Software Requirements and Design Document

HostelKonnect, one stop platform for hostels

Prepared by I22-1183, I22-1132, I22-0996

SCRAMIT

26-11-2024

Table of Contents

1. Introduction

1.1 Purpose

This document specifies the software requirements for **Hostel Konnect**, a platform aimed at connecting students with hostels. It covers the complete system requirements for **Version 1.0**, including student modules and hostel management functionalities. The scope of this SRS encompasses all features necessary to streamline processes for students and hostel administrators, ensuring an efficient and user-friendly experience.

1.2 Product Scope

Hostel Konnect is a platform designed to simplify hostel accommodations by enabling students to find suitable hostels and providing hostel administrators with tools to manage bookings and records.

- ***Purpose**: To enhance accessibility for students and streamline management for hostels.*
- ***Benefits**: Saves time, reduces administrative workload, and improves efficiency through digital transformation.*

1.3 Title

HostelKonnect, one stop platform for hostels

1.4 Objectives

*The major aims of the **Hostel Konnect** project are:*

- *To simplify the process for students to find and apply for hostels.*
- *To provide hostel administrators with tools for efficient booking and record management.*
- *To minimize manual effort and errors through digital automation.*
- *To create a user-friendly platform that ensures reliability and ease of use.*

1.5 Problem Statement

Students often face significant challenges when moving to a new city, particularly in finding suitable hostel accommodations. The process is time-consuming, involves visiting multiple locations, and lacks centralized information about availability, pricing, and amenities. This can lead to delays, frustration, and suboptimal choices.

Hostel administrators also face difficulties in managing inquiries, maintaining accurate records, and allocating rooms efficiently due to reliance on manual processes. This results in miscommunication, errors, and inefficient operations.

Feasibility:

With the widespread adoption of digital solutions, **Hostel Konnect** offers a feasible and scalable solution to these problems. It provides a centralized platform for students to search for hostels and for administrators to manage bookings, streamlining the entire process while saving time and effort for both parties.

2. Overall Description

2.1 Product Perspective

Context and Origin:

Hostel Konnect *is a new, self-contained product designed to address the challenges students face when finding hostel accommodations and the inefficiencies hostel administrators experience in managing their operations. This product is not a follow-on member of an existing product family, nor is it intended to replace any specific existing systems. Instead, it aims to fill a gap in the market by offering a digital solution to streamline the process of hostel search, application, and management.*

The application is designed to operate independently, providing essential features for both students and hostel administrators. It does not require integration with any pre-existing platforms, but can be extended in the future to work with third-party systems, such as payment gateways or external accommodation networks.

The major components of the system include:

- ***Student Module****: Allows students to search for hostels, view details, and submit applications.*
- ***Hostel Management Module****: Enables hostel administrators to manage room availability, bookings, and student data.*

In the future, integration with other platforms or systems can be considered to extend the functionality of ***Hostel Konnect****, such as incorporating payment processing or external accommodation listings.*

A simple diagram showing these components and their interfaces could be as follows:

2.2 Product Functions

Student Module:

- Search for hostels.
- View hostel details.
- Submit applications for hostels.
- Manage student profile.

Hostel Management Module:

- Register hostel details.

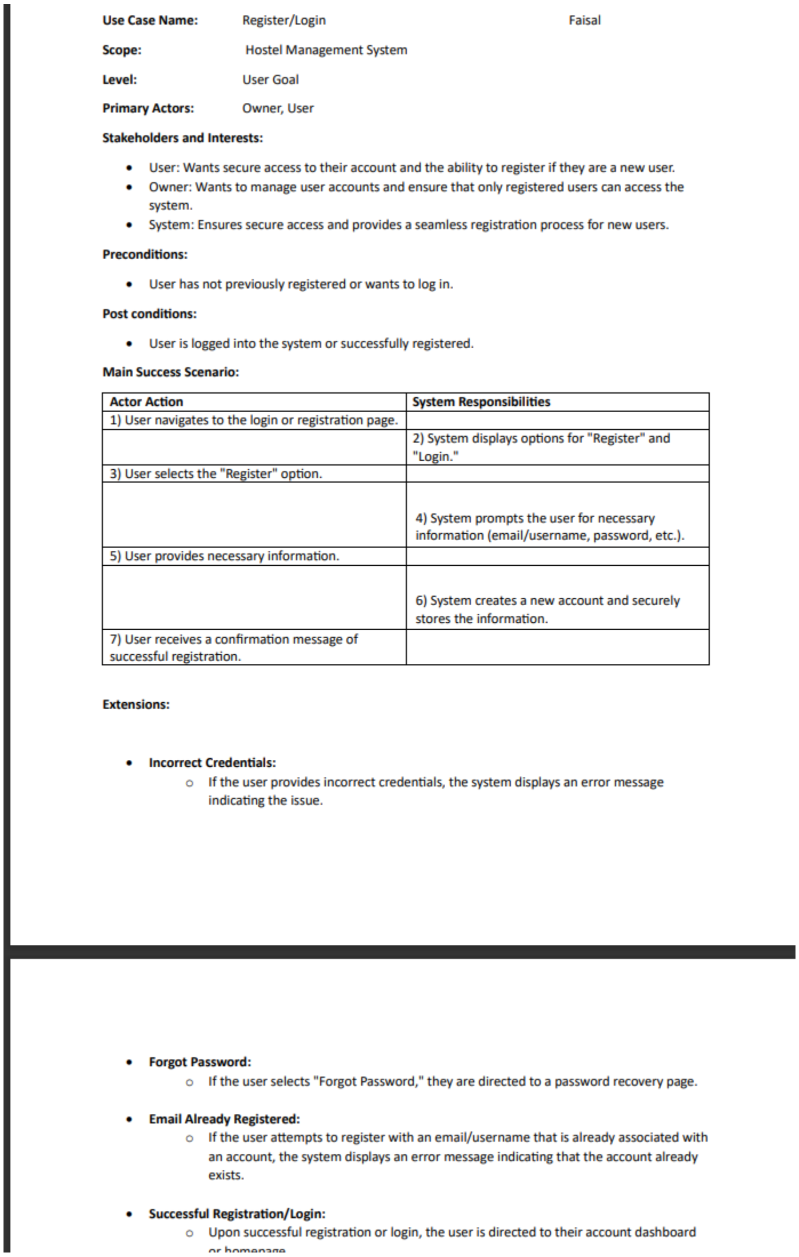
  - Update room availability.
  - Process student applications and bookings.

2.3 List of Use Cases

Here is a list of the use cases you provided:

1. Register/Login
2. Manage User Account
3. Manage Hostel Owner
4. List Hostel
5. Search Hostel
6. Check Room Availability
7. View Rooms
8. Book Room
9. Cancel Room Reservation
10. Check-in
11. Check-out
12. Review Hostel
13. Contact Support
14. Process Payment Transactions
15. Manage Payment

2.4 Extended Use Cases



Use Case Name: Register/Login Faisal

Scope: Hostel Management System

Level: User Goal

Primary Actors: Owner, User

Stakeholders and Interests:

- User: Wants secure access to their account and the ability to register if they are a new user.
- Owner: Wants to manage user accounts and ensure that only registered users can access the system.
- System: Ensures secure access and provides a seamless registration process for new users.

Preconditions:

- User has not previously registered or wants to log in.

Post conditions:

- User is logged into the system or successfully registered.

Main Success Scenario:

Actor Action	System Responsibilities
1) User navigates to the login or registration page.	
	2) System displays options for "Register" and "Login."
3) User selects the "Register" option.	
	4) System prompts the user for necessary information (email/username, password, etc.).
5) User provides necessary information.	
	6) System creates a new account and securely stores the information.
7) User receives a confirmation message of successful registration.	

Extensions:

- **Incorrect Credentials:**
  - If the user provides incorrect credentials, the system displays an error message indicating the issue.
- **Forgot Password:**
  - If the user selects "Forgot Password," they are directed to a password recovery page.
- **Email Already Registered:**
  - If the user attempts to register with an email/username that is already associated with an account, the system displays an error message indicating that the account already exists.
- **Successful Registration/Login:**
  - Upon successful registration or login, the user is directed to their account dashboard or homepage

Use Case Name: Manage User Account Ali

Scope: Hostel Management System

Level: User Goal

Primary Actor: admin

Stakeholders and Interests:

- Admin: Wants to manage user details.
- user: Wants to ensure their data is correct.

Preconditions

- Admin is logged in

Postconditions:

- User data is updated/verified.

Main Success Scenario

Actor Action	**System Responsibility**
1) Admin navigates to user management.	
	2) System displays users
3) Admin edits/approves owner details.	
	4) System validates and saves the changes.
5) user data is updated/verified.	
	6) System confirms the successful update.

Extensions

1) Invalid data: System prompts user to correct data.

Use Case Name:	Manage Hostel Owner	Ahmed
Scope:	Hostel Management System	
Level:	System Level	
Primary Actor:	Admin	

Stakeholders and Interests:

- Admin: Wants to manage owner details.
- Owners: Want to ensure their data is correct.

Preconditions:

- Admin is logged in.

Postconditions:

- Owner data is updated/verified.

Main Success Scenario:

Actor Action	**System Responsibility**
1) Admin navigates to owner management.	
	2) System displays the list of hostel owners.
3) Admin edits/approves owner details.	
	4) System validates and saves the changes.
5) Owner data is updated/verified.	
	6) System confirms the successful update.

Extensions:

1. Owner has incomplete info: Admin sends notification to owner.

Use Case Name: List Hostel Faisal

Scope: Hostel Management System

Level: User Goal

Primary Actor: Hostel Owner

Stakeholders and Interests:

- Owner: Wants to list hostel for booking.
- System: Ensures hostel details are valid.

Preconditions:

- Owner is logged in.

Postconditions:

- Hostel is listed and visible to users.

Main Success Scenario:

Actor Action	System Responsibility
1) Owner navigates to "List Hostel."	
	2) System displays the form for entering hostel details.
3) Owner fills in hostel details (name, location, price, etc.).	
	4) System verifies and saves the provided details.
5) Hostel is successfully listed.	
	6) System makes the hostel visible to users.

Extensions:

2a. Invalid data: System prompts owner to correct data.

Use Case Name: Search Hostel Ali

Scope: Hostel Management System

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants to browse available hostels.
- System: Presents accurate search results.

Preconditions

- Hostels are listed in the system.

Postconditions

- User browses hostels based on filters.

Main Success Scenario

Actor Action	System Responsibility
1) User navigates to hostel listings.	
	2) System displays all available hostels.
3) User applies filters (location, price, etc.).	
	4) System filters and displays matching hostels.
5) User views the filtered hostel results.	
	6) System updates results as per the filters.

Extensions

1. No results found: System displays a "No results" message.

Use Case Name: Check room Availability Ahmed

Scope: Hostel Management System

Level: User Goal

Primary Actor: owner

Stakeholders and Interests:

- Owner: updates if the rooms are available.
- System: Displays up-to-date availability info.

Preconditions:

- Hostel listings are available.

Postconditions:

- Availability status is shown to the user.

Main Success Scenario:

Actor Action	**System Responsibility**
1) Owner updates available rooms	
	2) System saves room availability
3) User views the availability status.	
	4) System displays availability status (available/not available).

Extensions:

1. No availability: System shows a "No availability" message.

Use Case Name:	View Rooms	Faisal
Scope:	Hostel Management System	
Level:	User Goal	
Primary Actor:	User	

Stakeholders and Interests:

- User: Wants to view detailed information about available rooms in a hostel.
- Hostel Owner: Ensures room details are accurately displayed for potential bookings.
- System: Ensures the user has access to up-to-date and accurate room information.

Preconditions:

- Rooms are available in the system, and the user has selected a hostel to browse.

Post conditions:

- User is presented with a detailed list of available rooms.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User selects a hostel from the listings.	
	2) System displays a list of available rooms with details (room type, price, amenities, availability).
3) User browses the available rooms.	
	4) User selects a specific room to view more detailed information
5) System shows detailed room information, including photos, reviews, and owner policies (check-in times, cancellation policies).	

Extensions:

1. No rooms available: System displays a "No rooms available" message and suggests other hostels.

Use Case Name:	Book Room	Ali
Scope:	Hostel Management System	
Level:	User Goal	
Primary Actor:	User	

Stakeholders and Interests:

- User: Wants to secure a room for a specific date.
- System: Manages room booking and ensures accurate bookings.

Preconditions:

- User is logged in and room availability is confirmed.

Postconditions:

- Room is booked successfully.

Main Success Scenario:

Action Actor	**System Responsibility**
1) User selects hostel and dates.	
	2) System checks room availability for the selected dates.
3) User confirms booking and proceeds to payment.	
	4) System processes payment and confirms the booking.
5) User receives booking confirmation.	
	6) System sends booking confirmation to the user.

Extensions:

1. Payment failed: System prompts user to retry.

Use Case Name:	Cancel Room reservation	Ahmed
Scope:	Hostel Management System	
Level:	User Goal	
Primary Actor:	User	

Stakeholders and Interests:

- User: Wants to cancel a previously made booking.
- System: Processes cancellation and refund (if applicable).

Preconditions:

- User has a confirmed booking.

Postconditions:

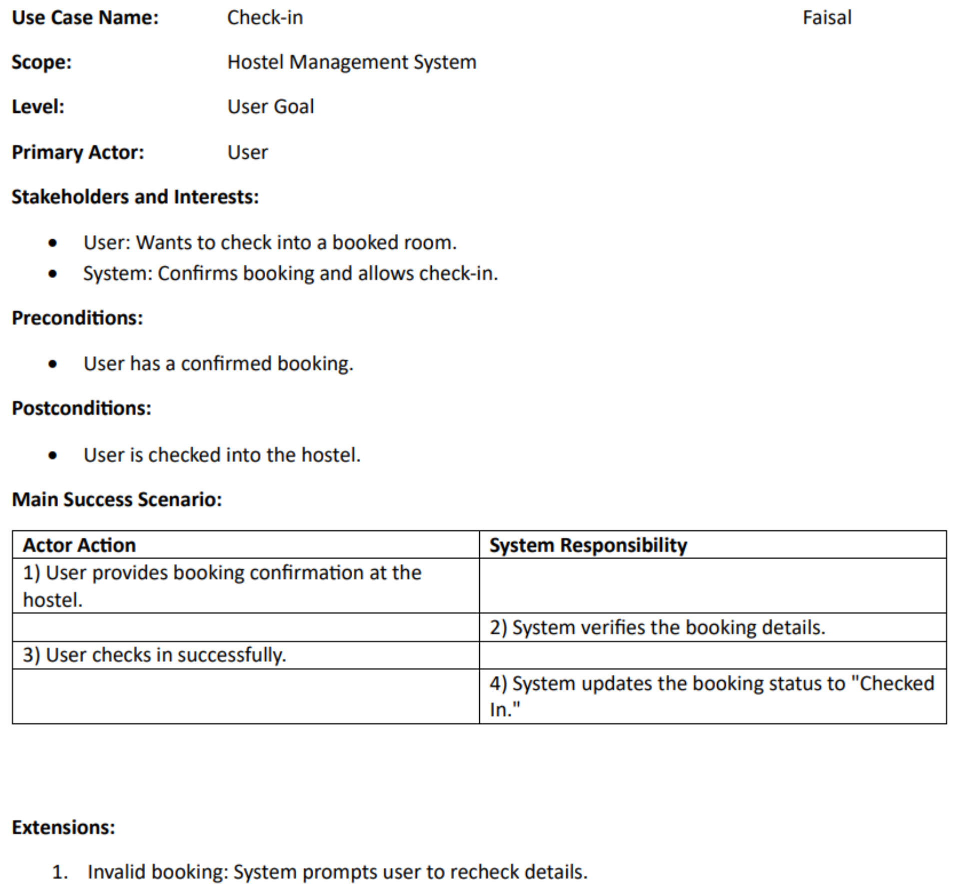
- Room is canceled, and user is refunded if applicable.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User navigates to "My Bookings."	
	2) System displays the user's bookings.
3) User selects a booking to cancel.	
	4) System processes the cancellation and checks refund eligibility.
5) User receives confirmation of the cancellation.	
	6) System sends confirmation and processes the refund if applicable.

