotel. Each room has a unique ID, type (like single or suite), price per night, and booking details. The methods facilitate retrieving room information, checking availability, and adjusting room prices.
- Discount: Represents promotional offers or discounts available in the system. Each discount has an ID, description, amount or percentage off, and its validity period. The applyDiscount method helps in adjusting prices based on these offers.
- Hotel: Represents individual hotel entities in the system. Each hotel has a unique ID, name, location, description, and star rating. The class provides methods to get a detailed overview of the hotel and fetch its rooms.

6 System States

In this section, we explore the state diagrams that illustrate the lifecycle of various entities within our system. Each diagram captures the dynamic behavior of an entity by depicting its states, the events that trigger transitions between these states, and the actions that result from these transitions. We'll examine the states involved in room availability for bookings, user profile verification, payment processing, and the stages of a chat message from being sent to deletion.

6.1 User Profile

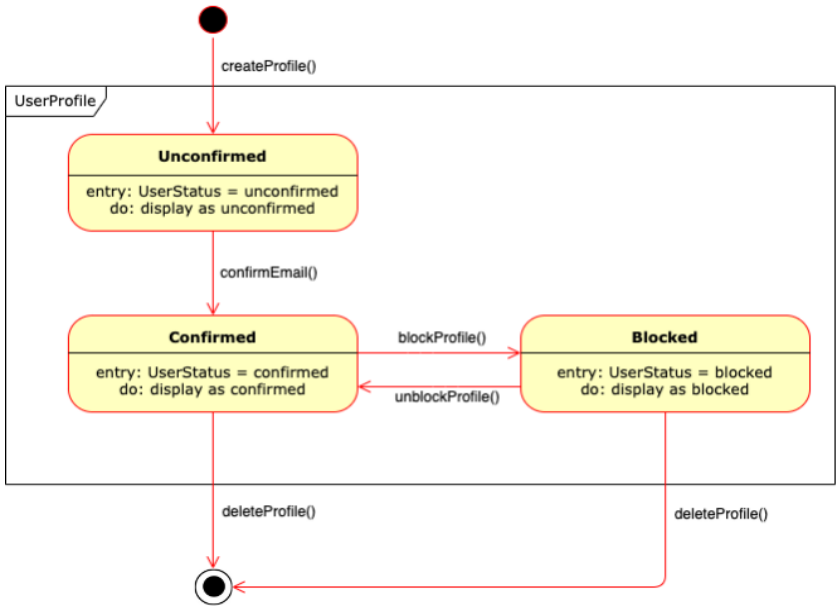

Figure 3: User profile states

States:

- Unconfirmed: A user profile is created but email confirmation is pending.
- Confirmed: The user's email is confirmed, fully activating the profile.
- Blocked: The profile is restricted and cannot be used until unblocked.

Transitions:

- createProfile() initializes an Unconfirmed profile.
- confirmEmail() changes the status from Unconfirmed to Confirmed.

- blockProfile() transitions a Confirmed profile to Blocked.
- unblockProfile() returns a Blocked profile to Confirmed.
- Profiles can be deleted from either Confirmed or Blocked state.

6.2 Room

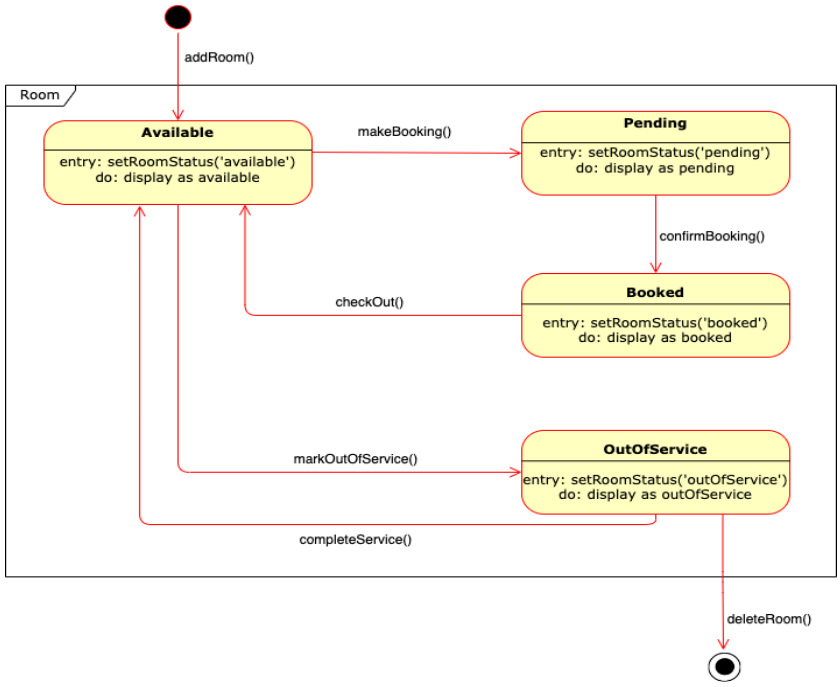

Figure 4: Room states

States:

- Available: The room is ready for booking.
- Pending: A booking is initiated but not confirmed.
- Booked: The booking is confirmed; the room is occupied.
- OutOfService: The room is unavailable for bookings due to maintenance or other reasons.

