HOTEL MANAGEMENT SYSTEM

Submitted By

SOWRAV MONDAL

221-35-819

Supervised By

Nuruzzaman Faruqui

Designation

Assistant Professor

This project report has been submitted in fulfilment of the requirements for the degree of **Bachelor of Science in Software Engineering**

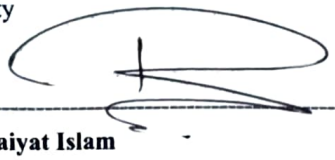
APPROVAL

This project titled on "**Hotel Management System**", submitted by **Sowrav Mondal (ID: 221-35-819)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

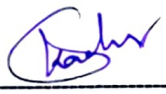
Chairman

Dr. A. H. M. Saifullah Sadi
Professor
Department of Software Engineering
Faculty of Science and Information Technology Daffodil International University



Internal Examiner 1

Dr. Rubaiyat Islam
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University



Internal Examiner 2

Dr. Md. Abdul Kader
Associate Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

Internal Examiner 3

Nuruzzaman Faruqui
Assistant Professor
Department of Software Engineering
Faculty of Science and Information Technology
Daffodil International University

External Examiner

Md. Mostafiz Khan
Managing Director
Tecognize Solutions Limited

PROJECT DECLARATION LETTER (OPTIONAL)

Librarian,
Daffodil International University,
Daffodil Smart City,
Ashulia.Dhaka,Bangladesh

Dear Sir,

CLASSIFICATION OF Project AS RESTRICTED

Please be informed that the following project is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name : Sowrav Mondal
Project Title : Roberto Hotel & Resort

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

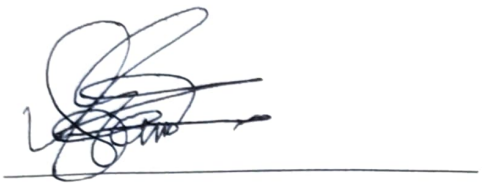
Date:

Stamp:

Note: This letter should be written by the supervisor and addressed to the Librarian, *Daffodil International University* with its copy attached to the thesis.

SUPERVISOR'S DECLARATION

I hereby declare that I have checked this project and, in my opinion, this project is adequate in terms of scope and quality for the award of the Bachelor of Science.



(Supervisor's Signature)

Full Name : **Nuruzzaman Faruqui**

Position : **Assistant Professor**

Date : 26/11/2025

STUDENT'S DECLARATION

I hereby declare that the work in this project is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Daffodil International University or any other institution.

(Student's Signature)

Full Name : **Sowrav Mondal**
ID Number : 221-35-819
Date : 26 November 2025

HOTEL MANAGEMENT SYSTEM

SOWRAV MONDAL

Project submitted in fulfillment of the requirements
for the award of the degree of

Bachelor of Science

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

NOVEMBER 2025

ACKNOWLEDGEMENTS

At the beginning, I express my gratitude to God for giving me strengths, patience and opportunities to complete this project.

I would like to thank my supervisor Nuruzzaman Faruqui, Assistant Professor, Department of Software Engineering, for his support, guidance, and valuable feedback to help me fulfill this project. His advance and valuable advice work as encouragement for me.

I am also sincerely thankful and grateful to all the faculty members of the Department of Software Engineering for their help and suggestions.

Finally, my deepest gratitude to my beloved parents for their continuous love, unconditional support and prayers which makes a foundation for my upcoming success.

DEDICATION

I therefore declare that I have done this project under the oversight of **Nuruzzaman Faruqui**, Assistant Professor, Department of Software Engineering, Daffodil International University. Also declare that neither entire record nor any portion of this record has been submitted somewhere else for my degree.

ABSTRACT

The **Roberto Hotel and Resort** system is a web-based application to automate hotel operations for medium and small hotels. The system supports multiple roles with valuable features like booking, payment, refund, and management. Customers can securely register, login, update the information, and manage booking online. The cart and checkout system are implemented to ensure safe and continental online payments, with loyalty reward feature, while the refund and cancellation system are also transparent processing. Admin can monitor and control all the system and activities efficiently, on the other hand can handle day to day booking and service tasks. The automation of check-in and check-out function provides extra power to the project. With a very strong security, data-protection, and user-friendly design, this system can reduce the manual work process and improve accuracy with customer satisfaction. Overall, it provides a modern and reliable solution for medium and small-sized hotel businesses.

TABLE OF CONTENT

DECLARATION
TITLE PAGE

ACKNOWLEDGEMENTS **II**

DEDICATION **III**

ABSTRACT **IV**

TABLE OF CONTENT v

LIST OF TABLES viii

LIST OF FIGURES x

LIST OF ABBREVIATIONS xiii

LIST OF APPENDICES **xiv**

CHAPTER 1 INTRODUCTION **1**

1.1 Background 1

1.1.1 Context and Relevance 1

1.1.2 Problem Identification 1

1.1.3 Purpose and Justification 2

1.1.4 Scope 2

1.2 Project Planning and Initiation 3

Feasibility Study (Step-by-Step) 3

1.3 Target User Profile and Tentative Elicitation Process 5

1.3.1 Target User 5

1.3.2 User profile 6

1.3.3 Elicitation Process 9

1.4 Project Block Diagram 12

1.5 System Requirements 13

1.5.1 Hardware Requirements 13

1.5.2 Software Requirements 13

1.5.3 Constraints and Dependencies 14

1.6 Project Scheduling 15

1.7 Summary 15

CHAPTER 2 DESIGN AND IMPLEMENTATION 16

2.1 Introduction 16

2.2 Functional Requirements 16

2.3 Non-Functional Requirements 21

2.3.1 Performance 21

2.3.2 Reliability 21

2.3.3 Portability 22

2.4 Object-oriented System design using UML 25

2.4.1 Use Case Diagram 25

2.4.2 Case Description 26

2.4.3 Activity Diagram 52

2.4.4 Sequence Diagram 77

2.4.5 Class Diagram 102

2.4.6 ER Diagram 103

2.5 Coding: Appendix A 104

2.6 Summary 142

CHAPTER 3 SOFTWARE TESTING 143

3.1 Introduction 143

3.2 Testing Features 143

3.2.1 Feature to Be Tested 143

3.3 Testing Strategies 144

3.3.1 Test Approach 144

3.3.2 Pass/Fail Criteria 145

3.4 System Testing (Test Cases with Report) 146

3.5 Summary 166

CHAPTER 4 DEPLOYMENT AND MAINTENANCE 167

4.1 Introduction 167

4.2 Try to follow the SRLC (software release life cycle) 167

CHAPTER 5 USER MANUAL 169

5.1 Introduction 169

5.2 Project Functionalities 169

5.3 Summary 189

CHAPTER 6 PROJECT SUMMARY 190

6.1 Introduction 190

6.2 Project Limitation 190

6.3 Scope 191

6.4 Future Work 191

6.5 Conclusion 192

REFERENCES 193

APPENDICES 194

LIST OF TABLES

Table 1.3.2.1: User Profile for Customer 6
Table 1.3.2.2: User Profile for Receptionist 7
Table 1.3.2.3: User Profile for Admin 8
Table 1.6.1 : Time Frame 15
Table 2.2.1: Functional Requirements 16-20
Table: 2.4.2.1 : Case Description-01: User Registration 26
Table: 2.4.2.2 : Case Description-02: User Login 27
Table: 2.4.2.3 : Case Description-03: Update information 28
Table: 2.4.2.4 : Case Description-04: Chatbot 29
Table: 2.4.2.5 : Case Description-05: Smart Room Recommendation 30
Table: 2.4.2.6 : Case Description-06: Add Room 31
Table: 2.4.2.7 : Case Description-07: Update Room 32
Table: 2.4.2.8: Case Description-08: Delete Room 33
Table: 2.4.2.9: Case Description-09: Add to Cart 34
Table: 2.4.2.10 : Case Description-10: Make Booking 35-36
Table: 2.4.2.11: Case Description-11: Make Payment 37
Table: 2.4.2.12: Case Description-12: Loyalty Reward 38
Table: 2.4.2.13: Case Description-13: Loyalty Discount 39
Table: 2.4.2.14: Case Description-14: Cancel Booking 40
Table: 2.4.2.15: Case Description-15: Get Refund 41
Table: 2.4.2.16: Case Description-16: Make Check-in 42
Table: 2.4.2.17: Case Description-17: Make Check-out 43
Table: 2.4.2.18: Case Description-18: Add Blog 44
Table: 2.4.2.1: Case Description-19: Update Blog 45
Table: 2.4.2.20: Case Description-20: Delete Blog 46
Table: 2.4.2.21: Case Description-21: View Booking 47
Table: 2.4.2.22 : Case Description-22: View Earning 48
Table: 2.4.2.23: Case Description-23: Resend Email 49
Table: 2.4.2.24: Case Description-24: Delete Contact 50
Table: 2.4.2.25: Case Description-25: Logout 51
Table: 3.4.1: Test case 01: User Registration 147
Table: 3.4.2: Test case 02: Login 148
Table: 3.4.3: Test case 03: Update User Information 149
Table: 3.4.4: Test case 04: Change Password 150
Table: 3.4.5: Test case 05: Smart Booking Recommendation 151
Table: 3.4.6: Test case 06: Cart System 152
Table: 3.4.7: Test case 07: Room Booking 153
Table: 3.4.8: Test case 08: Online Payment System 154
Table: 3.4.9: Test case 09: Booking Cancellation 155
Table: 3.4.10: Test case 10: Refund 156
Table: 3.4.11: Test case 11: Add Room 157
Table: 3.4.12: Test case 12: Room Management 158
Table: 3.4.13: Test case 13: Earning Management 159

Table: 3.4.14: Test case 14: User Management 160
Table: 3.4.15: Test case 15: Contact Management 161
Table: 3.4.16: Test case 16: Loyalty System Reward 162
Table: 3.4.17: Test case 17: Add Blog 163
Table: 3.4.18: Test case 18: Blog Management 164
Table: 3.4.19: Test case 19: Chatbot 165
Table: 3.4.20: Test case 20: Logout 166
Table 7.1: booking 195
Table 7.2: User 196
Table 7.3: rooms 196
Table 7.4: room_images 197
Table 7.5: user_preferences 197
197Table 7.6: payments 197
Table 7.7: earnings 198
Table 7.8: refund_requests 198
Table 7.9: coin_transactions 199
Table 7.10: chat_messages 199
Table 7.11: chat_sessions 199
Table 7.12: contact_submissions 200

LIST OF FIGURES

Figure 1: Tentative Elicitation Process 9

Figure 2: System Block Diagram 12

Figure 3: Use Case Diagram 25

Figure 3.1: Activity diagram for Registration 52

Figure 3.2: Activity diagram for Login 53

Figure 3.3: Activity diagram for Update Profile Information 54

Figure 3.4: Activity diagram for Chatbot 55

Figure 3.5: Activity diagram for Smart Room Recommendation 56

Figure 3.6: Activity diagram for Add Room 57

Figure 3.7: Activity diagram for Update Room 58

Figure 3.8: Activity diagram for Delete Room 59

Figure 3.9: Activity diagram for Add to cart 60

Figure 3.10: Activity diagram for Make Booking 61

Figure 3.11: Activity diagram for Make Payment 62

Figure 3.12: Activity diagram for Loyalty Reward 63

Figure 3.13: Activity diagram for Loyalty Discount 64

Figure 3.14: Activity diagram for Cancel Booking 65

Figure 3.15: Activity diagram for Get Refund 66

Figure 3.16: Activity diagram for Make Check-in 67

Figure 3.17: Activity diagram for Make Check-out 68

Figure 3.18: Activity diagram for Add Blog 67

Figure 3.19: Activity diagram for Update Blog 70

Figure 3.20: Activity diagram for Delete Blog 71

Figure 3.21: Activity diagram for View Booking 72

Figure 3.22: Activity diagram for View Earning 73

Figure 3.23: Activity diagram for Resend Email 74

Figure 3.24: Activity diagram for Delete Contact 75

Figure 3.25: Activity diagram for Logout 76

Figure 4.1: Sequence diagram for Registration 77

Figure 4.2: Sequence diagram for Login 78

Figure 4.3: Sequence diagram for Update Information 79

Figure 4.4: Sequence diagram for Chatbot 80

Figure 4.5: Sequence diagram for Smart Room Recommendation 81

Figure 4.6: Sequence diagram for Update Information 82
Figure 4.7: Sequence diagram for Blog Management 83
Figure 4.8: Sequence diagram for Delete Room 84
Figure 4.9: Sequence diagram for Add to Cart 85
Figure 4.10: Sequence diagram for Make Booking 86
Figure 4.11: Sequence diagram for Make Payment 87
Figure 4.12: Sequence diagram for Loyalty Reward 88
Figure 4.13: Sequence diagram for Loyalty Discount 89
Figure 4.14: Sequence diagram for Cancel Booking 90
Figure 4.15: Sequence diagram for Get Refund 91
Figure 4.16: Sequence diagram for Make Check-in 92
Figure 4.17: Sequence diagram for Make Check-out 93
Figure 4.18: Sequence diagram for Add Blog 94
Figure 4.19: Sequence diagram for Update Blog 95
Figure 4.20: Sequence diagram for Delete Blog 96
Figure 4.21: Sequence diagram for View Booking 97
Figure 4.22: Sequence diagram for View Earning 98
Figure 4.23: Sequence diagram for Resend Email 99
Figure 4.24: Sequence diagram for Delete Contact 100
Figure 4.25: Sequence diagram for Logout 101
Figure 5: Class Diagram 102
Figure 6: ER Diagram 103
Figure 7.1: User Dashboard 169
Figure 7.2: Admin Dashboard 170
Figure 7.3: Receptionist Dashboard 170
Figure 7.4: User Registration 171
Figure 7.5: Login for customer, admin & receptionist 172
Figure 7.6: Update Profile Information 173
Figure 7.7: Changed Password 174
Figure 7.8: Smart Booking Recommendation 175
Figure 7.9: Cart System 176
Figure 7.10: Room Booking 177
Figure 7.11: Online Payment System 178
Figure 7.12: Loyalty Reward 179

Figure 7.13: Loyalty Discount 180
Figure 7.14: Booking Cancellation 181
Figure 7.15: Refund Request 182
Figure 7.16: check-in and Check-out 182
Figure 7.17: Service Request 183
Figure 7.18: Add Room 183
Figure 7.19: Room Management 184
Figure 7.20: Booking Management 184
Figure 7.21: Earning Management 185
Figure 7.22: User Management 185
Figure 7.23: Contact Management 186
Figure 7.24: Chatbot System 187
Figure 7.25: Add Blog 188
Figure 7.26: Blog Management 189

LIST OF ABBREVIATIONS

HMS	Hotel Management System
UI	User Interface
UX	User Experience
API	Application Programming Interface
DB	Database
UML	Unified Modelling Language
MVC	Model-View-Controller
SSL	Secure Sockets Layer
HTTPS	Hypertext Transfer Protocol Secure
CMS	Content Management System
SQL	Structured Query Language
UXD	User Experience Design

LIST OF APPENDICES

Appendix A: Database Structure 19

CHAPTER 1 INTRODUCTION

1.1 Background

In today's era, hotel systems are rapidly growing and moving to automated systems from manual processes to improve the services and efficiency. But small and medium-sized hotels still rely on the handwritten documentation process to manage bookings and payments. Due to this, manual processes may cause some errors and issues.

To improve and solve these problem, Roberto Hotel Management System comes with a solution and develop a web-based platform to control a centralized system which allows customer to make hustle free booking, online payment with reward system and manage them systematically.

The system is designed to reduce the manual work process to improve accuracy and efficiency with customer satisfaction and provide a modern and secure platform to overcome traditional hotel management system.

1.1.1 Context and Relevance

The hospitality industry is growing rapidly over past decade for increasing digital transformation. The hotels and resort industry are moving on technological use to manage their day-to-day operation. The manual management systems are failing to fulfil the customer expectation in term of services, accuracy and efficiency.

A **Hotel Management System** provides a platform to handle their operations like booking, payments, room-allocation etc in organized manner way. With a web-based hotel management solution, the administrative task can be automated, where previously needs a huge human effort. The "**Roberto – Hotel & Resort**" aims to organized hotel operations for **small and medium-size**d hotel with secure and efficient online system with support of customers and staffs.

1.1.2 Problem Identification

Traditional small and medium-sized hotel system depends on manual registration, paper-based record-keeping and cash transactions, which may face human error and insufficiency. This limitation sometimes triggered double booking, delayed in check-in, mismanagement of customer valuables information and lack of a centralized control. In addition, unavailability of online payment system creates distribution on payments and refunds.

So, there is very strong need of a web-based automated system, which provide a centralized control and reduce manual workload for more accuracy.

1.1.3 Purpose and Justification

The main purpose of this hotel management system is to develop a web-based system to automates hotel operation like booking, check-in & check-out, payment, refund, loyalty reward, customer data handling with accuracy and their report generation. The system supports multiple stake holder like Admin, Receptionist and Customer to performs their desired task with efficiency without any manual work.

The project featured with Payment API for online and transparent transaction with building user trust. Moreover, with introduced **Smart Booking Recommendation** focus on customer satisfaction with their matching booking to enhance system operations easily.

The project adds values by producing a cost-effective, scalable and efficient system that is perfect for small and medium-sized hotels that lack digital infrastructure.

1.1.4 Scope

The project's scope covers all automation of hotels operations like booking, payment, report generation etc. It includes both front-end and backed development with a secure database system and a administrative and receptionist dashboard to manage all operations.

The system focuses on:

1. Secure Login and Registration
2. Room Booking and room management
3. Smart room booking recommendation
4. A special chatbot for getting information
5. Booking report generation
6. Online payment integration with stripe
7. Loyalty reward functions
8. Cancellation and refund processing
9. A Customer dashboard for managing operations
10. An Admin and Receptionist dashboard for controlling administrative operations

1.2 Project Planning and Initiation

Feasibility Study (Step-by-Step)

The phase focuses on the project idea, determining its feasibility and its purpose. A step-by-step feasibility was analyzed to ensure the system is suitable for the real world.

Phase 1 Preliminary Analysis & Project Scope Definition:

I identified major problems in small and medium sized hotels in preliminary analysis like manual booking errors, traditional cash payment system, and huge delay in refund policy. Also, there is no loyalty reward system to grab customer attention in today's hotel system.

To overcome these daily issues, the project scope was defined in below:

- Develop a web-based hotel management system to make it easily accessible for everyone using PHP, HTML, CSS, JS, MySQL
- Essential features like real-time room booking, room allocations, online payment system, reward program, and a centralized management system are added.
- Improve the accuracy and efficiency for customer and hotel staff

Phase 2 Market Feasibility Analysis:

During market research, I studied many exciting hotel systems, but they are more expensive and complicated for normal users. For that, local hotels cannot adopt them and stay with their manual or outdated software.

So, the study told a very strong market demand for:

- Affordable and user-friendly hotel system
- Real time booking and payment system
- Loyalty rewards for further booking
- Dedicated refund rules

The international hotel system offers many features but has a huge lack of affordability and localization. So, my proposed system can be the best choice for the local system to attract and satisfy customers.

Phase 3 Technical Feasibility Analysis:

In technical feasibility, I analyzed the system will be developed and run on available technology.

Those are:

- Frontend: **HTML, CSS, JavaScript**
- Backend: **PHP**
- Database: **MySQL**
- Server: **Apace or Xampp**
- Additional: **Stripe payment API**

All the tools are open-source and lightweight and have a wide area of support to maintain the system easily.

Phase 4 Financial Feasibility Analysis:

The system is financially stable and practical due to using open-source software and low development cost. Also, maintenance and further development will be cost effective.

The only expenses:

- Domain and hosting fee
- Further development and maintenance fee
- Stripe payment gateway fee (Now it is on test payment)

Compared to other hotel systems, where they charge a huge fee for adaption (listing your hotel) and a percentage of booking fee, my system can be cost effective and provide a provide a long-term financial benefit.

1.3 Target User Profile and Tentative Elicitation Process

1.3.1 Target User

1. **Customers**

- People who want to book room, check room availability, complete online transactions with managing their reservations.

2. **Receptionist**

- Hotel staffs to manage automate customers check-in, check-out, booking details, service request and handling their quires.

3. **Admin**

- The main administrative system controller who manages rooms, payment, refund, earning, customer quires and newsletter and system configurations.

1.3.2 User profile

Table 1.3.2.1 : User Profile for **Customer**

User Class	**Note on Characteristics**
Type of user	Customer
Age range	18 to 60 years
Frequency of use	Occasional using
Mandatory	Yes
Computer experience	Basic internet and mobile browsing
Education	Up to secondary level
Goal	To browse available room, room booking, making online payment and managing profile
Language skills	English, Bangla
Number of users	Up to hundred customers (depends on hotel size)
Training	No training needed
Others system use	May use another hotel website
Way of working	Individual usages

Table 1.3.2.2: User Profile for **Receptionist**

User Class	**Note on Characteristics**
Type of user	Receptionist (Hotel Staff)
Age range	20 to 40 years
Frequency of use	Daily Use
Mandatory	Yes, required to manage bookings with check-in and check-out
Computer experience	Intermediate
Education	Diploma or higher educations
Goal	To automate check-in and check-out feature and room assignment
Language skills	English and Bangla
Number of users	1 to 3 per hotel
Training	Required for system navigation and operations
Others system use	Experienced with hotel software
Way of working	Front desk usages, frequently interacting with customers

Table 1.3.2.3 : User Profile for **Admin**

User Class	**Note on Characteristics**
Type of user	Administrator
Age range	25 to 50 years
Frequency of use	Daily or weekly
Mandatory	Yes
Computer experience	Intermediate to advance computer experience
Education	Higher education or a bachelor's degree
Goal	To manage overall system
Language skills	English
Number of users	1 admin per hotels
Training	Required a basic system training
Others system use	Any financial system or hotel software
Way of working	Independent working and updating records regularly

1.3.3 Elicitation Process

Elicitation process is necessary to gather requirements to develop "Roberto – Hotel and Resort" system. Multiple elicitation process and method are used to gather requirements from different stakeholder to identify their expectation, functional needs and usability concern. The following methos are used:

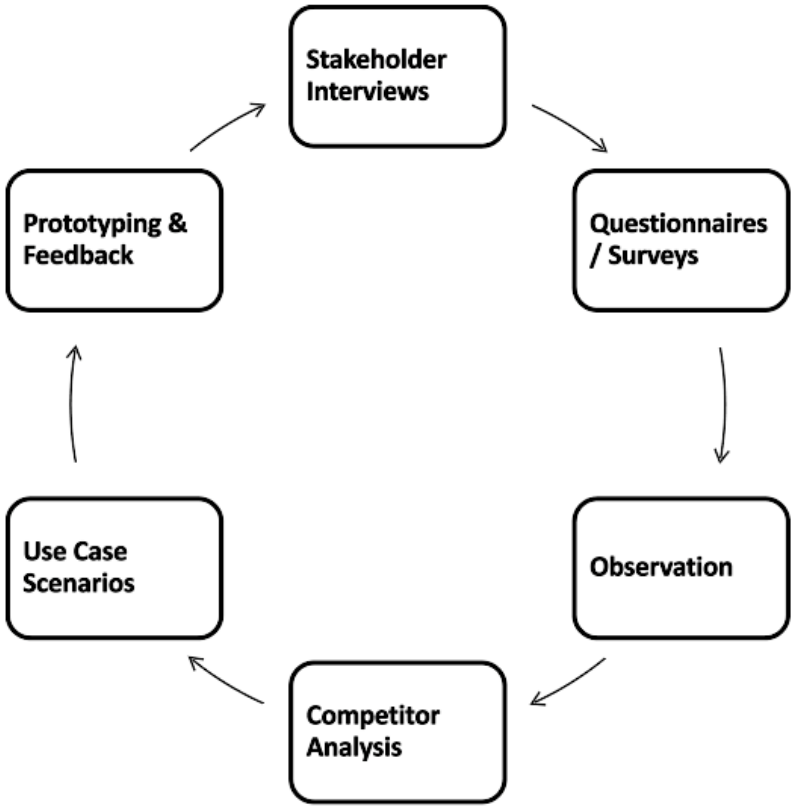


Figure 1 : Tentative Elicitation Process

1.3.3.1 Stakeholder Interviews

Interviews are conducted with hotel manager, hotel staffs and a group of frequent customers. These interviews are helpful to understand the lacking and challenges of manual booking system like double booking, payment information lacking, delay in refunds and paper-based customer data handling. The feedback guides to prioritize the essential features.

1.3.3.2 Surveys

Surveys were distributed to customers and hotel staffs to gather data. Customer surveys focus on booking system, their expected payment methods and user experience expectations. And staff surveys help to identify the issues or difficulties they are facing like manual entry, managing multiple customers during peak times etc.

1.3.3.3 Observation

Observation sessions are done on the front desk of a hotel to understand and study on the workflow. This session includes customer reservation, room allocation, payment process and check-in & check-out handling.

1.3.3.4 Competitor Analysis

Existing hotel management system (OYO, Booking.com) ware analysed to understand their functionality which are commonly offered to the customer. This analyzation helped me to identify their missing functionality and implement in my system:

Those are:

- Loyalty reward system
- Real time room booking and allocation
- Instant refund and cart system for room selection

1.3.3.5 Use Case Scenario

Use case scenario are designed and developed based on real hotel operation. User interaction process was captured on this.

They are:

- Customer browsing room, getting recommended room, adding to their cart and make a booking
- Customer gets loyalty reward after booking
- Receptionist performs check-in and check-out

- Admin adding new rooms and editing them as per needs

1.3.3.6 Prototyping and Feedback

Prototypes ware built for login page, booking page and admin dashboard and shown to some selected user.

Their feedback was used to improve:

- Layout and navigation structure
- Form usability
- Visual clarity

1.4 Project Block Diagram

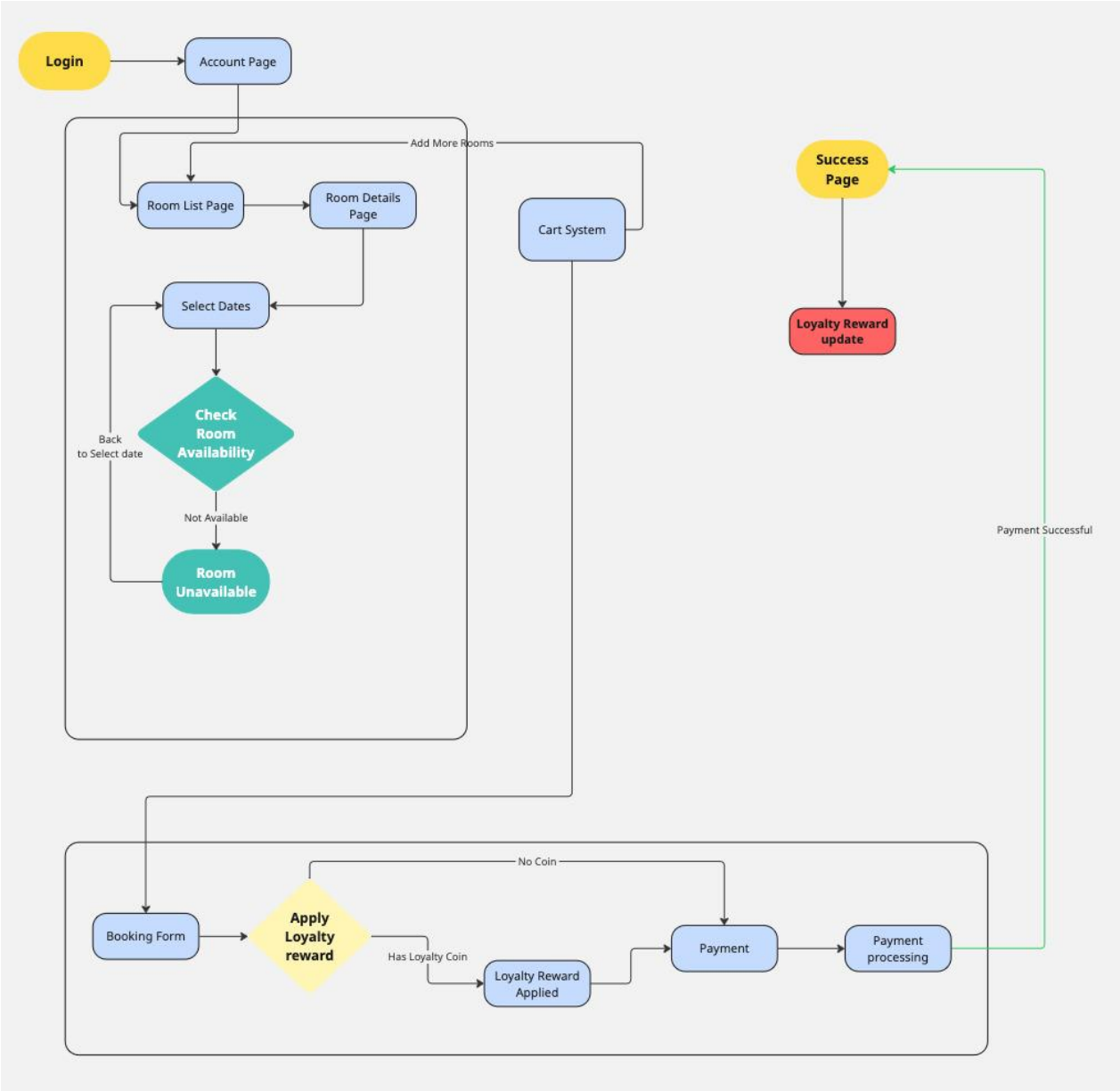


Figure 2: System Block Diagram

1.5 System Requirements

The system requirements outline the hardware and software dependencies that need to develop, deploy, operate, and maintain the Roberto Hotel system. These requirements ensure that the system can run easily and efficiently.

1.5.1 Hardware Requirements

For Development:

- Processor: Intel Core I3 or higher
- Ram: Minimum 4 GB
- Storage: Minimum 20 GB
- Display: Up to 1200 * 992 px
- Network: Stable internet connection

For Deployment:

- Processor: Dual-core CPUS
- Ram: Minimum 2 GB
- Storage: Minimum 20 GB
- Network: Stable internet connection with payment API access

For End User:

- Device: Desktop, Laptop, Mobile, Table
- Internet: Minimum 1Mbps speed

1.5.2 Software Requirements

For Development:

- Operating System: Windows / Mac OS / Linux
- Server: XAMPP (Apache and MYSQL)
- Code Editor: VS Code

For Deployment:

- Server: Apache or Nginx
- PHP version: 7.4 or above
- Database: MySQL
- SSL: SSL certificate need for secure payment

For End user:

- Browser: Cromer, Safari, Microsoft Edge, Opera Mini, Mozilla Firefox

1.5.3 Constraints and Dependencies

This section tells the limitations and dependencies of my system that will affect system functionality.

Constraints:

- The system is built for web, no mobile application support
- The server can produce a performance gap, depends on hosting environment
- The system is not optimized for any hotel chain

Dependencies:

- The system depends on stipe payment gateway to collect payments
- Required a valid SSL certificate
- The system maybe down due to server downtime

1.6 Project Scheduling

Table 1.6.1 : Time Frame

	Weeks				
Phase	1 - 4	4 -7	8 -14	14 -18	19 -24
Requirement Gathering	✓				
UI/UX Design	✓	✓			
Frontend Development		✓	✓		
Backend Development		✓	✓	✓	
Database Design			✓	✓	
Testing and Debugging			✓	✓	✓
Final Deployment & Documentation					✓

1.7 Summary

In Chapter 1, we can get an overview of Roberto Hotel Management System, including the background of the project, the existing problem in traditional hotel system and the project scope, and its purpose to develop. A details feasibility study is conducted to analyze the market demand and financial stability of the proposed project. Also, elicitation processes are analyzed to gather requirements from the customer and hotel staff. Furthermore, I discussed the system requirements and tools I used in this project. The project planning scheduling was also conducted in this chapter to get an idea of how much time is needed.

CHAPTER 2 DESIGN AND IMPLEMENTATION

2.1 Introduction

The chapter will describe the overall design and implementation of the Roberto Hotel system. It explains the different module are developed, structured and organized to ensure a secure and efficient operations

2.2 Functional Requirements

On this functional requirement section, it defines what the system must do. This section mapped the main functions the system will perform. These functional requirements ensure the project will fulfil the user's and business needs.