Extensions:

1. Refund eligibility: System checks cancellation policy before processing.



Use Case Name: Check-in Faisal

Scope: Hostel Management System

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants to check into a booked room.
- System: Confirms booking and allows check-in.

Preconditions:

- User has a confirmed booking.

Postconditions:

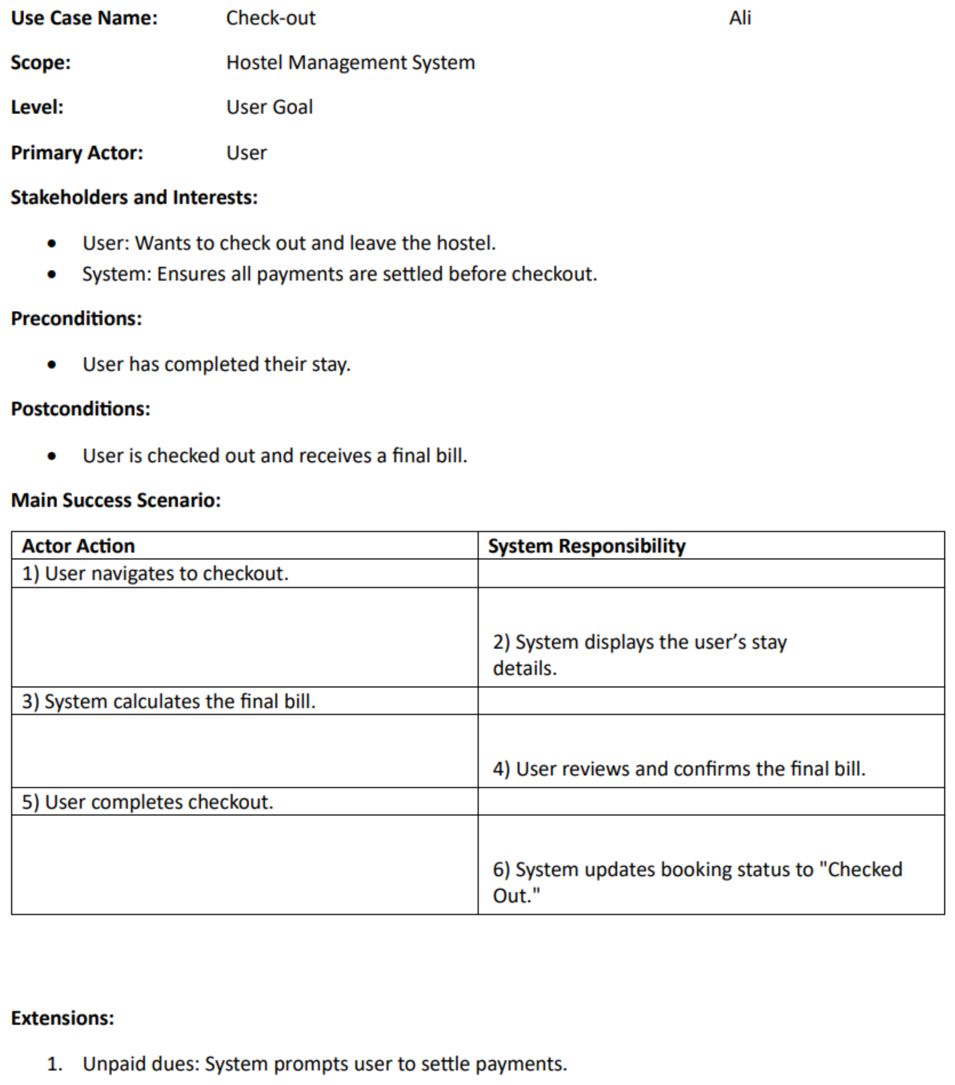
- User is checked into the hostel.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User provides booking confirmation at the hostel.	
	2) System verifies the booking details.
3) User checks in successfully.	
	4) System updates the booking status to "Checked In."

Extensions:

1. Invalid booking: System prompts user to recheck details.



Use Case Name: Check-out Ali

Scope: Hostel Management System

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants to check out and leave the hostel.
- System: Ensures all payments are settled before checkout.

Preconditions:

- User has completed their stay.

Postconditions:

- User is checked out and receives a final bill.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User navigates to checkout.	
	2) System displays the user's stay details.
3) System calculates the final bill.	
	4) User reviews and confirms the final bill.
5) User completes checkout.	
	6) System updates booking status to "Checked Out."

Extensions:

1. Unpaid dues: System prompts user to settle payments.

Use Case Name: Review Hostel Ahmed

Scope: Hostel Management System

Level: User Goal

Primary Actor: User

Stakeholders and Interests:

- User: Wants to leave feedback for a hostel.
- System: Collects reviews to help other users make informed decisions.

Preconditions:

- User has stayed at the hostel.

Postconditions:

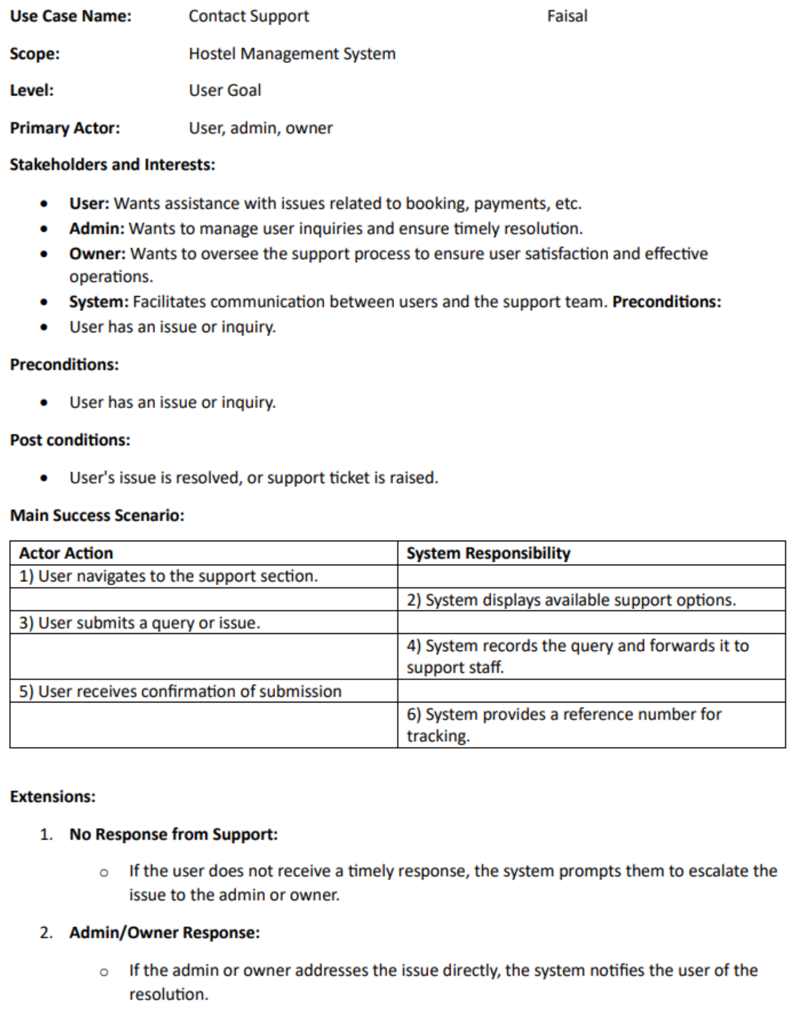
- Review is posted successfully.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User navigates to the review section.	
	2) System displays previous reviews and submission form.
3) User writes and submits a review.	
	4) System validates and posts the review.
5) User receives confirmation of successful submission.	
	6) System updates the review section with the new review.

Extensions

1. Inappropriate review: System filters inappropriate content.



Use Case Name:	Contact Support	Faisal
Scope:	Hostel Management System	
Level:	User Goal	
Primary Actor:	User, admin, owner	

Stakeholders and Interests:

- **User:** Wants assistance with issues related to booking, payments, etc.
- **Admin:** Wants to manage user inquiries and ensure timely resolution.
- **Owner:** Wants to oversee the support process to ensure user satisfaction and effective operations.
- **System:** Facilitates communication between users and the support team. **Preconditions:**
- User has an issue or inquiry.

Preconditions:

- User has an issue or inquiry.

Post conditions:

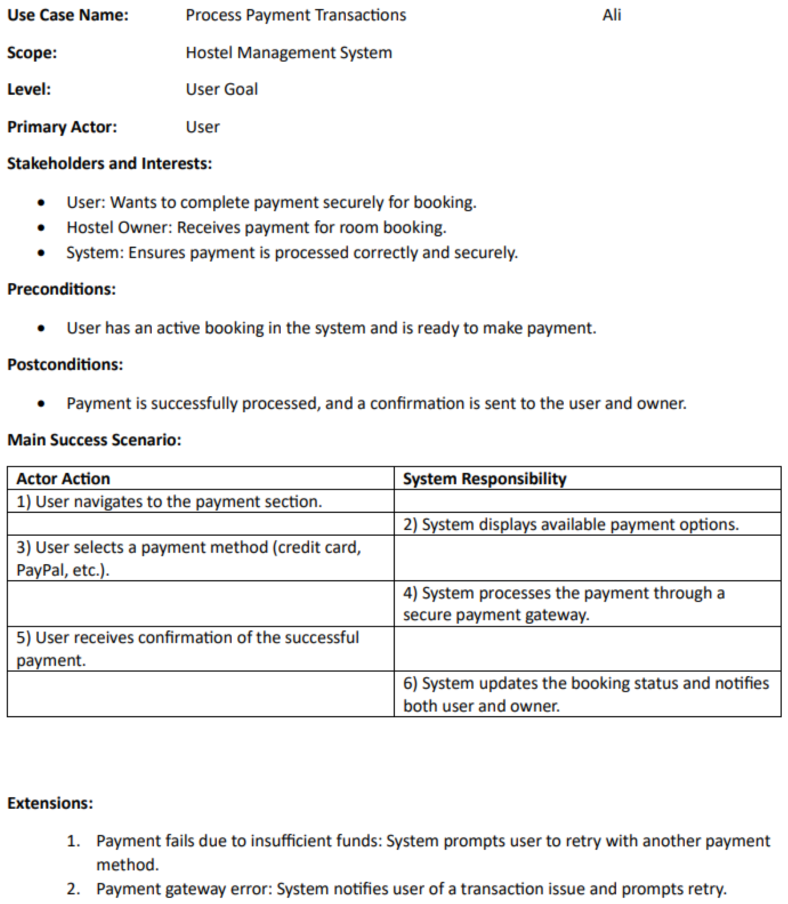
- User's issue is resolved, or support ticket is raised.

Main Success Scenario:

Actor Action	**System Responsibility**
1) User navigates to the support section.	
	2) System displays available support options.
3) User submits a query or issue.	
	4) System records the query and forwards it to support staff.
5) User receives confirmation of submission	
	6) System provides a reference number for tracking.

Extensions:

1. **No Response from Support:**
   - If the user does not receive a timely response, the system prompts them to escalate the issue to the admin or owner.
2. **Admin/Owner Response:**
   - If the admin or owner addresses the issue directly, the system notifies the user of the resolution.



Use Case Name:	Process Payment Transactions	Ali
Scope:	Hostel Management System	
Level:	User Goal	
Primary Actor:	User	

Stakeholders and Interests:

- User: Wants to complete payment securely for booking.
- Hostel Owner: Receives payment for room booking.
- System: Ensures payment is processed correctly and securely.

Preconditions:

- User has an active booking in the system and is ready to make payment.

Postconditions:

- Payment is successfully processed, and a confirmation is sent to the user and owner.

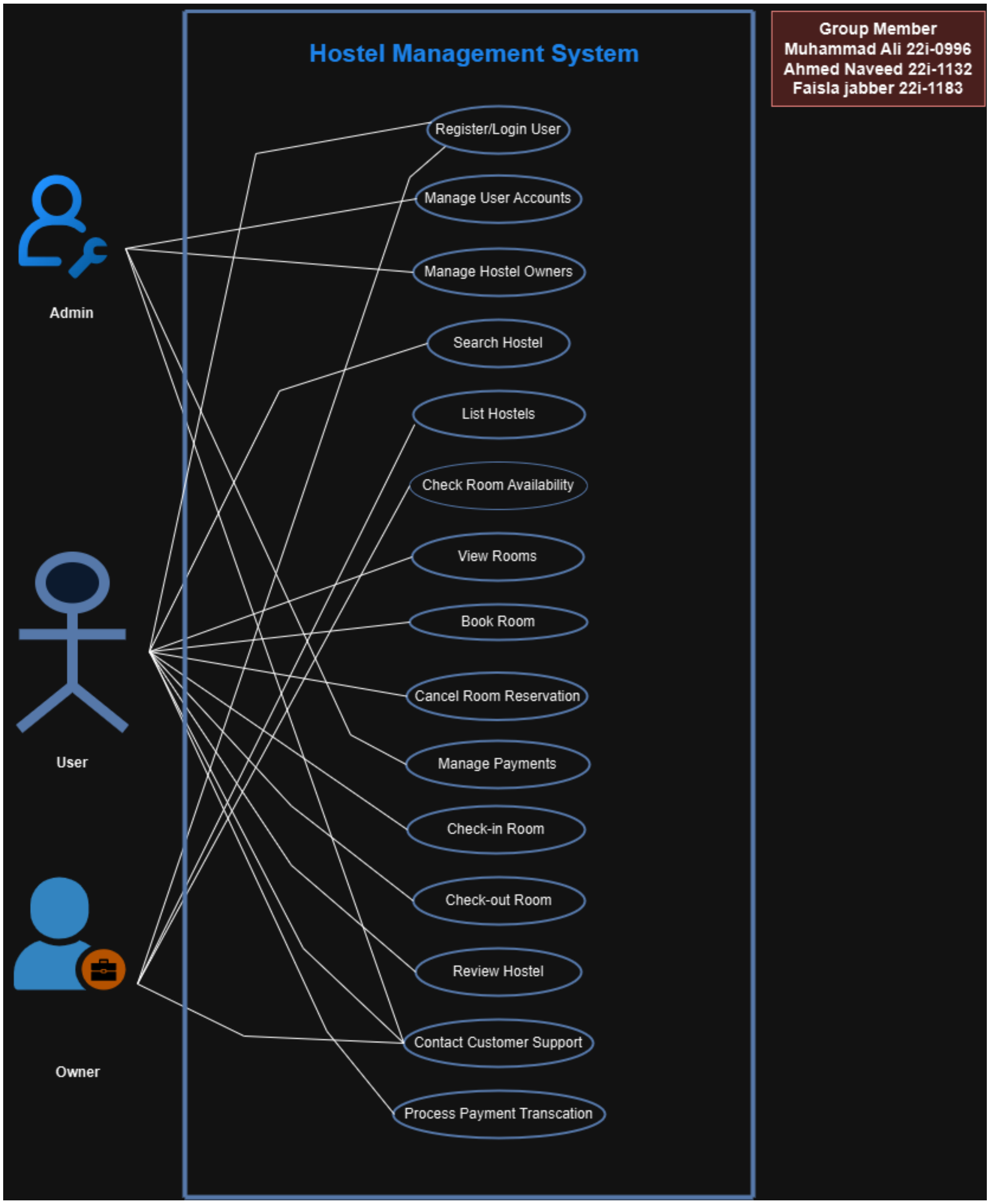
Main Success Scenario:

Actor Action	**System Responsibility**
1) User navigates to the payment section.	
	2) System displays available payment options.
3) User selects a payment method (credit card, PayPal, etc.).	
	4) System processes the payment through a secure payment gateway.
5) User receives confirmation of the successful payment.	
	6) System updates the booking status and notifies both user and owner.