Transitions:

- Rooms become Pending when someone starts the booking process.
- Pending rooms become Booked upon booking confirmation.
- Rooms return to Available after the guest checks out or maintenance is completed.
- Rooms can be marked OutOfService if not fit for use or deleted from the system if no longer needed.

6.3 Booking

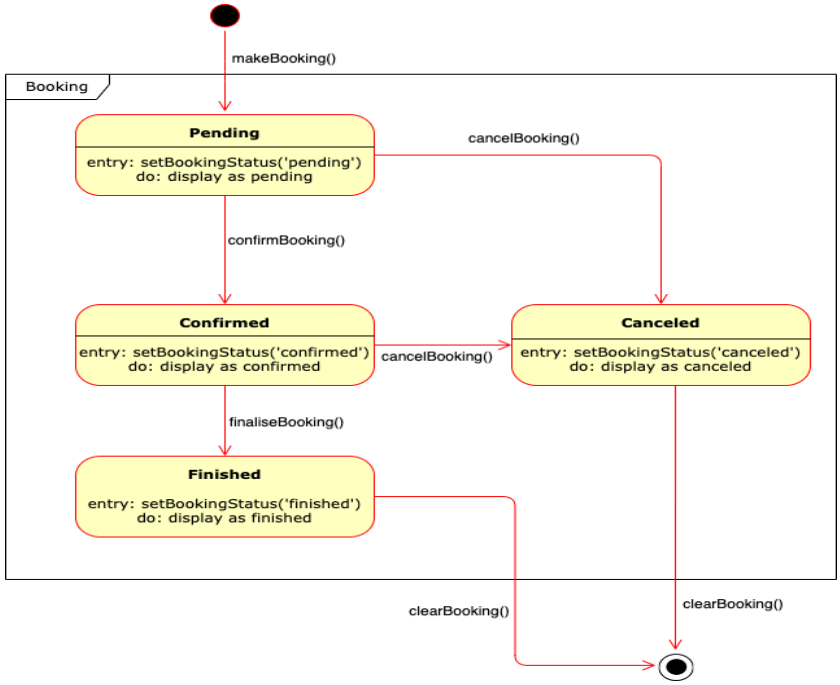

Figure 5: Booking states

States:

- Pending: A booking is made and awaiting confirmation.
- Confirmed: The booking has been confirmed and is currently active.
- Canceled: The booking has been canceled prior to completion.
- Finished: The booking process has been completed.

Transitions:

- makeBooking() starts a Pending booking.
- confirmBooking() changes Pending to Confirmed.
- cancelBooking() can be invoked on Pending or Confirmed bookings to change their status to Canceled.
- finaliseBooking() moves a Confirmed booking to Finished.
- clearBooking() is the final transition that can be applied to Canceled or Finished bookings to remove them from the system.

6.4 Payment

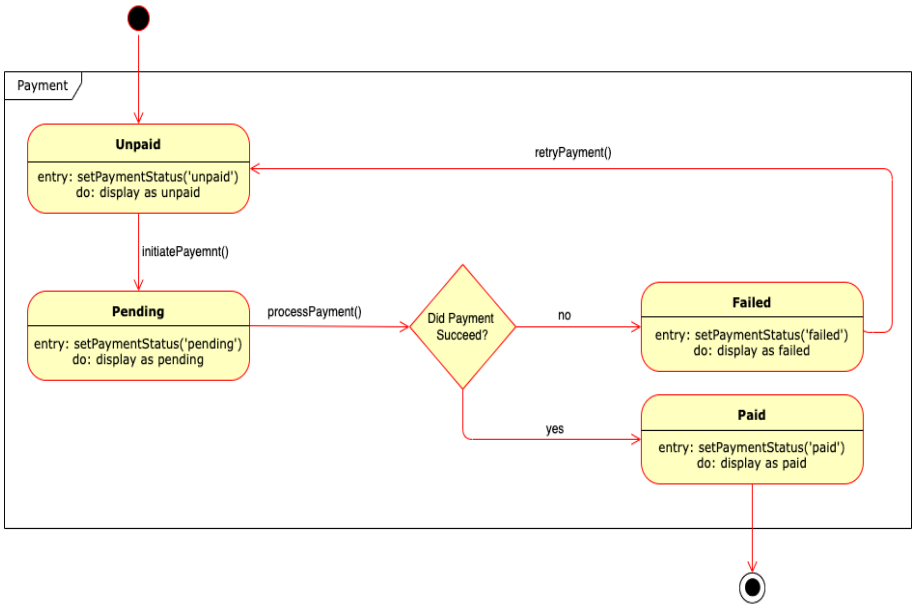

Figure 6: Payment states

States:

- Unpaid: The default state before payment initiation.
- Pending: Payment has been initiated and is being processed.
- Failed: Payment processing attempted but was unsuccessful.
- Paid: Payment has been successfully processed and completed.

Transitions:

- initiatePayment() moves from Unpaid to Pending.
- processPayment() decides whether the payment goes to Failed or Paid based on success.
- retryPayment() allows another attempt at processing from Failed, returning to Unpaid.