Table 2.2.1: Functional Requirements

FR01	**Registration**
Description	Using system functionality user need to register on the system by fill up the registration form
Stakeholder	Receptionist, Customer

FR02	**Login**
Description	Customer, Receptionist and Admin must be login to use the system, and they will go to their destination pages by login
Stakeholder	Receptionist, Customer, Admin

FR02	**Update Information**
Description	Customers and receptionists can update their profile information. For, customer it will be useful for checkout
Stakeholder	Customer, Receptionist

FR04	Add Room
Description	Admin can create a room from admin dashboard by clicking add room section
Stakeholder	Admin

FR05	Room Management
Description	Admin can get all information about room and can edit & delete the room from the system
Stakeholder	Admin

FR06	Smart Room Recommendation
Description	Customer can see their recommend room by inputting their information
Stakeholder	Customer

FR07	Room List
Description	The system will allow customer to see all available room with proper information
Stakeholder	Customer

FR08	Add to Cart
Description	The system will allow customer to add, view, update and remove their booking selection on cart will 30-minute expiry
Stakeholder	Customer

FR09	Booking Management
Description	Admin and receptionist can see the customer booking information with check-in and check-out date
Stakeholder	Admin, Receptionist

FR10	Check-in & Checkout
Description	The system will allow receptionist to make check-in and check-out for customer
Stakeholder	Receptionist

FR11	Loyalty Reward
Description	Customer will receive a loyalty reward in coins and can take discount next time by that
Stakeholder	Customer

FR12	Chatbot
Description	The system will allow the user to chat with the system to get information and solution of an issue
Stakeholder	Customer

FR13	Advance Analytics Report
Description	The system will allow the admin to generate report about from booking, user management, contact information
Stakeholder	Admin

FR14	Earning Management
Description	The system shall provide a clear and detailed view of net-earning, total cancellation, and refund to Admin with graphical representation.
Stakeholder	Admin

FR15	Cancel Booking
Description	The system will allow customer to cancel their booking before defined deadline
Stakeholder	Customer

FR16	Payment
Description	The system supports multiple secure payment methods, and customers can select their preferred payment method during checkout.
Stakeholder	Customer

FR17	Refund
Description	After cancelling a booking, customers can request a refund. The system shall deduct a 2% processing free from the total paid amount.
Stakeholder	Customer

FR18	Contact Management
Description	The system shall allow user to submit quarries via a contact form and admin can view the quarries from admin dashboard
Stakeholder	Customer, Admin

FR19	Newsletter Management
Description	The customer or user can subscribe newsletter to get full further information via mail and admin can see the subscriber details
Stakeholder	Customer, Admin

FR20	Add Blog
Description	The system will allow the admin to add a blog from the admin dashboard
Stakeholder	Admin

FR21	Blog Management
Description	The system will allow the admin to update and delete blogs for further improvement
Stakeholder	Admin

FR22	User Management
Description	The system will allow admin to view user details with proper information with their loyalty coins
Stakeholder	Admin

FR23	Logout
Description	The system will allow the Admin, Receptionist, and user to securely logout, ending the active session to prevent unauthorized access.
Stakeholder	Admin, Receptionist, Customer

2.3 Non-Functional Requirements

Non-functional requirements define the quality attributes of a project, not just focusing on a specific function. They ensure software perform efferently, stays stable and user-friendly over all operations. For Roberto Hotel Management system focus on bon-functional requirements like performance, reliability, portable, usability, scalability, security and maintainability.

2.3.1 Performance

The system will remain stable during data retrieval and focus on responsiveness and low latency during operations like room booking and payment to make sure efficiency and accuracy. The average page loading time is less than 3 seconds on a stable broadband connection which is slandered.

Database quires and transaction are optimized using indexing and caching to reduce the processing time. The system can handle **100 users** at a time without lacking in performance.

Key performance goals:

- Secure and quick login and navigation for all user
- Efficient data storages with MYSQL
- Lightweight front-end using HTML, CSS and JavaScript
- Smooth transaction on process and payment with PHP and API integration.

2.3.2 Reliability

Reliability ensures the system will perform its function without facing any error or failure. The Roberto Hotel Management system is available 24/7 without a slight downtime. All transactions like booking, payment etc are process and stored in database.

Error handling mechanism are introduced to provide meaningful information about error when a process fail and automatically recover minor issue like network interruptions. Regular backup is enabled to avoiding any data losses from system failure.

Key Reliability goals:

- 1. Error alert system
- 2. Regular backup of critical data

- 3. Transactions operations are maintained by automation
- 4. System availability will 99%

2.3.3 Portability

Portability ensures that the system can run on multiple platforms without any specific modification. The “Roberto – Hotel & Resort” system are developed to run on Google Crome, Mozilla Firefox, Safari, Opera Mini and Microsoft Edge.

The system is deployed in PHP and MYSQL server and can be accessible from any devices with internet connection. This flexibility allows the system run over on any local or web server.

Key Portability goals:

- Run on any web browser
- Responsive design for all type of media devices
- Minimal setup required on a new server

2.3.4 **Usability**

The system is designed to be user-friendly, a just need a minimal tanning for administrative to managing operations. Every page follows the colour schema and navigations to make customer’s registration, login and booking without any external support. Also, a chatbot feature is introduced to provide help if customer face any issue or can get information directly.

Key Usability goals:

- Clear interface for all user
- Responsive design for all type of media devices
- Clear feedback for all sections

2.3.5 **Security**

The system ensures secure login and operations because the system handle customer's sensitive data. Input validation to prevent unwanted SQL injections and SSL (HTTPS) for secure connections.

The stripe payment gate provides a secure API to make a payment to ensure customer data confidentiality.

Key Security goals:

1. Password encryption with hash function
2. Input validation
3. Secure payment API to collect payment
4. Session's timeout and logout system

2.3.6 **Scalability**

Scalability ensure the system can handle increasing workload without lacking on performance. Optimized database stored all the date and system manage them for further execution.

1. Cloud ready structure for improvement
2. Can handle more than 10000 records in database
3. Easy implementations of new modules

2.3.7 **Maintainability**

Maintainability ensures how a system can be updated or modified easily. The project is developed with MVC (Model-View-Controller) to separate the presentation part from the operations logic. The system is well structured to adopting any new functionality without affecting other modules.

Key Maintainability goals:

1. Centralized and organized file structure

2. Commented code to understand it’s function

3. A well and simple document for installation and troubleshooting

2.4 Object-oriented System design using UML

2.4.1 Use Case Diagram

Figure 3: Use case Diagram

2.4.2 Case Description

Table: 2.4.2.1 : Case Description-01: **User Registration**

<table>
<tr><td>Use Case</td><td>Registration</td></tr>
<tr><td>Goal</td><td>Allow customer to create an account on system</td></tr>
<tr><td>Precondition</td><td>Must use the registration page</td></tr>
<tr><td>Success End Condition</td><td>Notification “Registration Successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Username or email already taken”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System database</td></tr>
<tr><td>Trigger</td><td>User clicks the Registration button from navigation</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Click on Registration button</td></tr>
<tr><td>2.</td><td>System will display registration form</td></tr>
<tr><td>3.</td><td>Customer fill-ups all required field</td></tr>
<tr><td>4.</td><td>Click Complete Registration button</td></tr>
<tr><td>5.</td><td>System validates all information</td></tr>
<tr><td>6.</td><td>System saves the data on database and
Notification “Registration Successful”</td></tr>
</table>
</td></tr>
<tr><td></td><td>
<table>
<tr><td>2.1</td><td>System Error</td></tr>
<tr><td></td><td>2.1.a Try again</td></tr>
<tr><td>3.1</td><td>Have empty Field</td></tr>
<tr><td></td><td>3.1.a Fill up all required field</td></tr>
<tr><td>5.1</td><td>Validation filed</td></tr>
<tr><td></td><td>5.1.a Try again</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>User will need to fill up the form in a short period of time and process should be fast</td></tr>
</table>

Table: 2.4.2.2 : Case Description-02: **User Login**

<table>
<tr><td>Use Case</td><td>Login</td></tr>
<tr><td>Goal</td><td>Allow Customer, admin and Receptionist to access the system</td></tr>
<tr><td>Precondition</td><td>Must have an account on the system</td></tr>
<tr><td>Success End Condition</td><td>Notification “Login Successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid credentials”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer, Admin, Receptionist

Authenticator system, System Database</td></tr>
<tr><td>Trigger</td><td>User clicks on the Login button from navigation</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Click on Login button</td></tr>
<tr><td>2.</td><td>System will display Login form</td></tr>
<tr><td>3.</td><td>Customer fill-ups username / email and password</td></tr>
<tr><td>4.</td><td>System validates login credentials</td></tr>
<tr><td>5.</td><td>System redirects to their dashboard</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>Error message</td></tr>
<tr><td></td><td>4.1.a Credential not matched</td></tr>
<tr><td>6.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>6.1.a Login failed</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Login validation should be secure, and system must handle the incorrect credentials</td></tr>
</table>

Table: 2.4.2.3 : Case Description-03: **Update information**

<table>
<tr><td>Use Case</td><td>Update Information</td></tr>
<tr><td>Goal</td><td>Allow Customer to update their information including password</td></tr>
<tr><td>Precondition</td><td>Must be login in system</td></tr>
<tr><td>Success End Condition</td><td>Notification “Profile updated successfully”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid format”</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Customer
Authenticator system, System Database</td></tr>
<tr><td>Trigger</td><td>User clicks on the “Update Profile” Button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Customer Navigates to My Account page</td></tr>
<tr><td>2.</td><td>System will display all available profile data</td></tr>
<tr><td>3.</td><td>Customer modifies fields</td></tr>
<tr><td>4.</td><td>System checks all input filed</td></tr>
<tr><td>5.</td><td>System saves the data on database</td></tr>
<tr><td>6.</td><td>Notification “Profile updated successfully”</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>Empty field</td></tr>
<tr><td></td><td>4.1.a Fill up all required filed</td></tr>
<tr><td>5.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>5.1.a Update failed</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Input format should be validated correctly</td></tr>
</table>

Table: 2.4.2.4 : Case Description-04: **Chatbot**

<table>
<tr><td>Use Case</td><td>Chatbot</td></tr>
<tr><td>Goal</td><td>Allowing customer to get assistance</td></tr>
<tr><td>Precondition</td><td>Must visit website any page</td></tr>
<tr><td>Success End Condition</td><td>Chatbot assistance user to get direction or information</td></tr>
<tr><td>Failed End Condition</td><td>Chatbot failed to load</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System, Admin</td></tr>
<tr><td>Trigger</td><td>Customers click Chatbot icon from website</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Click Chatbot button</td></tr>
<tr><td>2.</td><td>Chatbot window open</td></tr>
<tr><td>3.</td><td>Chatbot display welcome message</td></tr>
<tr><td>4.</td><td>Customer type question or select option</td></tr>
<tr><td>5.</td><td>System processes the message</td></tr>
<tr><td>6.</td><td>Chatbot display appropriate answer</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>5.1</td><td>Server down</td></tr>
<tr><td></td><td>5.1.a Try again</td></tr>
<tr><td>6.1</td><td>System error</td></tr>
<tr><td></td><td>6.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Response must be immediate, and message will be clear</td></tr>
</table>

Table: 2.4.2.5 : Case Description-05: **Smart Room Recommendation**

<table>
<tr><td>Use Case</td><td>Smart Room Recommendation</td></tr>
<tr><td>Goal</td><td>Allowing customer to get assistance</td></tr>
<tr><td>Precondition</td><td>Must visit website any page</td></tr>
<tr><td>Success End Condition</td><td>Customer must be on website (Recommendation Page)</td></tr>
<tr><td>Failed End Condition</td><td>No recommendations are provided or system display error</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System</td></tr>
<tr><td>Trigger</td><td>Customers submit room preference information</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Navigates to Smart Room Recommendation</td></tr>
<tr><td>2.</td><td>System displays a preference form</td></tr>
<tr><td>3.</td><td>Customers enter preference information</td></tr>
<tr><td>4.</td><td>Then click Get Recommendation</td></tr>
<tr><td>5.</td><td>System analyzes input</td></tr>
<tr><td>6.</td><td>System generates a list of recommendation</td></tr>
<tr><td>7.</td><td>Recommended room are displayed</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>2.1</td><td>System error</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>Missing Input</td></tr>
<tr><td></td><td>4.1.a Please enter valid information</td></tr>
<tr><td>6.1</td><td>Recommendation fails</td></tr>
<tr><td></td><td>6.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Recommendation must be accurate</td></tr>
</table>

Table: 2.4.2.6 : Case Description-06: **Add Room**

<table>
<tr><td>Use Case</td><td>Add Room</td></tr>
<tr><td>Goal</td><td>Allow Admin to add new room</td></tr>
<tr><td>Precondition</td><td>Must be login on admin account</td></tr>
<tr><td>Success End Condition</td><td>Notification “Room added successfully”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid request method”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System Database</td></tr>
<tr><td>Trigger</td><td>Click on the Add Room button</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Click on Add Room section from admin dashboard</td></tr>
<tr><td>2.</td><td>System will display a room form</td></tr>
<tr><td>3.</td><td>Admin fill-ups all required filed with images</td></tr>
<tr><td>4.</td><td>Click on Add Room button</td></tr>
<tr><td>5.</td><td>System validates all filed</td></tr>
<tr><td>6.</td><td>Notification “Room Added Successfully”</td></tr>
<tr><td>7.</td><td>Automatically goes to Room Management section</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Try Again</td></tr>
<tr><td>5.1</td><td>Error message “Empty Filed”</td></tr>
<tr><td></td><td>5.1.a. Show all required filed</td></tr>
<tr><td>6.1</td><td>The system does not respond</td></tr>
<tr><td></td><td>6.1.a. Error message “Room cannot be added”</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>All fields must be validated, and system must be avoided to add duplicate rooms</td></tr>
</table>

Table: 2.4.2.7 : Case Description-07: **Update Room**

<table>
<tr><td>Use Case</td><td>Update Room</td></tr>
<tr><td>Goal</td><td>Allow Admin to edit existing room</td></tr>
<tr><td>Precondition</td><td>Must be login as</td></tr>
<tr><td>Success End Condition</td><td>Notification “Room updated successfully”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid request method”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System Database</td></tr>
<tr><td>Trigger</td><td>Click on the Edit button from Room Management sections</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Click on Room Management section from admin dashboard</td></tr>
<tr><td>2.</td><td>System will display all available rooms with all information's</td></tr>
<tr><td>3.</td><td>Click on the edit option</td></tr>
<tr><td>4.</td><td>System will display a popup with an editable form</td></tr>
<tr><td>5.</td><td>Admin modifies the room information</td></tr>
<tr><td>6.</td><td>Click on the Update Room button</td></tr>
<tr><td>7.</td><td>System confirms the update with “Room Updated Successfully” message</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>2.1</td><td>System Error</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>4.1.a. Try again</td></tr>
<tr><td>5.1</td><td>Empty field</td></tr>
<tr><td></td><td>5.1.a Required filed</td></tr>
<tr><td>7.1</td><td>System does not respond</td></tr>
<tr><td></td><td>7.1.a. Try again</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>Edit / delete must update the room immediately and interface must refresh without any errors</td></tr>
</table>

Table: 2.4.2.8: Case Description-08: **Delete Room**

<table>
<tr><td>Use Case</td><td>Delete Room</td></tr>
<tr><td>Goal</td><td>Allowing admin to delete room from the system</td></tr>
<tr><td>Precondition</td><td>Must have admin login and have one room in the system</td></tr>
<tr><td>Success End Condition</td><td>Room deleted Successfully</td></tr>
<tr><td>Failed End Condition</td><td>Error in room delete process</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System</td></tr>
<tr><td>Trigger</td><td>Admin click Room Management from the dashboard</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Login as admin</td></tr>
<tr><td>2.</td><td>Navigates to Room Management section</td></tr>
<tr><td>3.</td><td>System shows all available room list</td></tr>
<tr><td>4.</td><td>Click on a Delete</td></tr>
<tr><td>5.</td><td>Confirm on alert</td></tr>
<tr><td>6.</td><td>Contact Deleted</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>6.1</td><td>System error</td></tr>
<tr><td></td><td>6.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Room must be deleted immediately and updated to the system list</td></tr>
</table>

Table: 2.4.2.9: Case Description-09: **Add to Cart**

<table>
<tr><td>Use Case</td><td>Add to Cart</td></tr>
<tr><td>Goal</td><td>Allow customer to add and remove rooms before booking</td></tr>
<tr><td>Precondition</td><td>User must be logged-in as a customer</td></tr>
<tr><td>Success End Condition</td><td>Message “Room added successfully to cart”</td></tr>
<tr><td>Failed End Condition</td><td>Message “Sorry, encountered an Error, please try again”</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Customer
System</td></tr>
<tr><td>Trigger</td><td>Click on Add to Cart button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Visit a room single page</td></tr>
<tr><td>2.</td><td>Customer enters check-in and check-out date</td></tr>
<tr><td>3.</td><td>System will validate check-in and check-out date</td></tr>
<tr><td>4.</td><td>Click Add to Cart Button</td></tr>
<tr><td>5.</td><td>System adds the room on cart</td></tr>
<tr><td>6.</td><td>System shows a message “Room added Successfully” with View cart button</td></tr>
<tr><td>7.</td><td>Click on View cart button to go on cart page</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>Room not available</td></tr>
<tr><td></td><td>3.1.a. Select another date</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a Msg “Try gain “</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Cart will be expired in 30 minute to make the room available for other customer</td></tr>
</table>

Table: 2.4.2.10 : Case Description-10: **Make Booking**

<table>
<tr><td>Use Case</td><td>Make Booking</td></tr>
<tr><td>Goal</td><td>Allow customer to book a room</td></tr>
<tr><td>Precondition</td><td>Must be login on customer account and have rooms on cart</td></tr>
<tr><td>Success End Condition</td><td>Redirect to booking success page</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Booking Process Failed”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System</td></tr>
<tr><td>Trigger</td><td>Click on the Add Room button</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Navigates to cart page</td></tr>
<tr><td>2.</td><td>Click on the “Process to Checkout” button</td></tr>
<tr><td>3.</td><td>Customer redirected to checkout page</td></tr>
<tr><td>4.</td><td>Customer enters all necessary booking information</td></tr>
<tr><td>5.</td><td>Customer can add addition service</td></tr>
<tr><td>6.</td><td>Customer can use coin to get discount</td></tr>
<tr><td>7.</td><td>System validates all information</td></tr>
<tr><td>8.</td><td>Customer completes a payments</td></tr>
<tr><td>9.</td><td>System confirms the booking and redirects to Booking Successful page</td></tr>
<tr><td>10.</td><td>System add or subtract loyalty coin from customer</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>2.1</td><td>System does not respond</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>7.1</td><td>Error message “Empty Filed”</td></tr>
<tr><td></td><td>7.1.a. Show all required filed</td></tr>
<tr><td>9.1</td><td>The system does not respond</td></tr>
<tr><td></td><td>9.1.a. Try again</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>User will need to checkout to confirm the booking before expiring the cart. Booking confirmation should appear and no double booking occur</td></tr>
</table>

Table: 2.4.2.11: Case Description-11: **Make Payment**

<table>
<tr><td>Use Case</td><td>Make Payment</td></tr>
<tr><td>Goal</td><td>Allow user to make a payment to confirm the booking</td></tr>
<tr><td>Precondition</td><td>Must be login on admin account</td></tr>
<tr><td>Success End Condition</td><td>Notification “Payment Successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Payment processing error”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

Payment Gateway,</td></tr>
<tr><td>Trigger</td><td>Click on the Add Room button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Visit checkout page</td></tr>
<tr><td>2.</td><td>Customers select the payment method</td></tr>
<tr><td>3.</td><td>Customer enters the payment information</td></tr>
<tr><td>4.</td><td>Payment gateways validate the payment</td></tr>
<tr><td>5.</td><td>System confirms the payment</td></tr>
<tr><td>6.</td><td>System show notification “Payment successful”</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a. Try Again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Transaction process needs to be accurate and immediate</td></tr>
</table>

Table: 2.4.2.12: Case Description-12: **Loyalty Reward**

<table>
<tr><td>Use Case</td><td>Loyalty Reward</td></tr>
<tr><td>Goal</td><td>Allow customer to earn reward coin based on their payment</td></tr>
<tr><td>Precondition</td><td>Must be complete a booking with online payment</td></tr>
<tr><td>Success End Condition</td><td>Notification “Reward coin added”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Booking not complete”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

Loyalty Program</td></tr>
<tr><td>Trigger</td><td>Complete a successful paid booing</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Complete a paid booking</td></tr>
<tr><td>2.</td><td>System calculates loyalty point</td></tr>
<tr><td>3.</td><td>System update customer reward coin balance</td></tr>
<tr><td>4.</td><td>Go to the booking successful page</td></tr>
<tr><td>5.</td><td>System displays the reward point balance</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>Payment process failed</td></tr>
<tr><td></td><td>1.1.a. Try Again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Reward Coin must be calculated accurately and should add to the customer account</td></tr>
</table>

Table: 2.4.2.13: Case Description-13: **Loyalty Discount**

<table>
<tr><td>Use Case</td><td>Loyalty Discount</td></tr>
<tr><td>Goal</td><td>Allow customer to redeem their coin to get discount</td></tr>
<tr><td>Precondition</td><td>Must be complete a booking with online payment</td></tr>
<tr><td>Success End Condition</td><td>Discount amounts subtract from the total payable amount</td></tr>
<tr><td>Failed End Condition</td><td>Discount not added to payable amount</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

Loyalty Program</td></tr>
<tr><td>Trigger</td><td>Customer use coin during checkout</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Go to checkout page</td></tr>
<tr><td>2.</td><td>System displays all available coin</td></tr>
<tr><td>3.</td><td>Customers click “Use Max”</td></tr>
<tr><td>4.</td><td>System calculates the discount</td></tr>
<tr><td>5.</td><td>System updates total amount by subtracting discount amount</td></tr>
<tr><td>6.</td><td>Make a successful payment</td></tr>
<tr><td>7.</td><td>System deducts redeem coin from user account</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>Invalid coins</td></tr>
<tr><td></td><td>3.1.a. Input number of points</td></tr>
<tr><td>4.1</td><td>Calculation error</td></tr>
<tr><td></td><td>4.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Discount must apply immediately and system must check the coin balance</td></tr>
</table>

Table: 2.4.2.14: Case Description-14: **Cancel Booking**

<table>
<tr><td>Use Case</td><td>Cancel Booking</td></tr>
<tr><td>Goal</td><td>Allow user to make a booking cancelled</td></tr>
<tr><td>Precondition</td><td>Must have a confirmed booking</td></tr>
<tr><td>Success End Condition</td><td>Notification “Booking cancellation successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Cannot cancel the booking”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System database</td></tr>
<tr><td>Trigger</td><td>Click on the Add Room button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Navigate booking tab from my account page</td></tr>
<tr><td>2.</td><td>System will show all booking</td></tr>
<tr><td>3.</td><td>Customers click on the cancel booking option</td></tr>
<tr><td>4.</td><td>System will show an alert for confirmation</td></tr>
<tr><td>5.</td><td>System confirms the cancellation</td></tr>
<tr><td>6.</td><td>System show notification “Booking Cancellation Successful”</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>2.1</td><td>System error</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>System error</td></tr>
<tr><td></td><td>4.1.a Booing not found</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>System needs to mark the booking as cancelled immediately</td></tr>
</table>

Table: 2.4.2.15: Case Description-15: **Get Refund**

<table>
<tr><td>Use Case</td><td>Get Refund</td></tr>
<tr><td>Goal</td><td>Allow user to process refund</td></tr>
<tr><td>Precondition</td><td>User must have a cancelled booking</td></tr>
<tr><td>Success End Condition</td><td>Notification “Refund Successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Refund request cannot process”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer

System database, Payment gateway</td></tr>
<tr><td>Trigger</td><td>Click on the Add Room button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Navigate Booking tab from My Account page</td></tr>
<tr><td>2.</td><td>System will show all booking</td></tr>
<tr><td>3.</td><td>Customers click on the Refund button</td></tr>
<tr><td>4.</td><td>System will show a popup with refund and calculate refund amount</td></tr>
<tr><td>5.</td><td>Customer click Request Refund button</td></tr>
<tr><td>6.</td><td>System validates refunds request</td></tr>
<tr><td>7.</td><td>System processes the refund</td></tr>
<tr><td>8.</td><td>System update customer about refund</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>No refund option available</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>6.1</td><td>System error</td></tr>
<tr><td></td><td>4.1.a Booing not found</td></tr>
<tr><td>8.1</td><td>Payment process failed</td></tr>
<tr><td></td><td>8.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Refund process should be immediate and accurate</td></tr>
</table>

Table: 2.4.2.16: Case Description-16: **Make Check-in**

<table>
<tr><td>Use Case</td><td>Make Check-in</td></tr>
<tr><td>Goal</td><td>To verify customer booking and complete the check-in process to access the room</td></tr>
<tr><td>Precondition</td><td>Customer must have a valid paid booking</td></tr>
<tr><td>Success End Condition</td><td>Guest Check-in successfully</td></tr>
<tr><td>Failed End Condition</td><td>Check-in does not complete and room remains available</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Receptionist
Customer, Booking System</td></tr>
<tr><td>Trigger</td><td>Customer arrives in hotel and receptionist selects Check-in from system</td></tr>
<tr><td>Description / Main Success Scenario</td><td>
<table>
<tr><td>1.</td><td>Receptionist go to dashboard</td></tr>
<tr><td>2.</td><td>Navigates Check-in sections</td></tr>
<tr><td>3.</td><td>Show available Check-in</td></tr>
<tr><td>4.</td><td>Click on Check-in button</td></tr>
<tr><td>5.</td><td>Confirm check-in on alert</td></tr>
<tr><td>6.</td><td>Marked the booking as checked-in</td></tr>
<tr><td>7.</td><td>Notification Check-in successful</td></tr>
</table>
</td></tr>
<tr><td>Alternative Flows</td><td>
<table>
<tr><td>3.1</td><td>System Error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a. Try again</td></tr>
</table>
</td></tr>
<tr><td>Quality Requirements</td><td>System must be validated check-in and marked booking as checked-in immediately all over the system</td></tr>
</table>

Table: 2.4.2.17: Case Description-17: **Make Check-out**

<table>
<tr><td>Use Case</td><td>Make Check-out</td></tr>
<tr><td>Goal</td><td>Allow receptionist to complete the check-in process to access the room</td></tr>
<tr><td>Precondition</td><td>Customer must have a valid paid booking</td></tr>
<tr><td>Success End Condition</td><td>Customer is Checked-out successfully and marked status to checked-out</td></tr>
<tr><td>Failed End Condition</td><td>Customer remains not checked out</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Receptionist

Customer, Booking System</td></tr>
<tr><td>Trigger</td><td>Customer request to check-out and receptionist select check-out option from system</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Receptionist go to dashboard</td></tr>
<tr><td>2.</td><td>Navigates Check-out sections</td></tr>
<tr><td>3.</td><td>Show available Check-out list</td></tr>
<tr><td>4.</td><td>Click on Check-out button</td></tr>
<tr><td>5.</td><td>Confirm check-out on alert</td></tr>
<tr><td>6.</td><td>Marked the booking as checked out</td></tr>
<tr><td>7.</td><td>Notification Check-out successful</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System Error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>5.1</td><td>System Error</td></tr>
<tr><td></td><td>5.1.a. Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Booking status marked as checked out immediately all over the system</td></tr>
</table>

Table: 2.4.2.18: Case Description-18: **Add Blog**

<table>
<tr><td>Use Case</td><td>Add Blog</td></tr>
<tr><td>Goal</td><td>Allow Admin to add new blog</td></tr>
<tr><td>Precondition</td><td>Must be login on admin account</td></tr>
<tr><td>Success End Condition</td><td>Notification “Blog added successfully”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid request method”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System Database</td></tr>
<tr><td>Trigger</td><td>Click on the Add Blog button</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Click on Add Blog section from admin dashboard</td></tr>
<tr><td>2.</td><td>System will display a blog form</td></tr>
<tr><td>3.</td><td>Admin fill-ups all required filed with images</td></tr>
<tr><td>4.</td><td>Click on Add blog button</td></tr>
<tr><td>5.</td><td>System validates all filed</td></tr>
<tr><td>6.</td><td>Notification “Blog Added Successfully”</td></tr>
<tr><td>7.</td><td>Automatically goes to Blog Management section</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>1.1</td><td>System Error</td></tr>
<tr><td></td><td>1.1.a. Try Again</td></tr>
<tr><td>5.1</td><td>Error message “Missing Filed”</td></tr>
<tr><td></td><td>5.1.a. Show all required filed</td></tr>
<tr><td>6.1</td><td>The system does not respond</td></tr>
<tr><td></td><td>6.1.a. Error message “Blog cannot be added”</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>The blog field must be validated correctly, and post should be published immediately.</td></tr>
</table>

Table: 2.4.2.1: Case Description-19: **Update Blog**

<table>
<tr><td>Use Case</td><td>Update Blog</td></tr>
<tr><td>Goal</td><td>Allow Admin to edit existing blog</td></tr>
<tr><td>Precondition</td><td>Must be login as admin</td></tr>
<tr><td>Success End Condition</td><td>Notification “Blog updated successfully”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “Invalid request method”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System Database</td></tr>
<tr><td>Trigger</td><td>Click on the Edit button from Blog Management sections</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Click on Blog Management section from admin dashboard</td></tr>
<tr><td>2.</td><td>System will display all available blogs with all information's</td></tr>
<tr><td>3.</td><td>Click on the edit option</td></tr>
<tr><td>4.</td><td>System will display a popup with an editable form</td></tr>
<tr><td>5.</td><td>Admin modifies the blog information</td></tr>
<tr><td>6.</td><td>Click on the Update blog button</td></tr>
<tr><td>7.</td><td>System confirms the update “Blog Updated successfully” with message</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>2.1</td><td>System Error</td></tr>
<tr><td></td><td>2.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>The system did not respond</td></tr>
<tr><td></td><td>4.1.a. Try again</td></tr>
<tr><td>5.1</td><td>Empty field</td></tr>
<tr><td></td><td>5.1.a Required field</td></tr>
<tr><td>7.1</td><td>System does not respond</td></tr>
<tr><td></td><td>7.1.a. Error in update</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Blog edit function should be working smoothly and updated immediately.</td></tr>
</table>

Table: 2.4.2.20: Case Description-20: **Delete Blog**

<table>
<tr><td>Use Case</td><td>Delete Blog</td></tr>
<tr><td>Goal</td><td>Allowing admin to delete the existing blog</td></tr>
<tr><td>Precondition</td><td>Must have admin login and have blog on the system</td></tr>
<tr><td>Success End Condition</td><td>Blog deleted successfully</td></tr>
<tr><td>Failed End Condition</td><td>Error in process</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System</td></tr>
<tr><td>Trigger</td><td>Admin click Blog Management from the dashboard</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Login as admin</td></tr>
<tr><td>2.</td><td>Navigates to Blog Management</td></tr>
<tr><td>3.</td><td>System shows all blog in list</td></tr>
<tr><td>4.</td><td>Click on a Delete</td></tr>
<tr><td>5.</td><td>Confirm on alert</td></tr>
<tr><td>6.</td><td>Contact Deleted</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>6.1</td><td>System error</td></tr>
<tr><td></td><td>6.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Blog should delete from system and update the blog list</td></tr>
</table>

Table: 2.4.2.21: Case Description-21: **View Booking**

<table>
<tr><td>Use Case</td><td>View Booking</td></tr>
<tr><td>Goal</td><td>Allowing admin to view all available booking and their status</td></tr>
<tr><td>Precondition</td><td>Admin must be login and minimum 1 booking exist in the system</td></tr>
<tr><td>Success End Condition</td><td>Show all available booking</td></tr>
<tr><td>Failed End Condition</td><td>Bookings are showing incorrectly</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System, Customer</td></tr>
<tr><td>Trigger</td><td>User clicks on the booking menu from navigation</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table><tr><td>1.</td><td>Login as admin</td></tr><tr><td>2.</td><td>Navigates to booking section</td></tr><tr><td>3.</td><td>System shows all available booking with status</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>1.1</td><td>Redirection failed</td></tr><tr><td></td><td>1.1.a. Try Again</td></tr><tr><td>2.1</td><td>System error</td></tr><tr><td></td><td>2.1.a Try again</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>Booking filter must work smoothly</td></tr>
</table>