Extensions:

1. Payment fails due to insufficient funds: System prompts user to retry with another payment method.
2. Payment gateway error: System notifies user of a transaction issue and prompts retry.

Use Case Diagram



3. Other Nonfunctional Requirements

3.1 Performance Requirements

Response Time

- Login/Registration Process: The system should authenticate users within 3 seconds for login or registration.
    - *Rationale*: Users expect quick access to their accounts. Delayed responses could result in poor user experience.

Hostel Search

- Search Results Display: The system should return search results for available hostels within 5 seconds of applying filters.
    - *Rationale*: Users need fast access to relevant information when searching for hostels, particularly when applying multiple filters.

Room Booking Process

- Booking Confirmation: After booking a room, the system should confirm the reservation and process payment within 10 seconds.
    - *Rationale*: Users expect instant confirmation and a smooth, uninterrupted booking process.

Scalability

- Concurrent Users: The system should handle at least 1,000 concurrent users without significant performance degradation.
    - *Rationale*: The platform is expected to grow and must be able to support large numbers of users at once, especially during peak booking times.

Data Storage and Retrieval

- Database Access Time: Queries to retrieve hostel or user data should be completed within 2 seconds under normal load.
    - *Rationale*: Quick data access ensures a responsive user interface and smooth navigation, reducing the likelihood of users abandoning the system.

Payment Processing

- Transaction Processing: The payment transaction should be completed in under 5 seconds.
    - *Rationale*: Payment processing is a critical operation, and delays can result in frustration and abandoned bookings.

Room Availability Updates

- Real-time Updates: Room availability information should be updated in real-time as bookings are made or canceled.
    - *Rationale*: Accurate availability data is necessary to prevent overbooking and ensure that users are only shown available rooms.

3.2 Safety Requirements

Data Security

- ***Protect User Data****: Encrypt all sensitive information like passwords and payment details.*
    - *Why: To keep user data safe from theft or unauthorized access.*

    - ***Safeguards**: Use strong encryption and store passwords securely.*
    - ***Prevention**: Prevent unauthorized access to accounts and data.*

Payment Security

- ***Secure Payments**: Use a secure payment gateway for processing transactions.*
    - *Why: To protect payment information and prevent fraud.*
    - ***Safeguards**: Use trusted, PCI-compliant payment processors.*
    - ***Prevention**: Prevent fraudulent transactions.*

Account Protection

- ***Login Safety**: Require multi-factor authentication (MFA) or CAPTCHA during login.*
    - *Why: To stop hackers from gaining access to accounts.*
    - ***Safeguards**: Enforce strong passwords and limit login attempts.*
    - ***Prevention**: Prevent unauthorized access to accounts.*

System Reliability

- ***Handle Failures**: Ensure the system can recover quickly from errors or crashes.*
    - *Why: To avoid data loss and system downtime.*
    - ***Safeguards**: Regular backups and real-time monitoring.*
    - ***Prevention**: Prevent data loss during failures.*

Privacy Protection

- ***Follow Data Privacy Laws**: Comply with regulations like GDPR to protect user privacy.*
    - *Why: To ensure users' personal information is handled legally.*
    - ***Safeguards**: Collect only necessary information and allow users to control their data.*
    - ***Prevention**: Prevent misuse or unnecessary collection of personal data.*

Content Safety

- ***Prevent Harmful Content**: Monitor and filter harmful or inappropriate user-submitted content.*
    - *Why: To maintain a safe and respectful environment.*
    - ***Safeguards**: Use content moderation tools to detect inappropriate posts.*
    - ***Prevention**: Prevent harmful or abusive content from being shared.*

3.3 Security Requirements

1. ***User Login**:*
    - *Users must log in with a secure email/username and password.*
    - *Passwords must be stored securely (hashed).*
    - *Two-factor authentication (2FA) must be enabled for admin users.*
    - *Users should automatically log out after a period of inactivity (e.g., 30 minutes).*

2. ***Data Privacy**:*
    - *The system must follow privacy laws like GDPR and CCPA.*
    - *Personal data must be encrypted (protected) both during transfer and storage.*
    - *Users must be able to view, update, or delete their personal data.*
    - *Clear privacy policies must be provided to users.*

3. ***Payment Security**:*
    - *Payments must be processed securely using PCI DSS-compliant methods.*

- *Payment details should never be stored; only secure tokens are kept.*

4. ***Access Control**:*
   - *Use role-based access, so admins have more privileges than regular users.*
   - *Only authorized users can access sensitive information.*

5. ***Monitoring and Auditing**:*
   - *The system must track and log important actions (e.g., logins, data changes, payments).*
   - *Admins should be alerted to unusual activity.*

6. ***Compliance**:*
   - *Follow security standards (e.g., ISO/IEC 27001) and local data protection laws (e.g., GDPR).*
   - *Ensure the system meets industry-specific security standards.*

7. ***Security Updates**:*
   - *Regular security checks (e.g., audits, tests) must be performed.*
   - *Apply security patches and updates regularly.*
   - *All data transfer must be secured using encryption (SSL/TLS).*

8. ***Incident Response**:*
   - *In case of a data breach, users must be notified within the required time (e.g., 72 hours for GDPR).*
   - *Users can report security issues through a dedicated support system.*

3.4 Software Quality Attributes

Key quality characteristics for the product:

1. ***Usability***
   - *The system must be easy to navigate, with tasks like booking a room completed in under 5 minutes.*
   - *A help section should be available within 2 clicks.*

2. ***Reliability***
   - *The system must have an uptime of 99.5% or higher and recover from errors within 30 seconds without losing data.*

3. ***Performance***
   - *The system must respond to user actions (e.g., loading pages, bookings) within 3 seconds under normal load.*

4. ***Maintainability***
   - *Code should be modular, well-documented, and easy to update without significant downtime.*

5. ***Adaptability***
   - *The system should allow easy addition of new features or changes (e.g., new payment methods) with minimal changes to core functionality.*

3.5 Business Rules

Operating principles for the product:

1. ***Role-Based Access Control***
   - *Only admins can manage user accounts, including updating user or owner details. Regular users and owners cannot access admin functionalities.*

2. ***User Authentication***
   - *All users must be authenticated before accessing any sensitive information, such as bookings or personal data.*

3. ***Owner Responsibilities***
   - *Only hostel owners can list their hostels and update room availability. They cannot modify other owners' hostel data.*

4. ***Booking Permissions***
   - *Users can only book rooms if they have a valid account and are logged in. Guests must register before making a booking.*

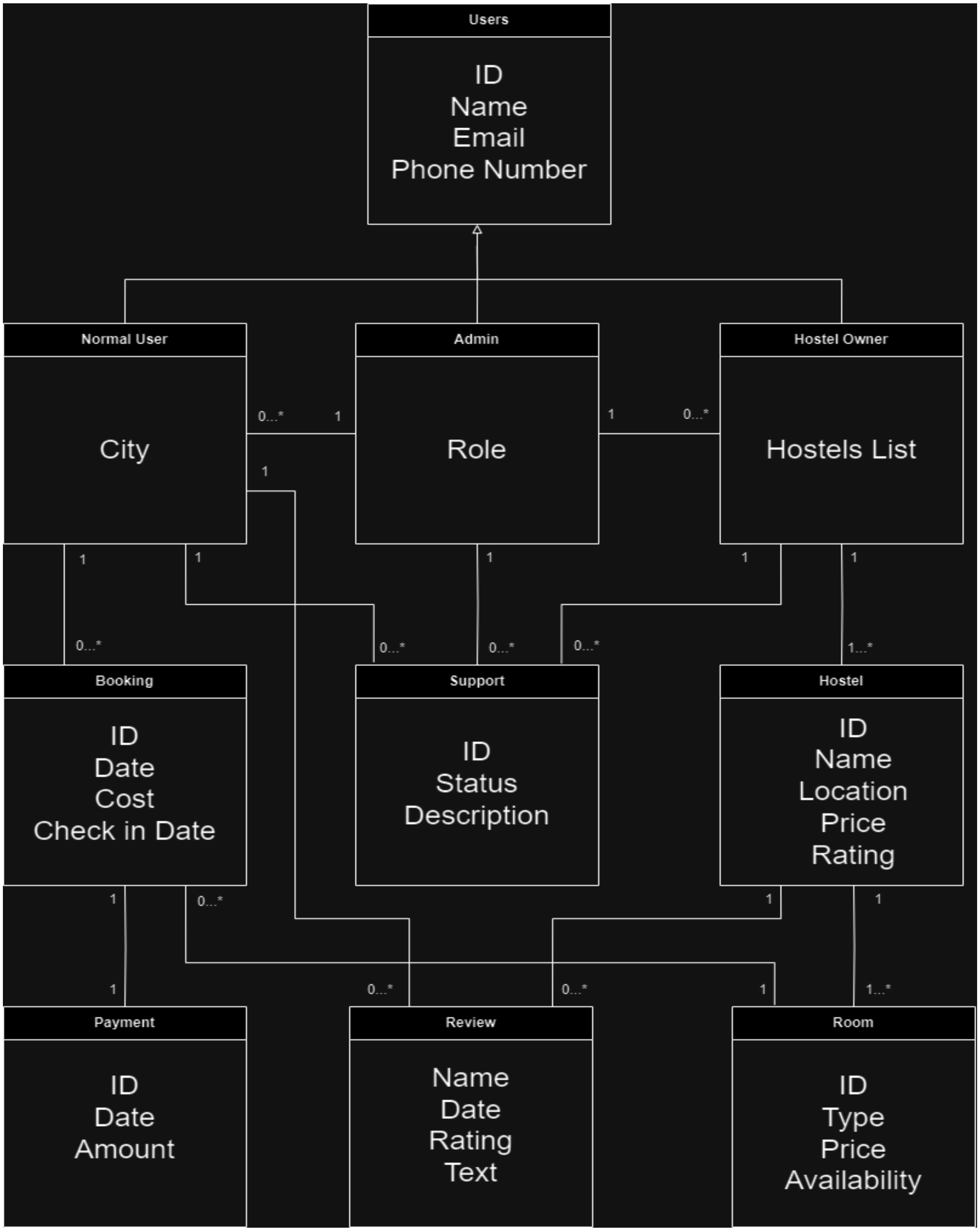
5. ***Payment Security***
   - *Only users with successful payments can complete a booking. Payment information is processed securely through a trusted gateway, and no payment details are stored in the system.*

3.6 Operating Environment

1. ***Hardware Platform:***
   - ***Server:*** *Web server with:*
     - *Multi-core processor (Intel Xeon or equivalent)*
     - *8 GB RAM (minimum)*
     - *100 GB SSD storage (minimum)*
     - *High-speed internet (100 Mbps or faster)*

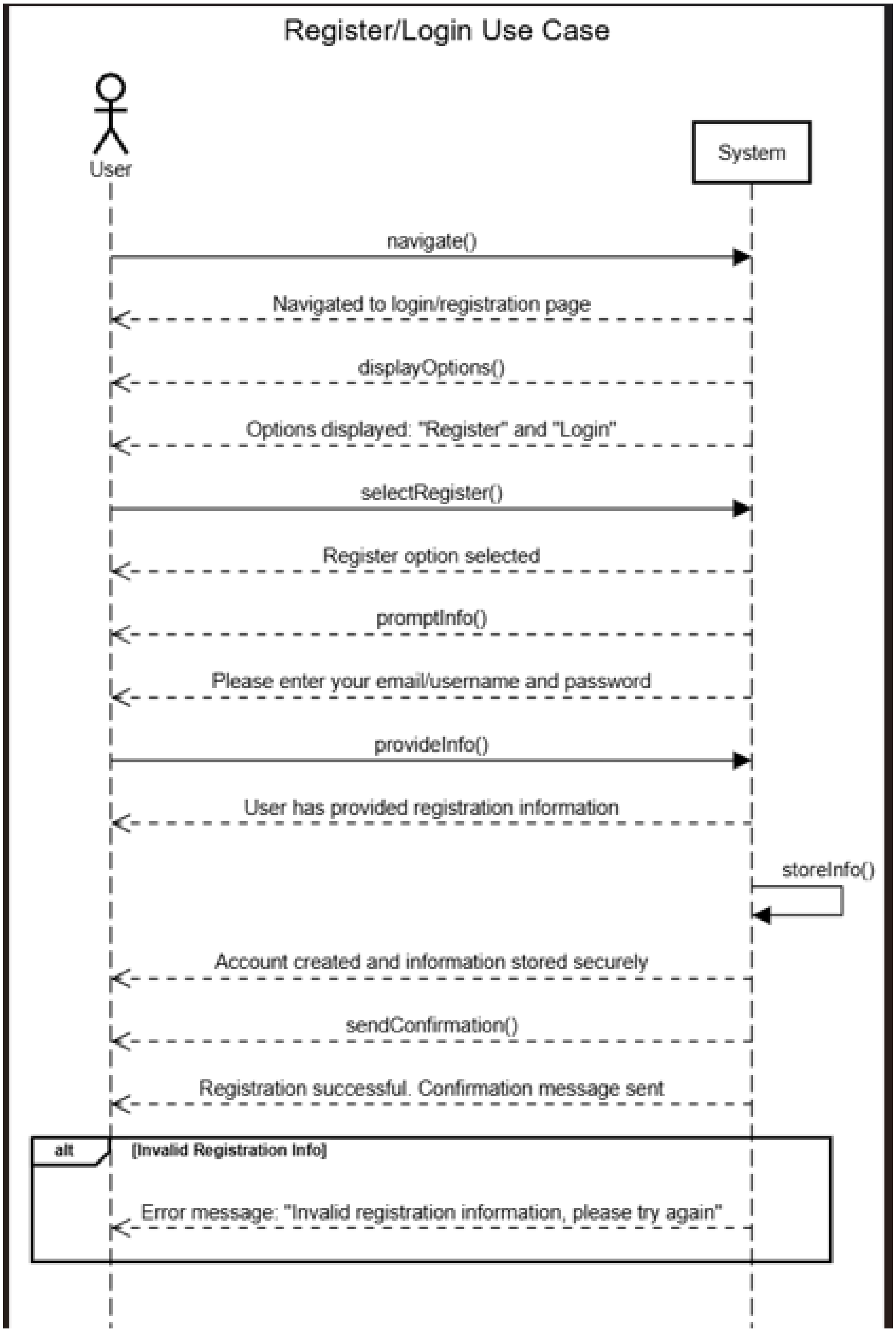
2. ***Operating System:***
   - ***Server OS:*** *Linux (Ubuntu 20.04 or later, CentOS 7 or later)*
   - ***Client OS:*** *Any modern OS, including:*
     - *Windows 10/11*
     - *macOS 10.13 or later*
     - *Linux (Ubuntu, Fedora, etc.)*
   - ***Browsers:*** *Google Chrome, Microsoft Edge (latest versions)*