6.5 Chat and Messages

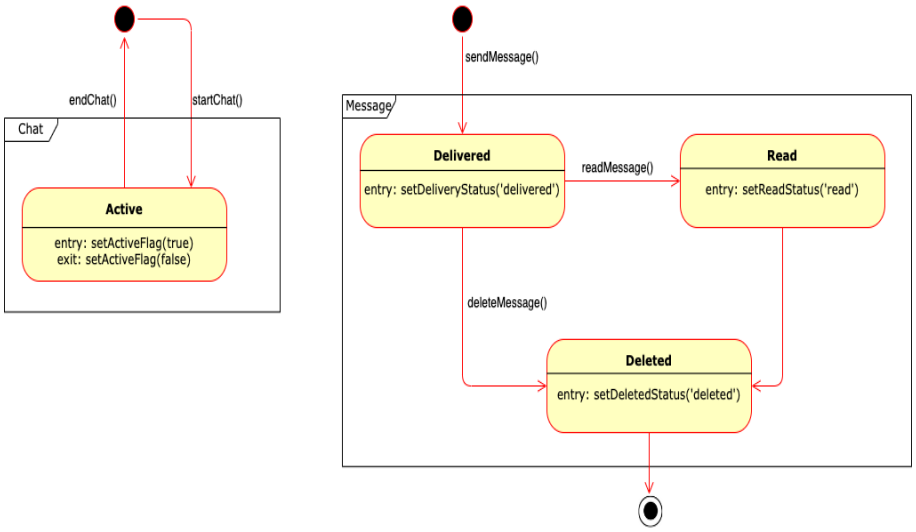

Figure 7: Chat and messages states

Chat states:

- Active: A chat session is currently ongoing.

Chat transitions:

- startChat(): Initiates an active chat session.
- endChat(): Terminates the chat session.

Message states:

- Delivered: A message has been sent successfully.
- Read: The message has been opened and read by the recipient.
- Deleted: The message has been removed from the chat.

Message transitions:

- sendMessage(): Changes message status from any prior state to Delivered.
- readMessage(): Marks a Delivered message as Read.
- deleteMessage(): Removes a message, changing its status to Deleted from either Delivered or Read.

7 System Activities

This section outlines the system activities for our application, detailing the workflows for booking, payment, discounts, and customer service chat. The booking flow guides the user from initiating a booking to confirming or canceling based on room availability and payment success. The payment flow captures the process from selecting a payment method to the final transaction outcome. The discount flow details how discounts are applied and room prices updated by the hotel manager. Lastly, the chat flow describes the interactive communication between the customer and the hotel manager, from the initiation of the chat to its conclusion. Each flow is designed to streamline the user experience and administrative operations within the hotel booking service.

7.1 Booking Flow

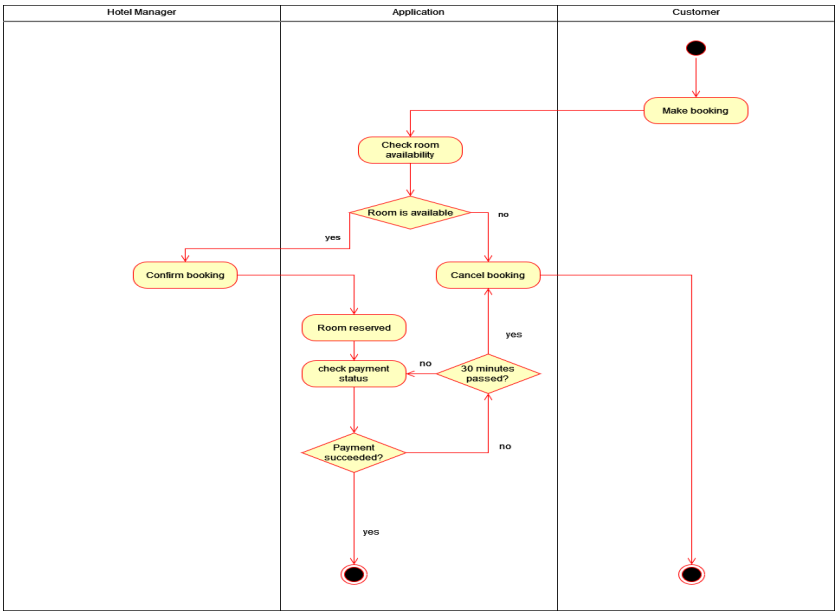

Figure 8: Booking flow

In the booking flow, the customer initiates the process by creating a booking, which prompts the application to check for room availability. If a room is available, the application proceeds to reserve it and then checks the payment status. The booking is confirmed only if the payment is successful. However, if at any point the room is not available or the payment fails, the application cancels the booking.