Table: 2.4.2.22 : Case Description-22: **View Earning**

<table>
<tr><td>Use Case</td><td>View Earning</td></tr>
<tr><td>Goal</td><td>Allowing admin to view all earning, cancellation, net earning with graphical view</td></tr>
<tr><td>Precondition</td><td>Must have admin login and one confirm booking need to exist in the system</td></tr>
<tr><td>Success End Condition</td><td>Graphs are loading smoothly</td></tr>
<tr><td>Failed End Condition</td><td>Earning graphs failed to load or conflicting between calculating graph value</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Admin

System</td></tr>
<tr><td>Trigger</td><td>Admin click Earning from the dashboard</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Login as admin</td></tr>
<tr><td>2.</td><td>Navigates to Earning section</td></tr>
<tr><td>3.</td><td>System shows all available earning data with graph</td></tr>
<tr><td>4.</td><td>Click on a specific graph point</td></tr>
<tr><td>5.</td><td>The system displays that point display details in tooltip</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>Graph fails to load</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>System error</td></tr>
<tr><td></td><td>4.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Graph must be load smoothly and calculation must be accurates</td></tr>
</table>

Table: 2.4.2.23: Case Description-23: **Resend Email**

<table>
<tr><td>Use Case</td><td>Resend Email</td></tr>
<tr><td>Goal</td><td>Allowing admin to send email to reply contact message</td></tr>
<tr><td>Precondition</td><td>Must have admin login and have one contact submission</td></tr>
<tr><td>Success End Condition</td><td>Email send link working fine</td></tr>
<tr><td>Failed End Condition</td><td>Email addresses are not valid or fond.
Email processing error</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System</td></tr>
<tr><td>Trigger</td><td>Admin click Contact Management from the dashboard</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Login as admin</td></tr>
<tr><td>2.</td><td>Navigates to Contact Management section</td></tr>
<tr><td>3.</td><td>System shows all available contact data</td></tr>
<tr><td>4.</td><td>Click on a Send email</td></tr>
<tr><td>5.</td><td>Reply email sent</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>4.1</td><td>System error</td></tr>
<tr><td></td><td>4.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Email sending must be reliable and fast and contact massage load correctly</td></tr>
</table>

Table: 2.4.2.24: Case Description-24: **Delete Contact**

<table>
<tr><td>Use Case</td><td>Delete Contact</td></tr>
<tr><td>Goal</td><td>Allowing admin to delete the contact messages</td></tr>
<tr><td>Precondition</td><td>Must have admin login and have one contact submission</td></tr>
<tr><td>Success End Condition</td><td>Contact deleted Successfully</td></tr>
<tr><td>Failed End Condition</td><td>Contact not deleted</td></tr>
<tr><td>Primary Actors:
Secondary Actors:</td><td>Admin
System</td></tr>
<tr><td>Trigger</td><td>Admin click Contact Management from the dashboard</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table>
<tr><td>1.</td><td>Login as admin</td></tr>
<tr><td>2.</td><td>Navigates to Contact Management section</td></tr>
<tr><td>3.</td><td>System shows all available contact data</td></tr>
<tr><td>4.</td><td>Click on a Delete</td></tr>
<tr><td>5.</td><td>Confirm on alert</td></tr>
<tr><td>6.</td><td>Contact Deleted</td></tr>
</table></td></tr>
<tr><td>Alternative Flows</td><td><table>
<tr><td>3.1</td><td>System error</td></tr>
<tr><td></td><td>3.1.a. Try Again</td></tr>
<tr><td>6.1</td><td>System error</td></tr>
<tr><td></td><td>6.1.a Try again</td></tr>
</table></td></tr>
<tr><td>Quality Requirements</td><td>Contact should delete from system and disappear from the contact list</td></tr>
</table>

Table: 2.4.2.25: Case Description-25: **Logout**

<table>
<tr><td>Use Case</td><td>Logout</td></tr>
<tr><td>Goal</td><td>Allow Customer, admin and Receptionist to securely logout</td></tr>
<tr><td>Precondition</td><td>Must have an account on the system</td></tr>
<tr><td>Success End Condition</td><td>Notification “Logout Successful”</td></tr>
<tr><td>Failed End Condition</td><td>Notification “System Error”</td></tr>
<tr><td>Primary Actors:

Secondary Actors:</td><td>Customer, Admin, Receptionist

Authenticator system, System Database</td></tr>
<tr><td>Trigger</td><td>User clicks on the Logout button from navigation</td></tr>
<tr><td>Description / Main Success Scenario</td><td><table><tr><td>1.</td><td>Click on Logout button</td></tr><tr><td>2.</td><td>System validates logout credentials</td></tr><tr><td>3.</td><td>Logout Successful</td></tr><tr><td>4.</td><td>System redirects login page</td></tr></table></td></tr>
<tr><td>Alternative Flows</td><td><table><tr><td>2.1</td><td>System Error</td></tr><tr><td></td><td>2.1.a. Try Again</td></tr></table></td></tr>
<tr><td>Quality Requirements</td><td>User sessions data should be cleared to prevent unauthorized access</td></tr>
</table>

2.4.3 Activity Diagram

For Registration:

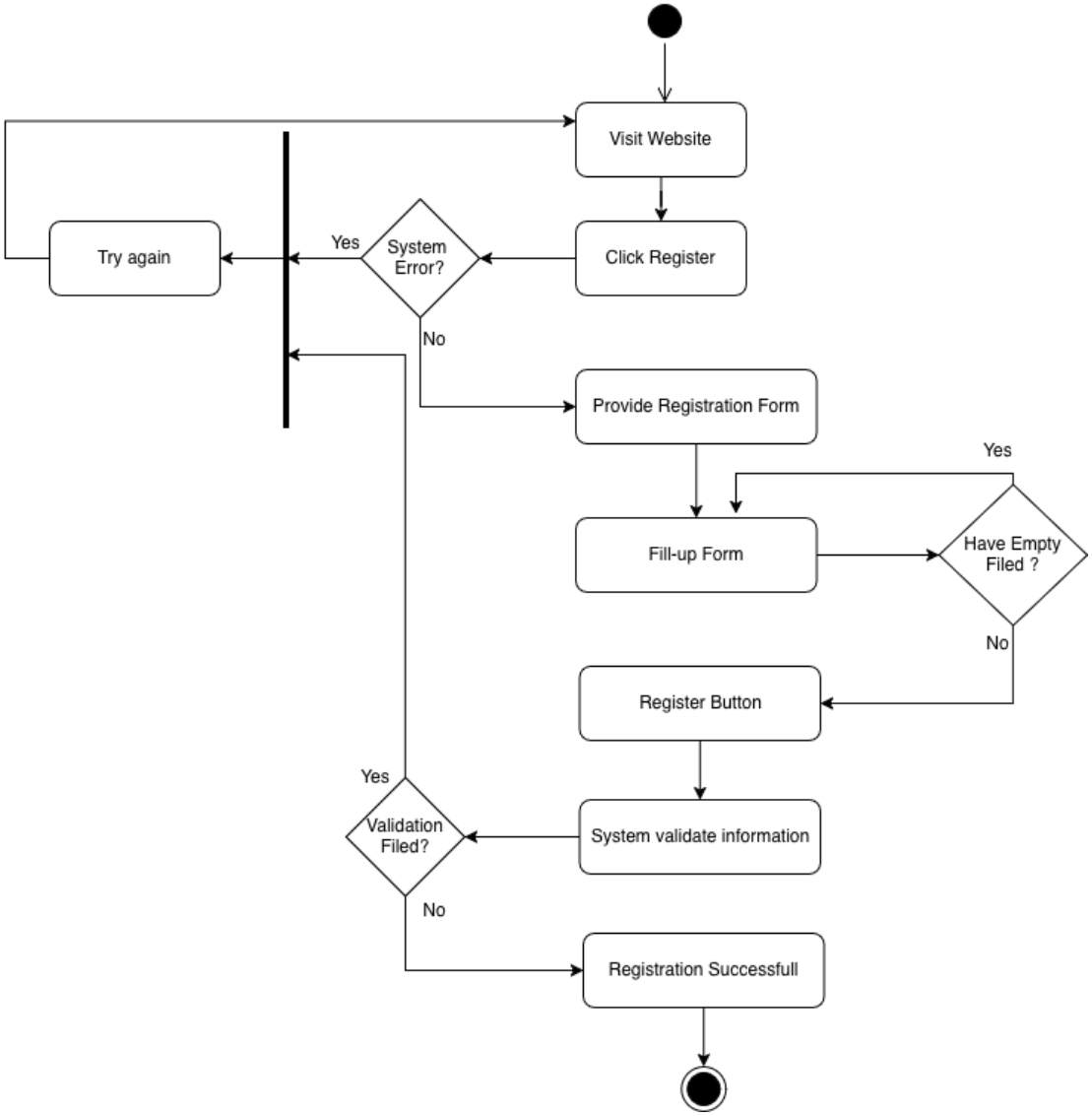


Figure 3.1: Activity diagram for Registration

For Login:

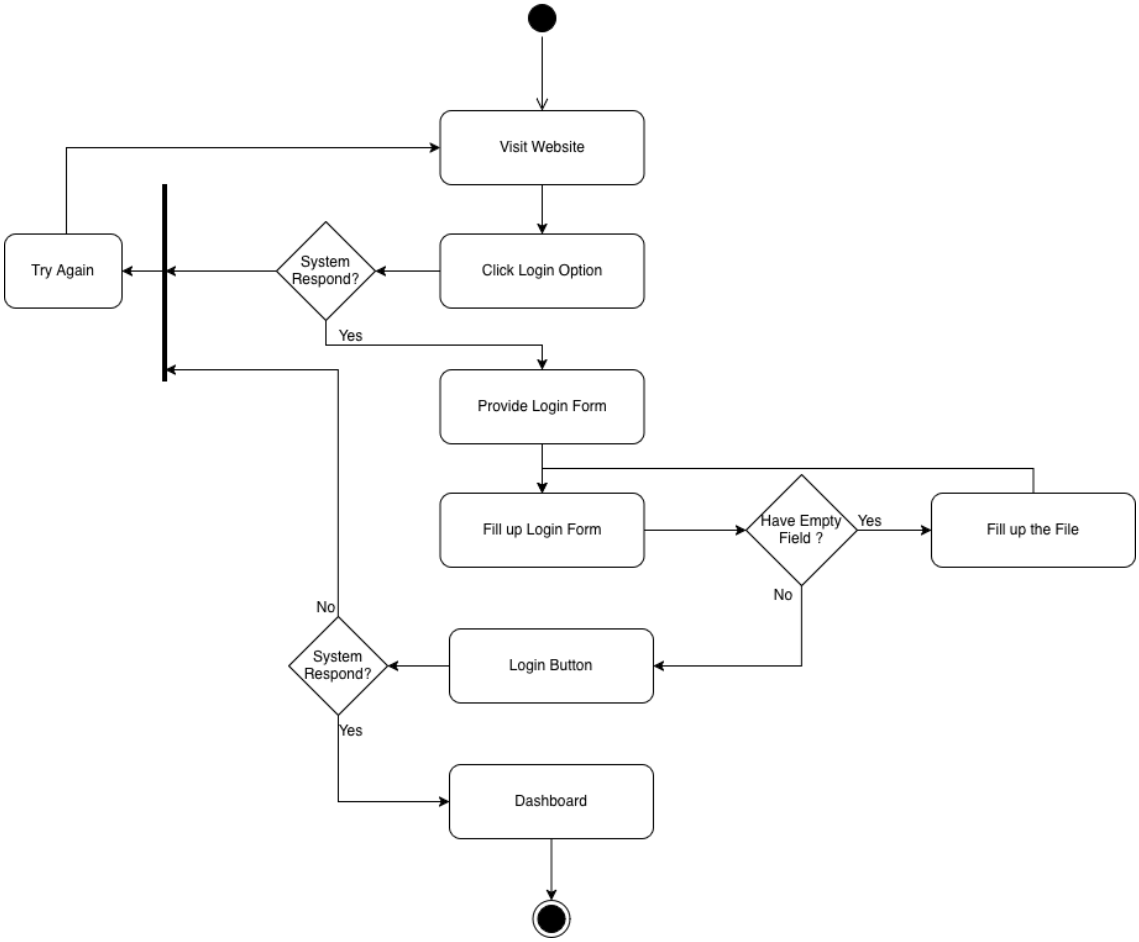


Figure 3.2: Activity diagram for Login

For Update Profile Information:

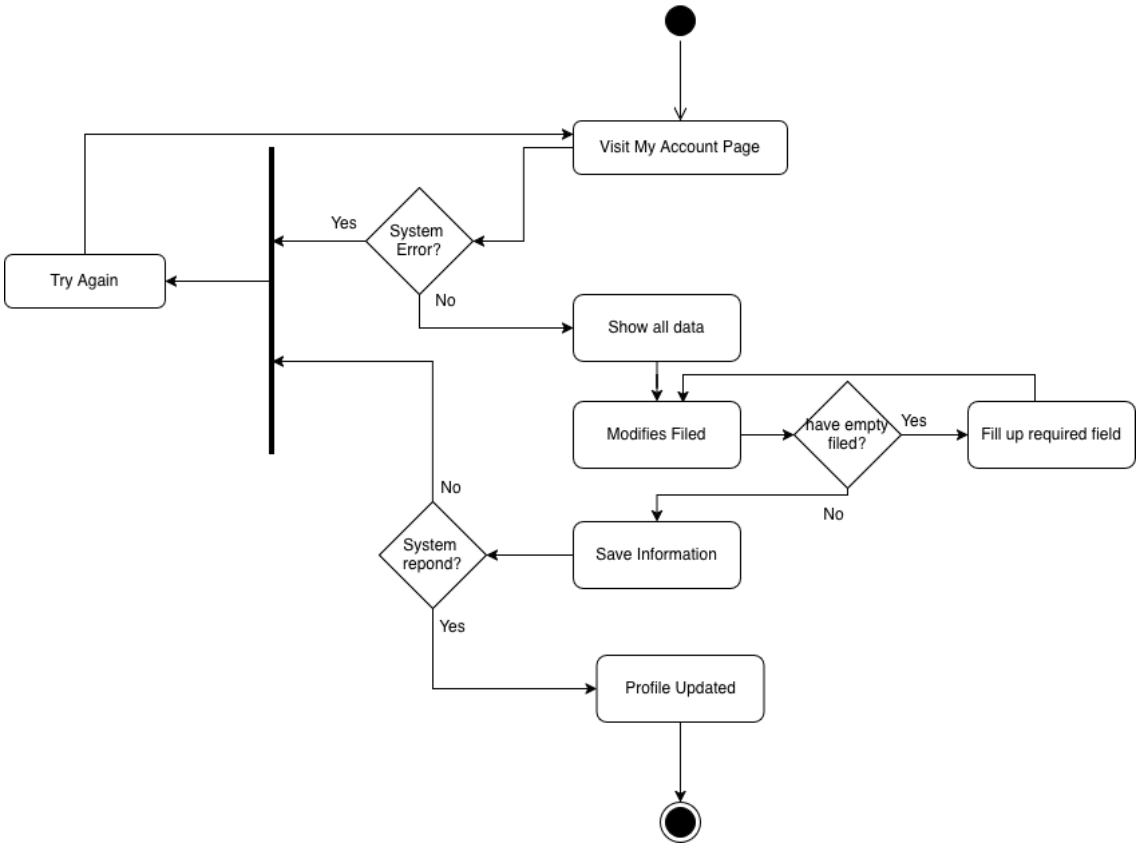


Figure 3.3: Activity diagram for Update Profile Information

For Chatbot

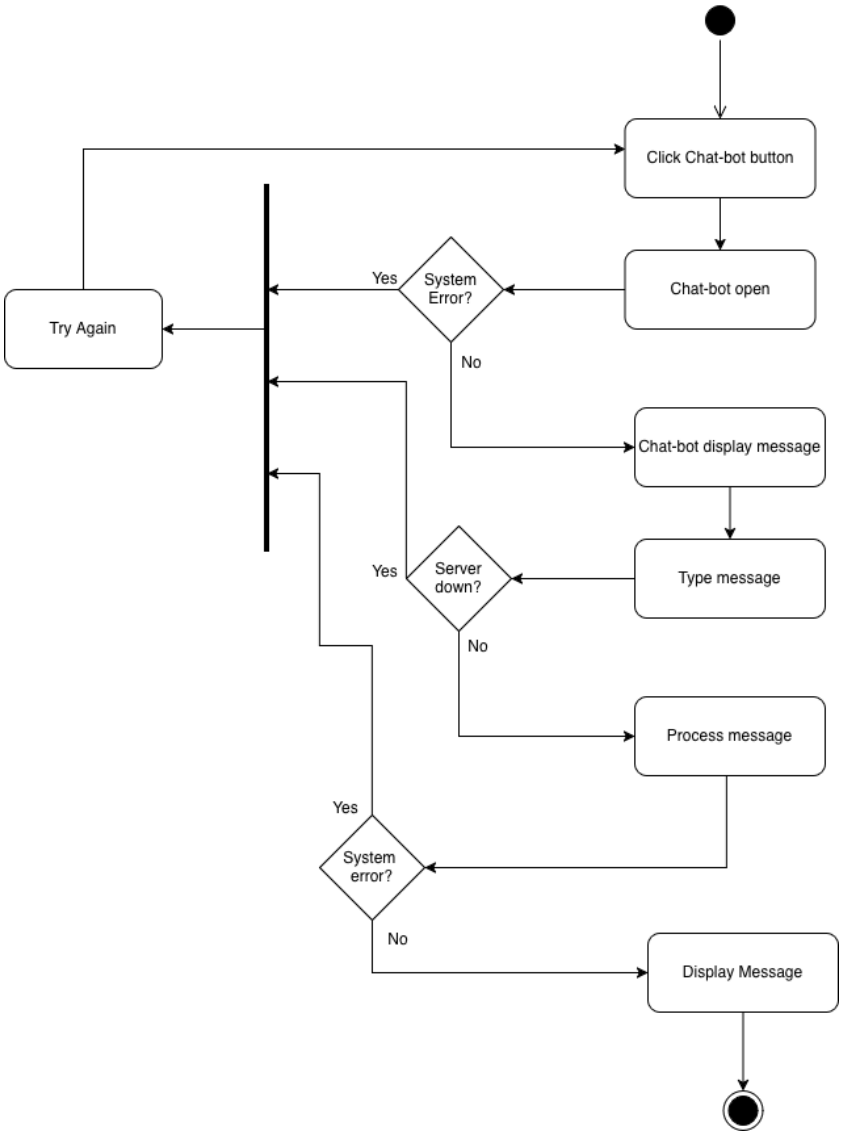


Figure 3.4: Activity diagram for Chatbot

For Smart Room Recommendation:

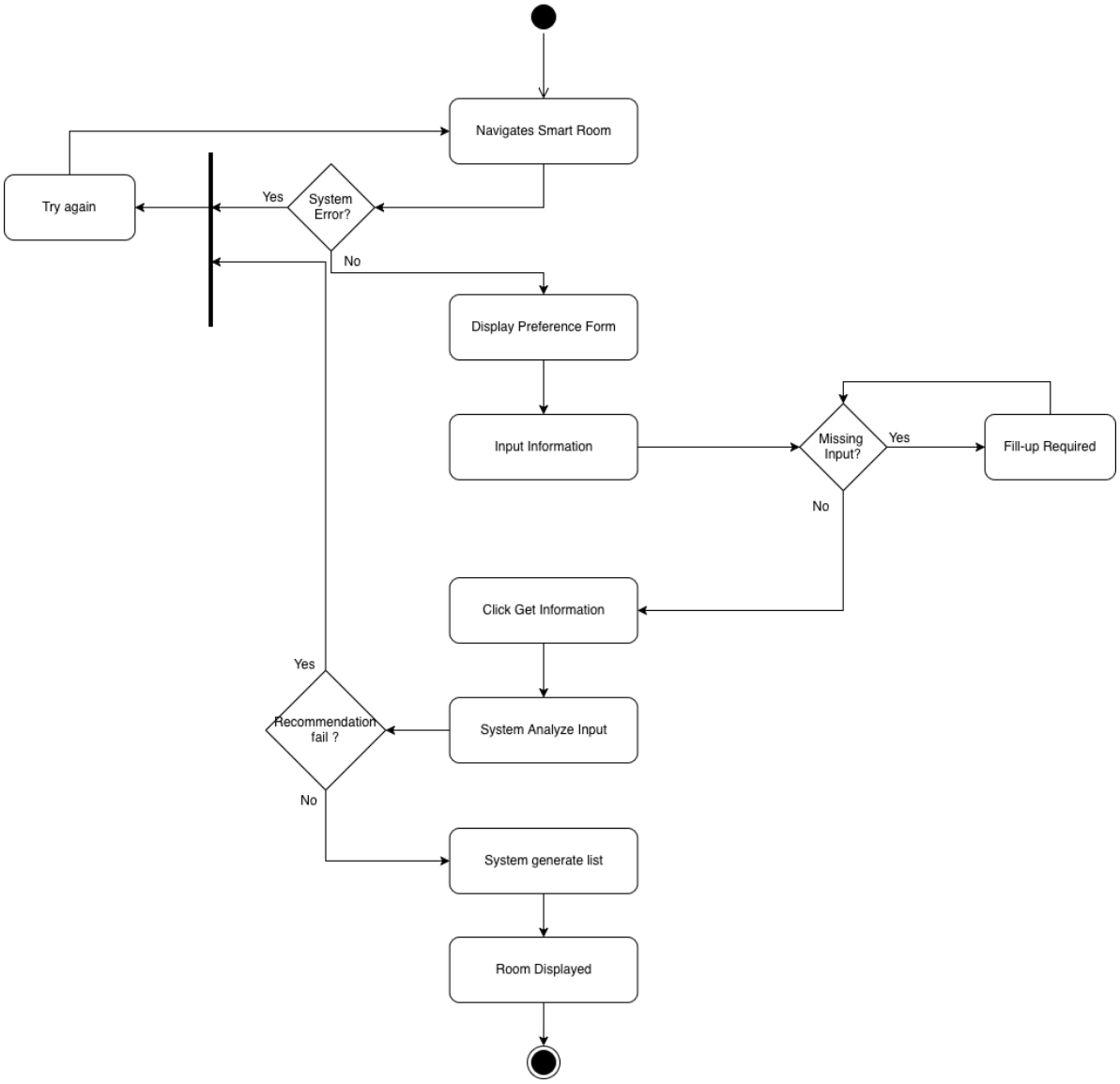


Figure 3.5: Activity diagram for Smart Room Recommendation

For Add Room:

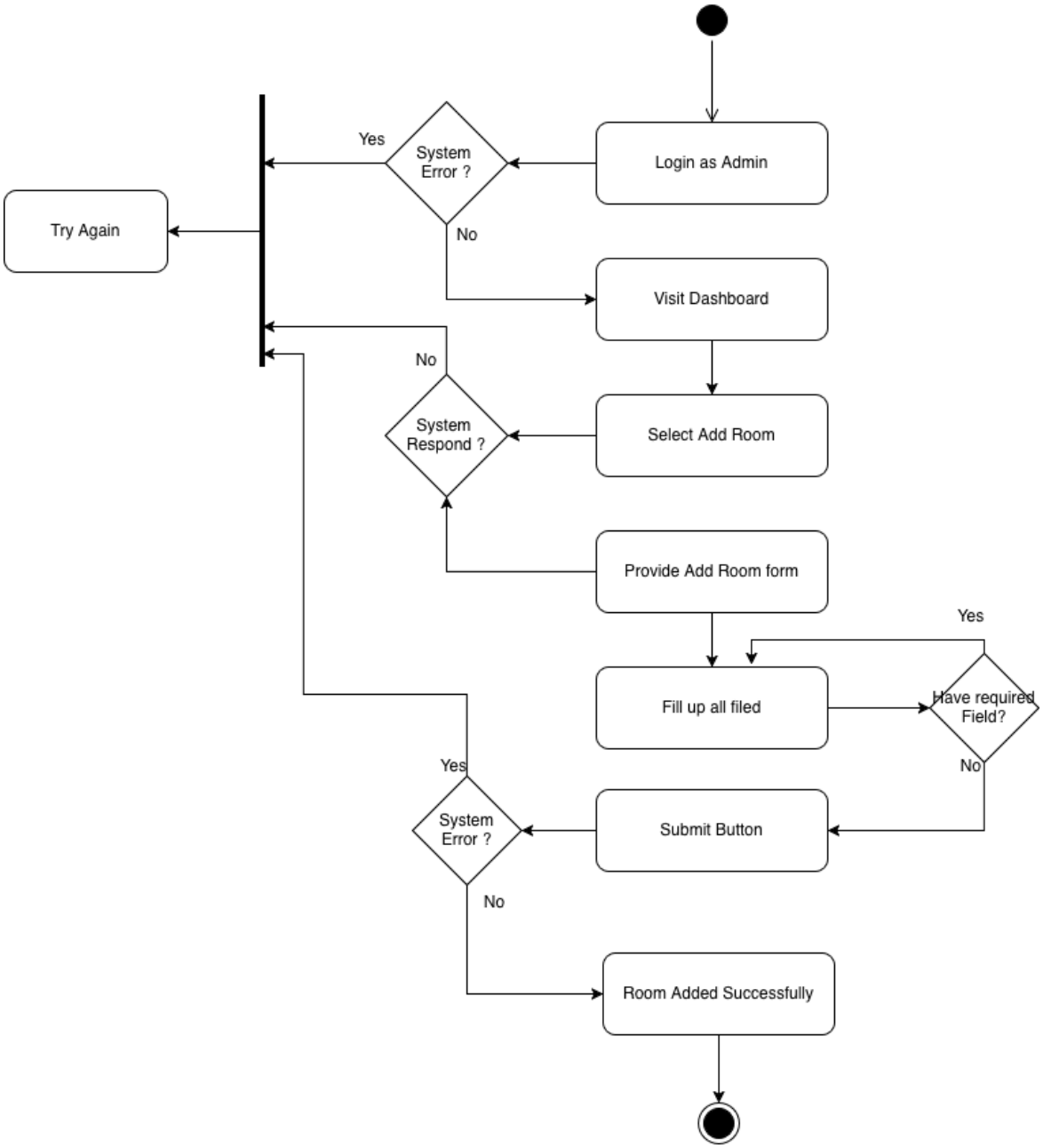


Figure 3.6: Activity diagram for Add Room

For Update Room:

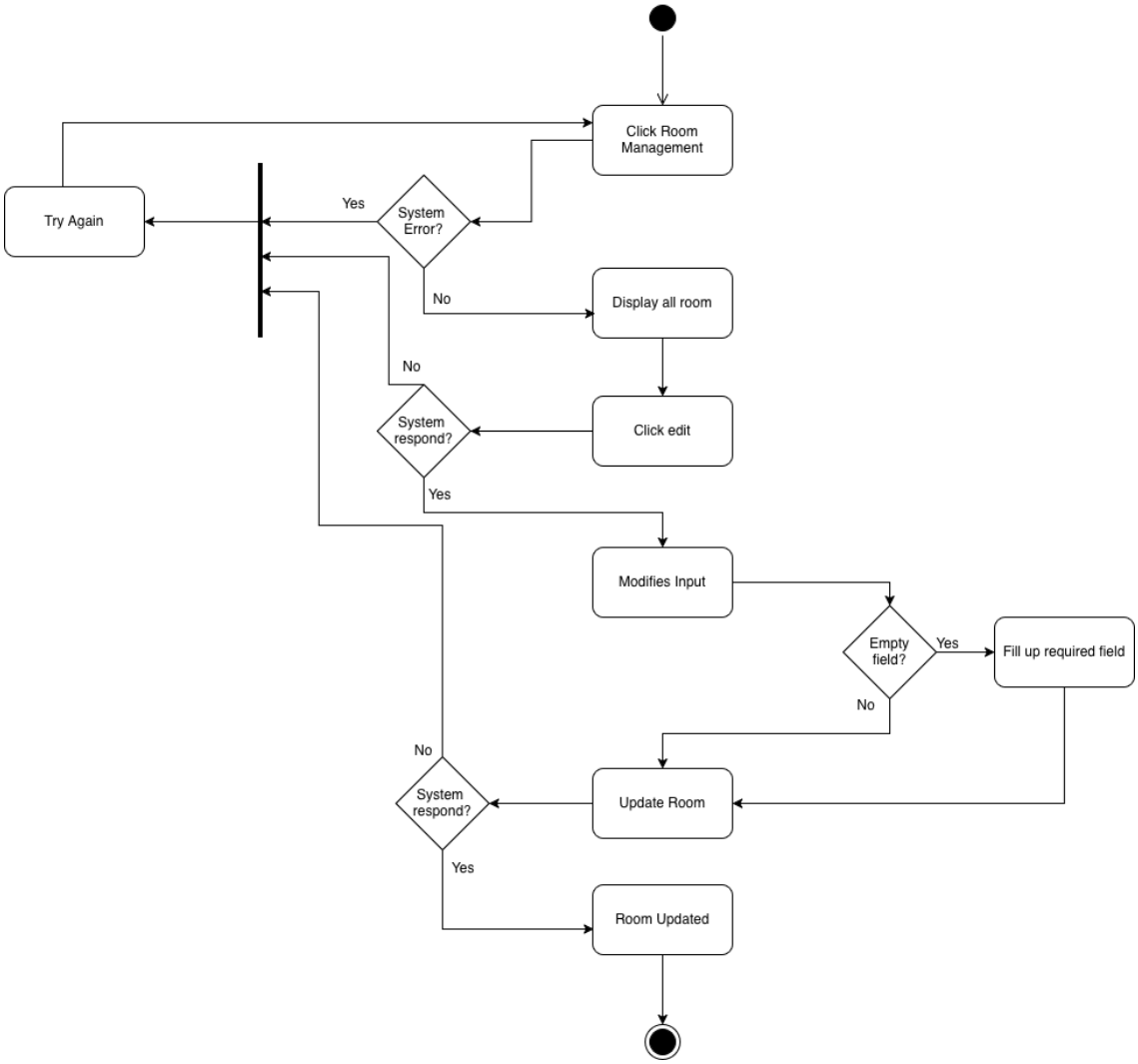


Figure 3.7: Activity diagram for Update Room

For Delete Room:

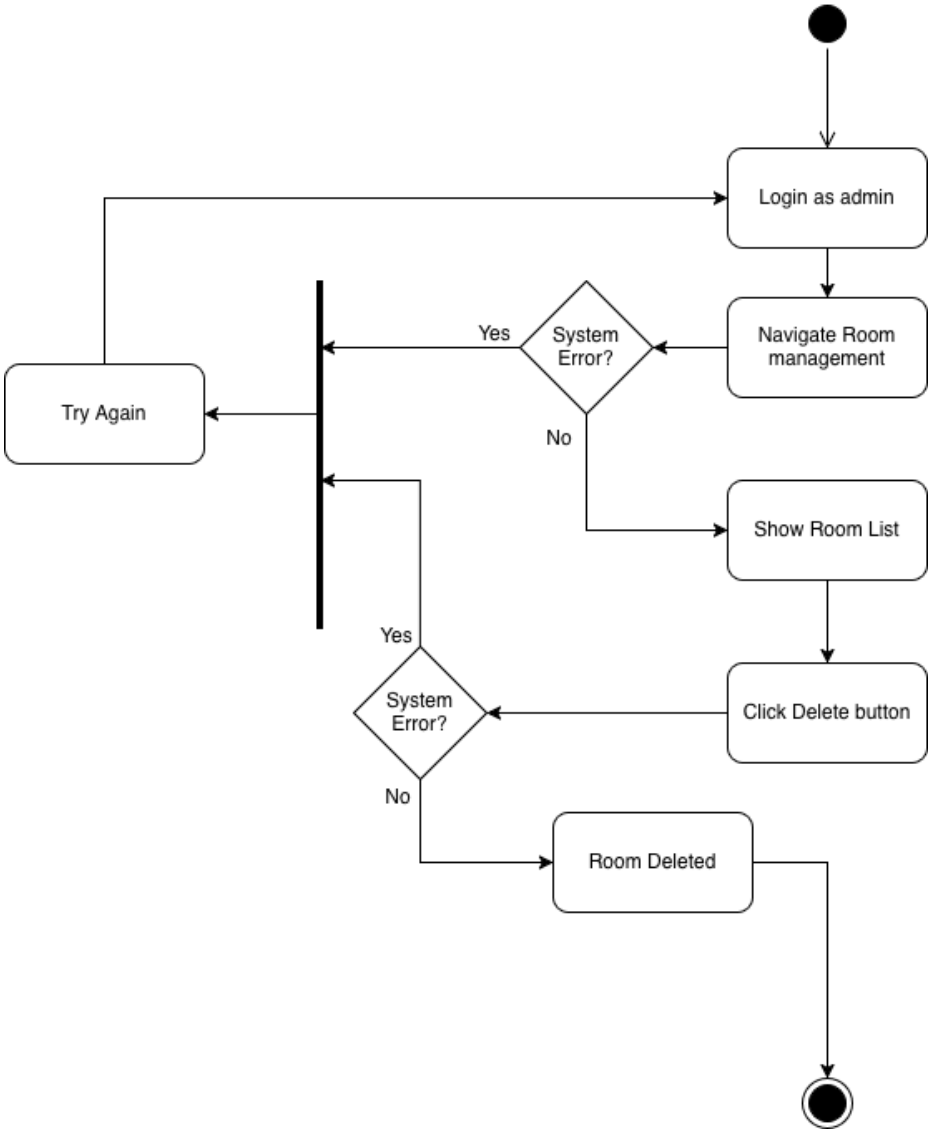


Figure 3.8: Activity diagram for Delete Room

For Add to Cart:

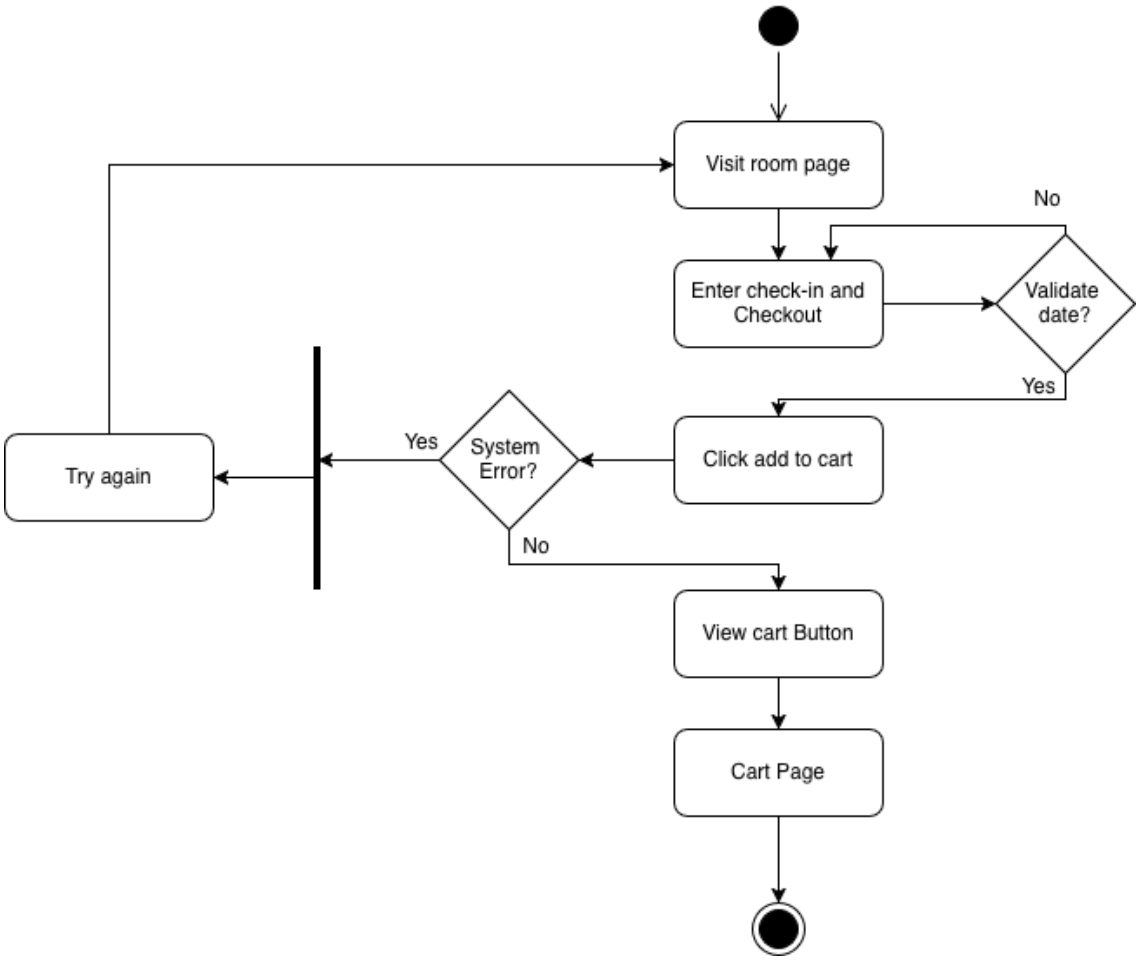


Figure 3.9: Activity diagram for Add to cart

For Make Booking:

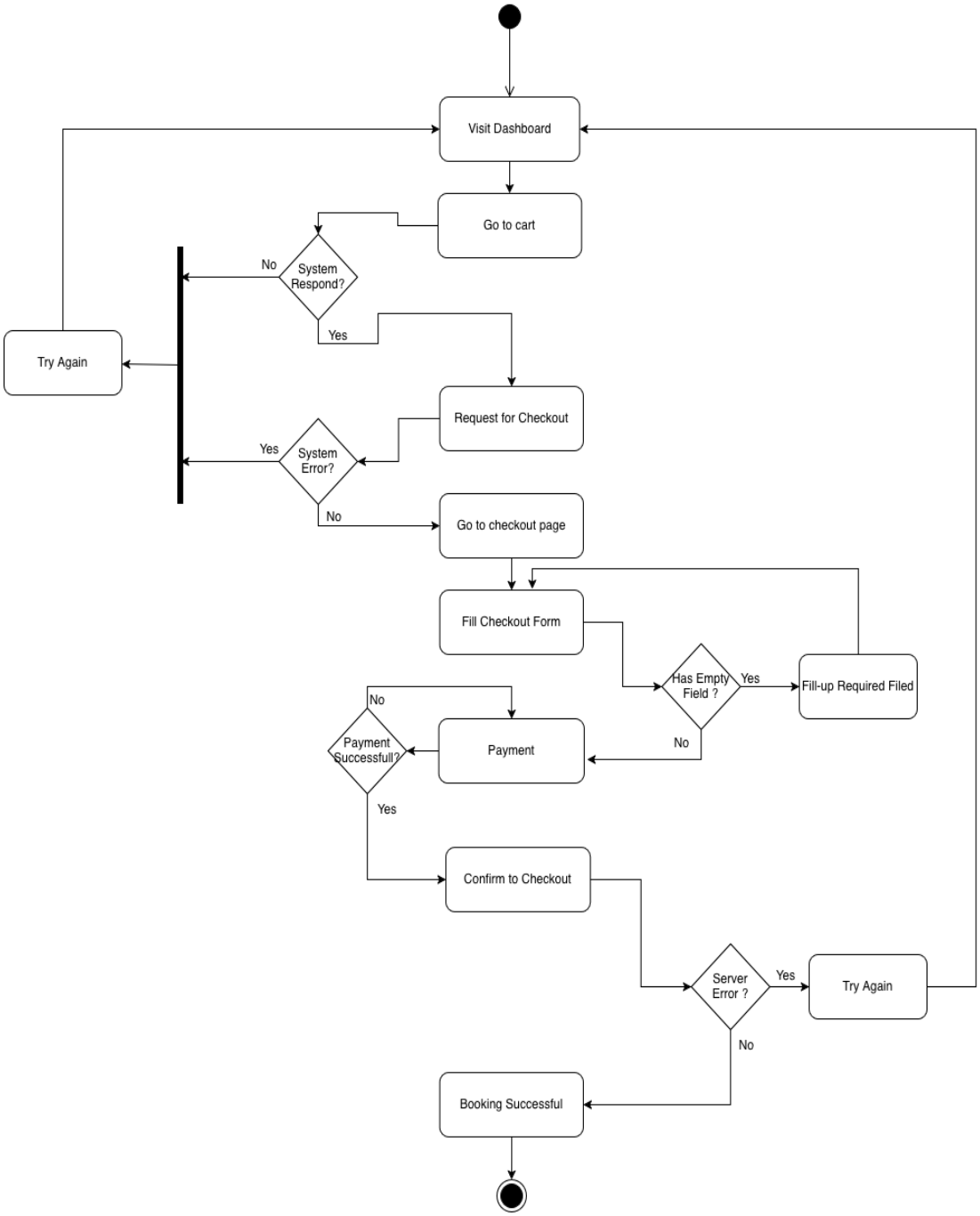


Figure 3.10: Activity diagram for Make Booking

For Make Payment:

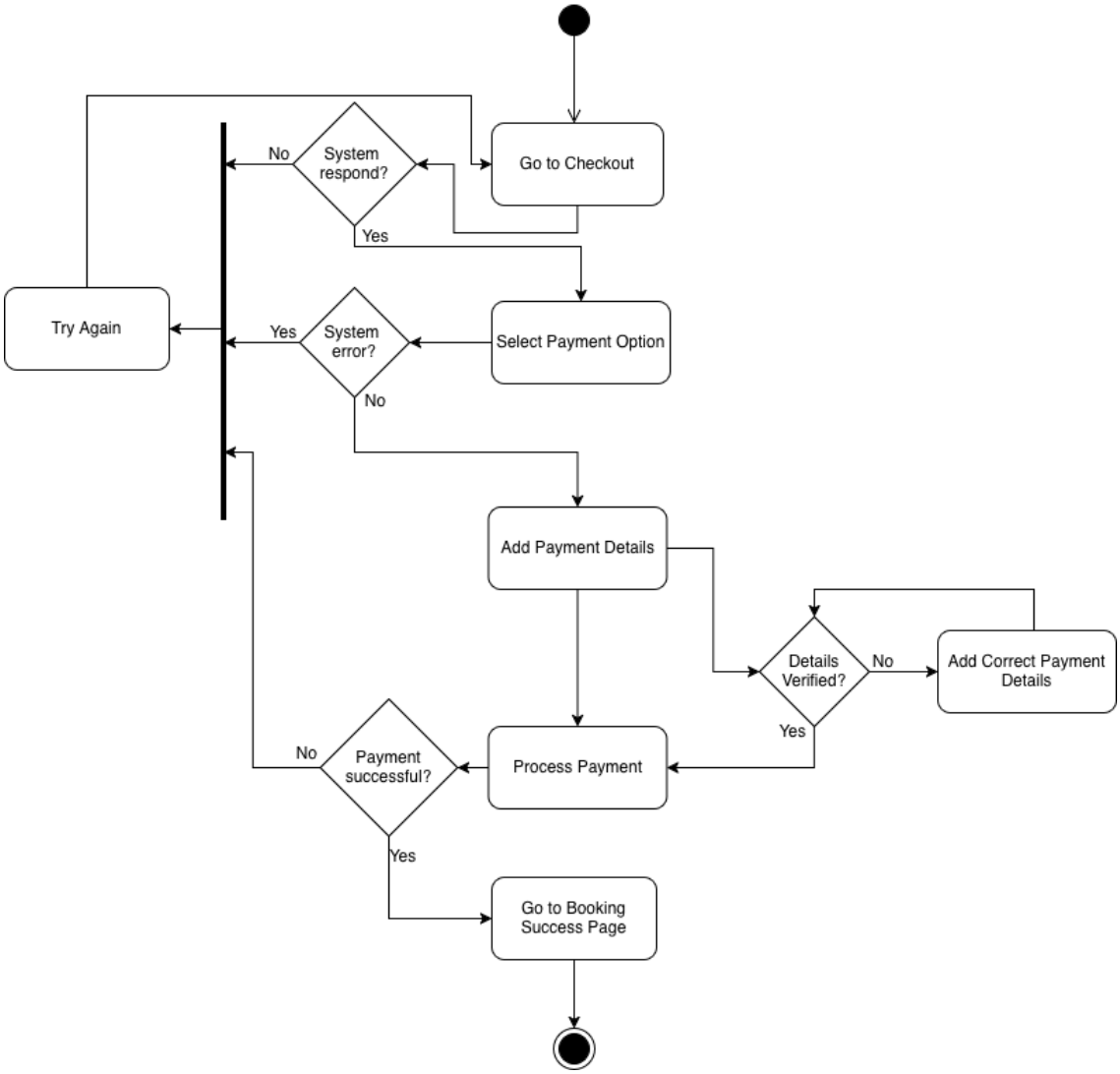


Figure 3.11: Activity diagram for Make Payment

For Loyalty Reward:

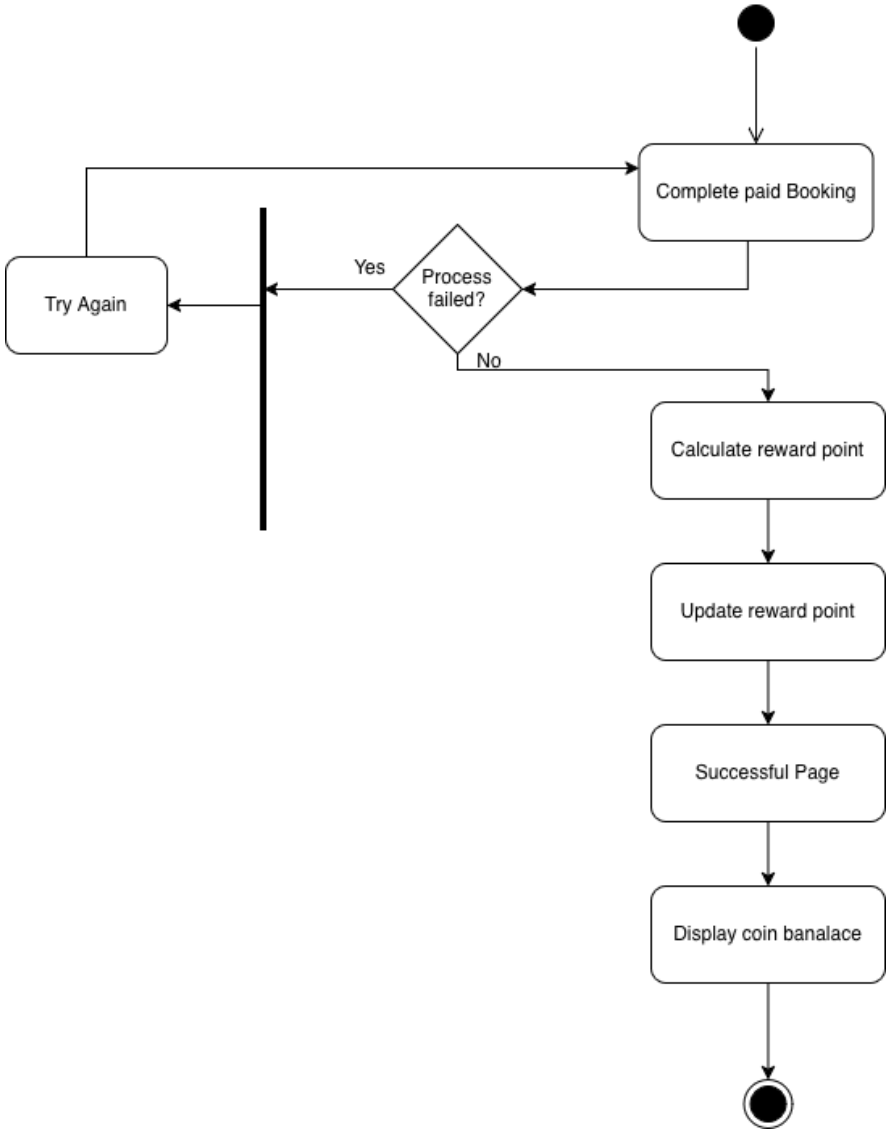


Figure 3.12: Activity diagram for Loyalty Reward

For Loyalty Discount:

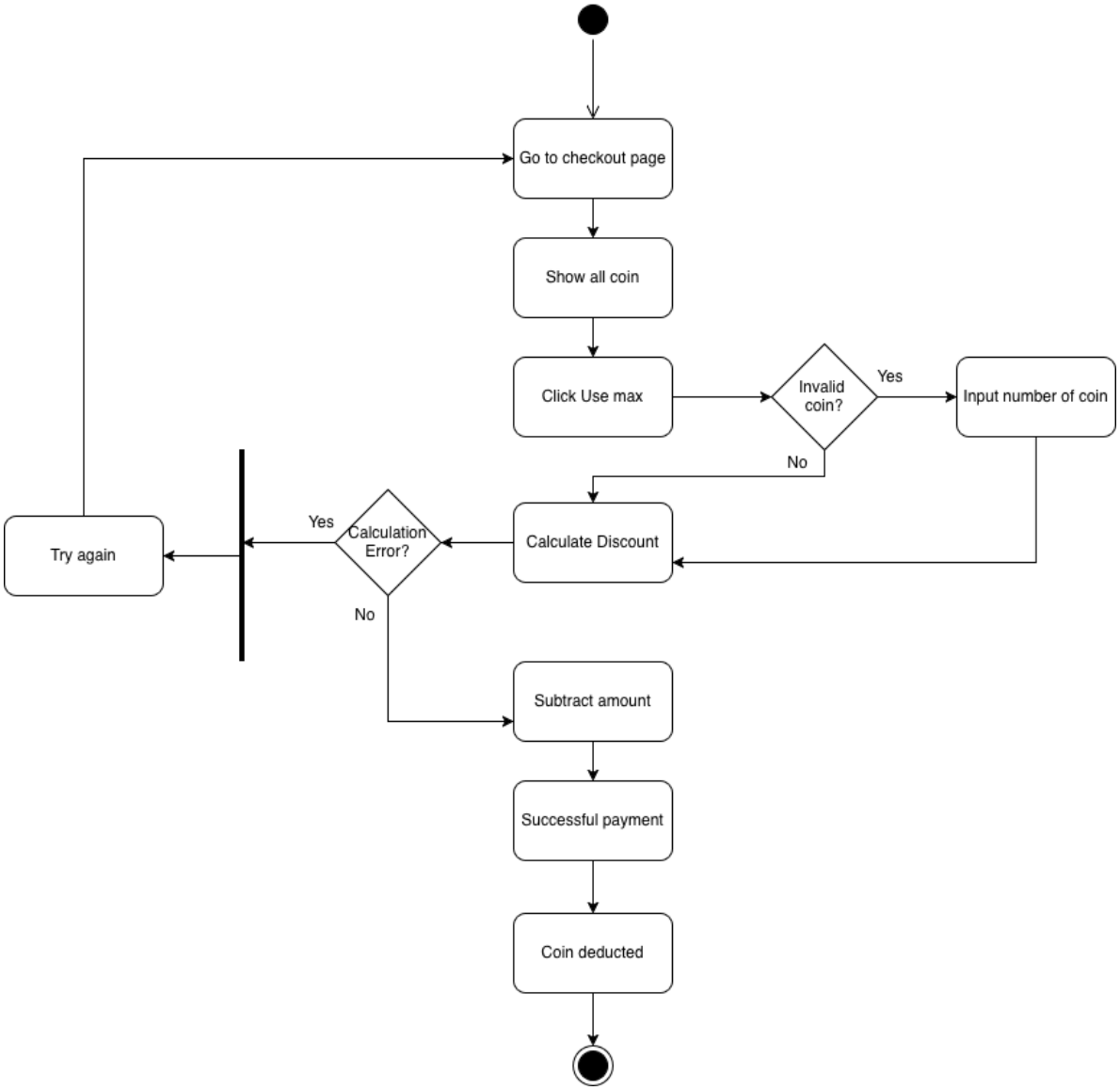


Figure 3.13: Activity diagram for Loyalty Discount

For Cancel Booking:

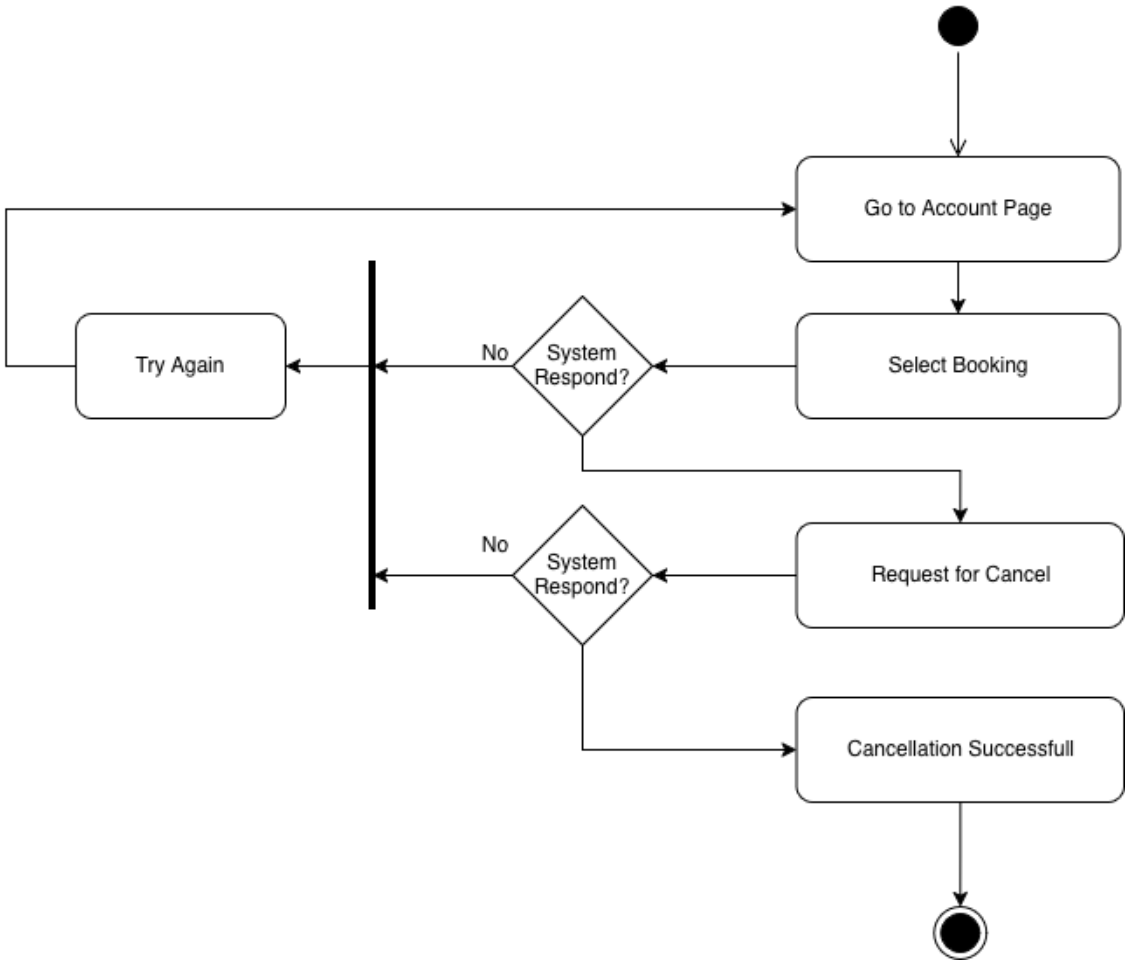


Figure 3.14: Activity diagram for Cancel Booking

For Get Refund:

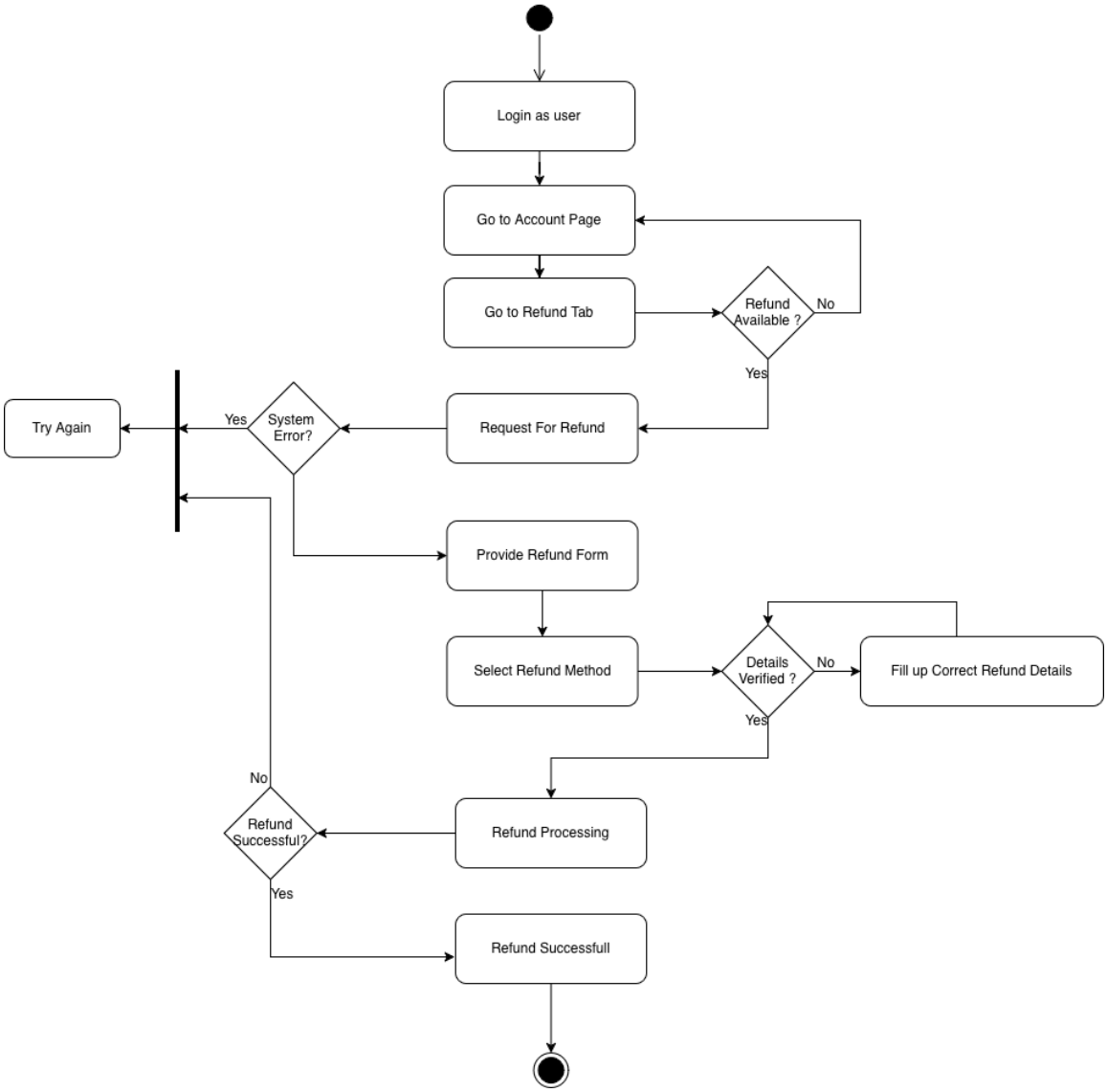


Figure 3.15: Activity diagram for Get Refund

For Make Check-in:

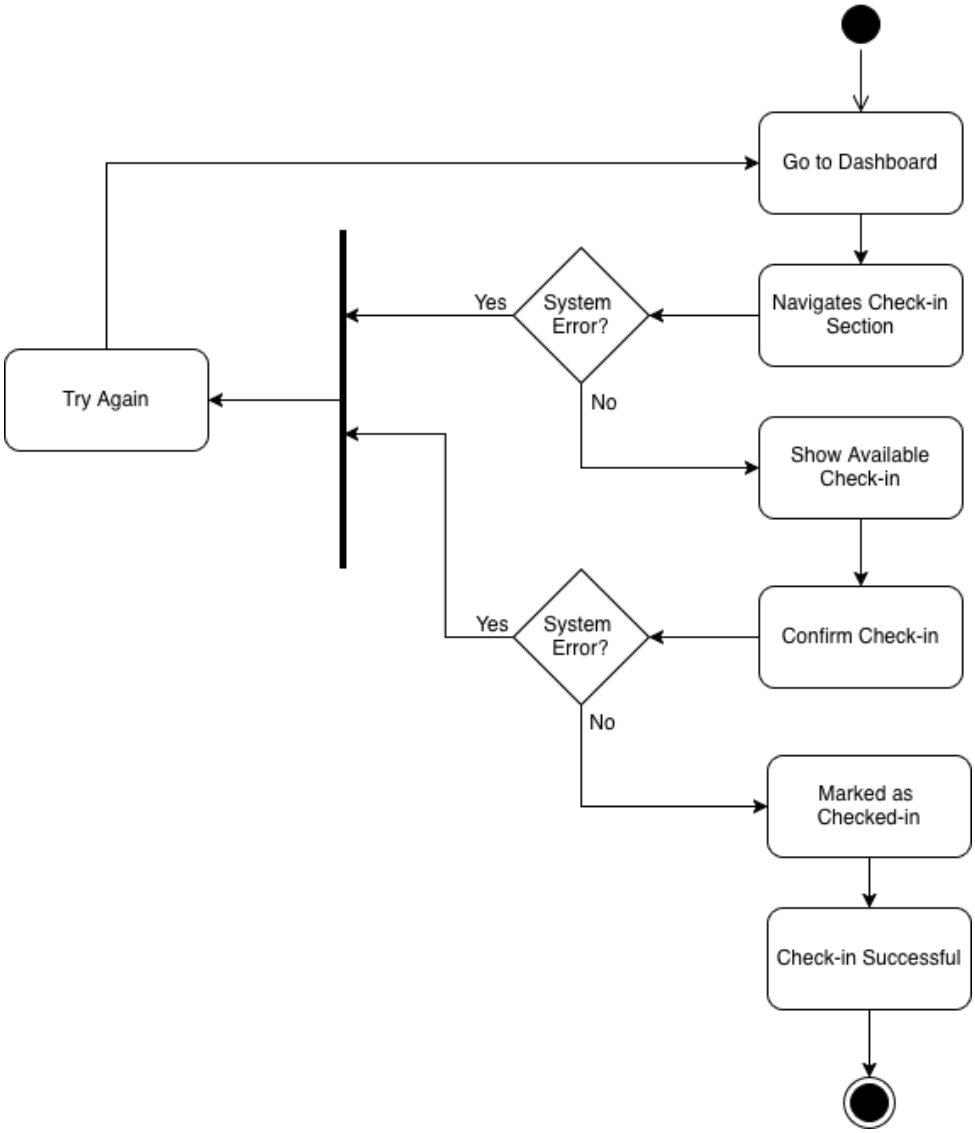


Figure 3.16: Activity diagram for Make Check-in

For Make Check-out:

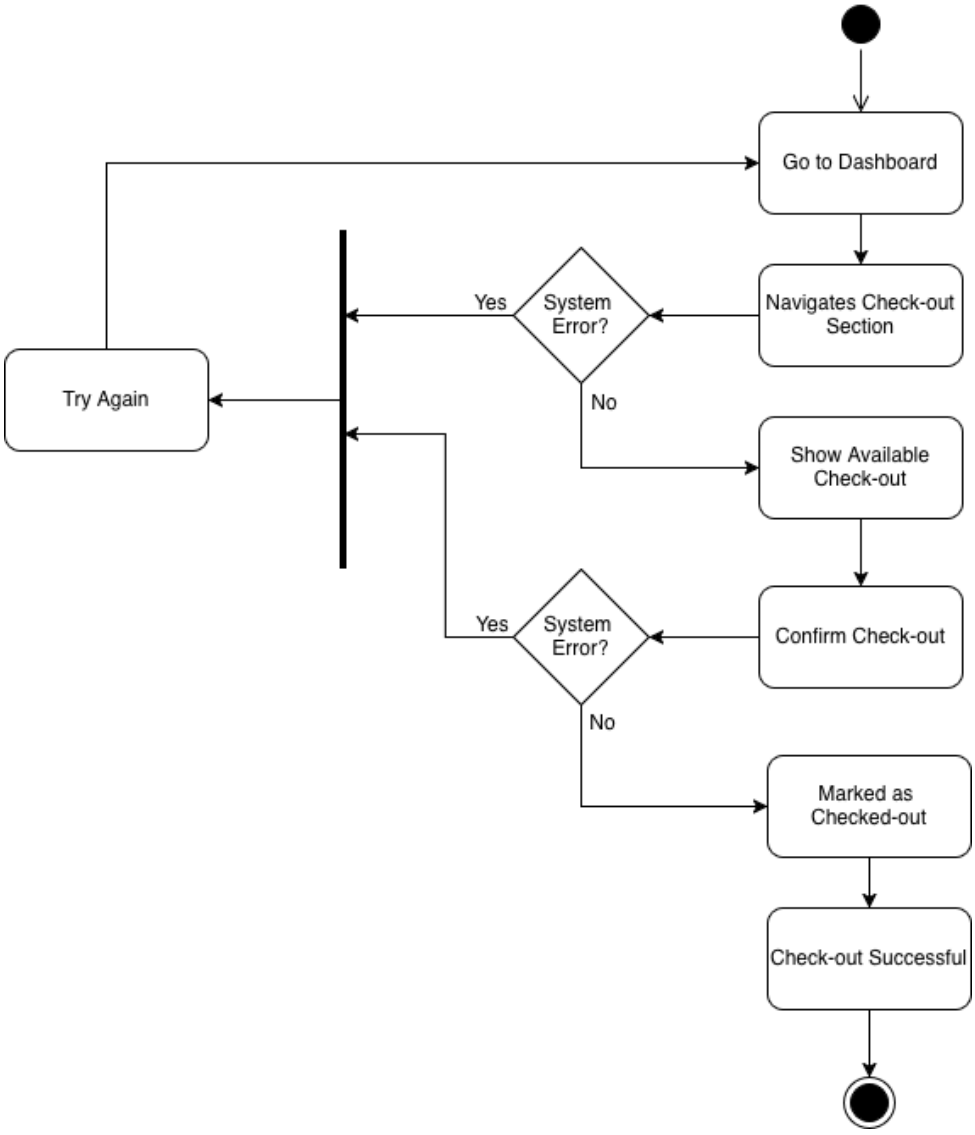


Figure 3.17: Activity diagram for Make Check-out

For Add Blog:

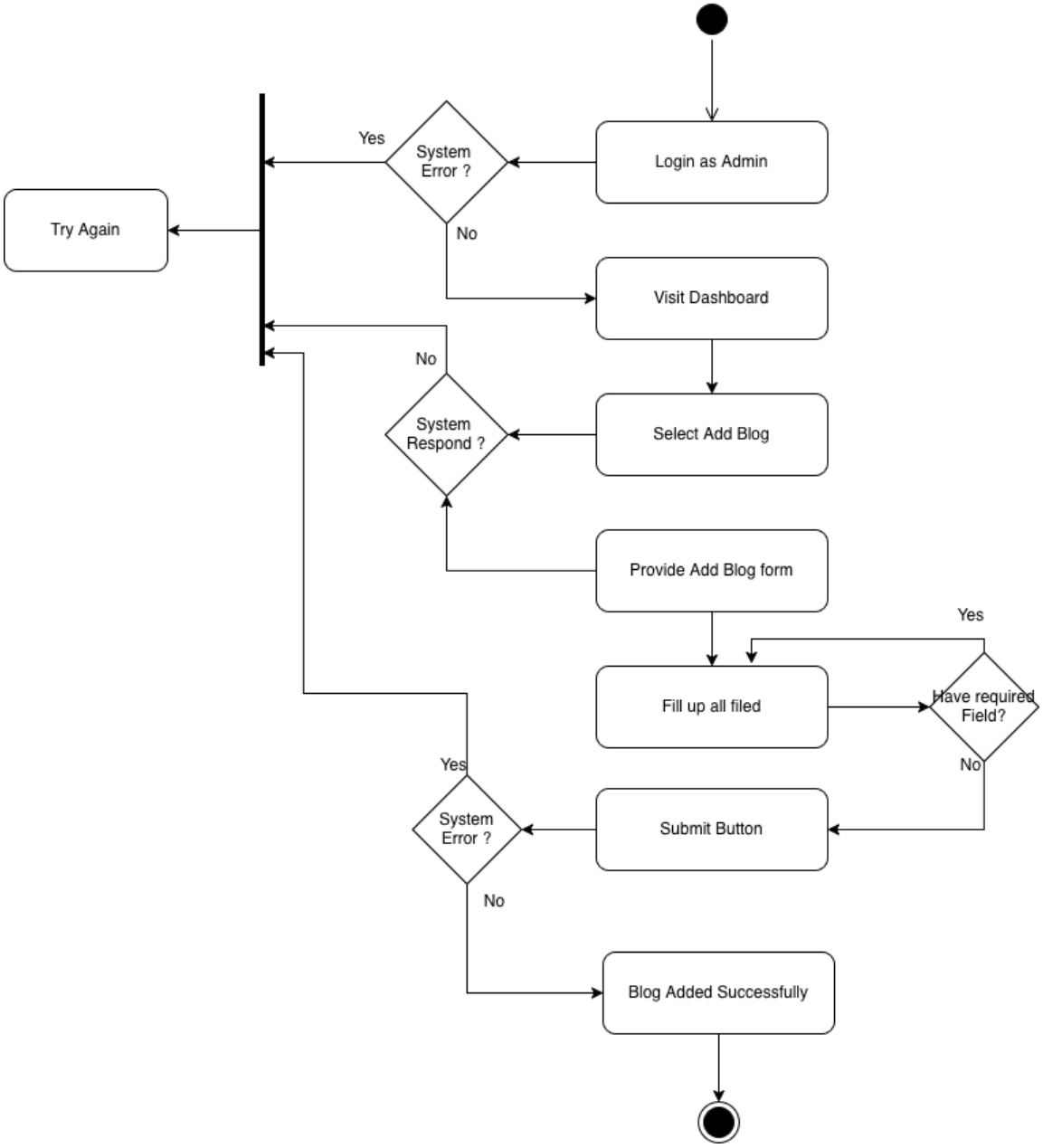


Figure 3.18: Activity diagram for Add Blog

For Update Blog:

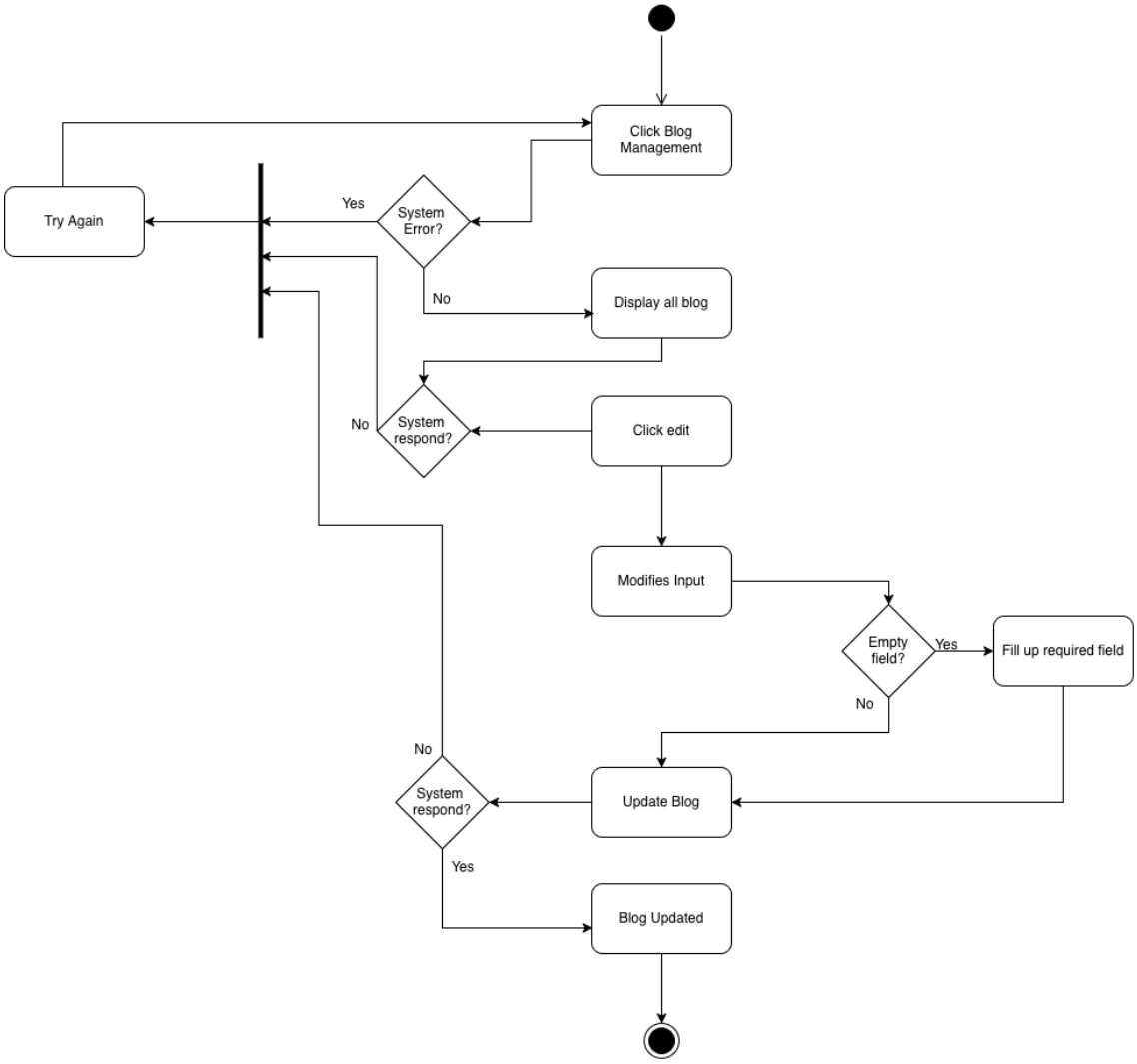


Figure 3.19: Activity diagram for Update Blog

For Delete Blog:

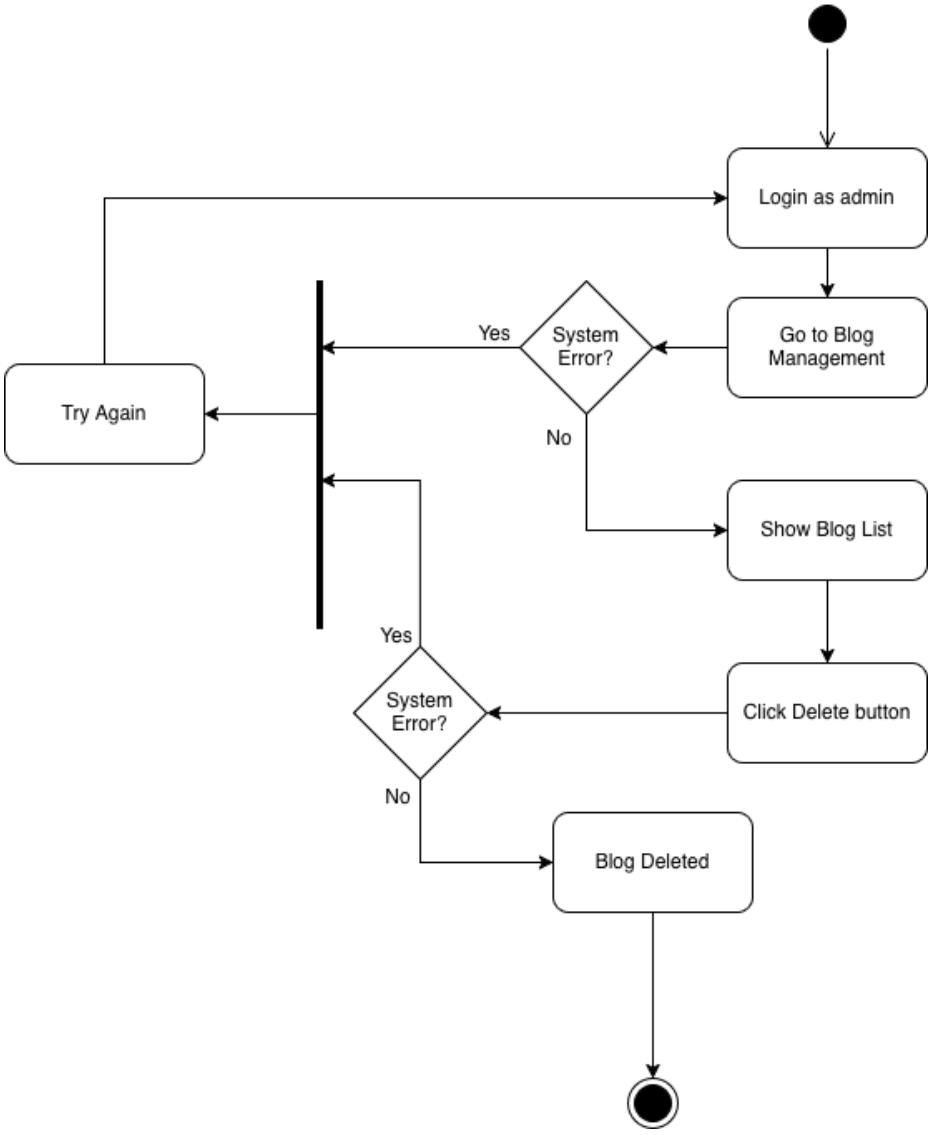


Figure 3.20: Activity diagram for Delete Blog

For View Booking:

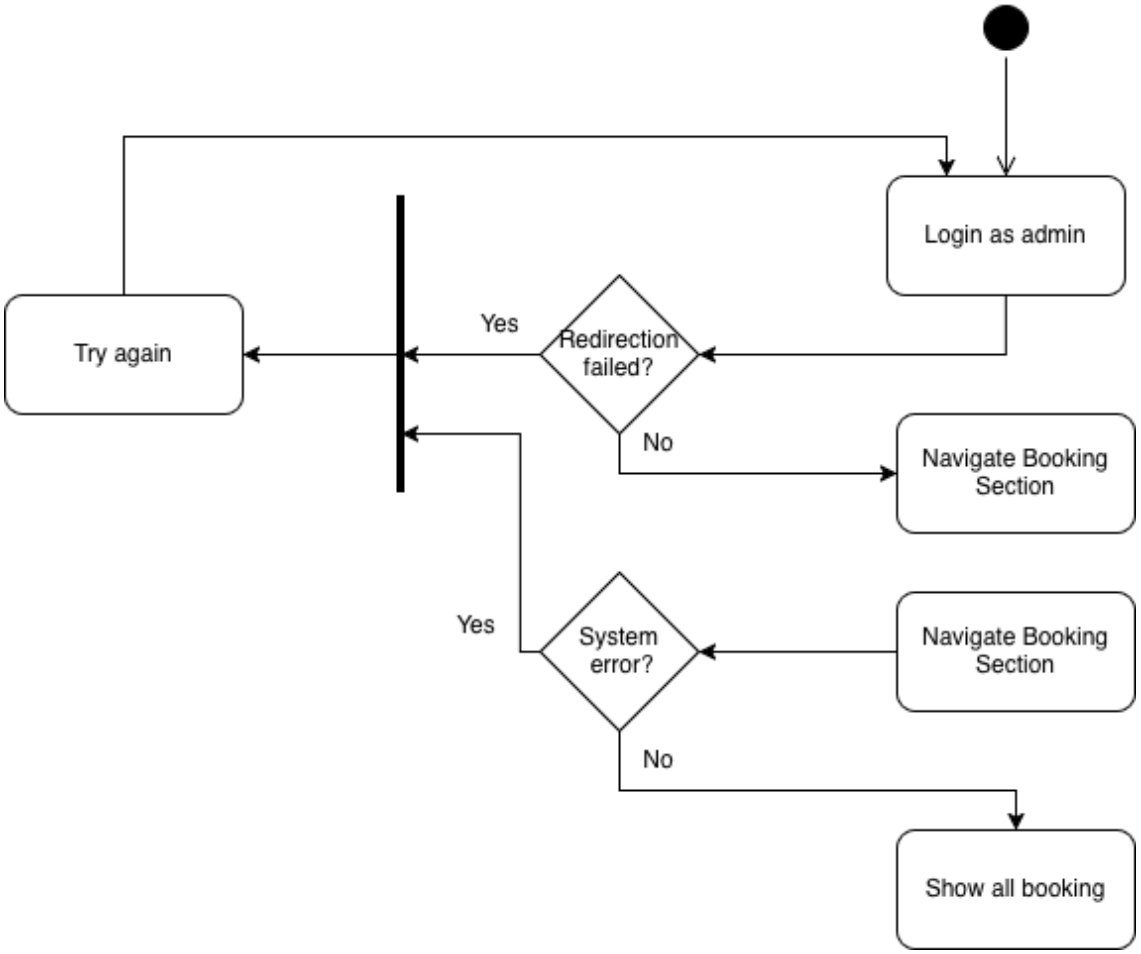


Figure 3.21: Activity diagram for View Booking

For View Earning:

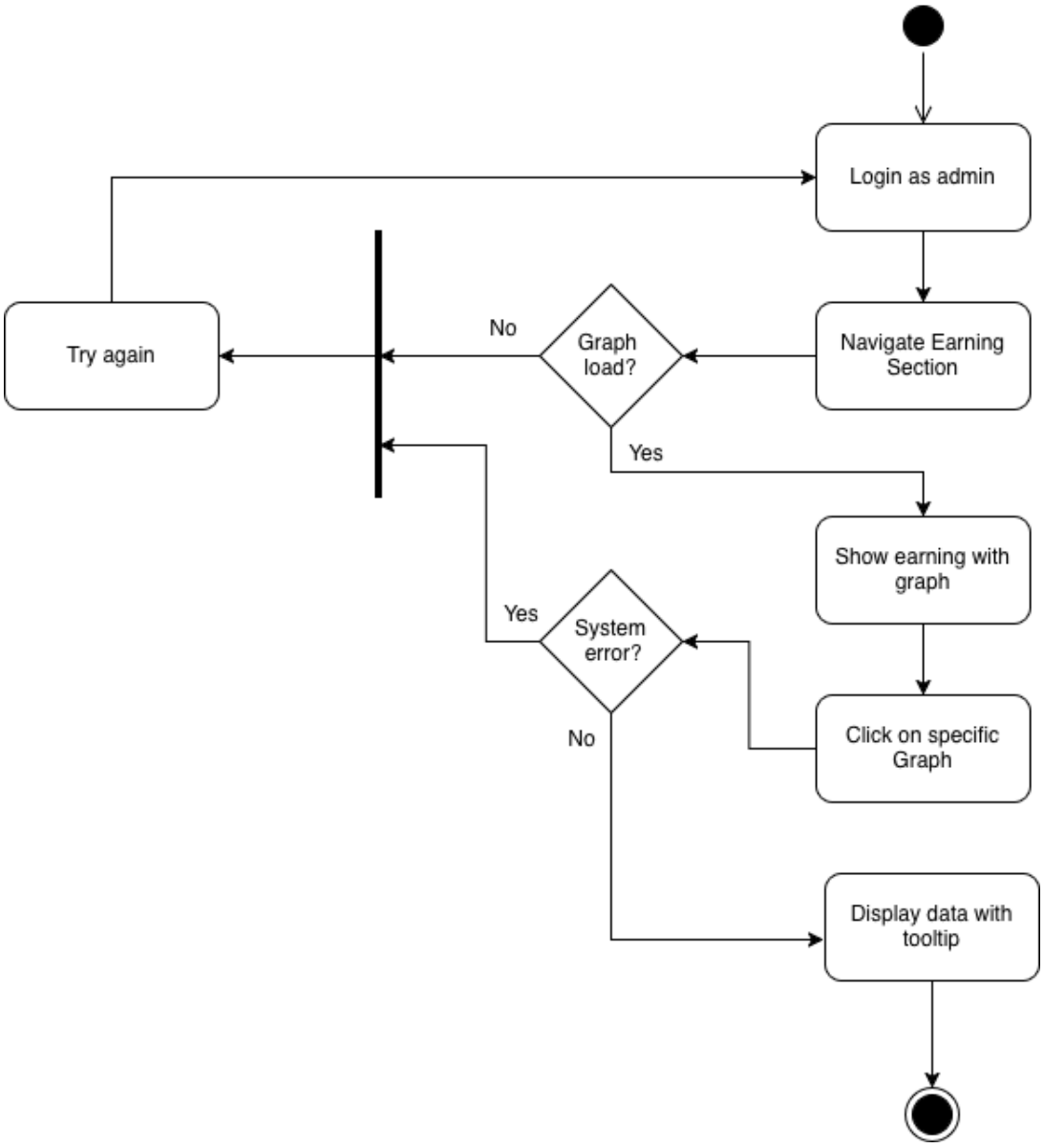


Figure 3.22: Activity diagram for View Earning

For Resend Email:

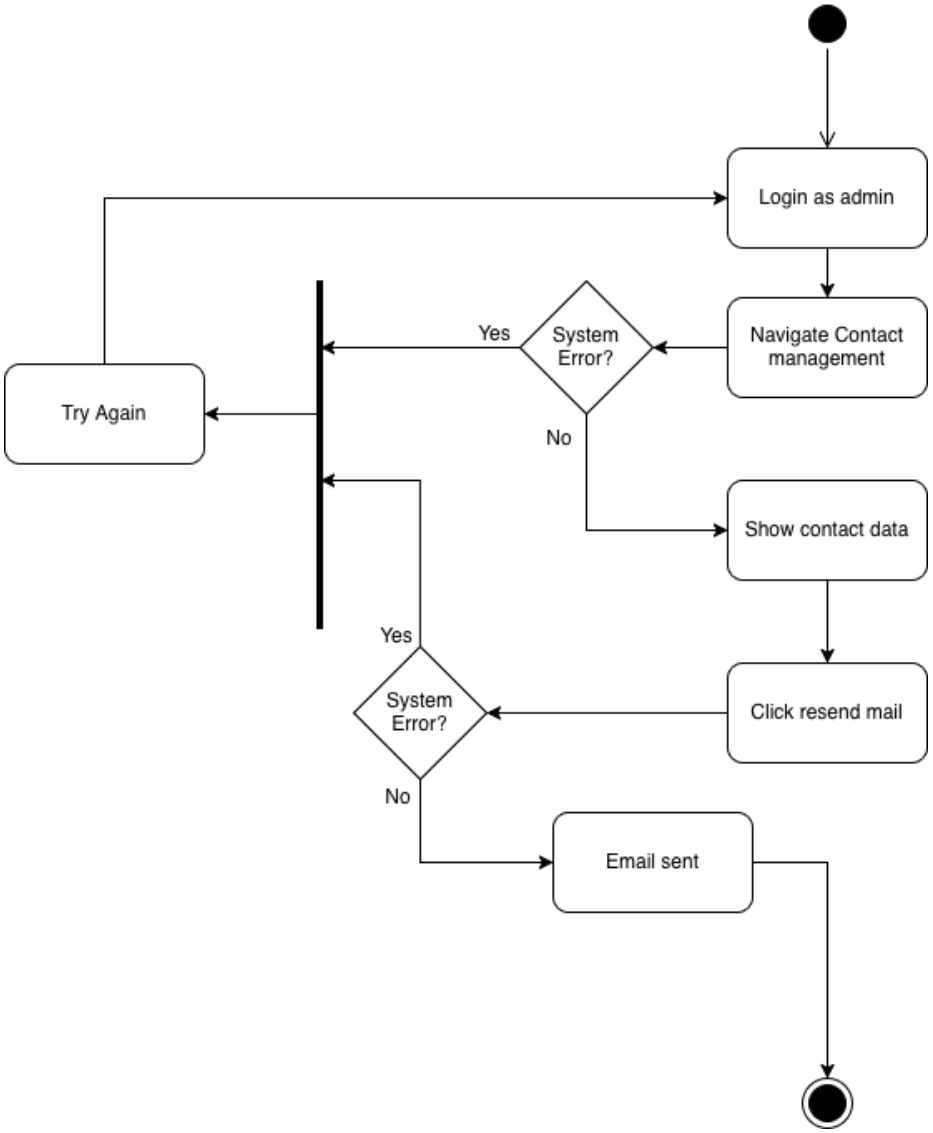


Figure 3.23: Activity diagram for Resend Email

For Delete Contact:

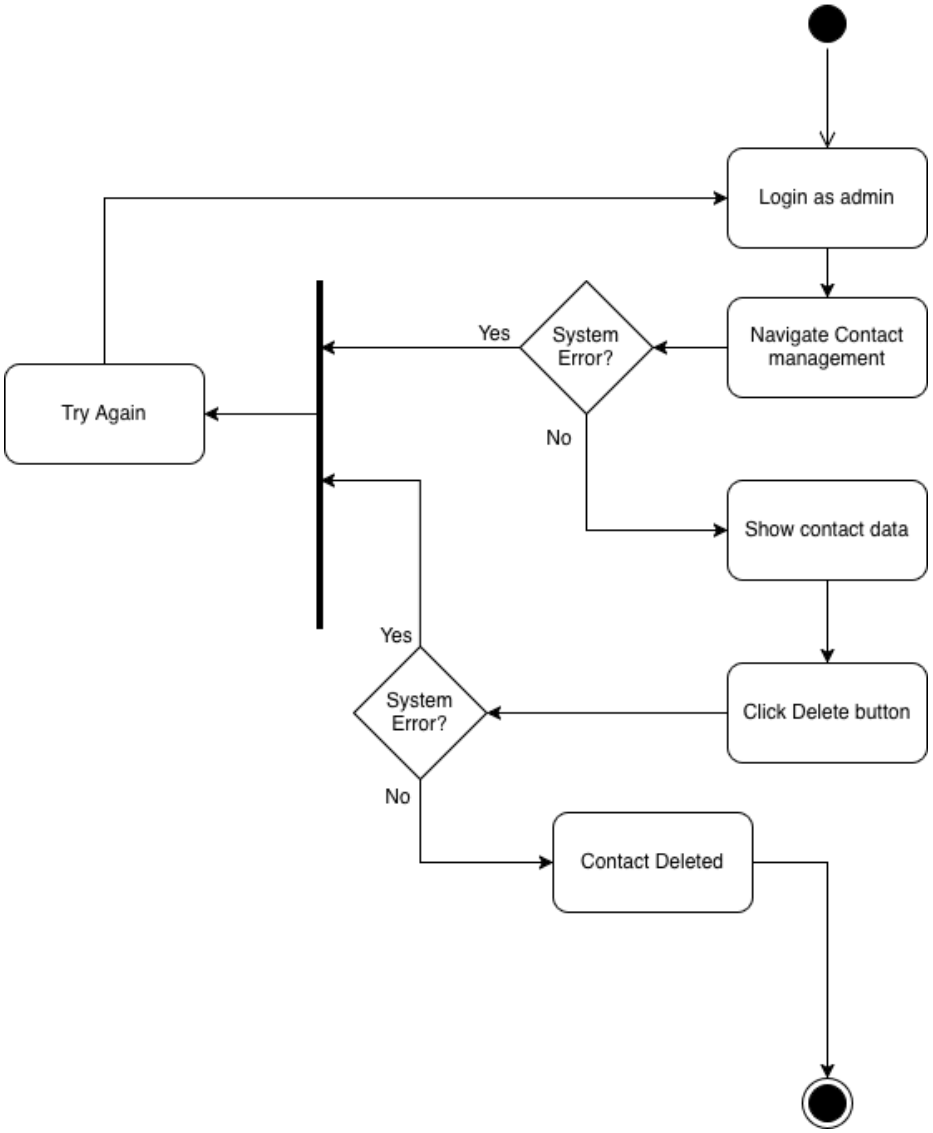


Figure 3.24: Activity diagram for Delete Contact

For Logout:

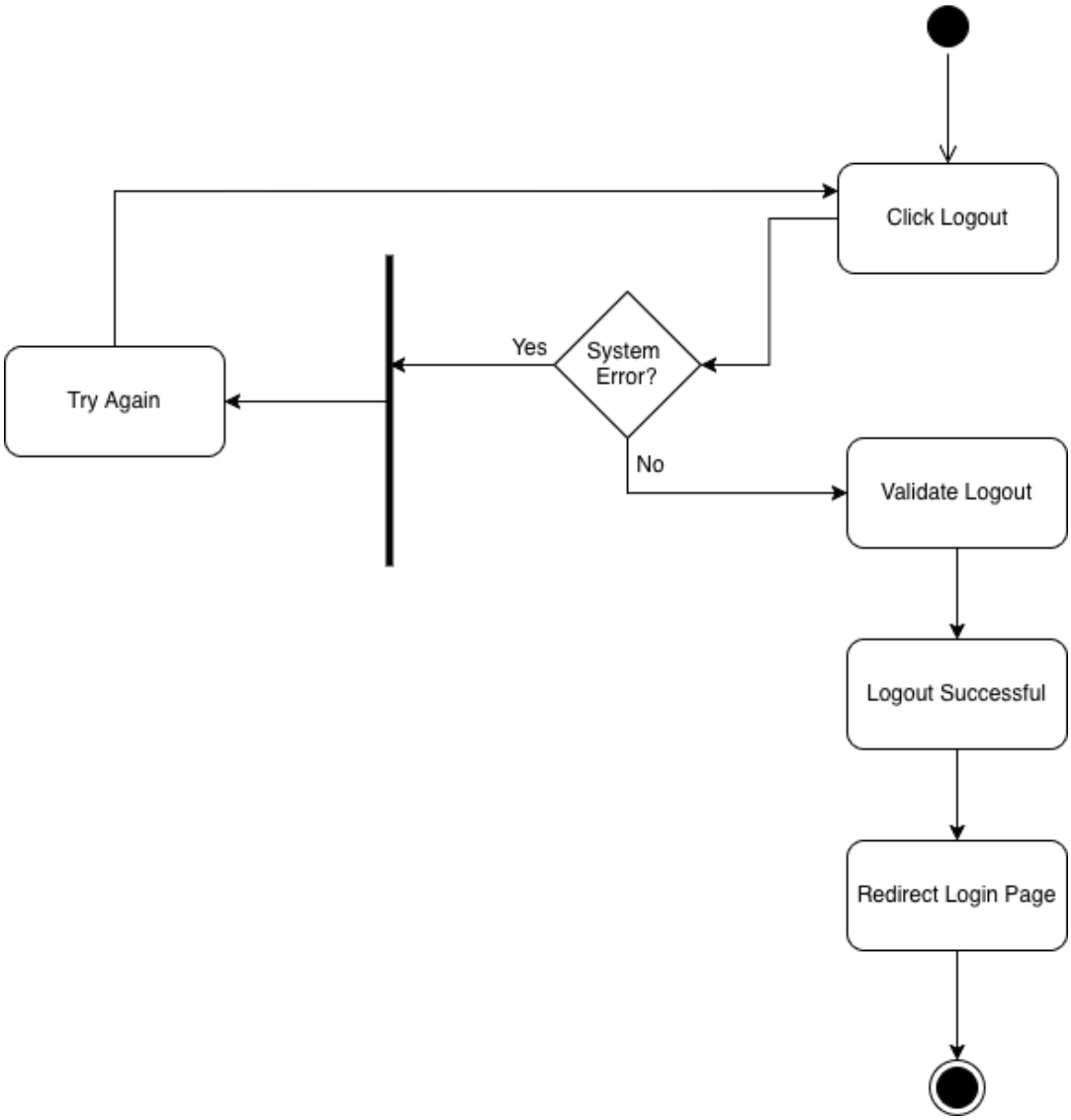


Figure 3.25: Activity diagram for Logout

2.4.4 Sequence Diagram

For User Registration

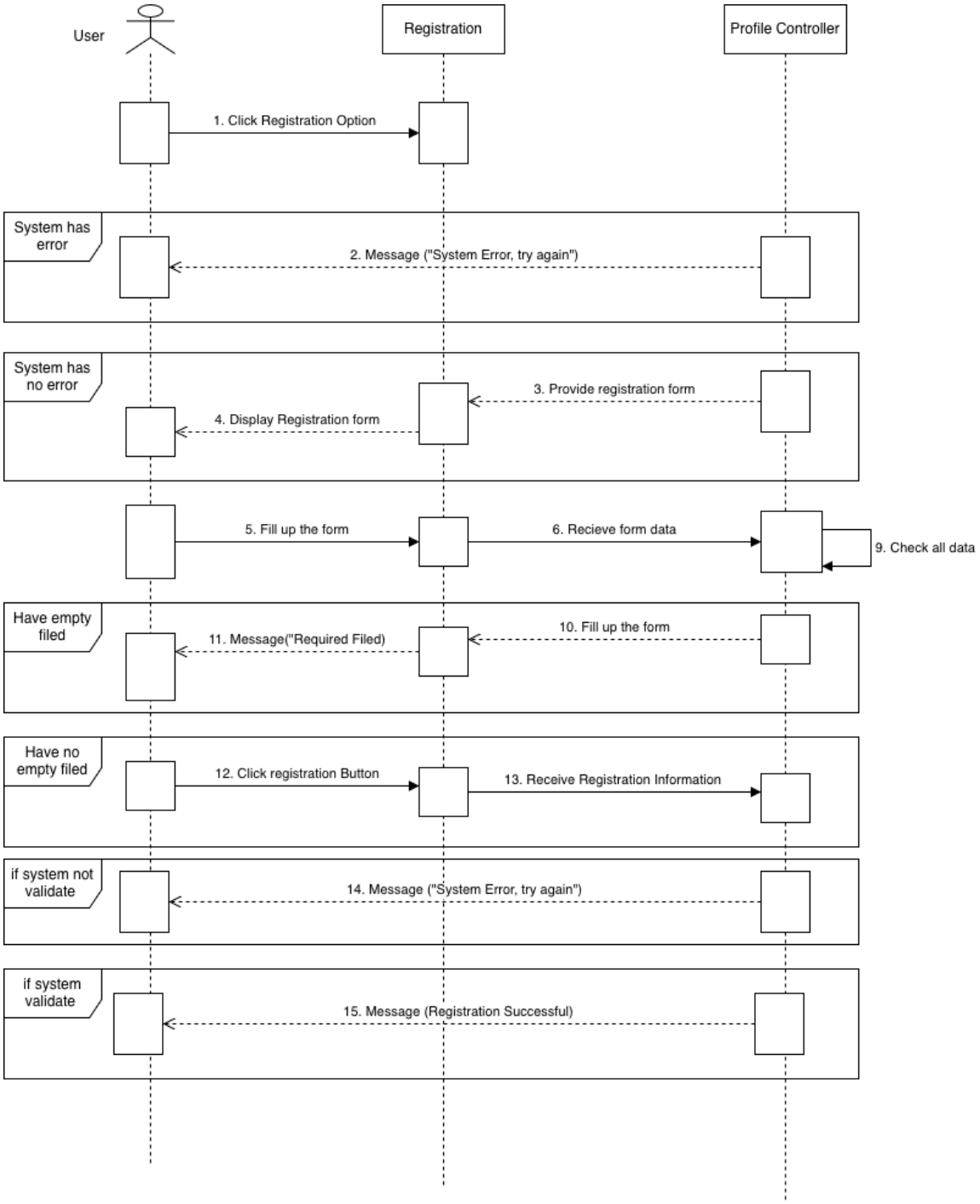


Figure 4.1: Sequence diagram for Registration

For Login

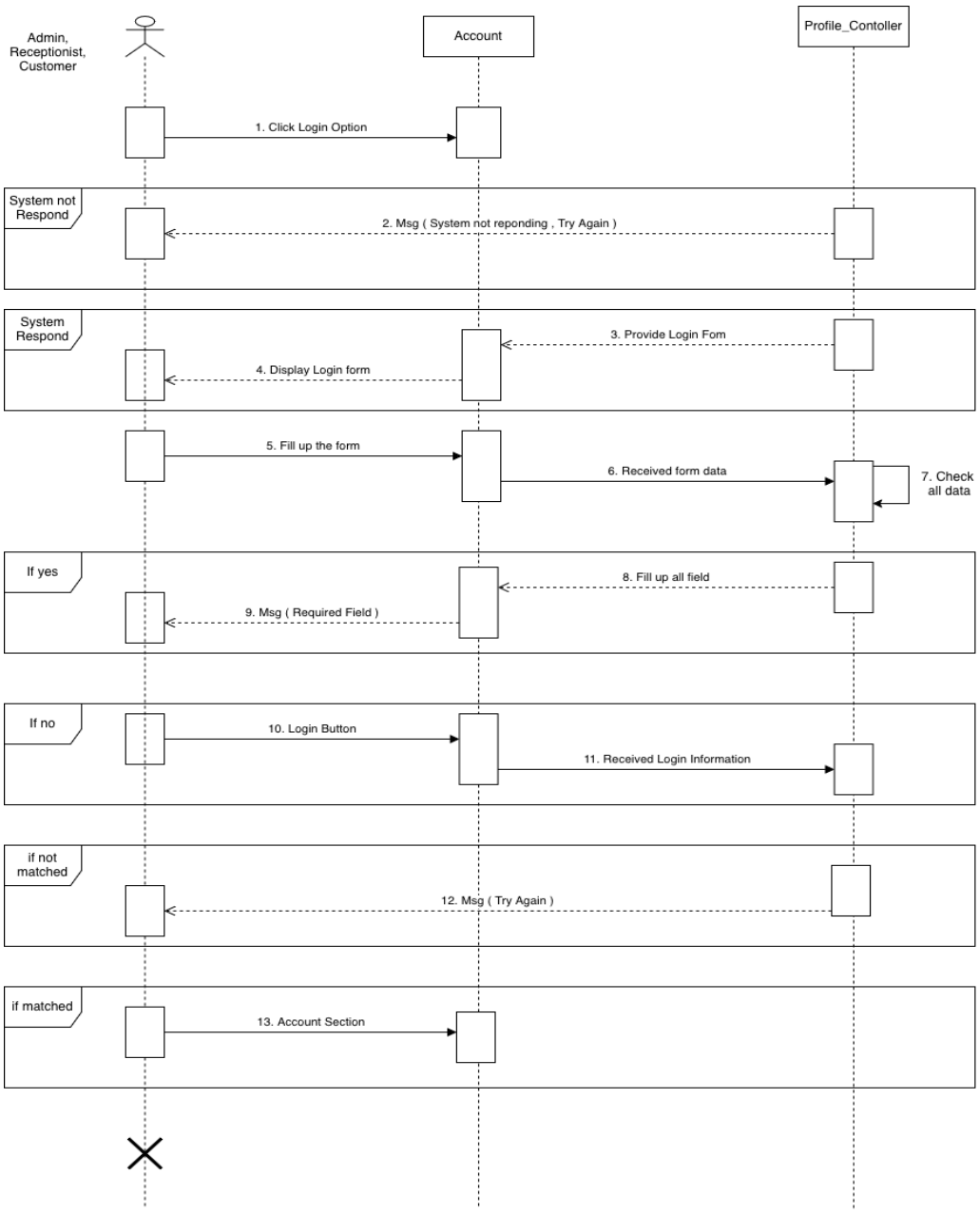


Figure 4.2: Sequence diagram for Login

For Update Information

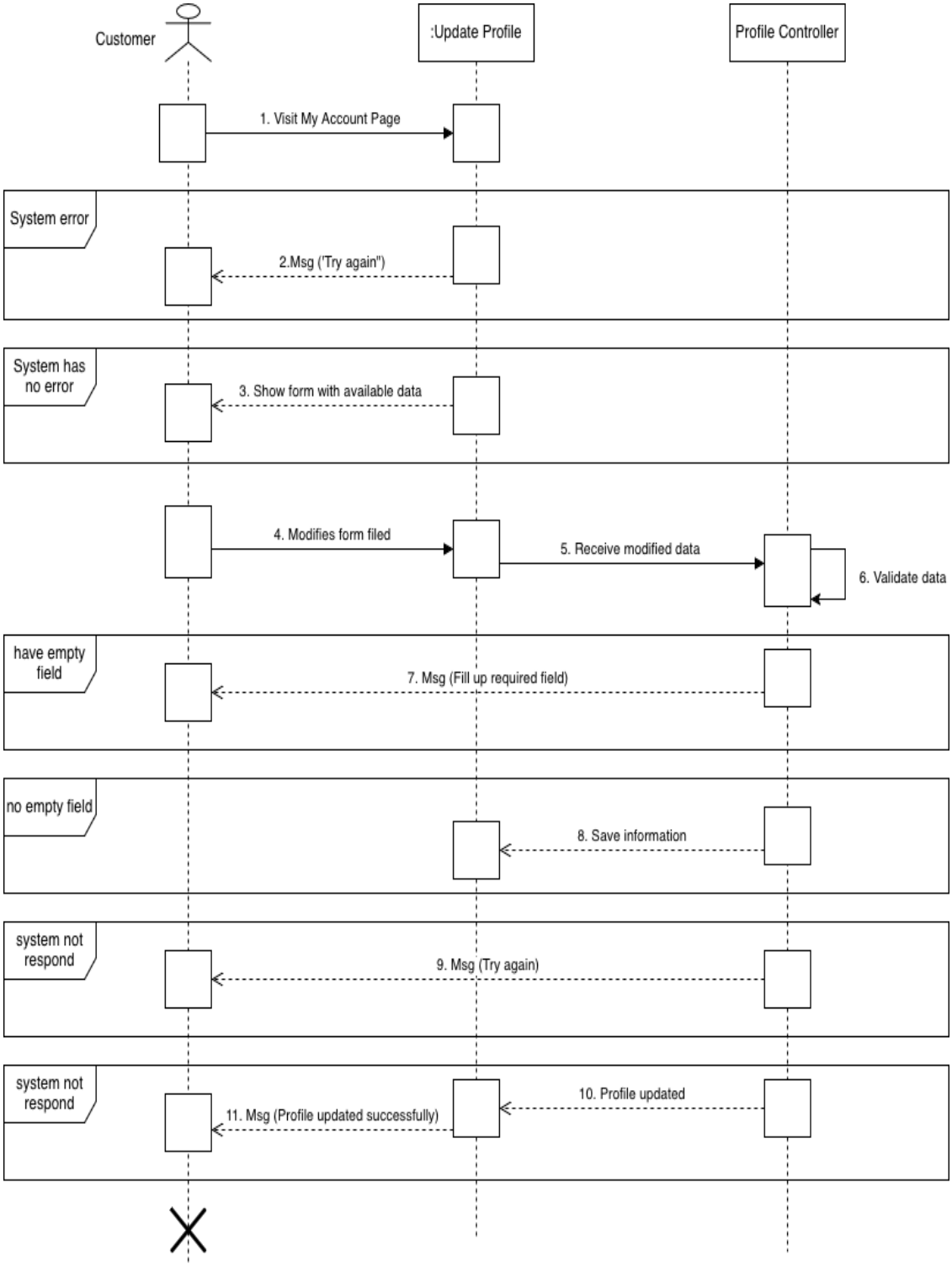


Figure 4.3: Sequence diagram for Update Information

For Chatbot

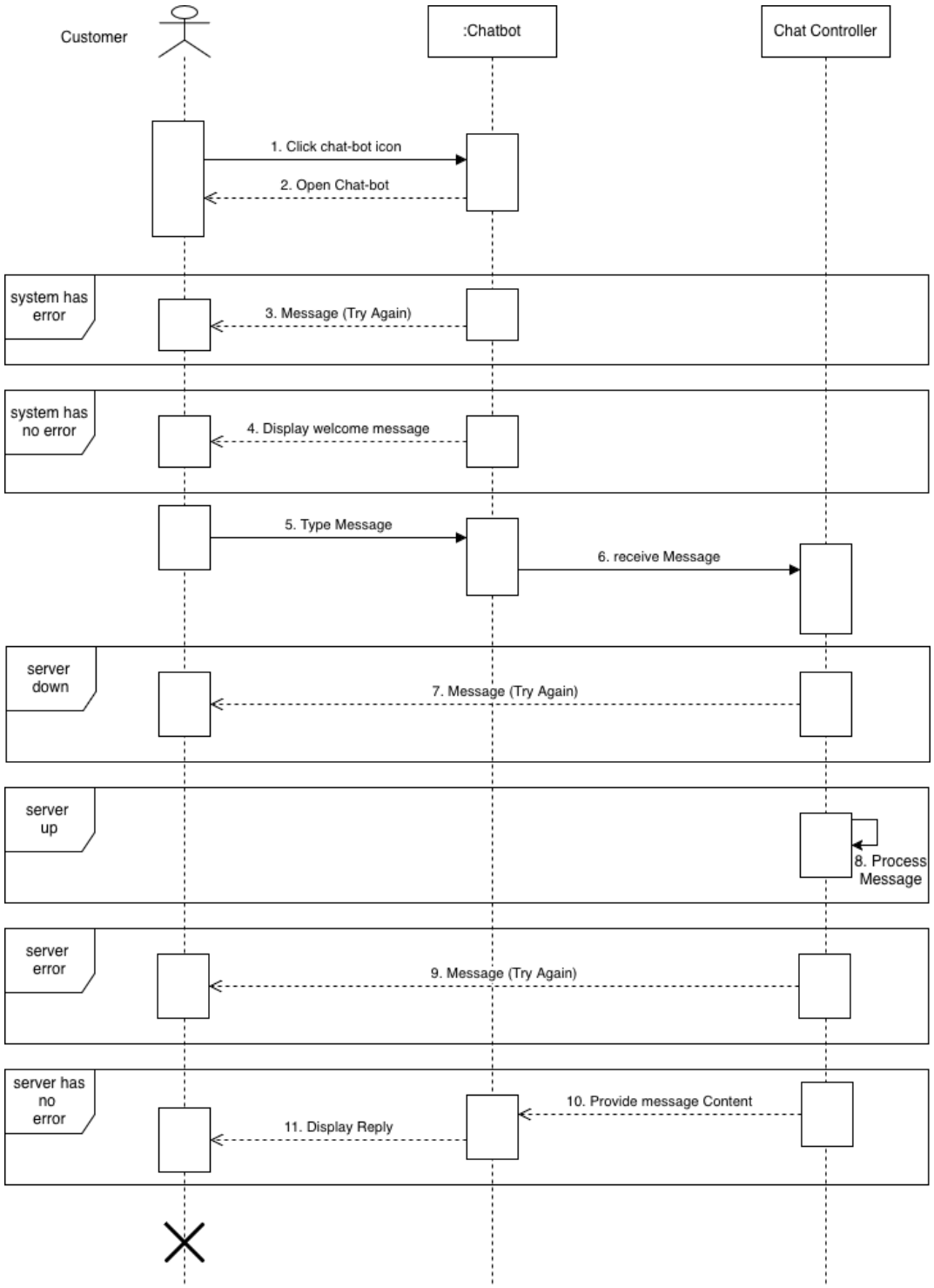


Figure 4.4: Sequence diagram for Chatbot

For Smart Room Recommendation:

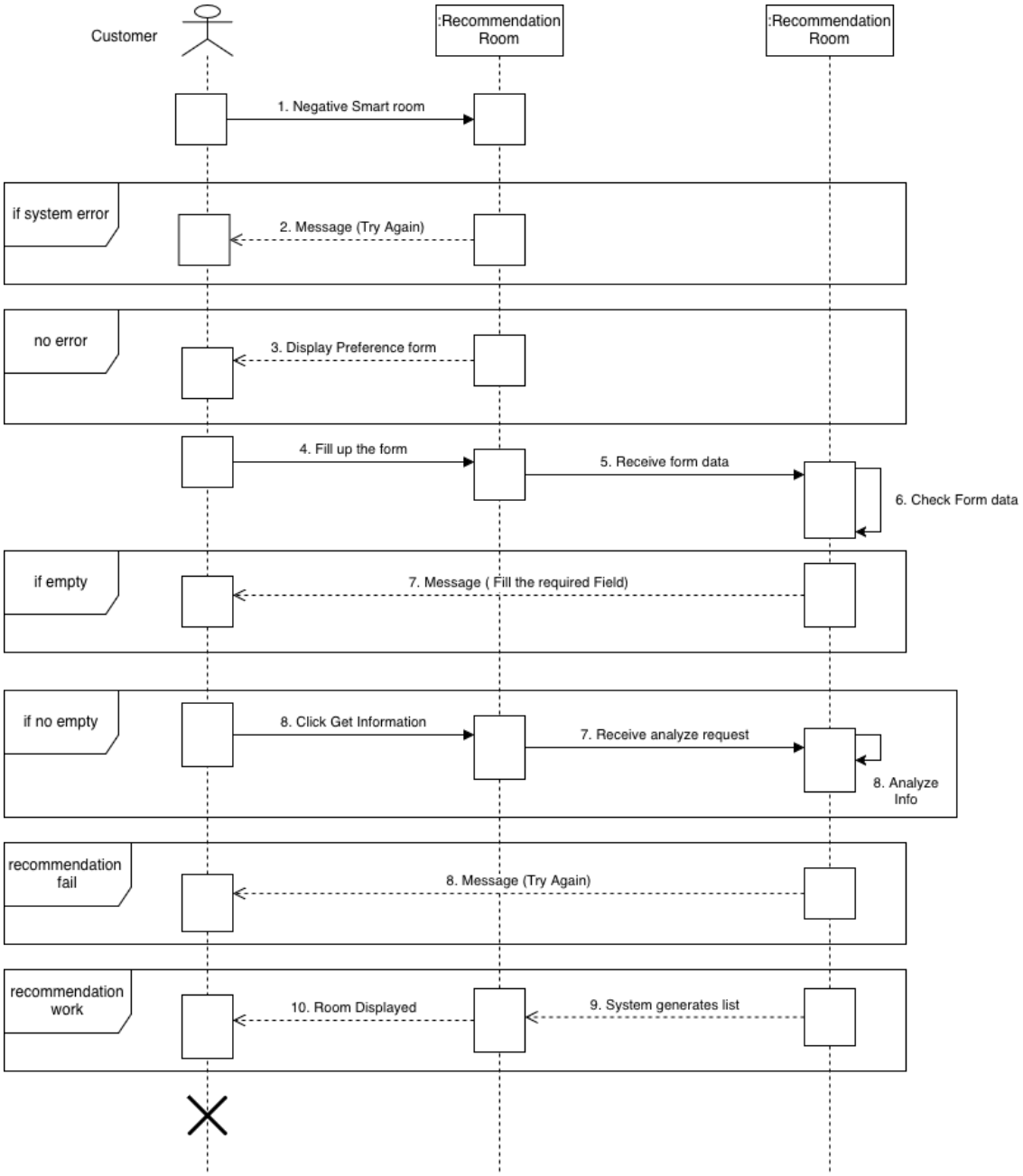


Figure 4.5: Sequence diagram for Smart Room Recommendation

For Add Room

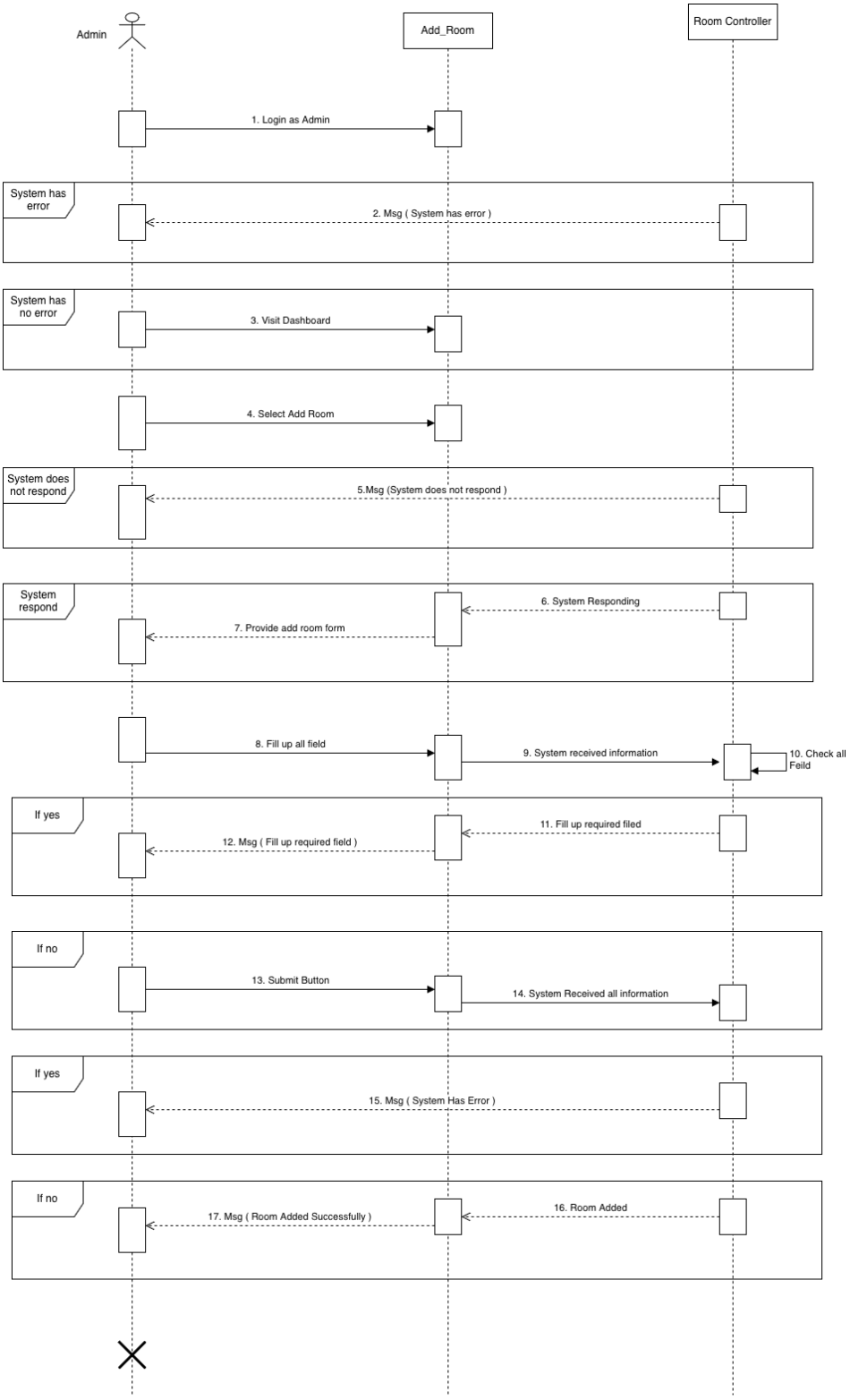


Figure 4.6: Sequence diagram for Update Information

For Update Room

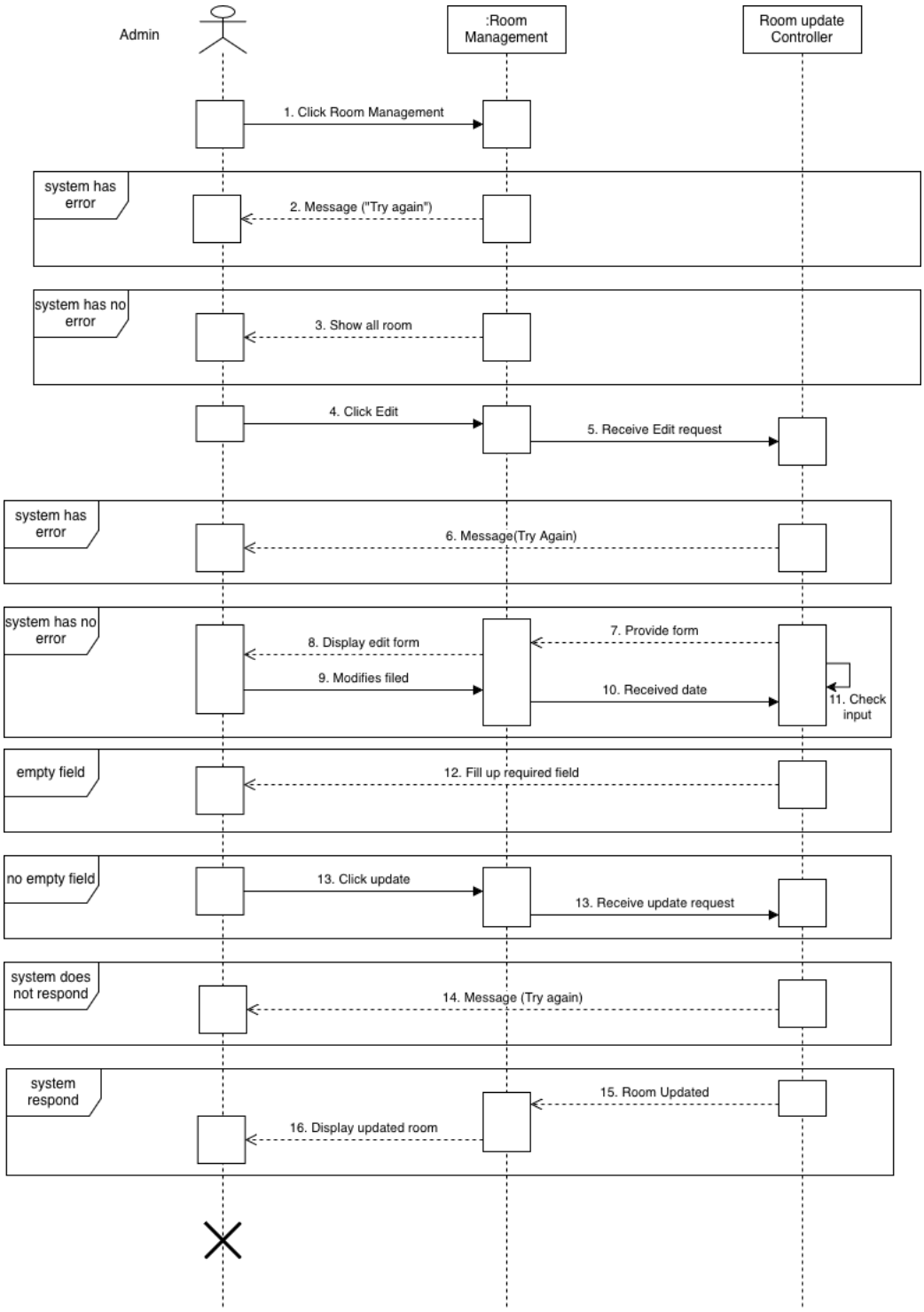


Figure 4.7: Sequence diagram for Blog Management

For Delete Room:

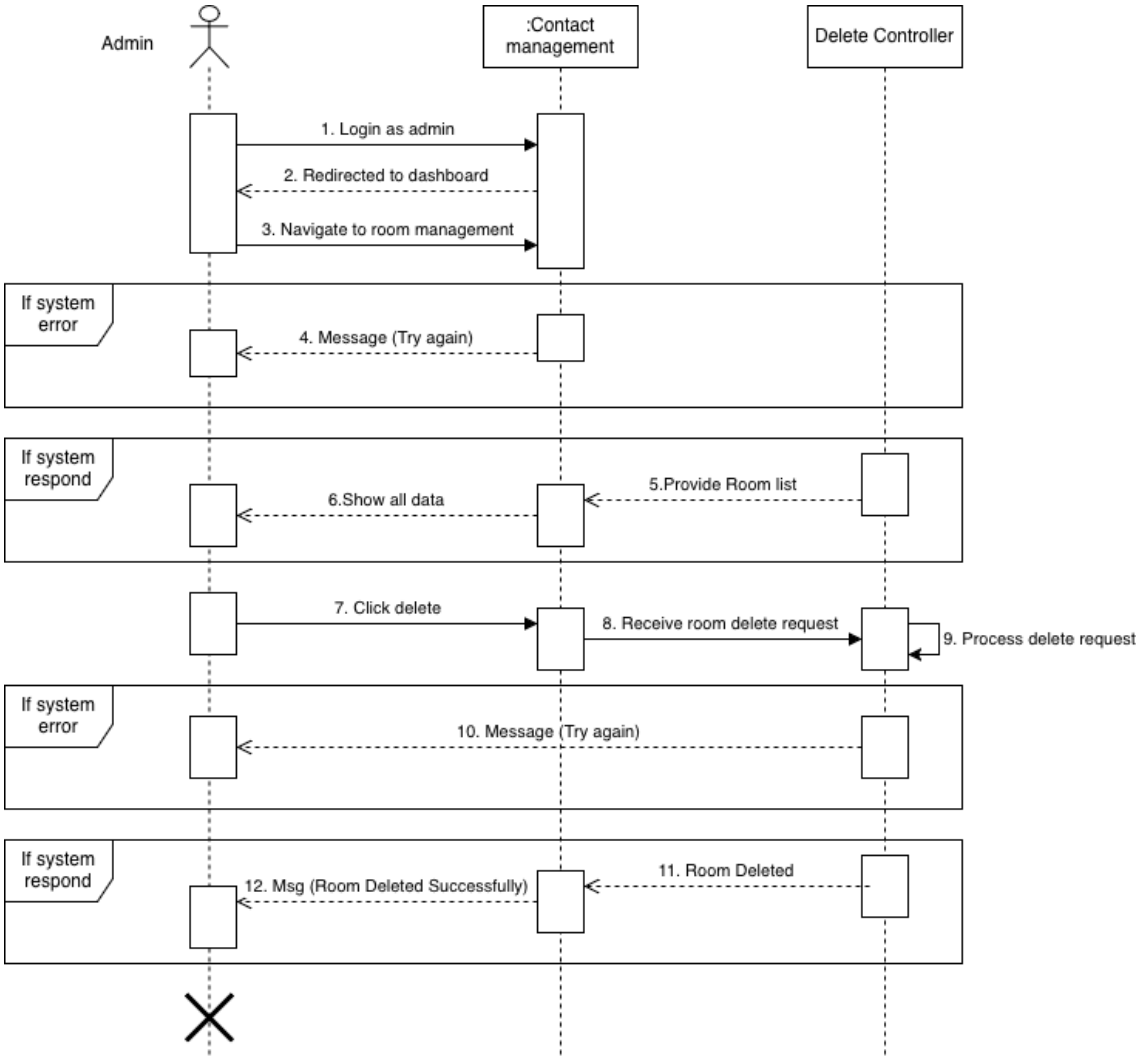


Figure 4.8: Sequence diagram for Delete Room

For Add to Cart

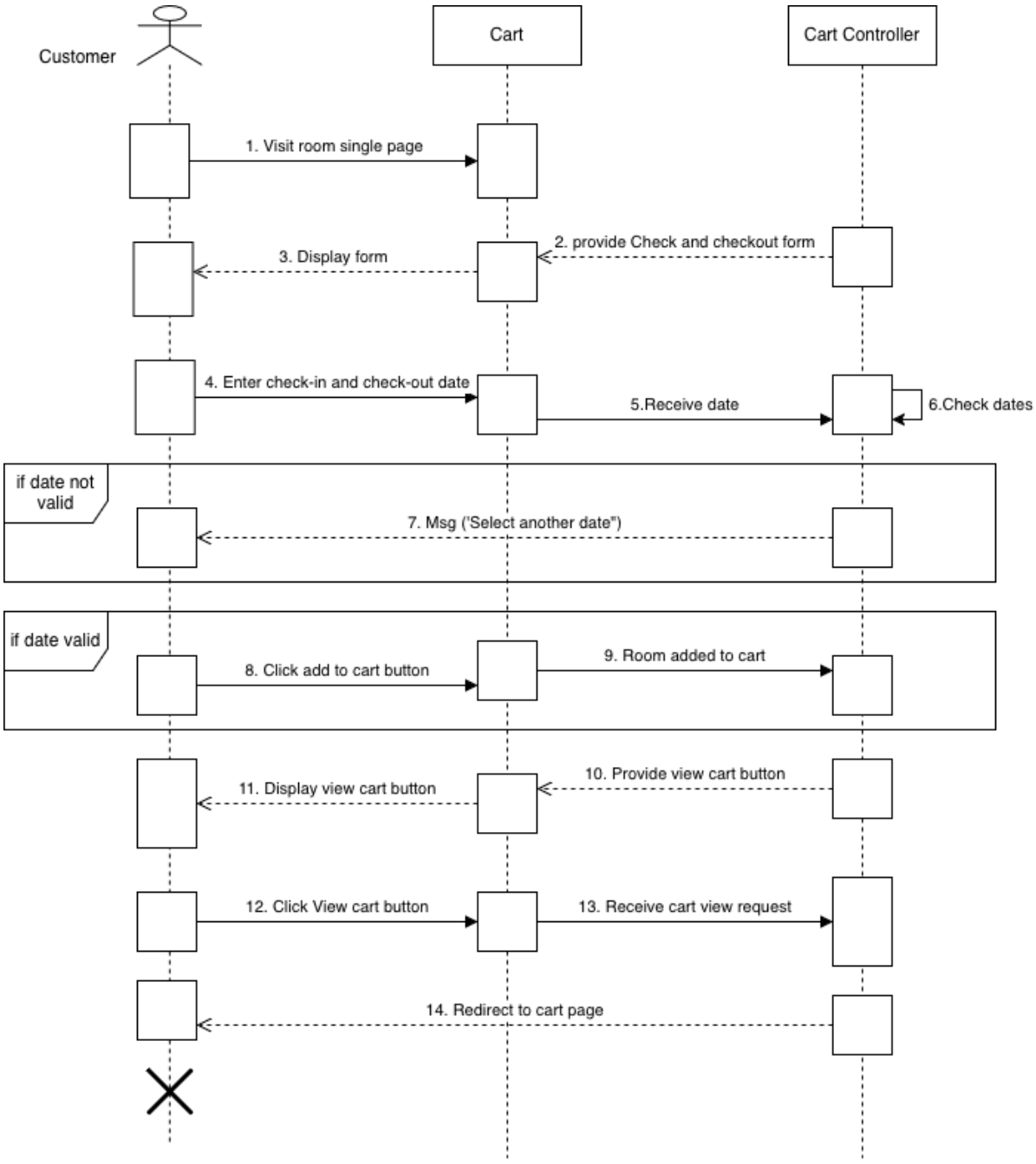


Figure 4.9: Sequence diagram for Add to Cart

For Make Booking

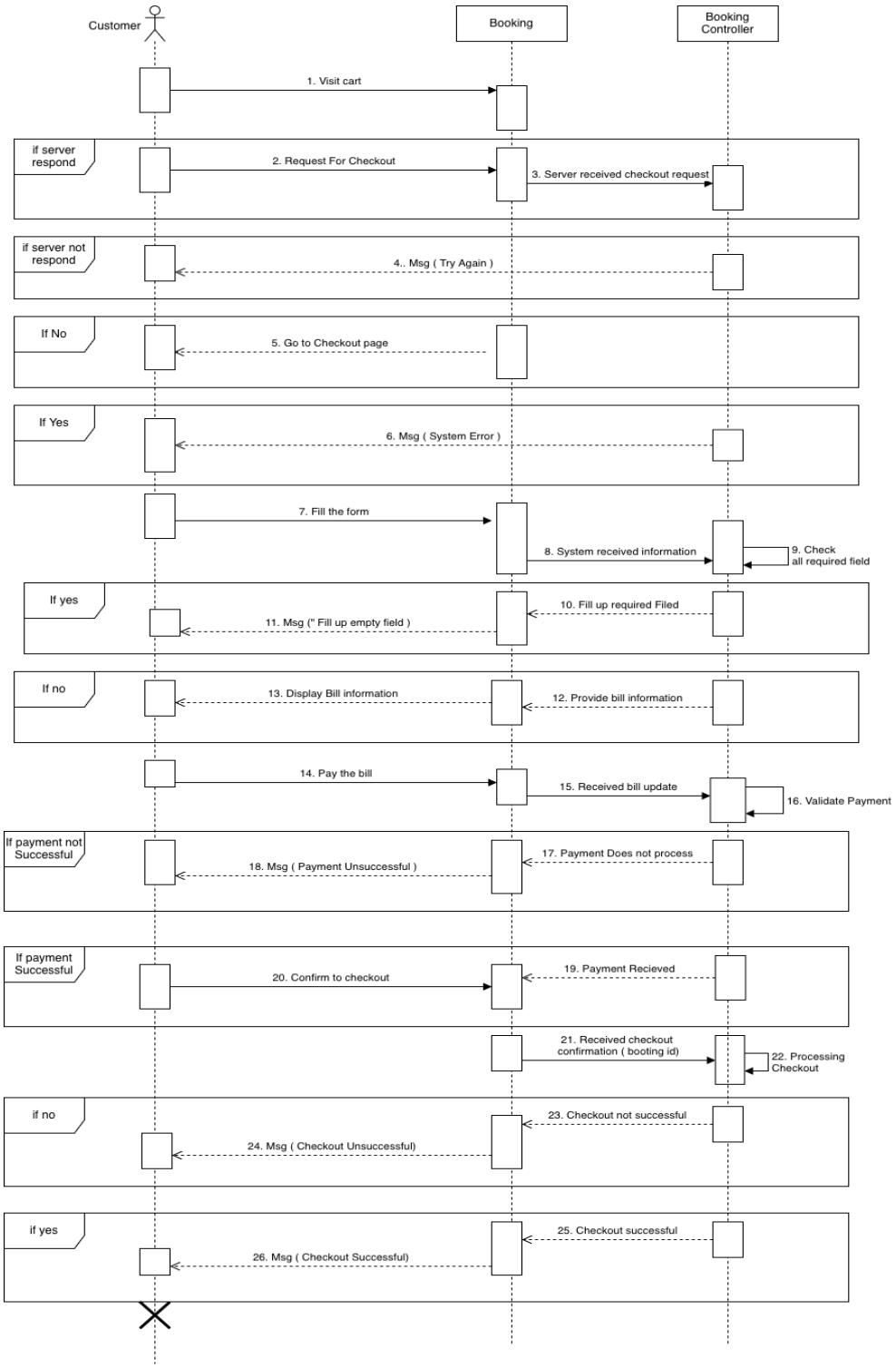


Figure 4.10: Sequence diagram for Make Booking

For Make Payment

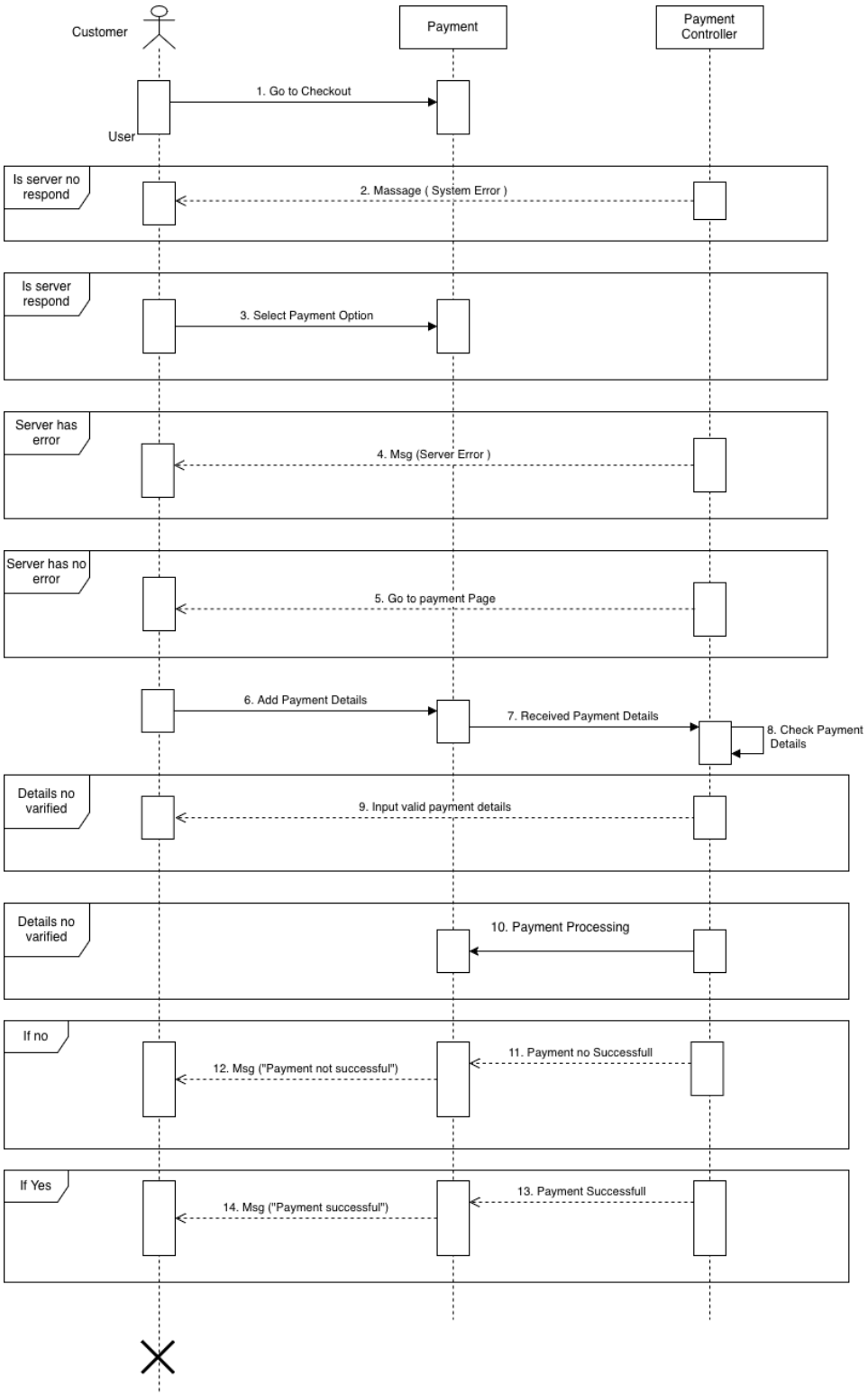


Figure 4.11: Sequence diagram for Make Payment

For Loyalty Reward

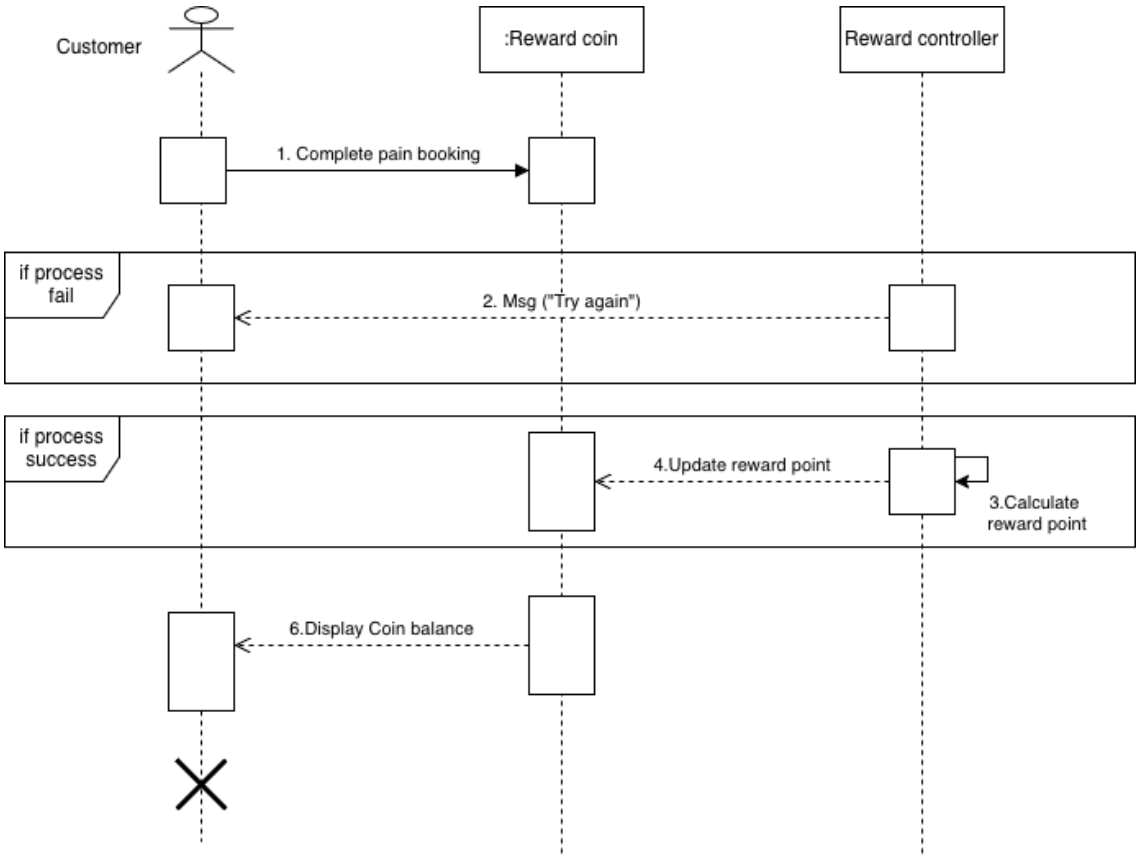


Figure 4.12: Sequence diagram for Loyalty Reward

For Loyalty Discount

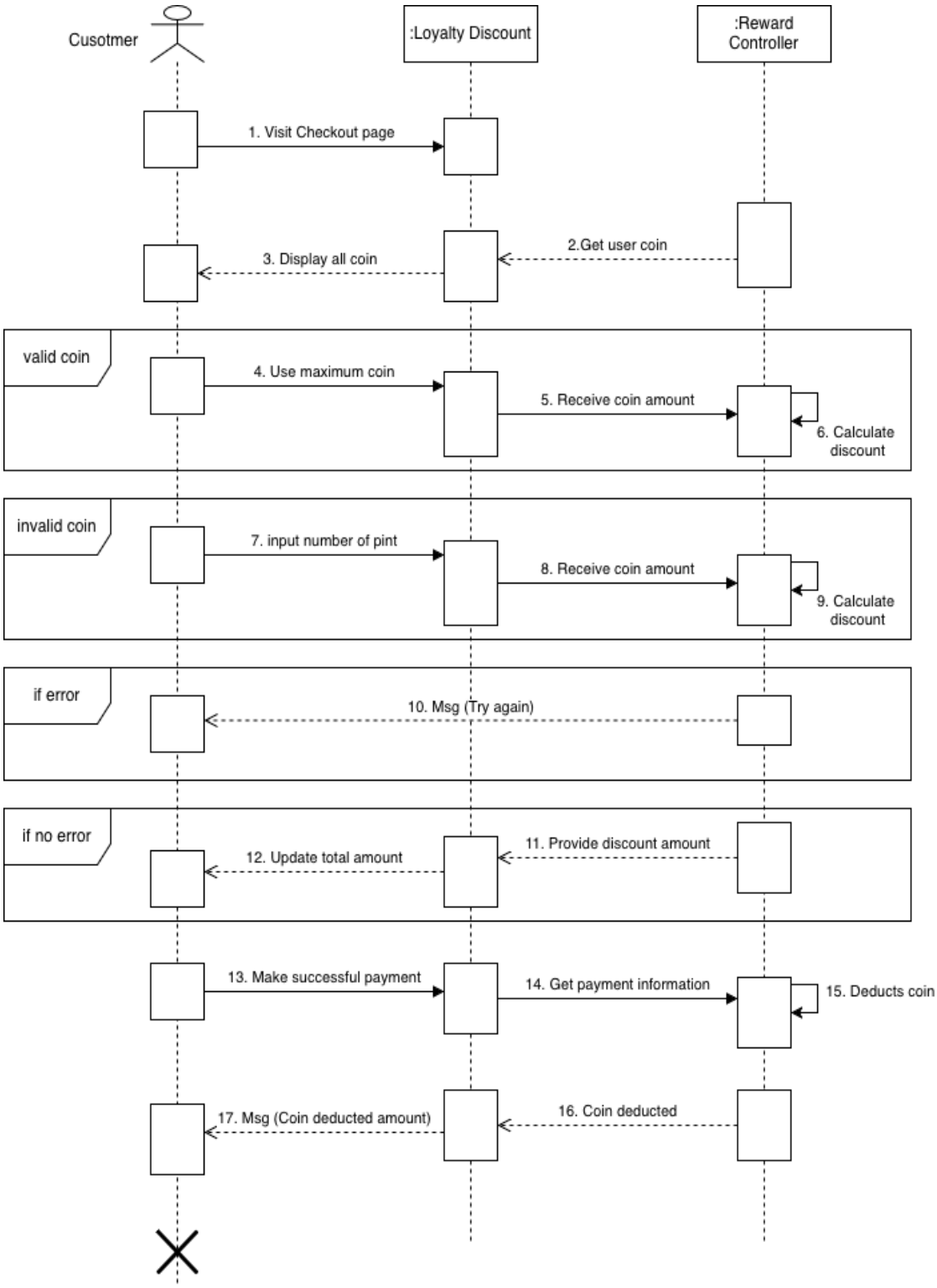


Figure 4.13: Sequence diagram for Loyalty Discount

For Cancel Booking

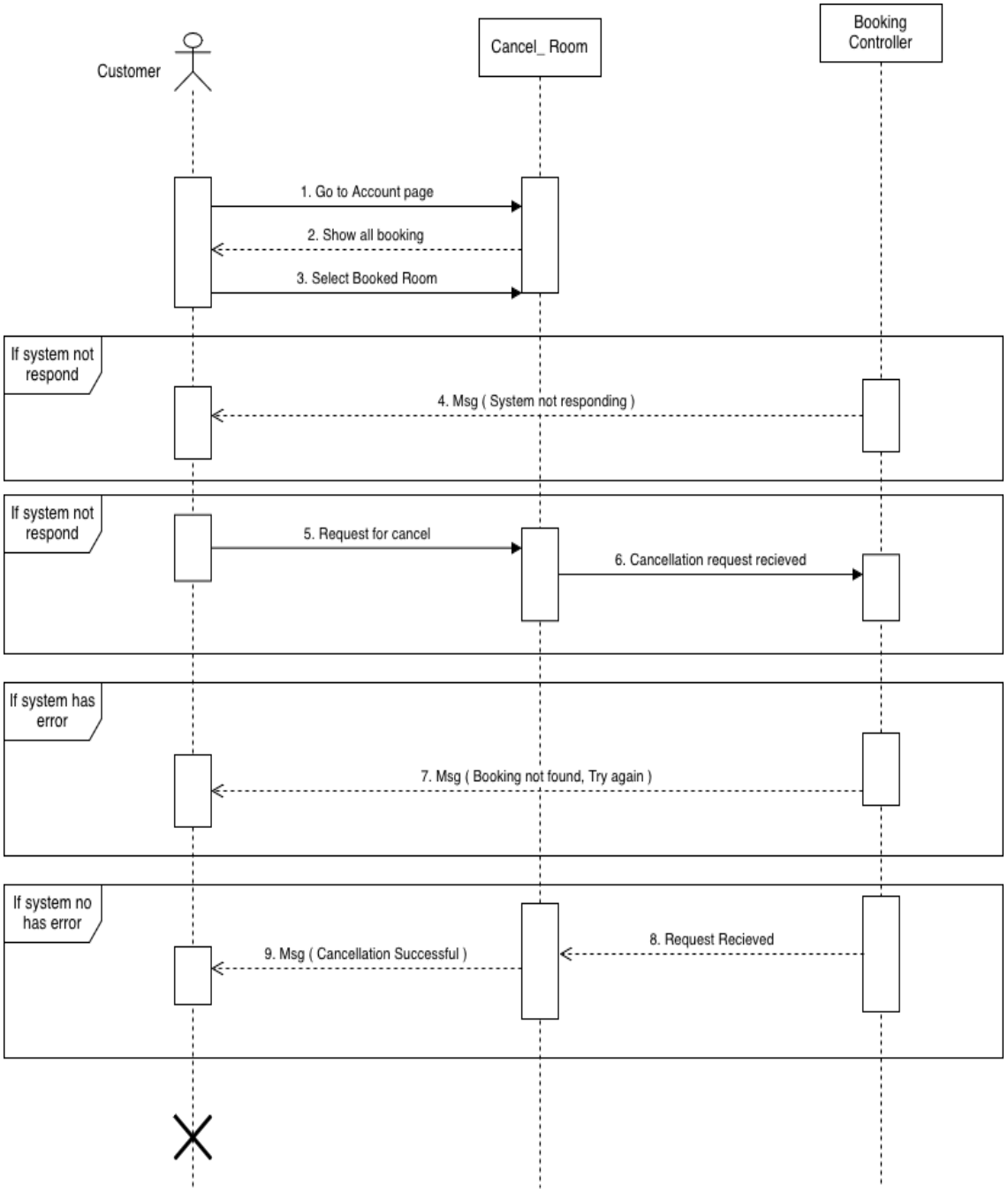


Figure 4.14: Sequence diagram for Cancel Booking

For Get Refund

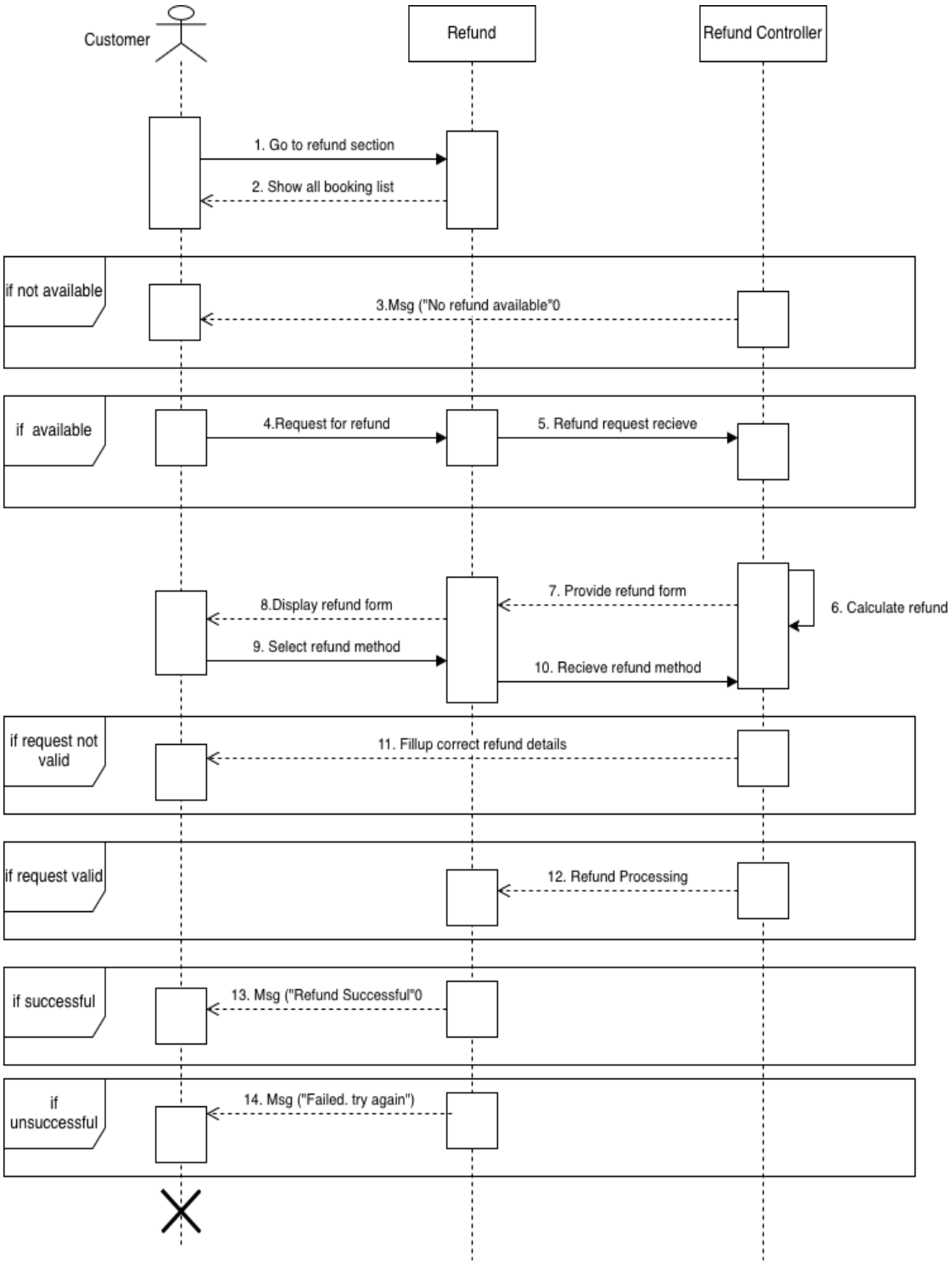


Figure 4.15: Sequence diagram for Get Refund

For Make Check-in

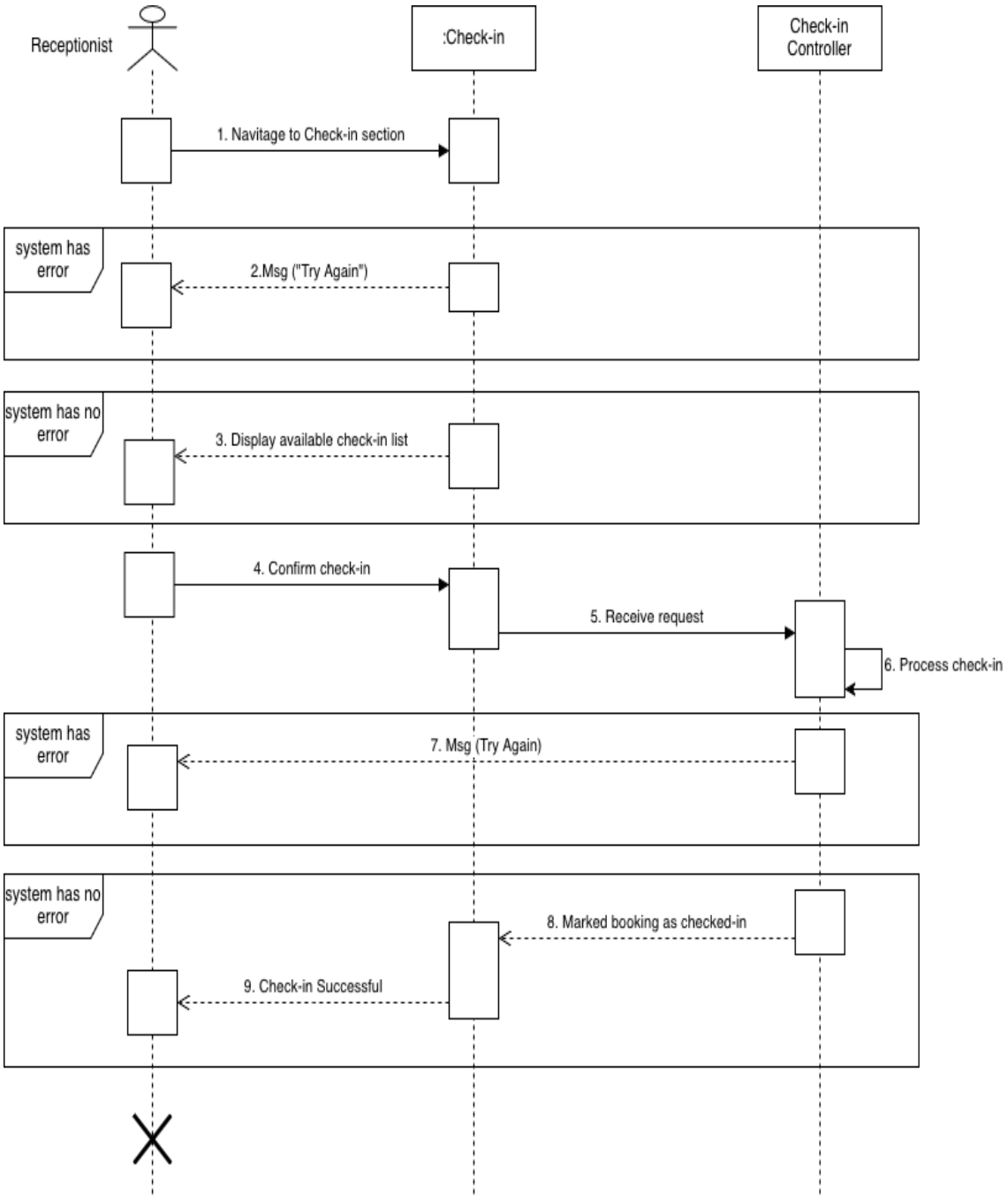


Figure 4.16: Sequence diagram for Make Check-in

For Make Check-out

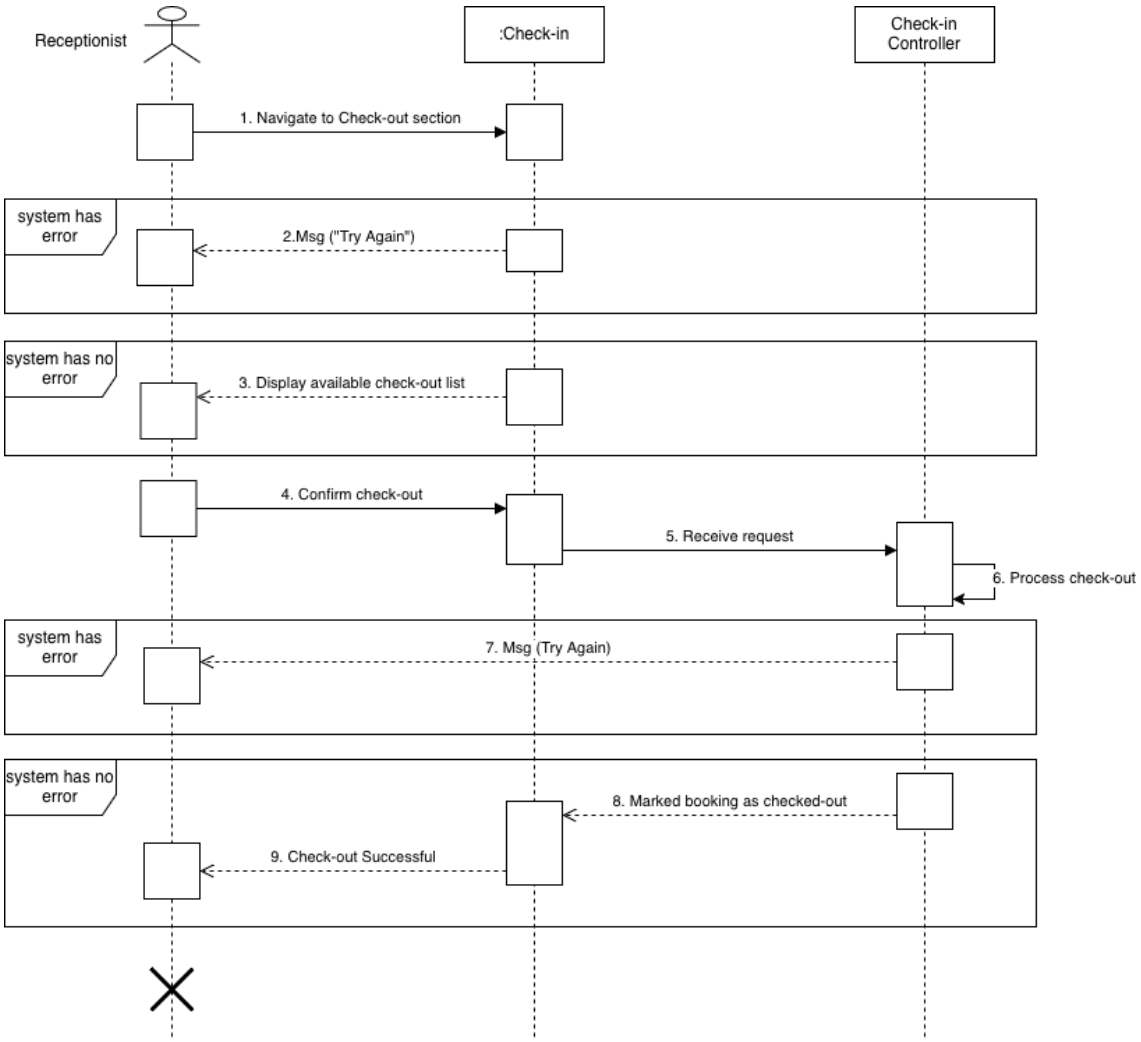