4. Domain Model

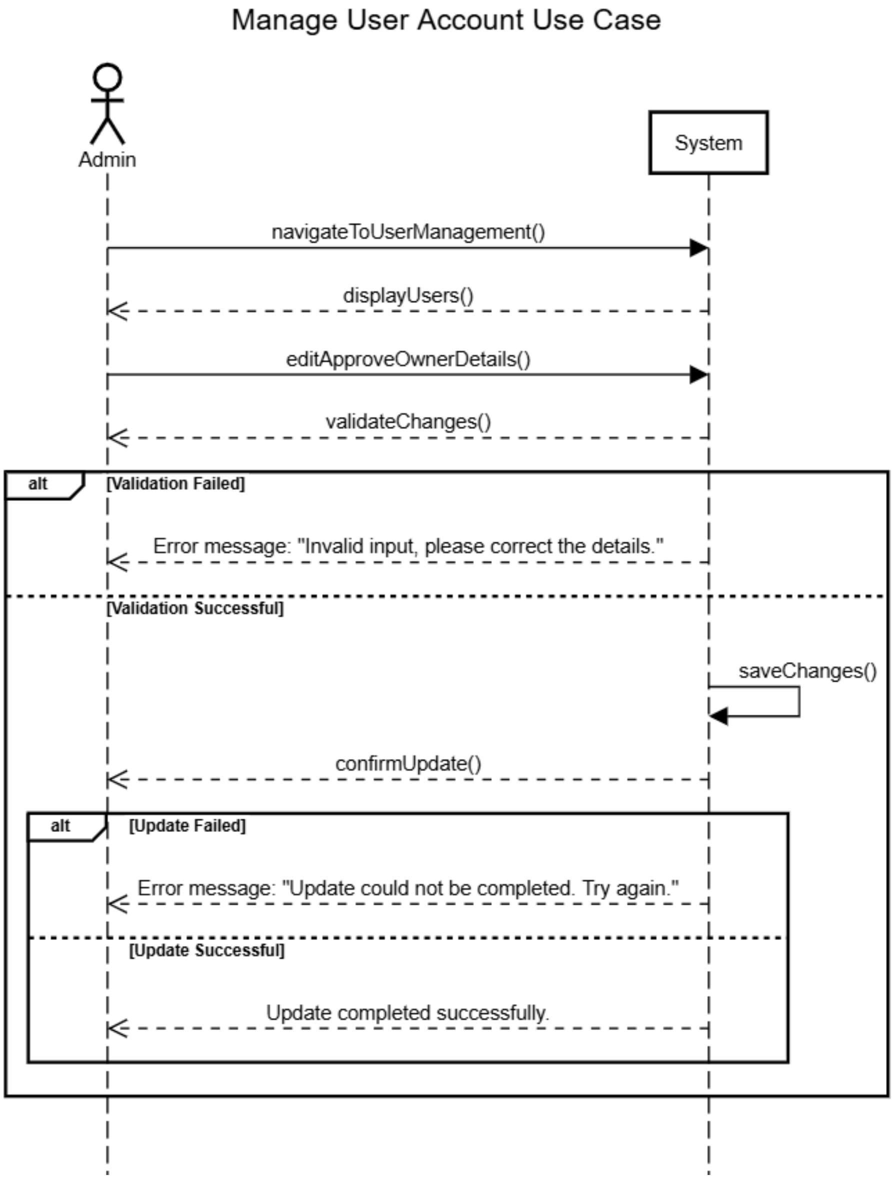


System Sequence Diagram

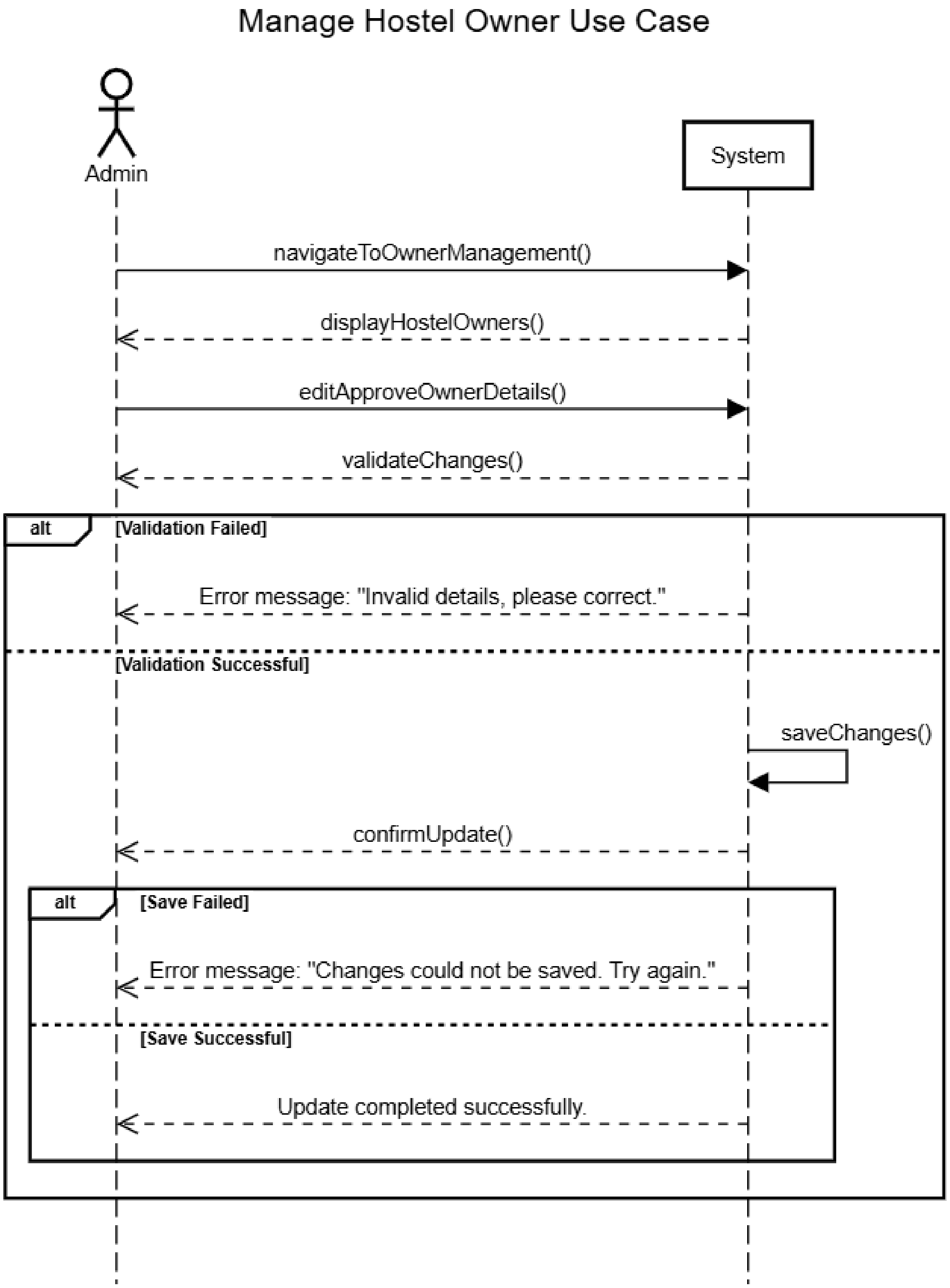
4.1


Register/Login Use Case
User
System
navigate()
Navigated to login/registration page
displayOptions()
Options displayed: "Register" and "Login"
selectRegister()
Register option selected
promptInfo()
Please enter your email/username and password
provideInfo()
User has provided registration information
storeInfo()
Account created and information stored securely
sendConfirmation()
Registration successful. Confirmation message sent
alt
[Invalid Registration Info]
Error message: "Invalid registration information, please try again"

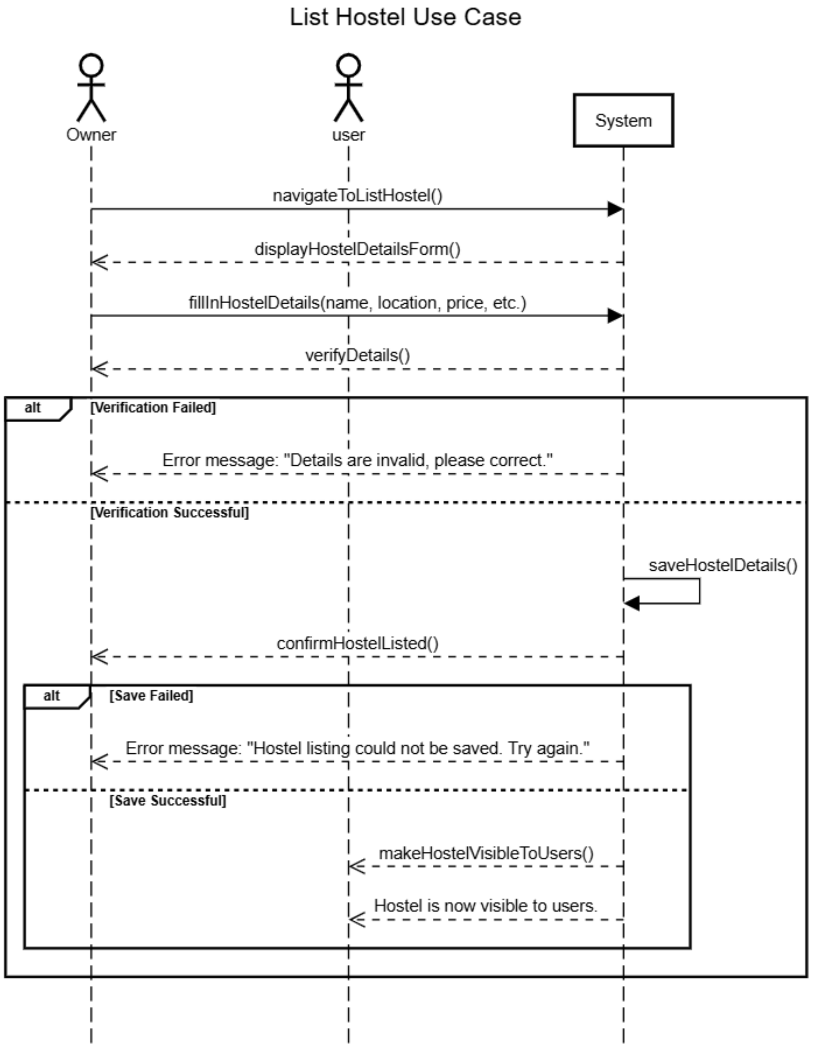
4.2


Manage User Account Use Case
Admin
System
navigateToUserManagement()
displayUsers()
editApproveOwnerDetails()
validateChanges()
alt
[Validation Failed]
Error message: "Invalid input, please correct the details."
[Validation Successful]
saveChanges()
confirmUpdate()
alt
[Update Failed]
Error message: "Update could not be completed. Try again."
[Update Successful]
Update completed successfully.

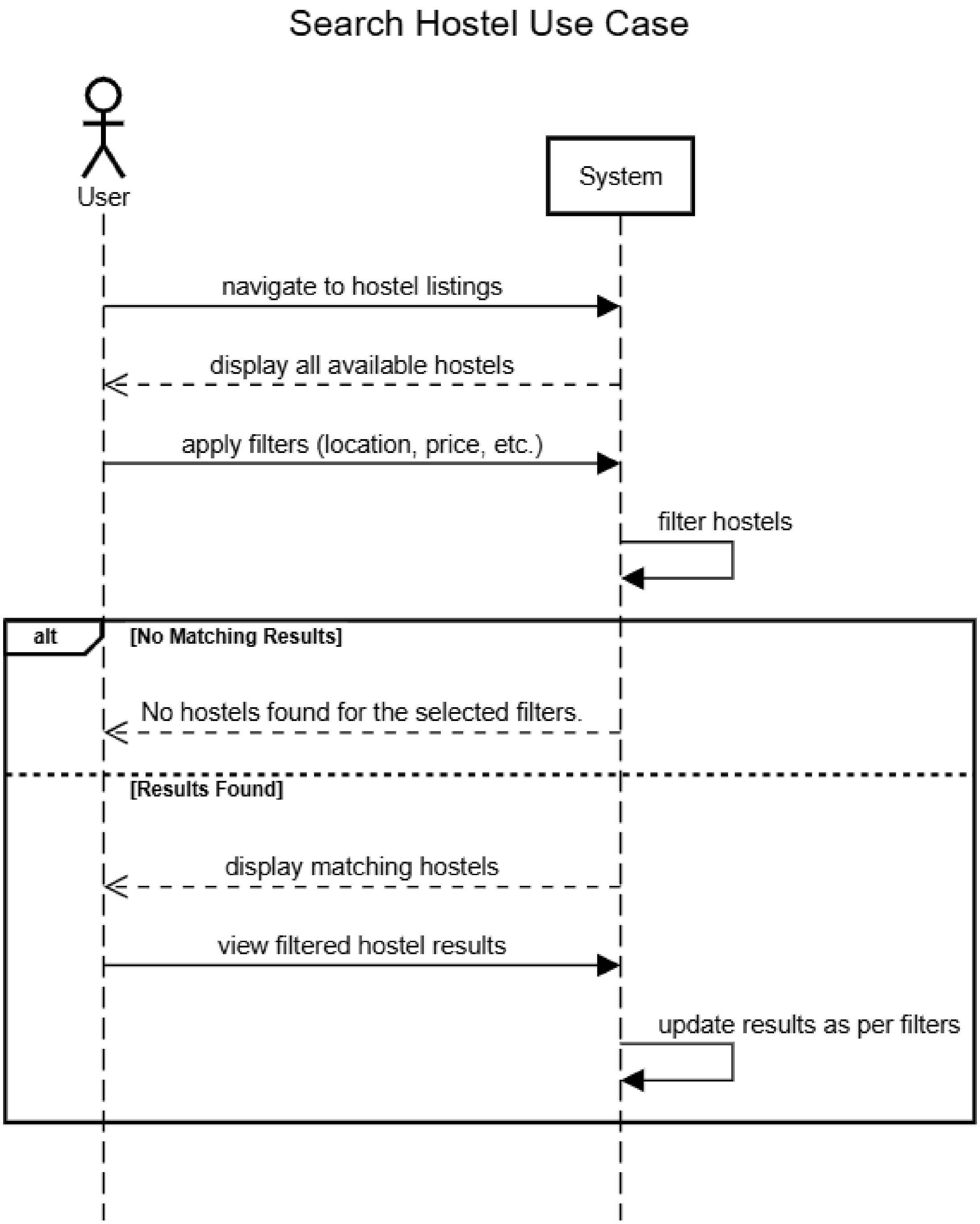
4.3


Manage Hostel Owner Use Case
Admin
System
navigateToOwnerManagement()
displayHostelOwners()
editApproveOwnerDetails()
validateChanges()
alt
[Validation Failed]
Error message: "Invalid details, please correct."
[Validation Successful]
saveChanges()
confirmUpdate()
alt
[Save Failed]
Error message: "Changes could not be saved. Try again."
[Save Successful]
Update completed successfully.