7.2 Payment Flow

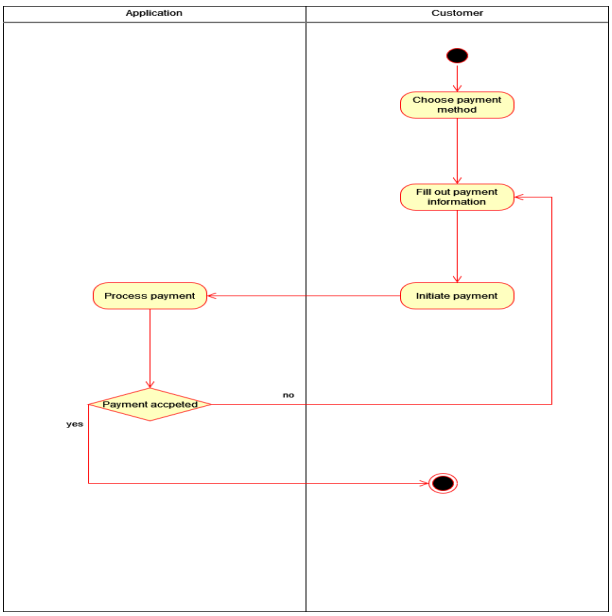

Figure 9: Payment flow

The payment flow begins with the customer selecting a payment method and providing the requisite payment details. The payment is then initiated and processed by the application. On successful processing, the payment is accepted, concluding the transaction. In the event of processing failure, the transaction is terminated.

7.3 Discount Flow

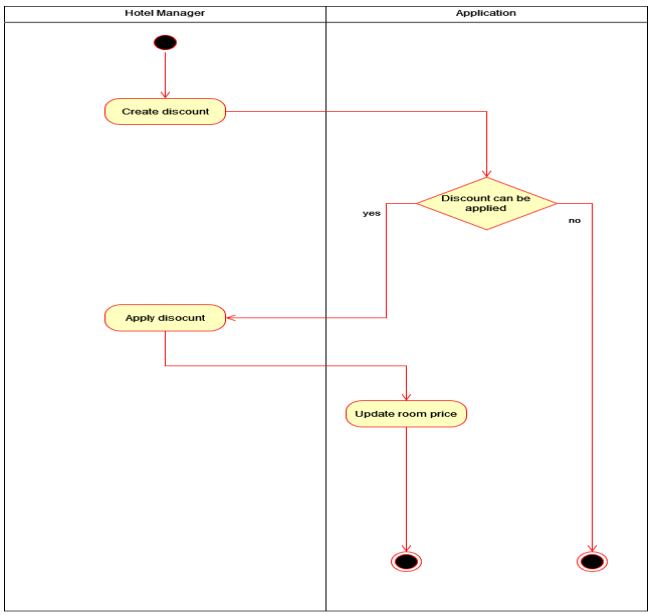

Figure 10: Discount flow

The discount flow is initiated by the hotel manager creating a discount. The application then determines whether the discount can be applied. If applicable, the discount is applied and the room price is updated accordingly. If not, the process concludes without any changes to the pricing.

7.4 Chat Flow

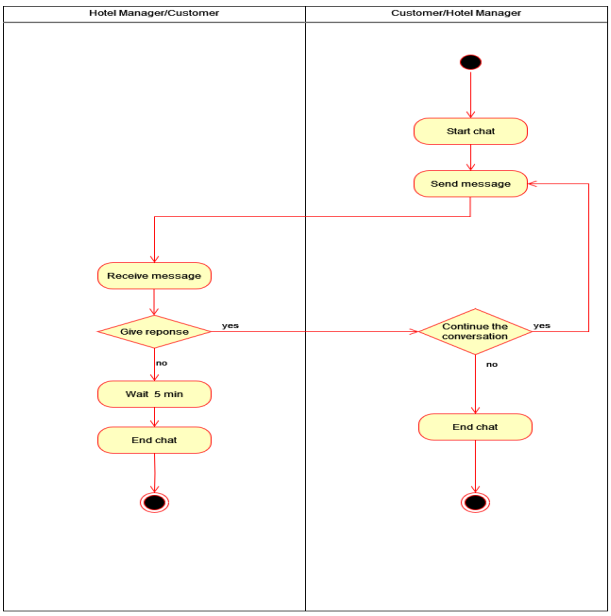

Figure 11: Chat flow

The chat flow begins with the customer starting a chat session and sending a message. The hotel manager receives this message and provides a response. If the conversation continues, the exchange of messages persists. Otherwise, if there is no further communication from either side, the chat session is concluded.

8 Sequence Diagrams

8.1 Booking Sequence

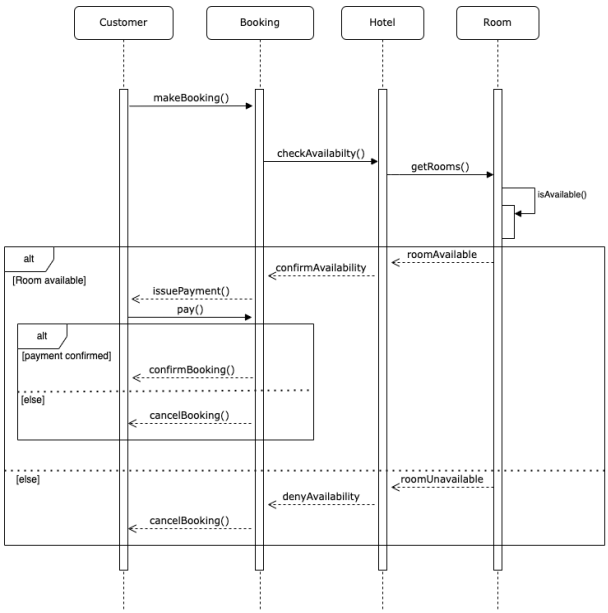

Figure 12: Discount sequence

The sequence diagram illustrates a room booking process. It begins with the Customer initiating a makeBooking() request to the Booking service. The Booking service then checks room availability with the Hotel, which in turn verifies this via the Room service. If a room is available, the Hotel confirms this to the Booking service, which prompts the Customer to issue a payment. If the payment is confirmed, the Booking service confirms the booking with confirmBooking() method back to the Customer. Alternatively, if the payment cannot be confirmed or no room is available, the Booking service cancels the booking with cancelBooking() method to the Customer, and the Hotel may deny the booking with a denyAvailability() message if the room is not available.

8.2 Discount Sequence

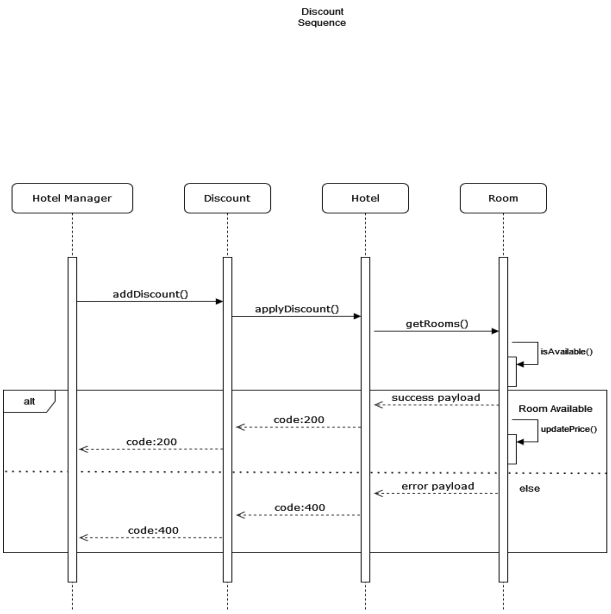

Figure 13: Discount sequence

In the provided sequence diagram, we observe the workflow for applying a discount to a hotel room through a hotel booking system. The process is initiated by the Hotel Manager, who issues an addDiscount() command to the Discount service. Upon receiving this command, the Discount service proceeds to invoke the applyDiscount() method on the Hotel service. The Hotel service, in turn, requests available rooms by calling the getRooms() method on the Room service. The Room service checks for availability and, if a room is available, it sends back the relevant data payload, which includes the updated price reflecting the discount. This updated information is passed back through the Hotel service to the Discount service, which concludes the transaction. The sequence diagram includes alternative paths to handle different responses, in case rooms are not available(booked for the period of the discount) 400 code is returned.

8.3 Review Sequence

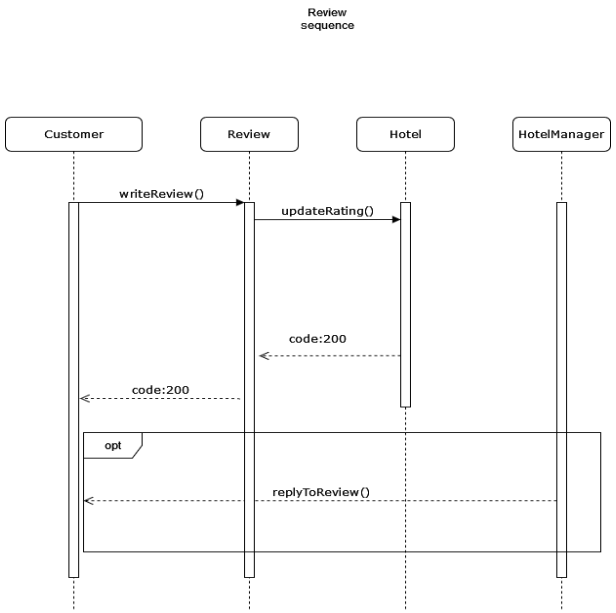

Figure 14: Review Sequence

The sequence diagram outlines a review process. It starts with a Customer writing a review, which triggers the Review object to prompt the Hotel to update its rating. Once the update is successful, indicated by a 200 OK message, the HotelManager has the option to reply to the review, which completes the interaction loop back to the Customer. The 'opt' label indicates that the Hotel Manager's reply is not mandatory but an additional feature.

9 Server Application endpoints API

9.1 UserProfile

9.1.1 Register User

Endpoint: /api/user/register
Method: POST
Response Status:

- `200`: OK (User Registered)
- `403`: Forbidden (User is not allowed to register, invalid format or already registered etc.)