4.4

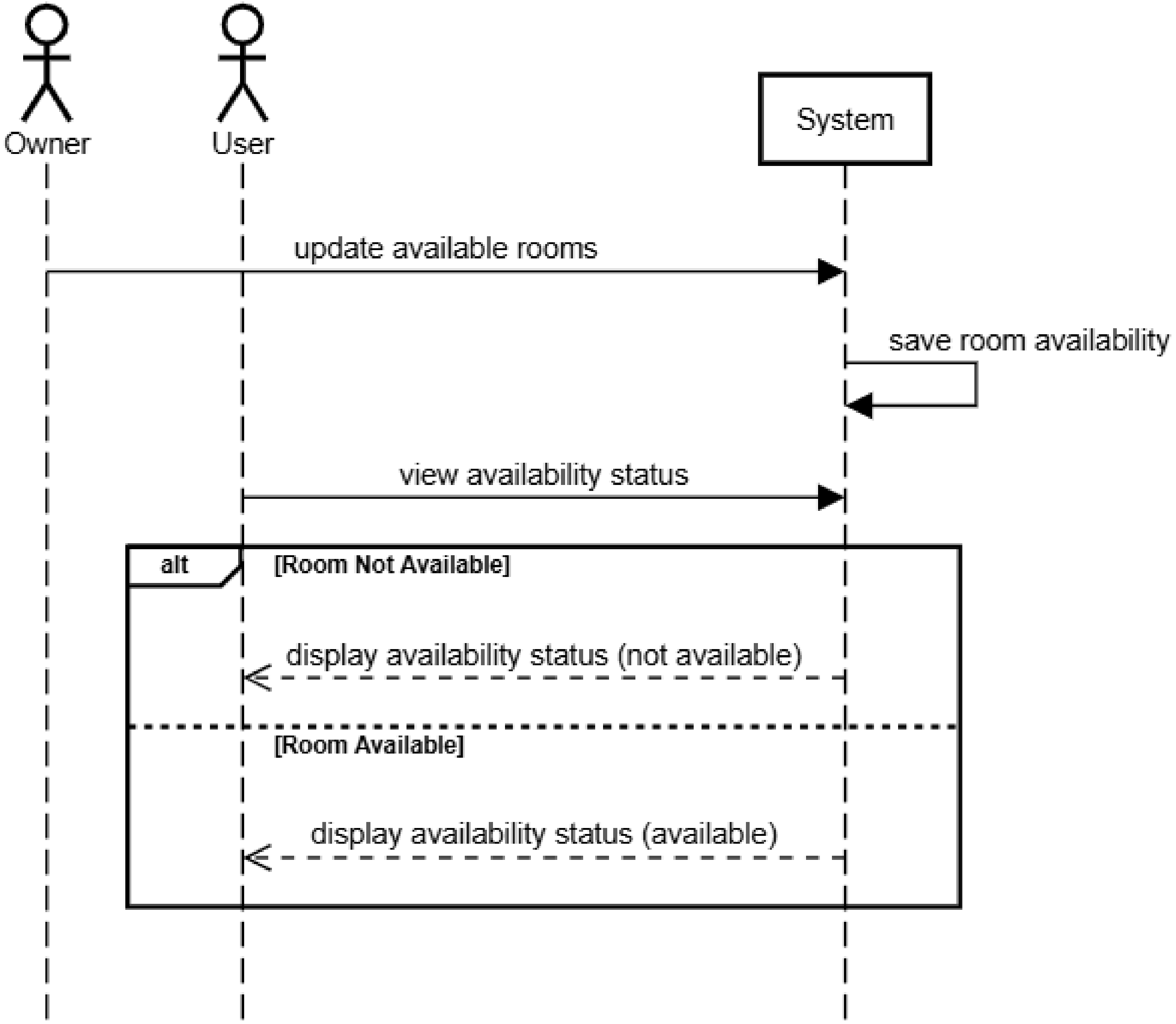

List Hostel Use Case
Owner
user
System
navigateToListHostel()
displayHostelDetailsForm()
fillInHostelDetails(name, location, price, etc.)
verifyDetails()
alt
[Verification Failed]
Error message: "Details are invalid, please correct."
[Verification Successful]
saveHostelDetails()
confirmHostelListed()
alt
[Save Failed]
Error message: "Hostel listing could not be saved. Try again."
[Save Successful]
makeHostelVisibleToUsers()
Hostel is now visible to users.

4.5

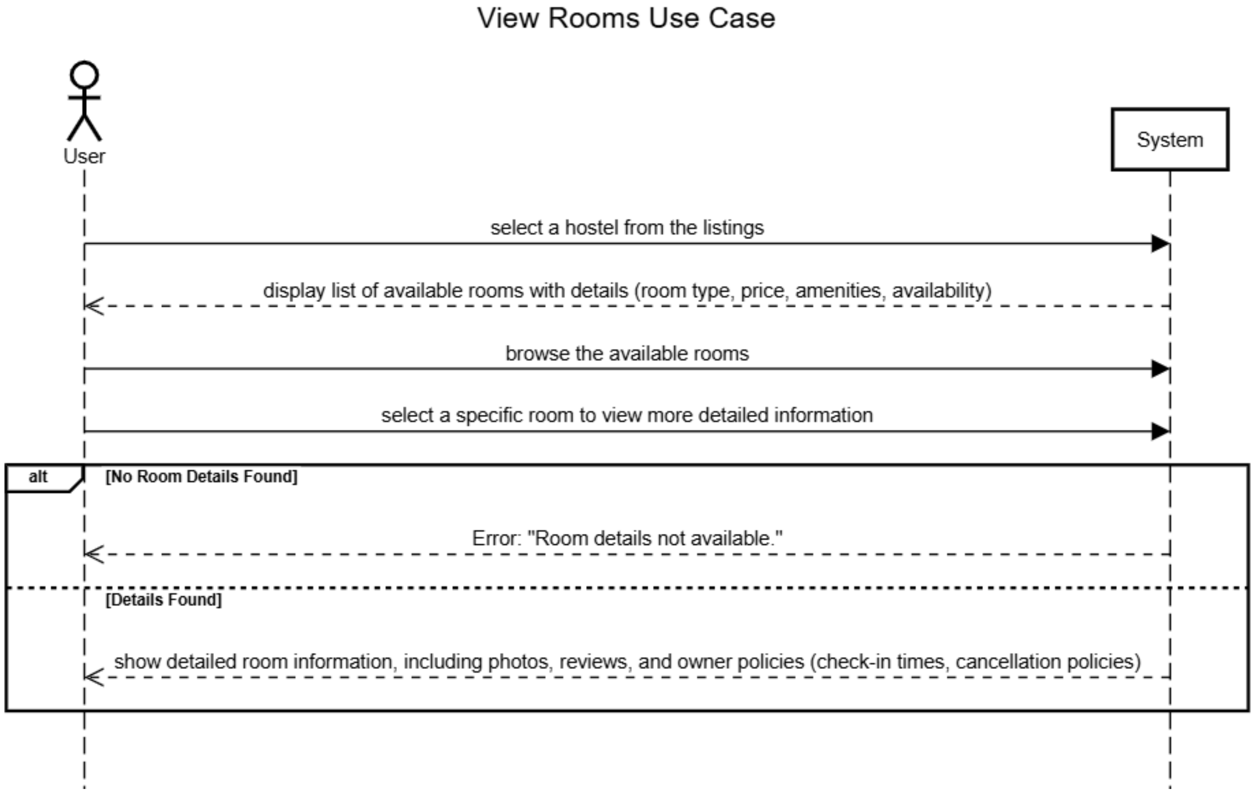

Search Hostel Use Case
User
System
navigate to hostel listings
display all available hostels
apply filters (location, price, etc.)
filter hostels
alt
[No Matching Results]
No hostels found for the selected filters.
[Results Found]
display matching hostels
view filtered hostel results
update results as per filters

4.6

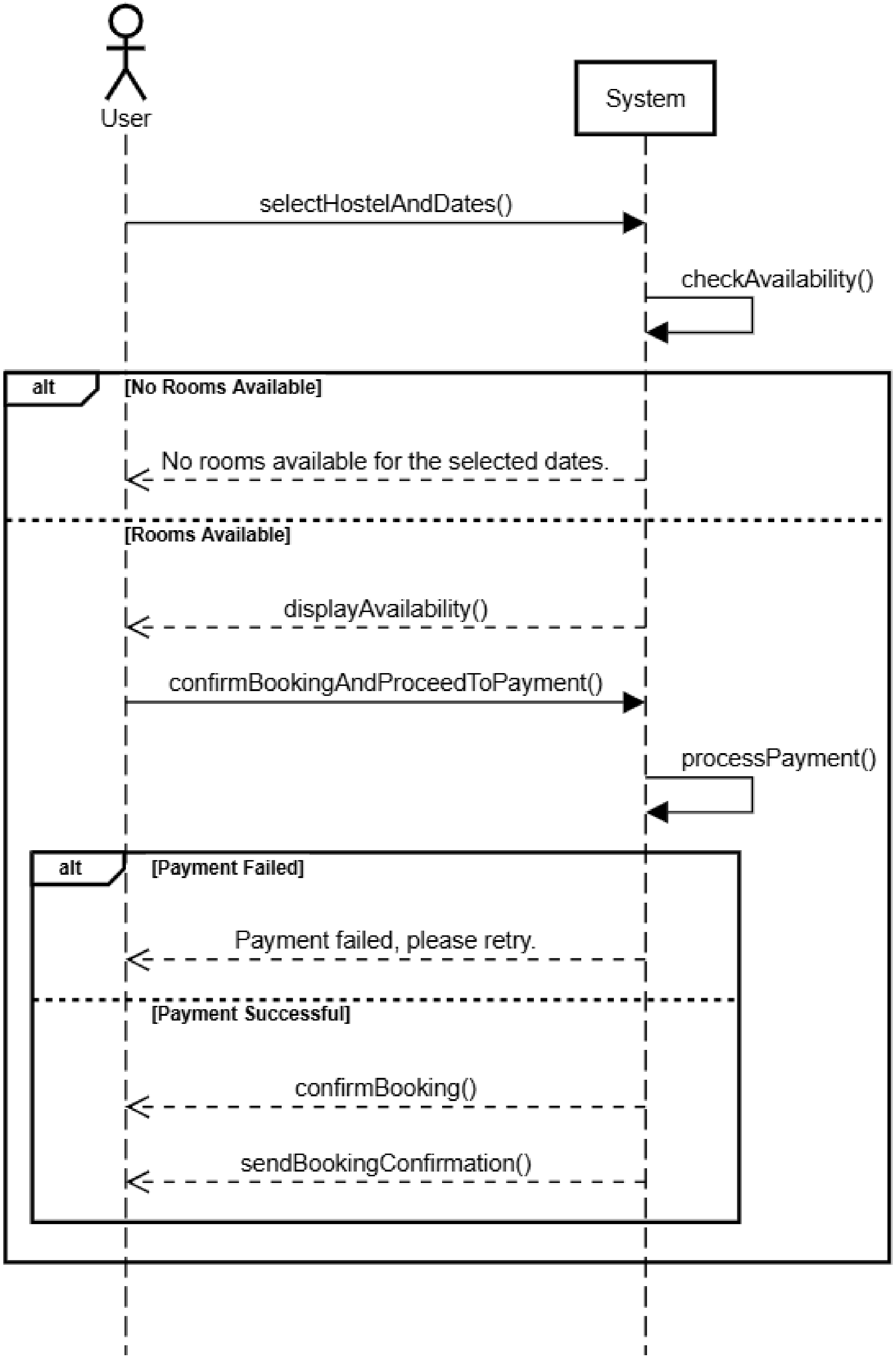
Check Room Availability Use Case



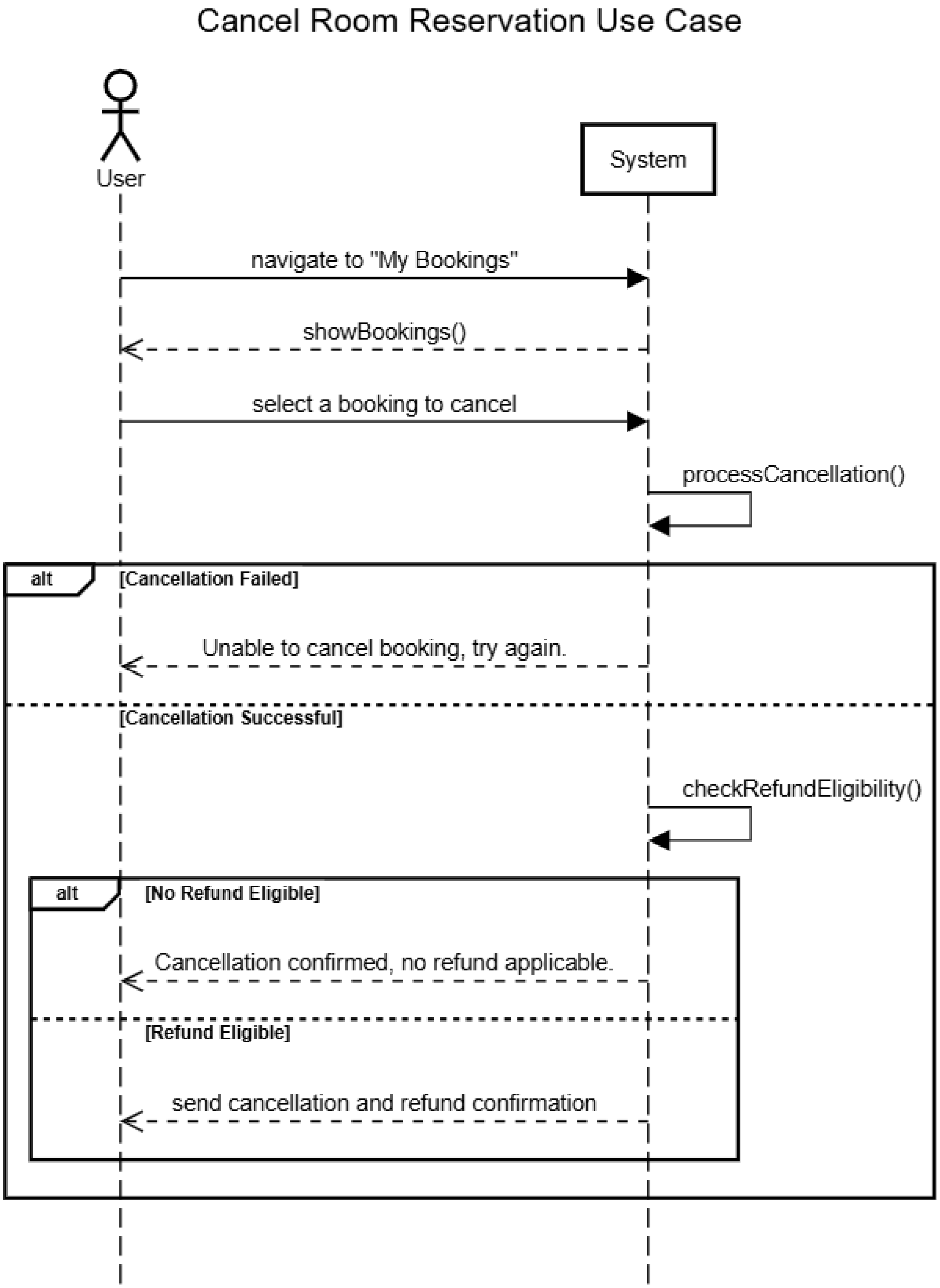
4.7


View Rooms Use Case
User
System
select a hostel from the listings
display list of available rooms with details (room type, price, amenities, availability)
browse the available rooms
select a specific room to view more detailed information
alt
[No Room Details Found]
Error: "Room details not available."
[Details Found]
show detailed room information, including photos, reviews, and owner policies (check-in times, cancellation policies)