9.1.2 Delete User

Endpoint: /api/user/delete
Method: POST
Response Status:

- `200`: OK (Deletion successful)

9.1.3 User Login

Endpoint: /api/user/login
Method: POST
Parameters:

- `Username (string)`: The username of the user
- `Password (string)`: The password of the user

Response Status:

- `200`: OK (Login successful)
- `401`: Unauthorized (Username or password is incorrect)
- `403`: Forbidden (User is not allowed to log in, perhaps due to a ban or not having confirmed email)
- `404`: Not Found (User does not exist)

9.1.4 User Logout

Endpoint: /api/user/logout
Method: POST
Parameters:

- `UserID (int)`: The identifier of the user

Response Status:

- `200:` OK (Logout successful)
- `401:` Unauthorized (UserID is invalid or user not logged in)
- `404:` Not Found (User does not exist)

9.2 Customer

9.2.1 Update Profile

Endpoint: /api/user/update-profile
Method: POST
Parameters:

- `UserID (int):` The identifier of the customer
- `Address (string):` The new address of the customer
- `DateOfBirth (datetime):` The new date of birth of the customer

Response Status:

- `201:` OK (Profile updated successfully)
- `400:` Invalid ID (Parameters are invalid)
- `401:` Forbidden (UserID is invalid or unauthorized access)
- `404:` Not Found (Customer not found)

9.2.2 Make Booking

Endpoint: /api/customer/make-booking
Method: POST
Parameters:

- `UserID (int):` The identifier of the customer
- `HotelID (int):` The identifier of the hotel
- `StartDate (datetime):` The start date of the booking
- `EndDate (datetime):` The end date of the booking

Response Status:

- `201:` Created (Booking made successfully)
- `400:` Bad Request (Parameters are invalid)
- `401:` Unauthorized (UserID is invalid or unauthorized access)
- `403:` Forbidden (Booking not allowed)
- `404:` Not Found (Hotel or customer not found)

9.2.3 Cancel Booking

Endpoint: /api/customer/cancel-booking
Method: POST
Parameters:

- `UserID (int)`: The identifier of the customer
- `BookingID (int)`: The identifier of the booking

Response Status:

- `200`: OK (Booking cancelled successfully)
- `401`: Unauthorized (UserID or BookingID is invalid or unauthorized access)
- `404`: Not Found (Booking or customer not found)

9.2.4 Modify Booking

Endpoint: /api/customer/modify-booking
Method: POST
Parameters:

- `UserID (int)`: The identifier of the customer
- `BookingID (int)`: The identifier of the booking
- `StartDate (datetime)`: The start date of the booking
- `EndDate (datetime)`: The end date of the booking

Response Status:

- `200`: OK (Booking modified successfully)
- `400`: Bad Request (Parameters are invalid)
- `401`: Unauthorized (UserID or BookingID is invalid or unauthorized access)
- `404`: Not Found (Booking or customer not found)

9.2.5 Write Review

Endpoint: /api/customer/write-review
Method: POST
Parameters:

- `UserID (int)`: The identifier of the customer
- `BookingID (int)`: The identifier of the booking

- `HotelID (int)`: The identifier of the hotel
- `Content (string)`: The content of the review
- `Rating (int)`: The rating given to the hotel

Response Status:

- `201`: Created (Review submitted successfully)
- `400`: Bad Request (Parameters are invalid)
- `401`: UserID is invalid or unauthorized access
- `404`: Not Found (Hotel or customer not found)

9.2.6 Search Hotels

Endpoint: /api/customer/search-hotels
Method: GET
Parameters:

- `SearchTerm (string)`: The term used to search for hotels
- `Location (string)`: The location where to search for hotels
- `StartDate (datetime)`: The start date for availability
- `EndDate (datetime)`: The end date for availability
- `Price (int)`: The price for the room
- `NumberOfGuests (int)`: The number of guests in the room

Response Status:

- `200`: OK (Search completed successfully)
- `400`: Bad Request (Parameters are invalid)
- `404`: Not Found (No hotels found matching the criteria)

9.3 Hotel Manager

9.3.1 Add Hotel

Endpoint: /api/hotel-manager/add-hotel
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `HotelDetails (JSON/object)`: The details of the hotel to be added

Response Status:

- `201:` Created (Hotel added successfully)
- `400:` Bad Request (Parameters are invalid or insufficient)
- `401:` Unauthorized (ManagerID is invalid or unauthorized access)
- `403:` Forbidden (Manager is not allowed to add hotels)

9.3.2 Remove Hotel

Endpoint: /api/hotel-manager/update-hotel
Method: POST
Parameters:

- `ManagerID (int):` The identifier of the manager
- `HotelID (int):` The identifier of the hotel to be updated
- `UpdatedDetails (JSON/object):` The new details for the hotel

Response Status:

- `200:` OK (Hotel updated successfully)
- `400:` Bad Request (Parameters are invalid or insufficient)
- `401:` Unauthorized (ManagerID is invalid or unauthorized access)
- `404:` Not Found (Hotel not found)

9.3.3 Add Discount

Endpoint: /api/hotel-manager/add-discount
Method: POST
Parameters:

- `ManagerID (int):` The identifier of the manager
- `HotelID (int):` The identifier of the hotel to be updated
- `DiscountDetails (JSON/object):` The details of the discount to be added

Response Status:

- `201:` Created (Discount added successfully)
- `400:` Bad Request (Parameters are invalid or insufficient)
- `401:` Unauthorized (ManagerID is invalid or unauthorized access)
- `404:` Not Found (Hotel not found)

9.3.4 Remove Discount

Endpoint: /api/hotel-manager/remove-discount
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `DiscountID (int)`: The identifier of the discount to be removed

Response Status:

- `200`: OK (Discount removed successfully)
- `401`: Unauthorized (ManagerID is invalid or unauthorized access)
- `404`: Not Found (Discount not found)

9.3.5 Add Room

Endpoint: /api/hotel-manager/add-room
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `HotelID (int)`: The identifier of the hotel
- `RoomDetails (JSON/object)`: The details of the room to be added

Response Status:

- `201`: Created (Room added successfully)
- `400`: Bad Request (Parameters are invalid or insufficient)
- `401`: Unauthorized (ManagerID is invalid or unauthorized access)
- `404`: Not Found (Hotel not found)

9.3.6 Delete Room

Endpoint: /api/hotel-manager/delete-room
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `RoomID (int)`: The identifier of the room to be deleted

Response Status:

- `200`: OK (Room deleted successfully)
- `401`: Unauthorized (ManagerID is invalid or unauthorized access)
- `404`: Not Found (Room not found)

9.3.7 Update Room

Endpoint: /api/hotel-manager/update-room
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `RoomID (int)`: The identifier of the room to be deleted
- `UpdatedDetails (JSON/object)`: The new details for the room

Response Status:

- `200`: OK (Room updated successfully)
- `400`: Bad Request (Parameters are invalid or insufficient)
- `401`: Unauthorized (ManagerID is invalid or unauthorized access)
- `404`: Not Found (Room not found)

9.3.8 Confirm Booking

Endpoint: /api/hotel-manager/confirm-booking
Method: POST
Parameters:

- `ManagerID (int)`: The identifier of the manager
- `BookingID (int)`: The identifier of the booking to be confirmed

Response Status:

- `200`: OK (Booking confirmed successfully)
- `400`: Bad Request (Parameters are invalid or insufficient)
- `401`: Unauthorized (ManagerID is invalid or unauthorized access)
- `404`: Not Found (Booking not found)

9.4 Review

9.4.1 Get Review

Endpoint: /api/review
Method: POST
Parameters:

- `ReviewID (int)`: The identifier of the review to retrieve

Response Status:

- `200:` OK (Review retrieved successfully)
- `400:` Bad Request (ReviewID is invalid)
- `404:` Not Found (Review does not exist)

9.5 Discount

9.5.1 Apply Discount

Endpoint: /api/discount/apply
Method: POST
Parameters:

- `DiscountID (int):` The identifier of the discount to apply
- `OriginalPrice (double):` The original price before the discount is applied

Response Status:

- `200:` OK (Discount applied successfully)
- `400:` Bad Request (Parameters are invalid)
- `404:` Not Found (Discount does not exist)

9.5.2 Remove Discount

Endpoint: /api/discount/remove
Method: POST
Parameters:

- `DiscountID (int):` The identifier of the discount to remove

Response Status:

- `200:` OK (Discount removed successfully)
- `404:` Not Found (Discount does not exist)

9.6 Chat

9.6.1 Create Chat

Endpoint: /api/chat
Method: POST
Parameters:

- `ChatID (int):` The identifier of the chat session.
- `StartTime (datetime):` The start time of the chat session.

- `EndTime (datetime)`: The end time of the chat session.

Response Status:

- `201`: OK (Chat session started successfully)
- `401`: Invalid ID (ChatID is invalid)
- `403`: Forbidden (Unauthorized access)
- `404`: Not Found (Chat session not found)

9.6.2 Start Chat

Endpoint: /api/chat/start
Method: POST
Parameters:

- `ChatID (int)`: The identifier of the chat session.

Response Status:

- `201`: OK (Chat session started successfully)
- `401`: Invalid ID (ChatID is invalid)
- `403`: Forbidden (Unauthorized access)
- `404`: Not Found (Chat session not found)

9.6.3 End Chat

Endpoint: /api/chat/end
Method: POST
Parameters:

- `ChatID (int)`: The identifier of the chat session.

Response Status:

- `200`: OK (Chat session ended successfully)
- `401`: Invalid ID (ChatID is invalid)
- `403`: Forbidden (Unauthorized access)
- `404`: Not Found (Chat session not found)

9.6.4 Report Chat

Endpoint: /api/chat/report
Method: POST
Parameters:

- `ChatID (int)`: The identifier of the chat session.

Response Status:

- `201`: OK (Chat session reported successfully)
- `404`: Not Found (Chat session not found)
- `403`: Forbidden (Unauthorized access)

9.7 Message

9.7.1 Send Message

Endpoint: /api/message/send
Method: POST
Parameters:

- `MessageID (int)`: The identifier of the message.
- `Content (string)`: The content of the message.
- `Timestamp (datetime)`: The timestamp of the message.

Response Status:

- `201`: OK (Message sent successfully)
- `401`: Invalid ID (MessageID is invalid)
- `403`: Forbidden (Unauthorized access)
- `404`: Not Found (Chat or Message not found)

9.7.2 Delete Message

Endpoint: /api/message/delete
Method: POST
Parameters:

- `MessageID (int)`: The identifier of the message to be deleted.

Response Status:

- `200:` OK (Message deleted successfully)
- `401:` Invalid ID (MessageID is invalid)
- `403:` Forbidden (Unauthorized access)
- `404:` Not Found (Chat or Message not found)

9.7.3 Report Message

Endpoint: /api/message/report
Method: POST
Parameters:

- `MessageID (int):` The identifier of the message.