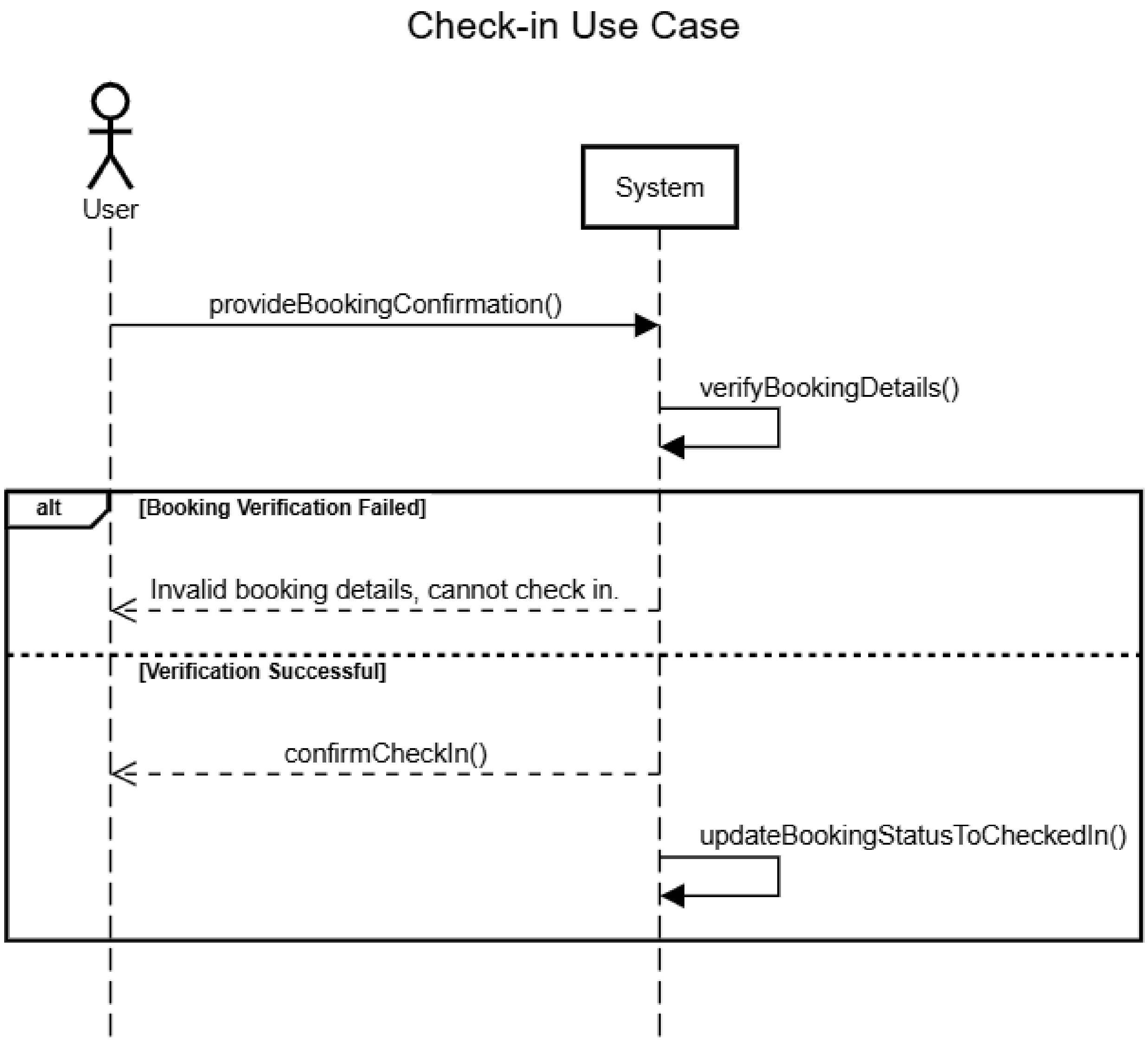
4.8 Book Room Use Case



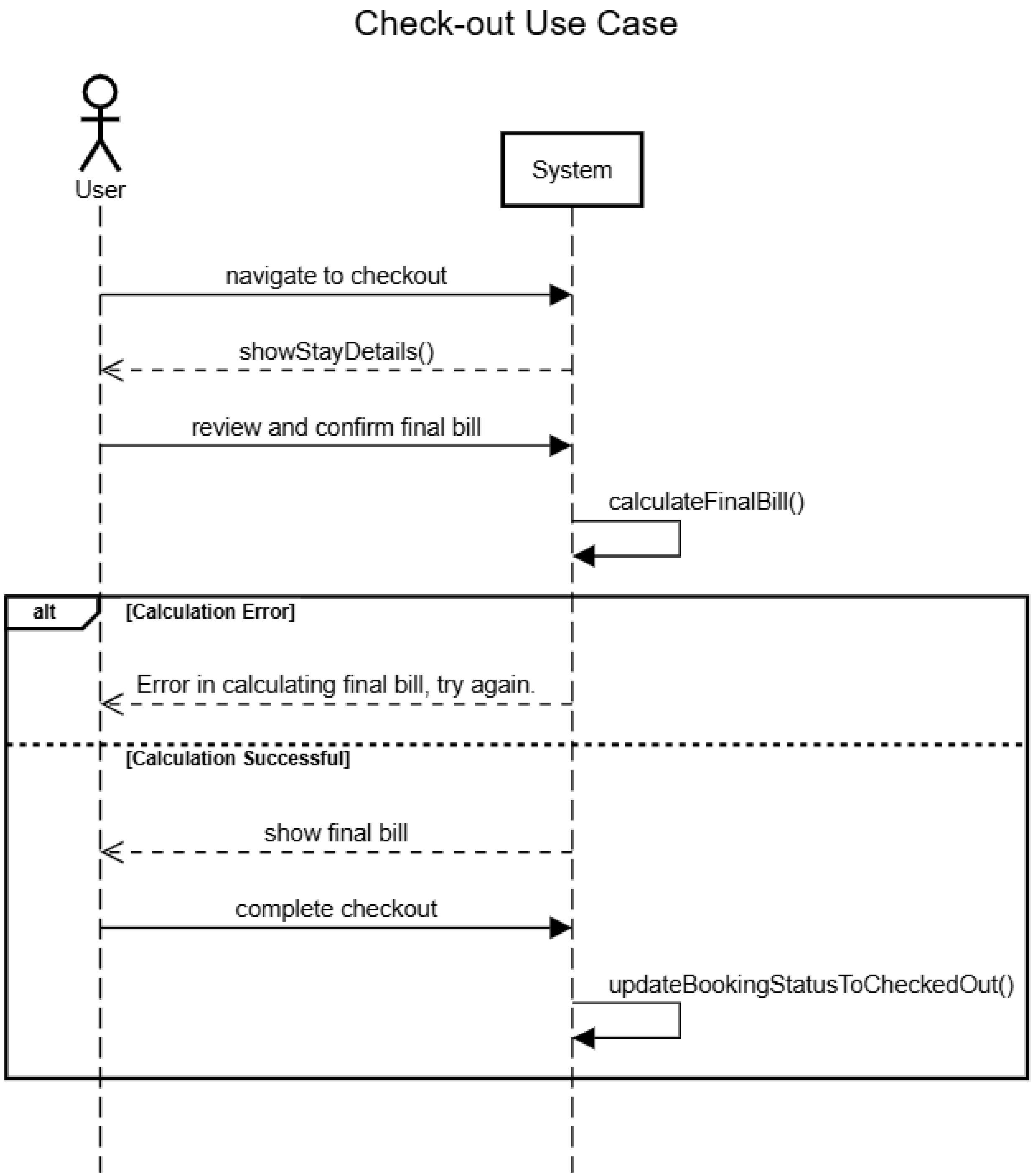
4.9


Cancel Room Reservation Use Case
User
System
navigate to "My Bookings"
showBookings()
select a booking to cancel
processCancellation()
alt
[Cancellation Failed]
Unable to cancel booking, try again.
[Cancellation Successful]
checkRefundEligibility()
alt
[No Refund Eligible]
Cancellation confirmed, no refund applicable.
[Refund Eligible]
send cancellation and refund confirmation

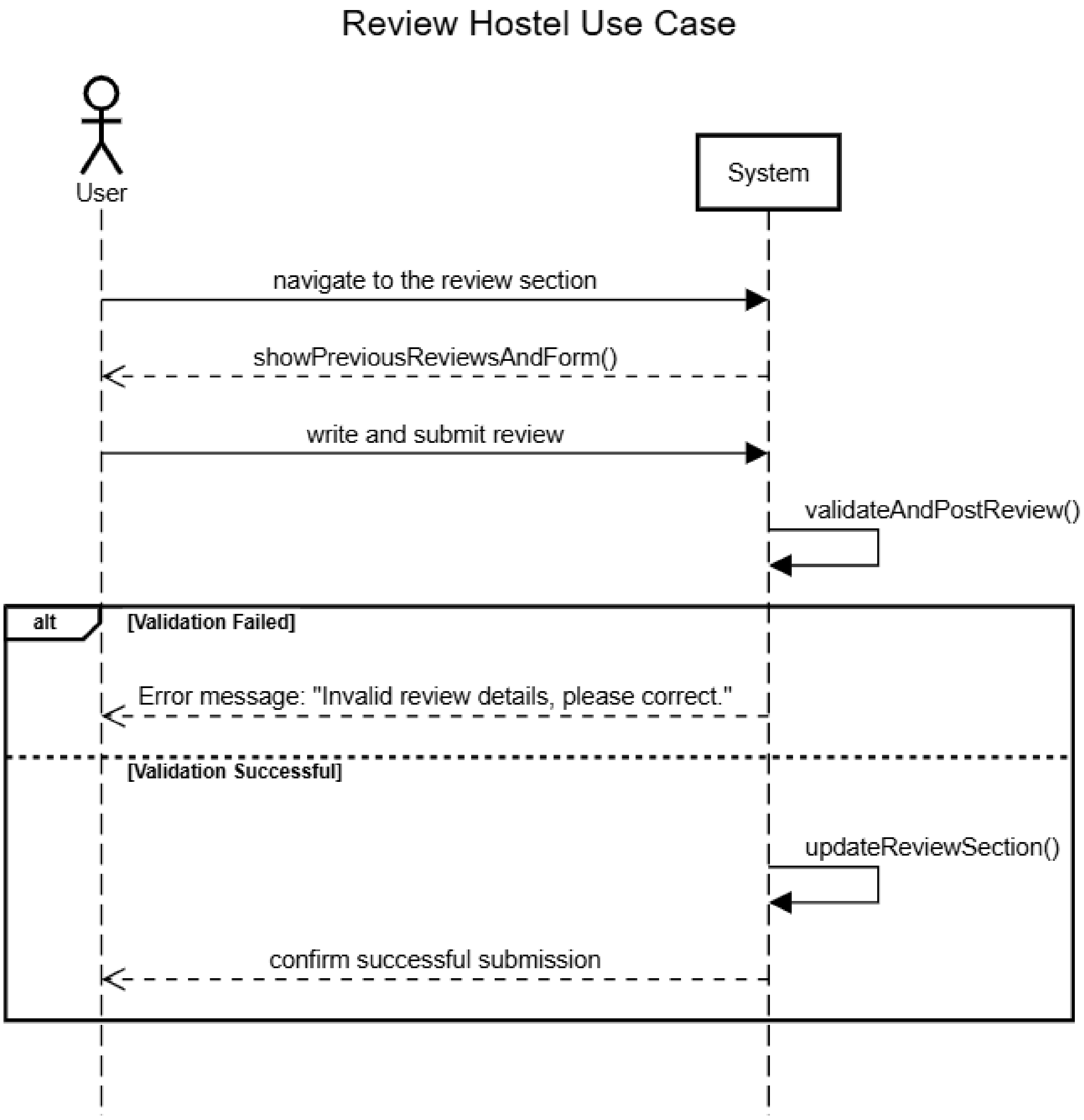
4.10


Check-in Use Case
User
System
provideBookingConfirmation()
verifyBookingDetails()
alt
[Booking Verification Failed]
Invalid booking details, cannot check in.
[Verification Successful]
confirmCheckIn()
updateBookingStatusToCheckedIn()

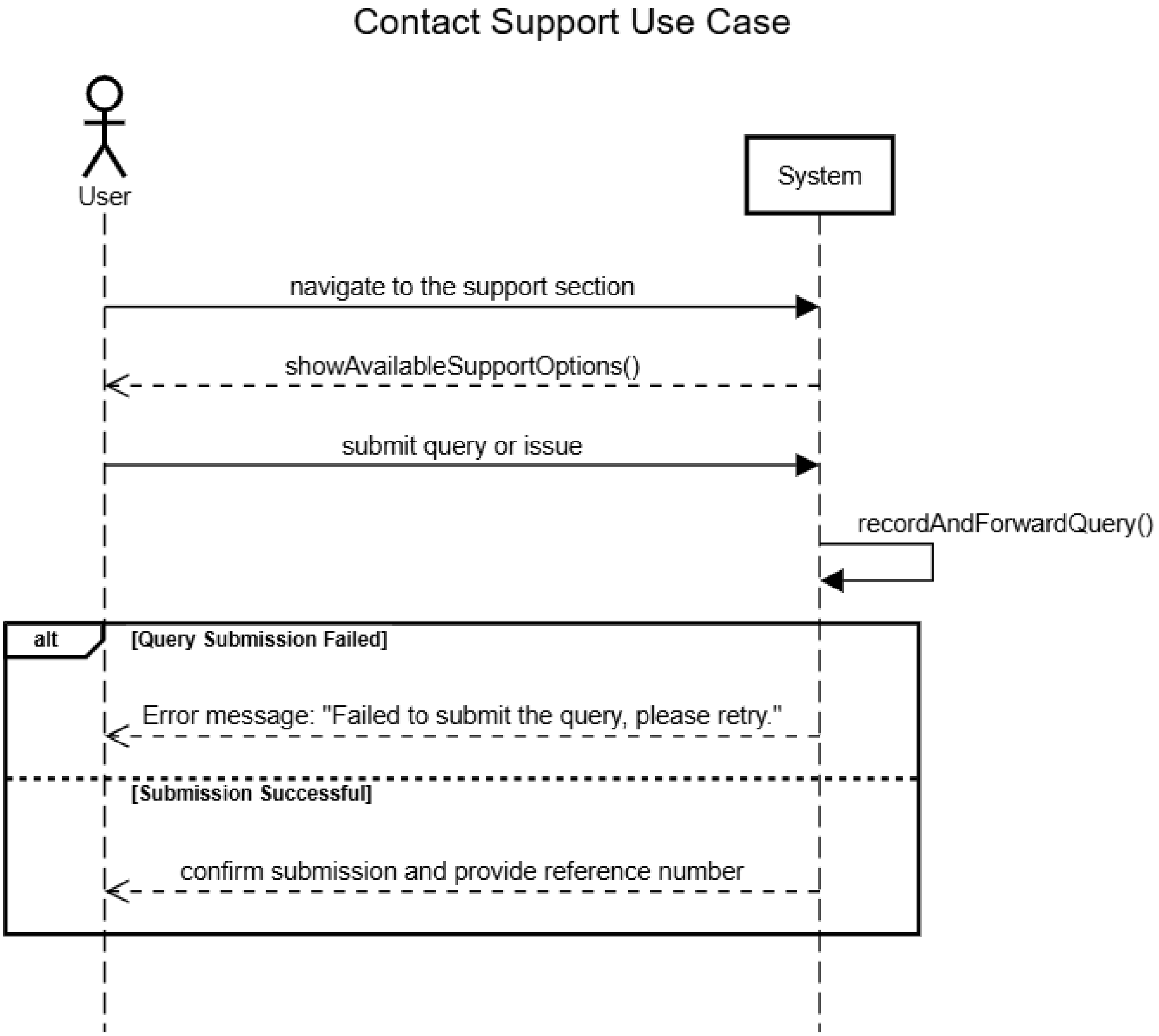
4.11


Check-out Use Case
User
System
navigate to checkout
showStayDetails()
review and confirm final bill
calculateFinalBill()
alt
[Calculation Error]
Error in calculating final bill, try again.
[Calculation Successful]
show final bill
complete checkout
updateBookingStatusToCheckedOut()