Response Status:

- `201:` OK (Message reported successfully)
- `404:` Not Found (Chat or Message not found)
- `403:` Forbidden (Unauthorized access)

9.8 Booking

9.8.1 Calculate Cost

Endpoint: /api/booking/calculate_cost
Method: POST
Parameters:

- `BookingDate (date):` The date when the booking is made.
- `StartDate (date):` The start date of the booking period.
- `EndDate (date):` The end date of the booking period.

Response Status:

- `200:` OK (Cost calculated successfully)
- `400:` Bad Request (Invalid date range or other request error)
- `404:` Not Found (Booking not found)

9.8.2 Check Availability

Endpoint: /api/booking/check_availability
Method: POST
Parameters:

- `StartDate (date)`: The start date of the booking period.
- `EndDate (date)`: The end date of the booking period.

Response Status:

- `200`: OK (Booking is available for the given date range)
- `400`: Bad Request (Invalid date range or other request error)
- `404`: Not Found (Booking not found)

9.9 Payment

9.9.1 Process Payment

Endpoint: /api/payment/process
Method: POST
Parameters:

- `PaymentID (int)`: The identifier of the payment.
- `PaymentMethod (string)`: The payment method used (e.g., credit card, PayPal).
- `PaymentDate (date)`: The date when the payment is processed.
- `Amount (float)`: The amount of the payment.

Response Status:

- `201`: Created (Payment processed successfully)
- `400`: Bad Request (Invalid payment details or other request error)
- `404`: Not Found (Payment not found)

9.10 Hotel

9.10.1 Get Hotel Details

Endpoint: /api/hotel/details
Method: GET
Parameters:

- `HotelID (int)`: The identifier of the hotel.

Response Status:

- `200`: OK (Hotel details retrieved successfully)
- `400`: Bad Request (Invalid HotelID or other request error)
- `404`: Not Found (Hotel not found)

9.10.2 Get Room Information

Endpoint: /api/hotel/room
Method: GET
Parameters:

- `HotelID (int)`: The identifier of the hotel.

Response Status:

- `200`: OK (Room information retrieved successfully)
- `400`: Bad Request (Invalid HotelID or other request error)
- `404`: Not Found (Hotel not found)

9.11 Room

9.11.1 Get Room Details

Endpoint: /api/room/details
Method: GET
Parameters:

- `RoomID (int)`: The identifier of the room.

Response Status:

- `200`: OK (Room details retrieved successfully)
- `400`: Bad Request (Invalid RoomID or other request error)
- `404`: Not Found (Room not found)

9.11.2 Check Room Availability

Endpoint: /api/room/availability
Method: POST
Parameters:

- `RoomID (int)`: The identifier of the room.
- `CheckInDate (date)`: The start date for availability check.
- `CheckOutDate (date)`: The end date for availability check.

Response Status:

- `200`: OK (Room is available for the given date range)
- `400`: Bad Request (Invalid RoomID or other request error)
- `404`: Not Found (Room not found)

9.11.3 Update Room Price

Endpoint: /api/room/update_price
Method: PUT
Parameters:

- `RoomID (int)`: The identifier of the room.
- `NewPrice (float)`: The new price per night for the room.

Response Status:

- `200`: OK (Room price updated successfully)
- `400`: Bad Request (Invalid RoomID or other request error)
- `404`: Not Found (Room not found)

10 Abnormal Behavior

A flawed or incomplete execution of the application life cycle can lead to adverse outcomes, including poor user experience, loss of data, or a lack of responsiveness.

10.1 Invalid Input Data

When updating profiles, making or modifying bookings, writing reviews, or searching hotels, there's a risk of receiving invalid input (like incorrect date formats, invalid addresses, non-existent hotel IDs, etc.). This might lead to failures in processing requests or incorrect data being stored.

10.2 Database or Network Failures

Operations like updating profiles, making bookings, and writing reviews involve database transactions. Network or database issues could lead to incomplete or failed transactions. For example, a booking might be half-processed if the database fails mid-transaction.

10.3 Concurrency Issues

If multiple customers are trying to book the same hotel room for overlapping dates, it could lead to concurrency problems. Similarly, modifying bookings simultaneously could cause data inconsistencies.

10.4 Authentication/Authorization Failures

If the system requires customers to be authenticated, there could be issues where a customer is unable to log in or is wrongly authenticated. Unauthorized access to modify or cancel bookings could lead to security breaches.

10.5 Handling of Non-existent Resources

Attempts to modify or cancel bookings that don't exist, or writing reviews for hotels that are not in the database. This could cause errors or exceptions in the system.

10.6 Data Validation Issues

Reviews with inappropriate content or ratings that are outside an acceptable range. Incorrect date ranges for bookings (e.g., start date is after end date) or past dates for new bookings.

10.7 System Overload

If the system receives too many requests simultaneously (like during a promotional event), it could become overloaded, leading to slow response times or crashes.

10.8 Partial System Failures

In a distributed system, part of the system (like the hotel search service) might fail while others continue to work. This can lead to partial functionality being available to the customer.