4.12

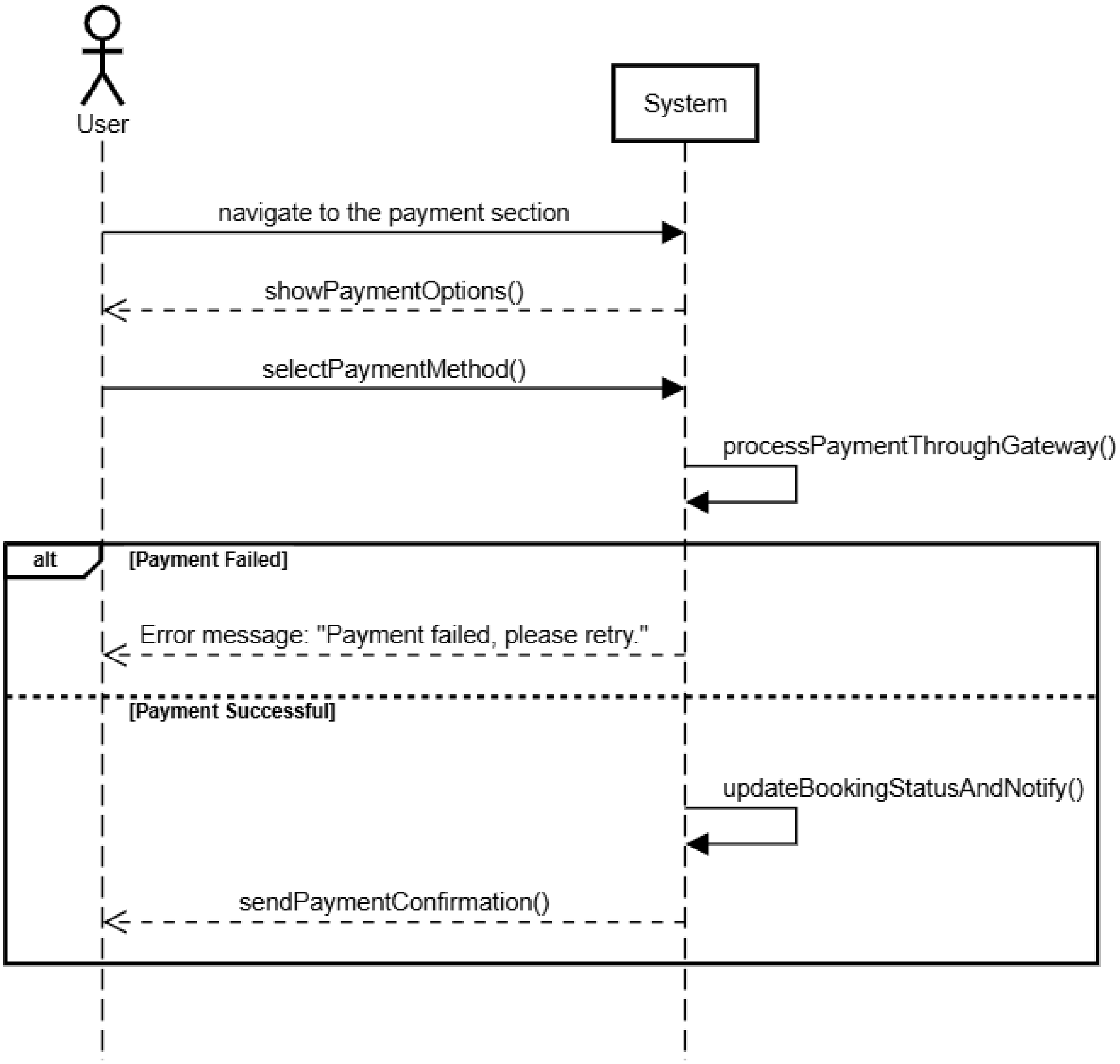

Review Hostel Use Case
User
System
navigate to the review section
showPreviousReviewsAndForm()
write and submit review
validateAndPostReview()
alt
[Validation Failed]
Error message: "Invalid review details, please correct."
[Validation Successful]
updateReviewSection()
confirm successful submission

4.13

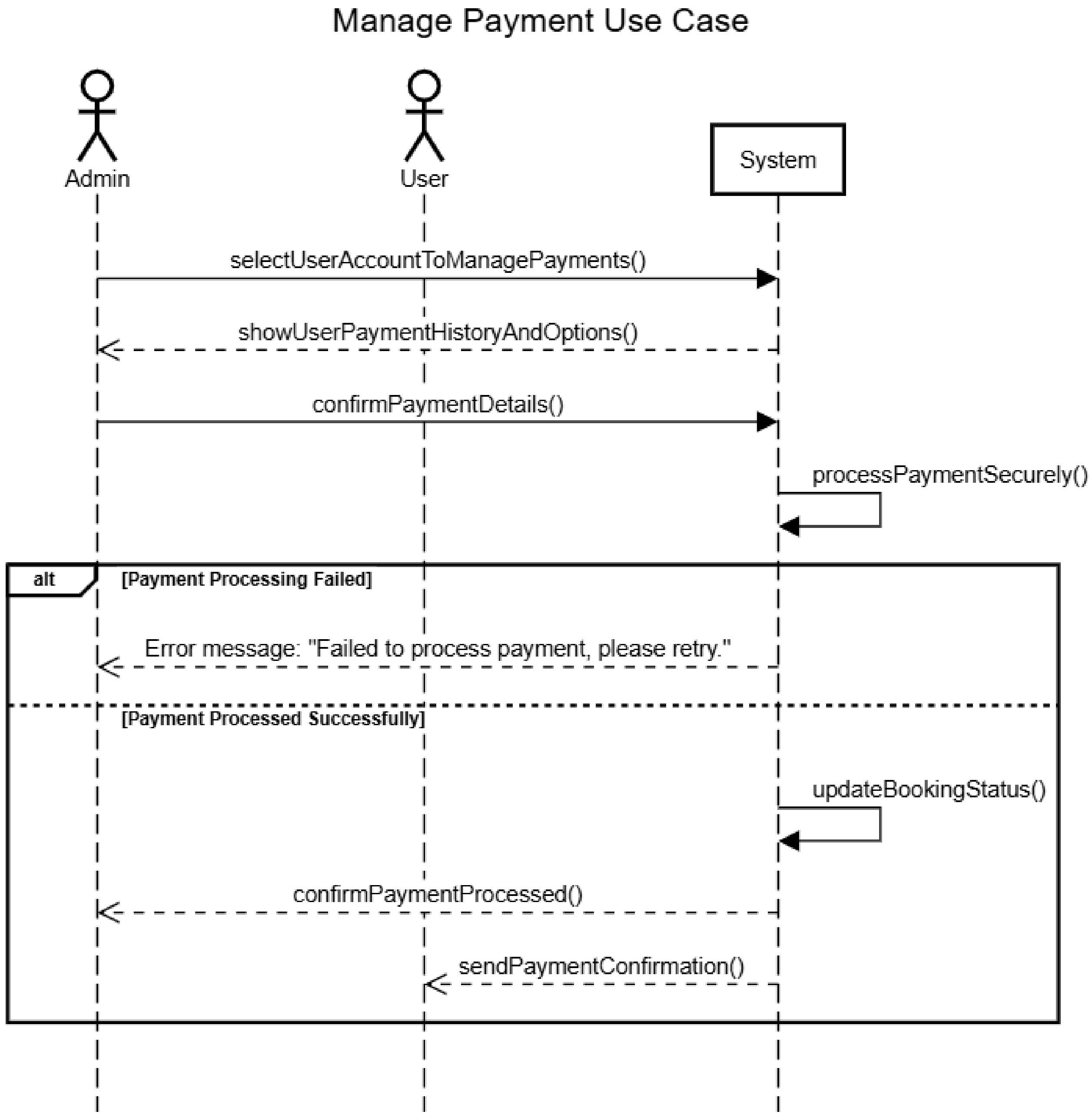

Contact Support Use Case
User
System
navigate to the support section
showAvailableSupportOptions()
submit query or issue
recordAndForwardQuery()
alt
[Query Submission Failed]
Error message: "Failed to submit the query, please retry."
[Submission Successful]
confirm submission and provide reference number

4.14

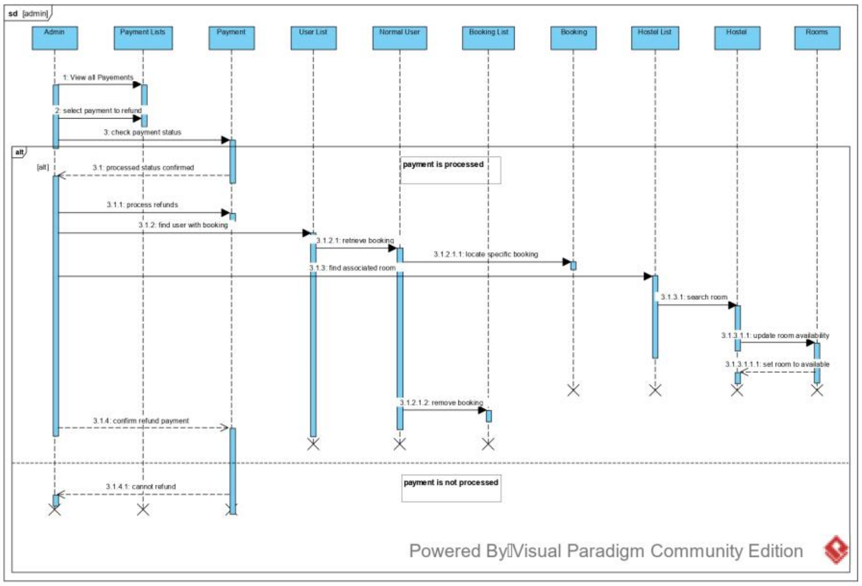
Process Payment Transactions Use Case

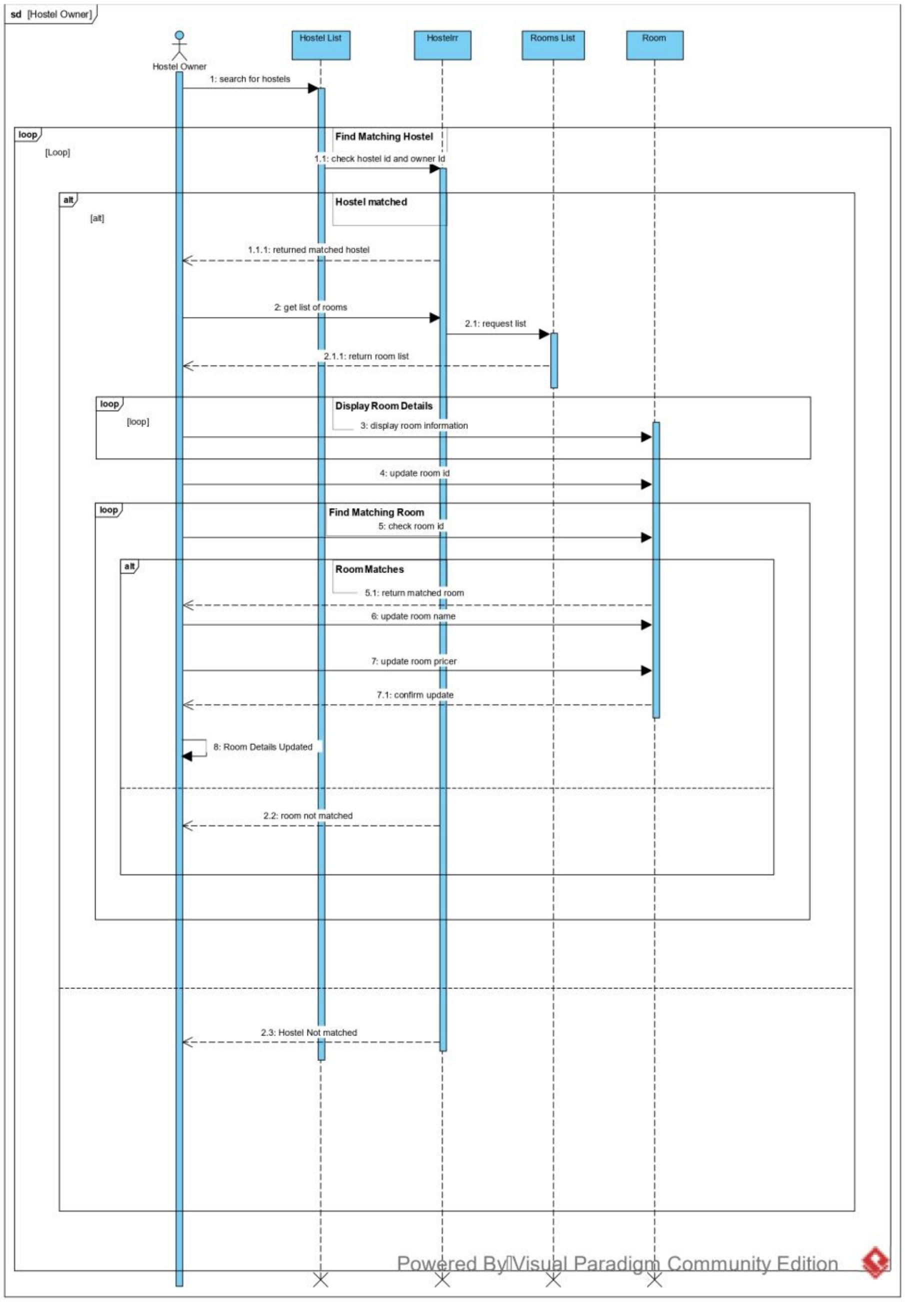

User
System
navigate to the payment section
showPaymentOptions()
selectPaymentMethod()
processPaymentThroughGateway()
alt
[Payment Failed]
Error message: "Payment failed, please retry."
[Payment Successful]
updateBookingStatusAndNotify()
sendPaymentConfirmation()

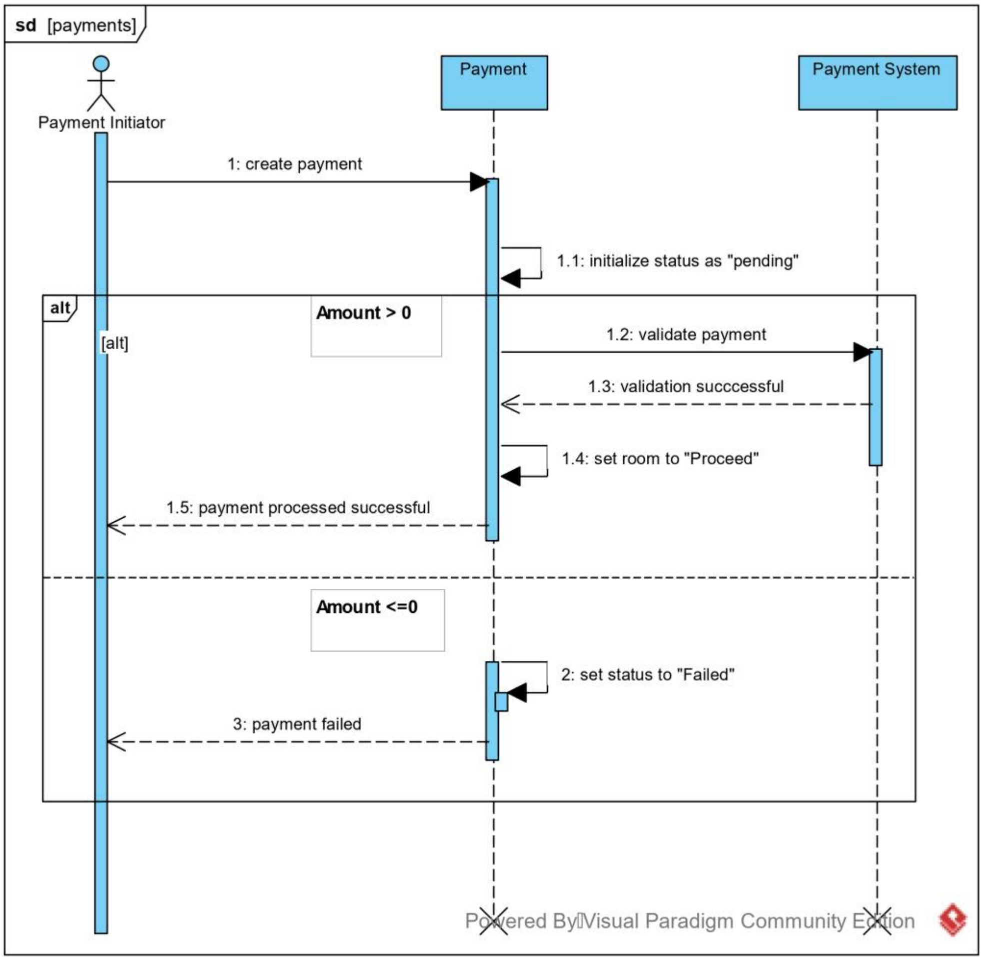
4.15


Manage Payment Use Case
Admin
User
System
selectUserAccountToManagePayments()
showUserPaymentHistoryAndOptions()
confirmPaymentDetails()
processPaymentSecurely()
alt
[Payment Processing Failed]
Error message: "Failed to process payment, please retry."
[Payment Processed Successfully]
updateBookingStatus()
confirmPaymentProcessed()
sendPaymentConfirmation()

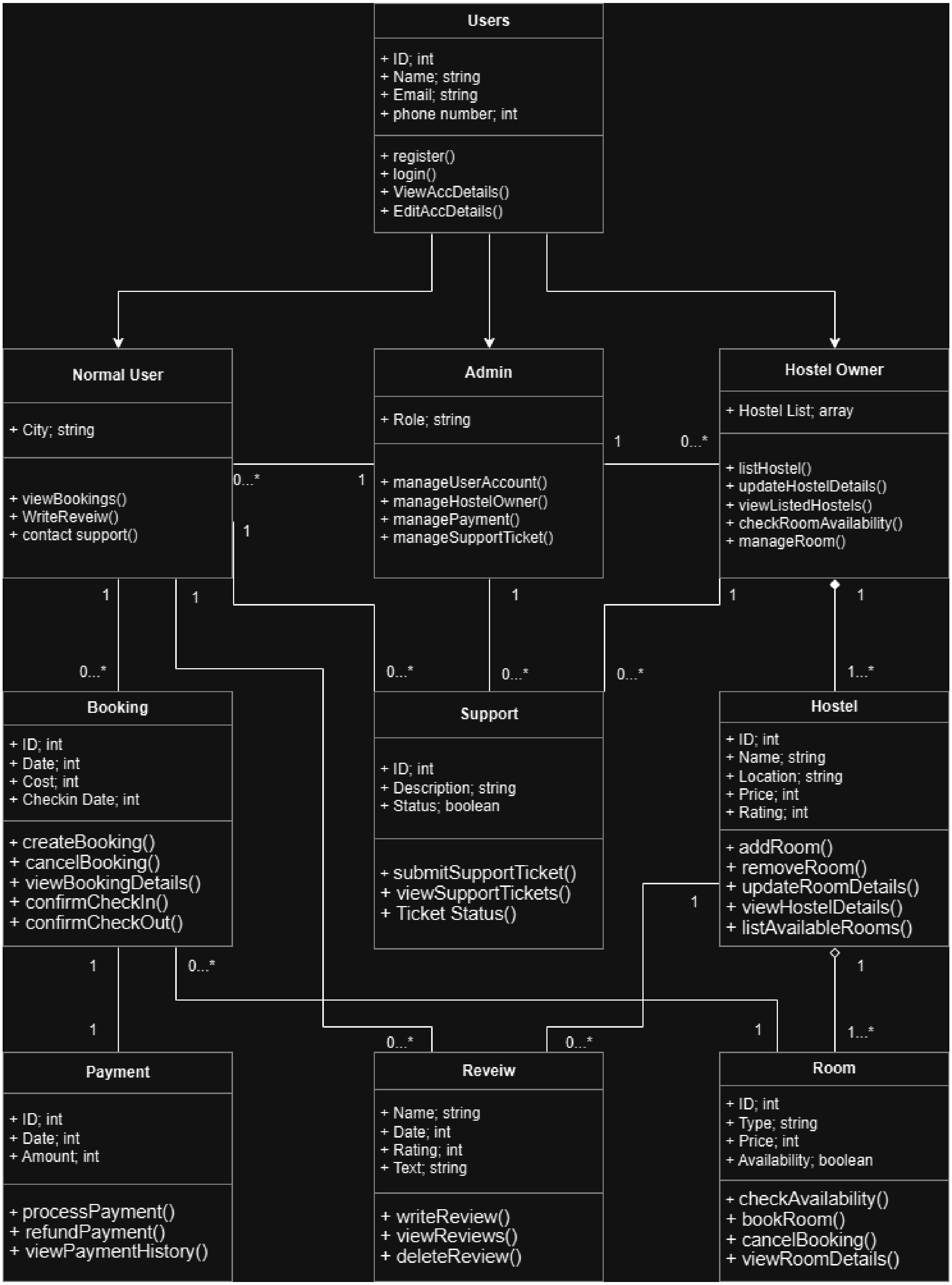
5. Sequence Diagram


sd [admin]
Admin
Payment Lists
Payment
User List
Normal User
Booking List
Booking
Hostel List
Hostel
Rooms
1: View all Payements
2: select payment to refund
3: check payment status
alt
[alt]
3.1: processed status confirmed
payment is processed
3.1.1: process refunds
3.1.2: find user with booking
3.1.2.1: retrieve booking
3.1.2.1.1: locate specific booking
3.1.3: find associated room
3.1.3.1: search room
3.1.3.1.1: update room availability
3.1.3.1.1.1: set room to available
3.1.2.1.2: remove booking
3.1.4: confirm refund payment
payment is not processed
3.1.4.1: cannot refund
Powered By Visual Paradigm Community Edition

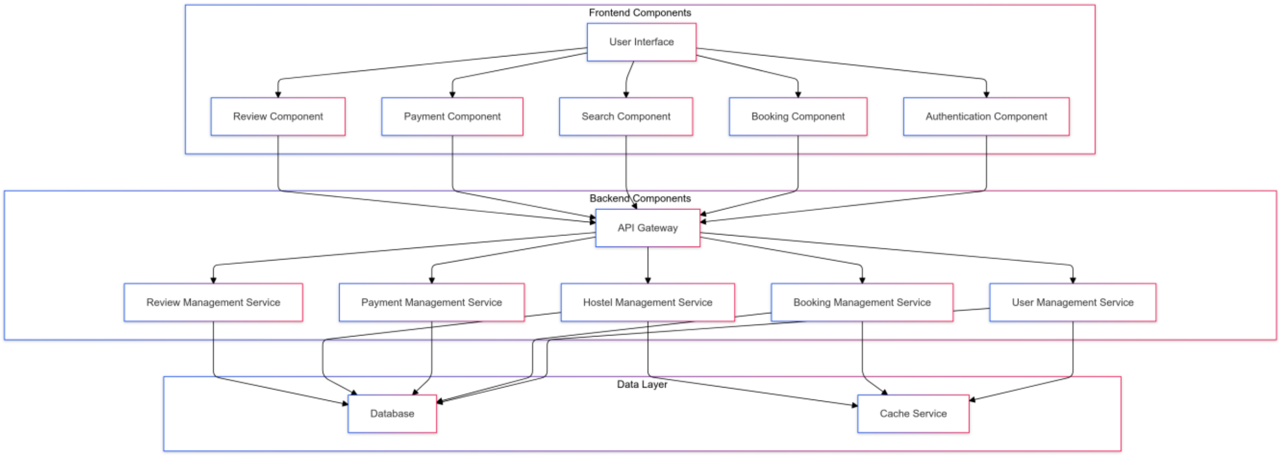

sd [Hostel Owner]
Hostel Owner
Hostel List
Hostelrr
Rooms List
Room
1: search for hostels
loop
[Loop]
Find Matching Hostel
1.1: check hostel id and owner Id
alt
[alt]
Hostel matched
1.1.1: returned matched hostel
2: get list of rooms
2.1: request list
2.1.1: return room list
loop
[loop]
Display Room Details
3: display room information
4: update room id
loop
Find Matching Room
5: check room id
alt
Room Matches
5.1: return matched room
6: update room name
7: update room pricer
7.1: confirm update
8: Room Details Updated
2.2: room not matched
2.3: Hostel Not matched
Powered By Visual Paradigm Community Edition


sd [payments]
Payment Initiator
Payment
Payment System
1: create payment
1.1: initialize status as "pending"
alt
[alt]
Amount > 0
1.2: validate payment
1.3: validation succcessful
1.4: set room to "Proceed"
1.5: payment processed successful
Amount <=0
2: set status to "Failed"
3: payment failed
Powered By Visual Paradigm Community Edition

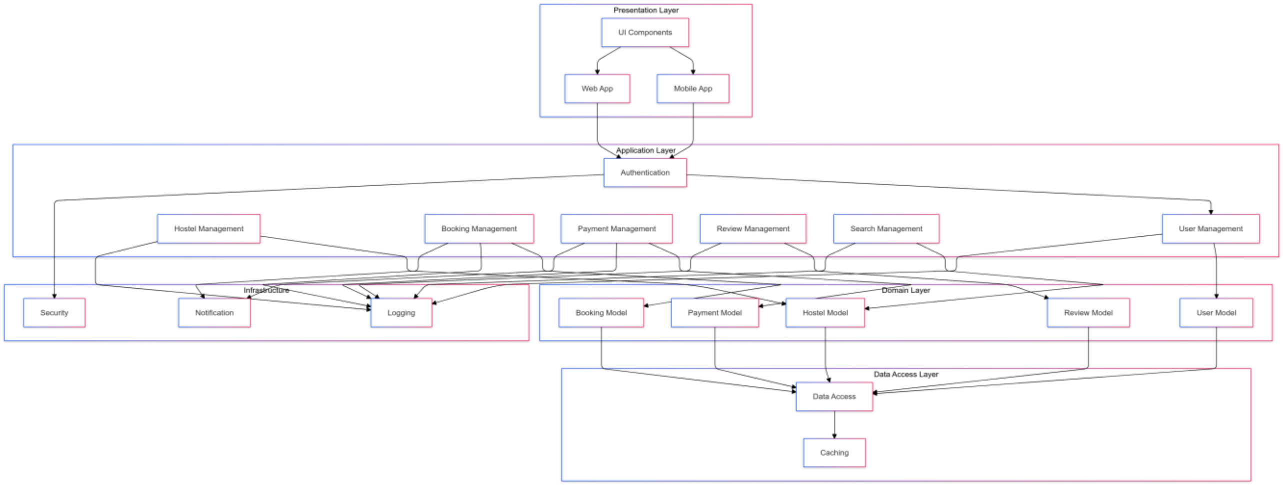
6. Class Diagram



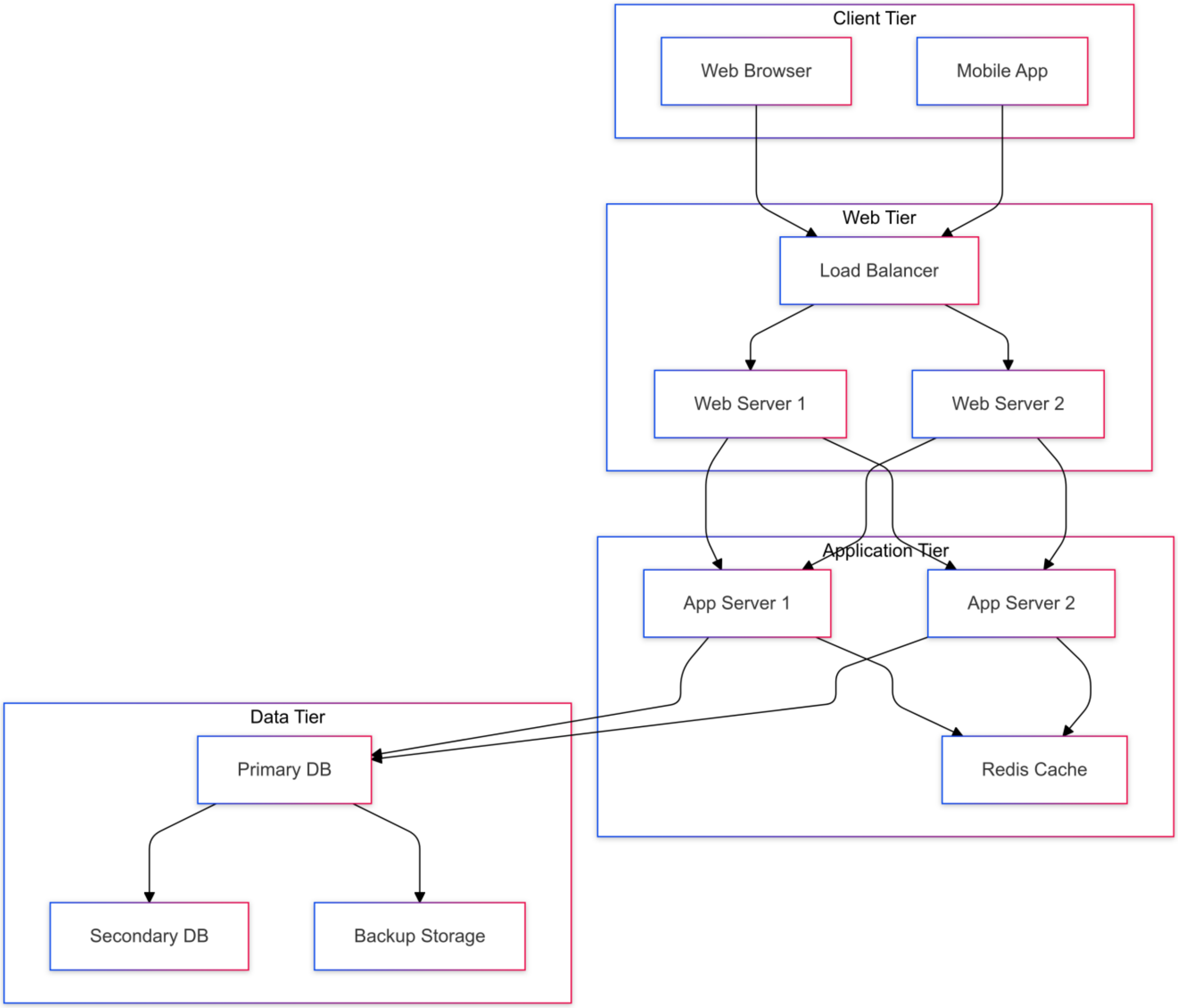
7. Component Diagram



8. Package Diagram



9. Deployment Diagram


Client Tier
Web Browser
Mobile App
Web Tier
Load Balancer
Web Server 1
Web Server 2
Application Tier
App Server 1
App Server 2
Redis Cache
Data Tier
Primary DB
Secondary DB
Backup Storage