Figure 4.17: Sequence diagram for Make Check-out

For Add Blog

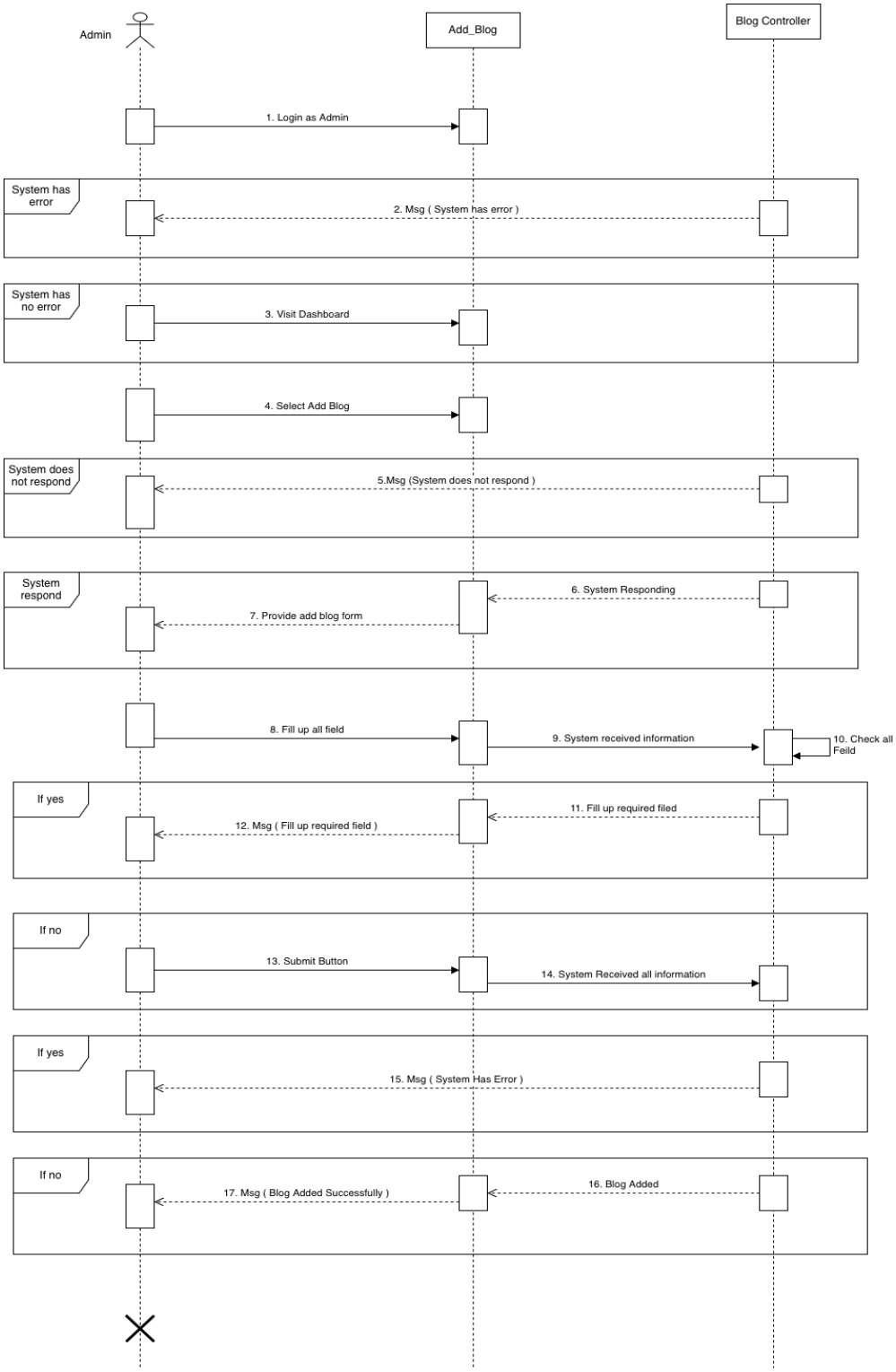


Figure 4.18: Sequence diagram for Add Blog

For Update Blog

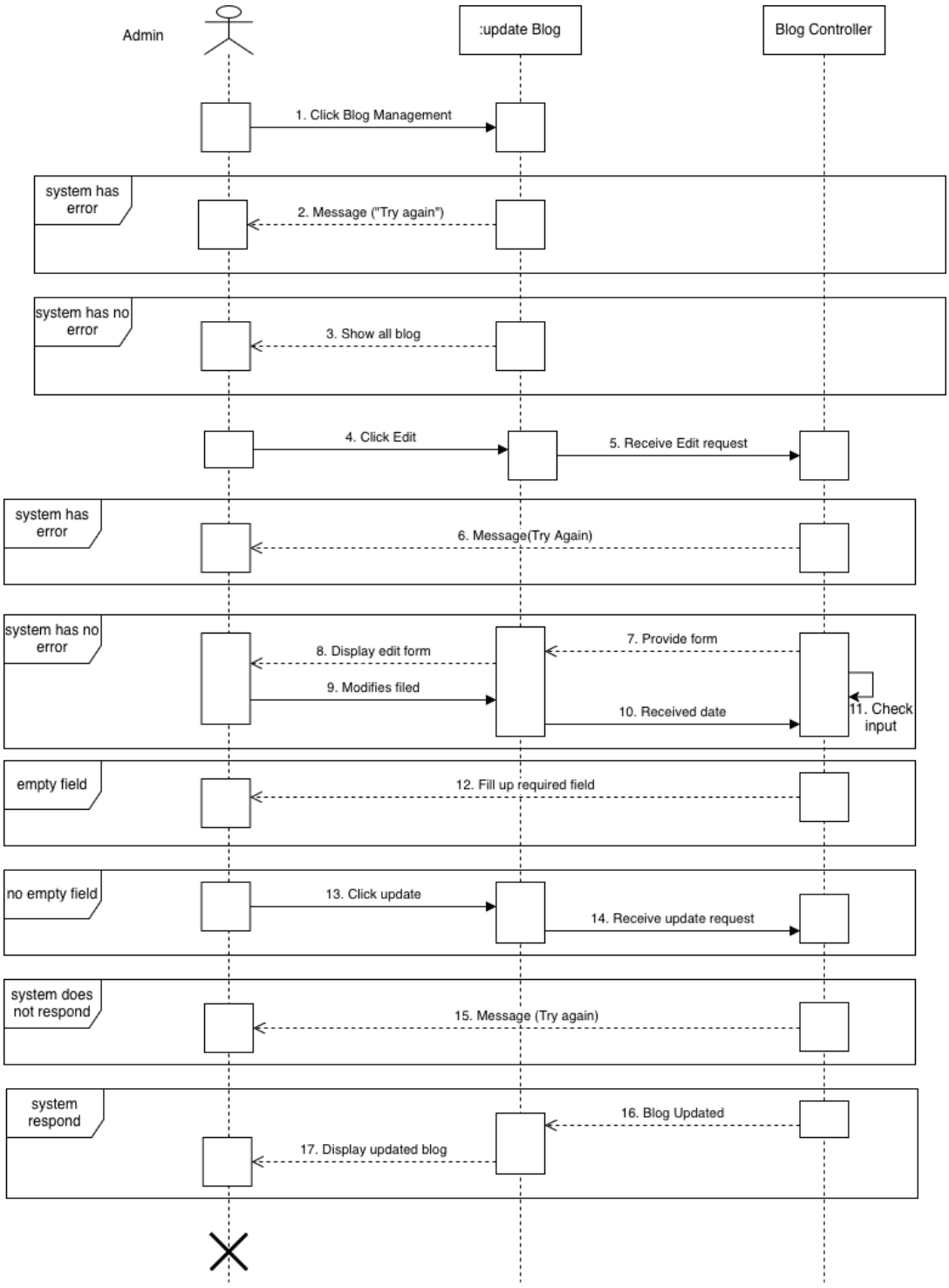


Figure 4.19: Sequence diagram for Update Blog

For Delete Blog:

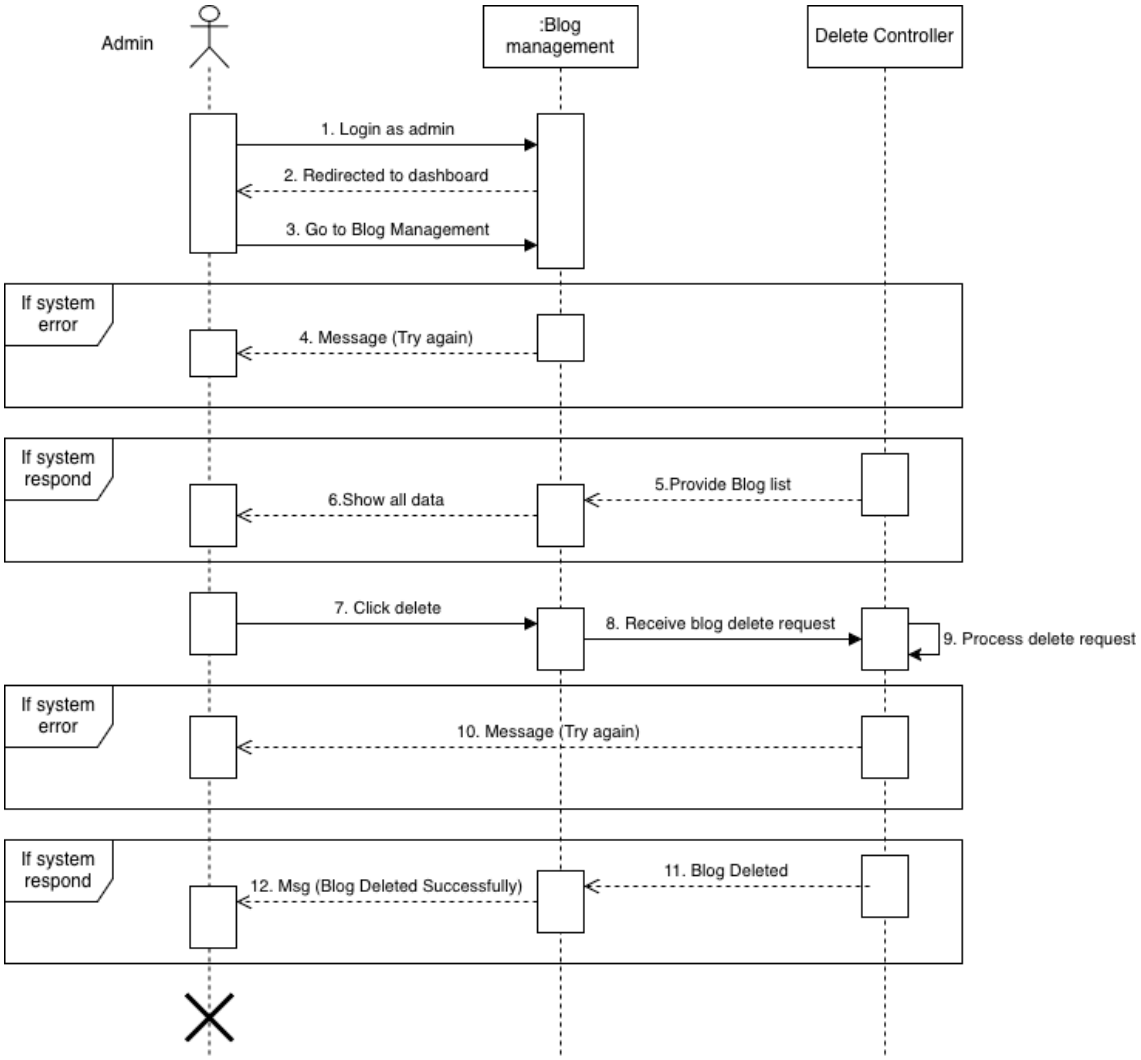


Figure 4.20: Sequence diagram for Delete Blog

For View Booking

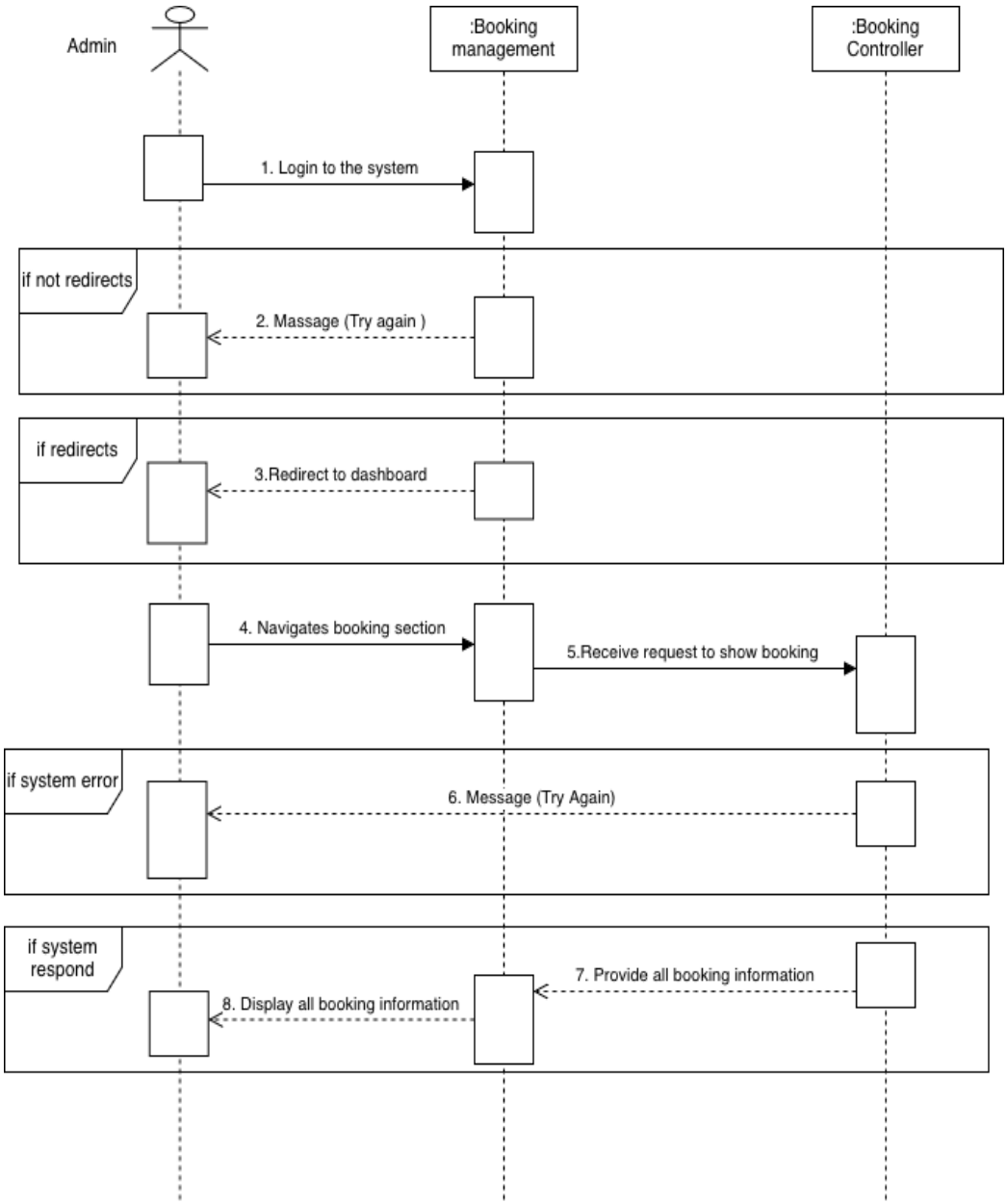


Figure 4.21: Sequence diagram for View Booking

For View Earning

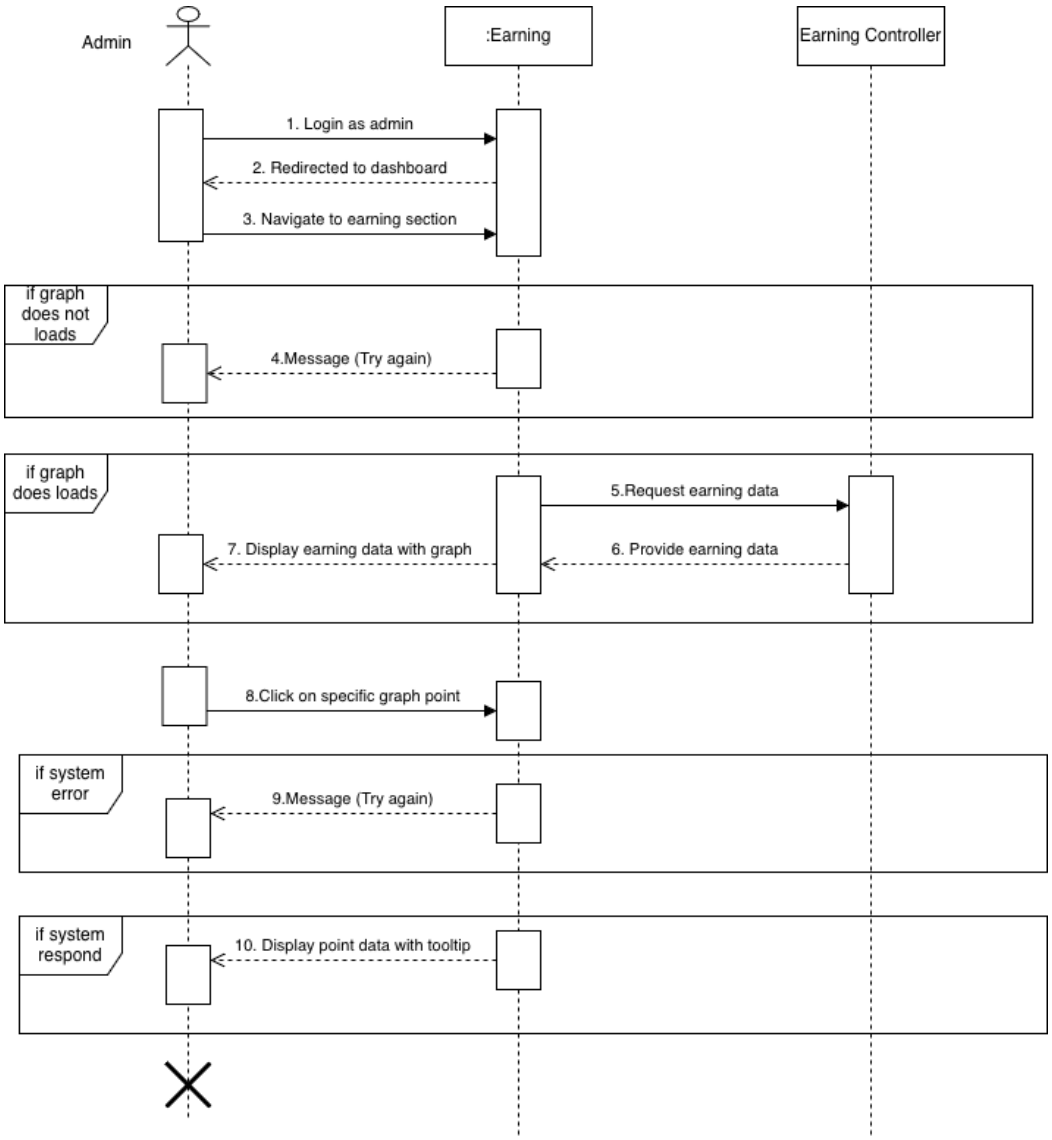


Figure 4.22: Sequence diagram for View Earning

For Resend Email

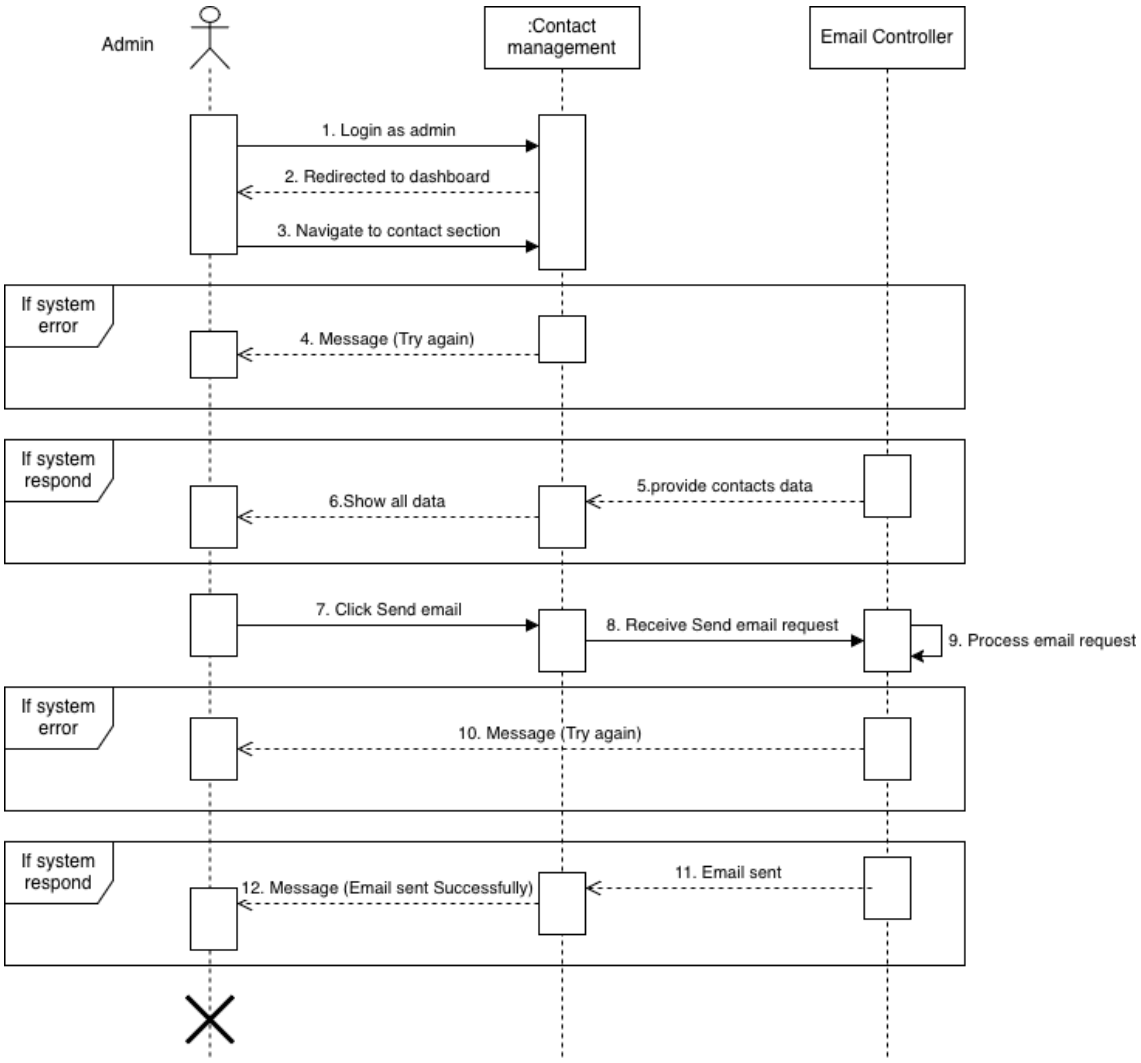


Figure 4.23: Sequence diagram for Resend Email

For Delete Contact

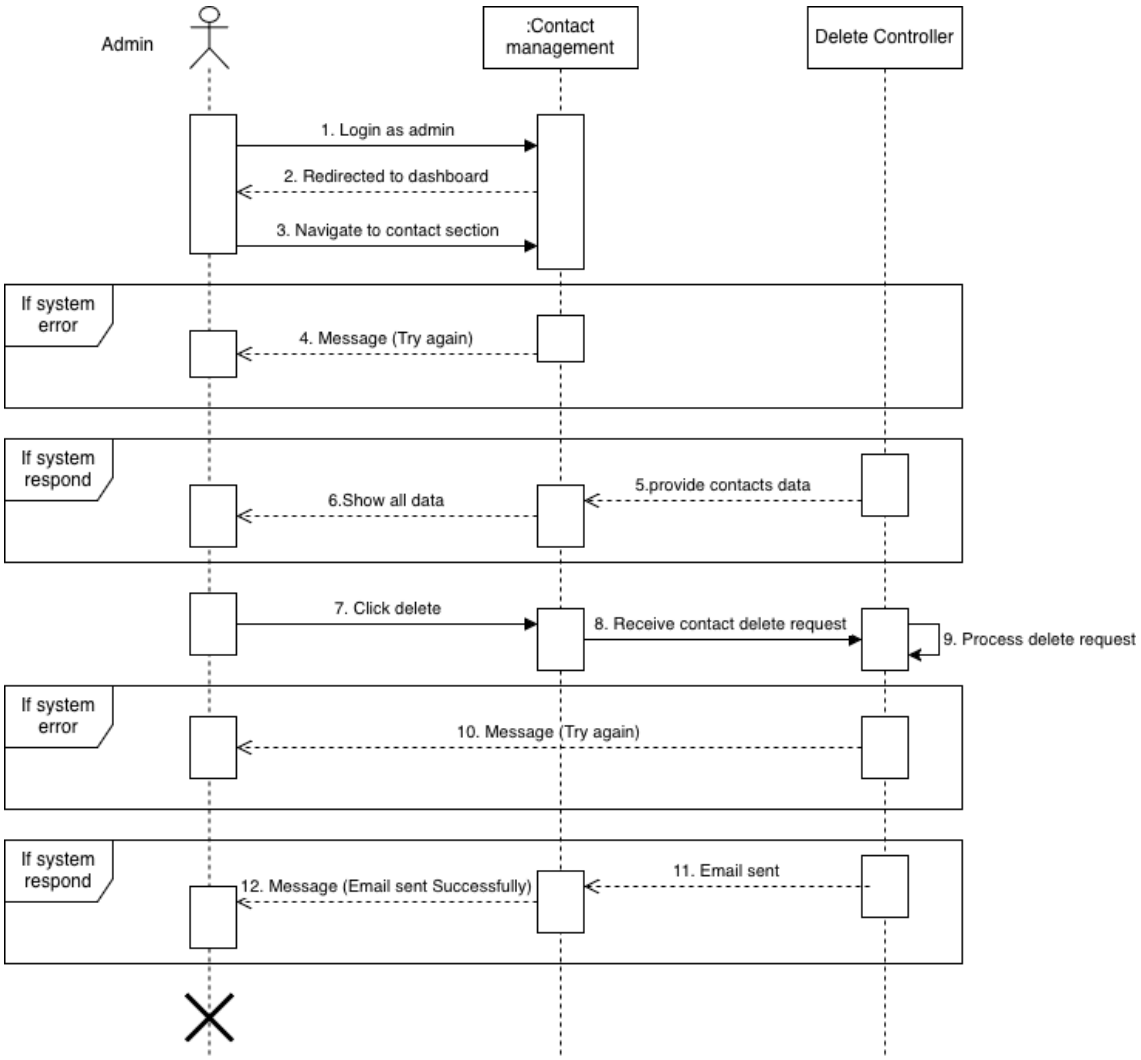


Figure 4.24: Sequence diagram for Delete Contact

For Logout:

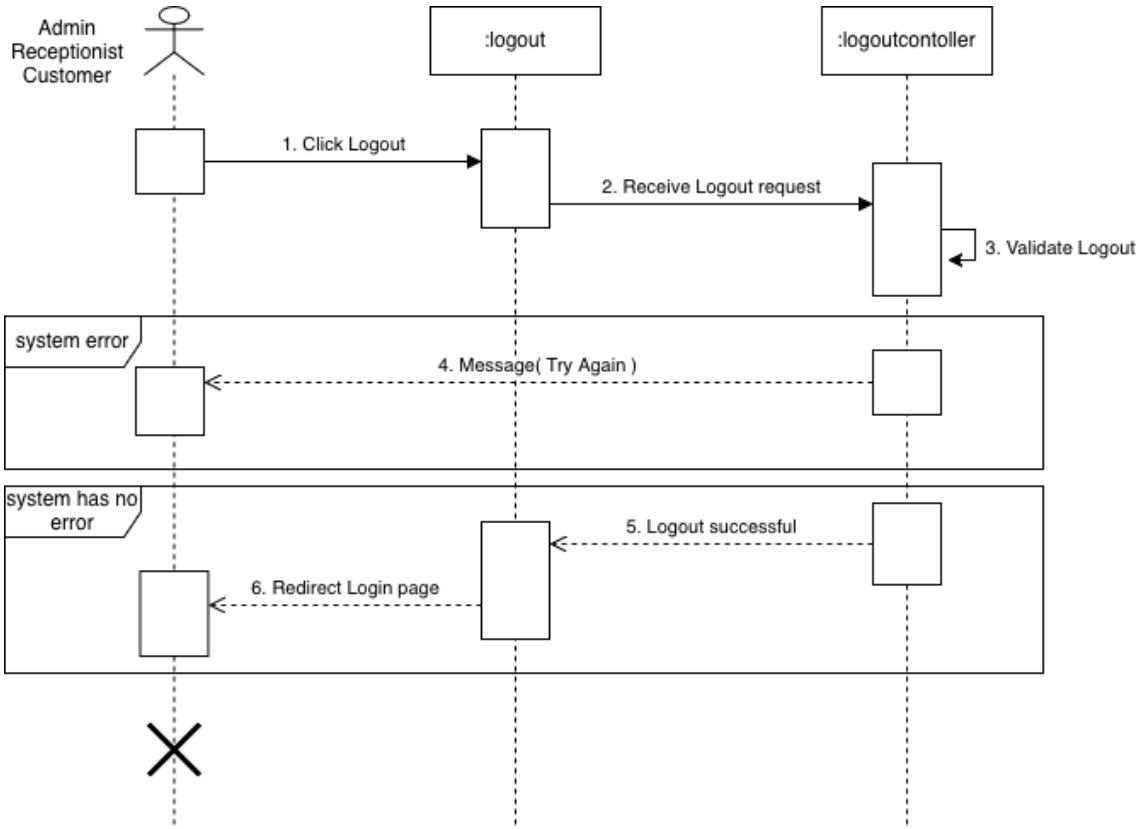


Figure 4.25: Sequence diagram for Logout

2.4.5 Class Diagram

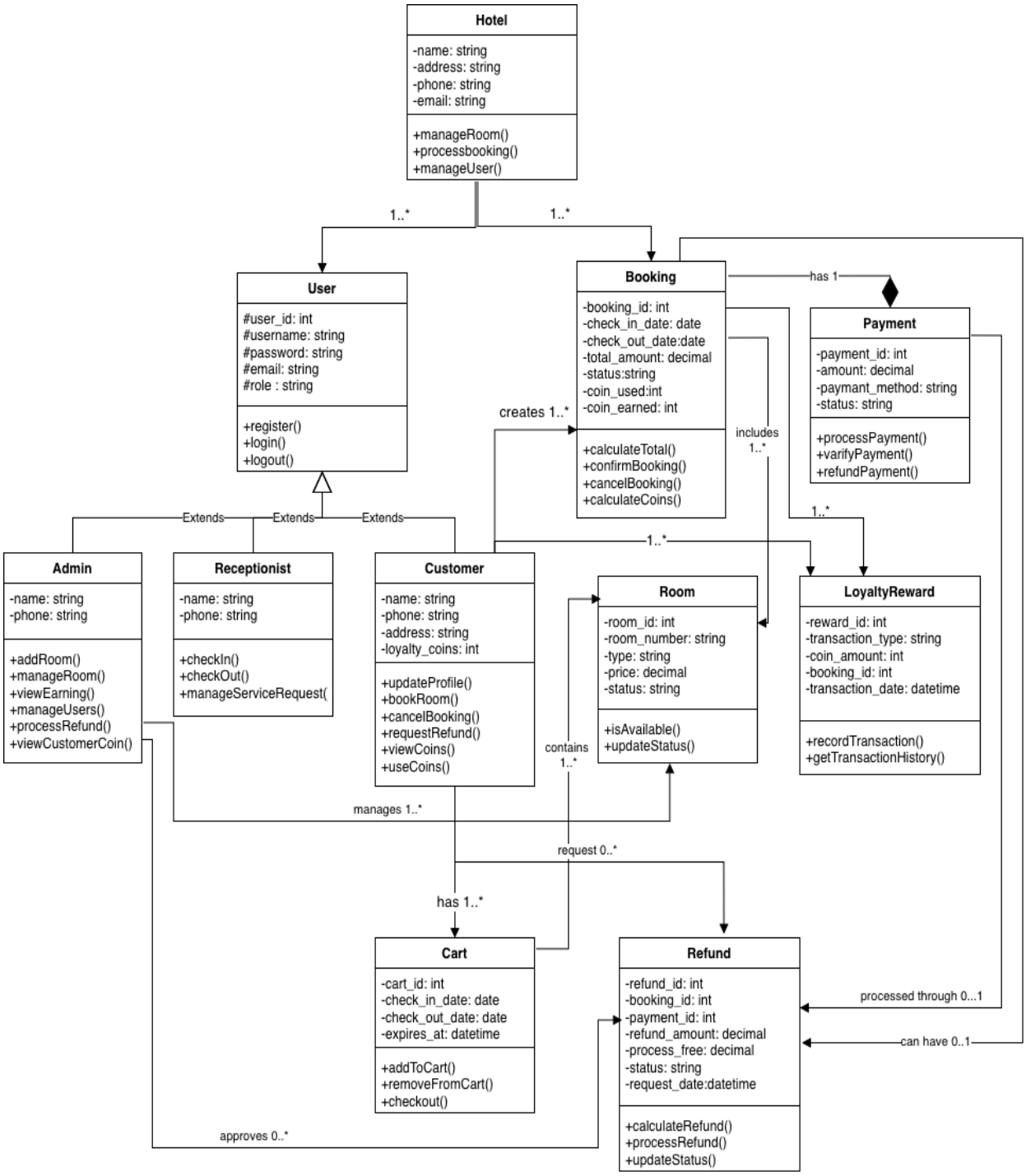


Figure 5: Class Diagram

2.4.6 ER Diagram

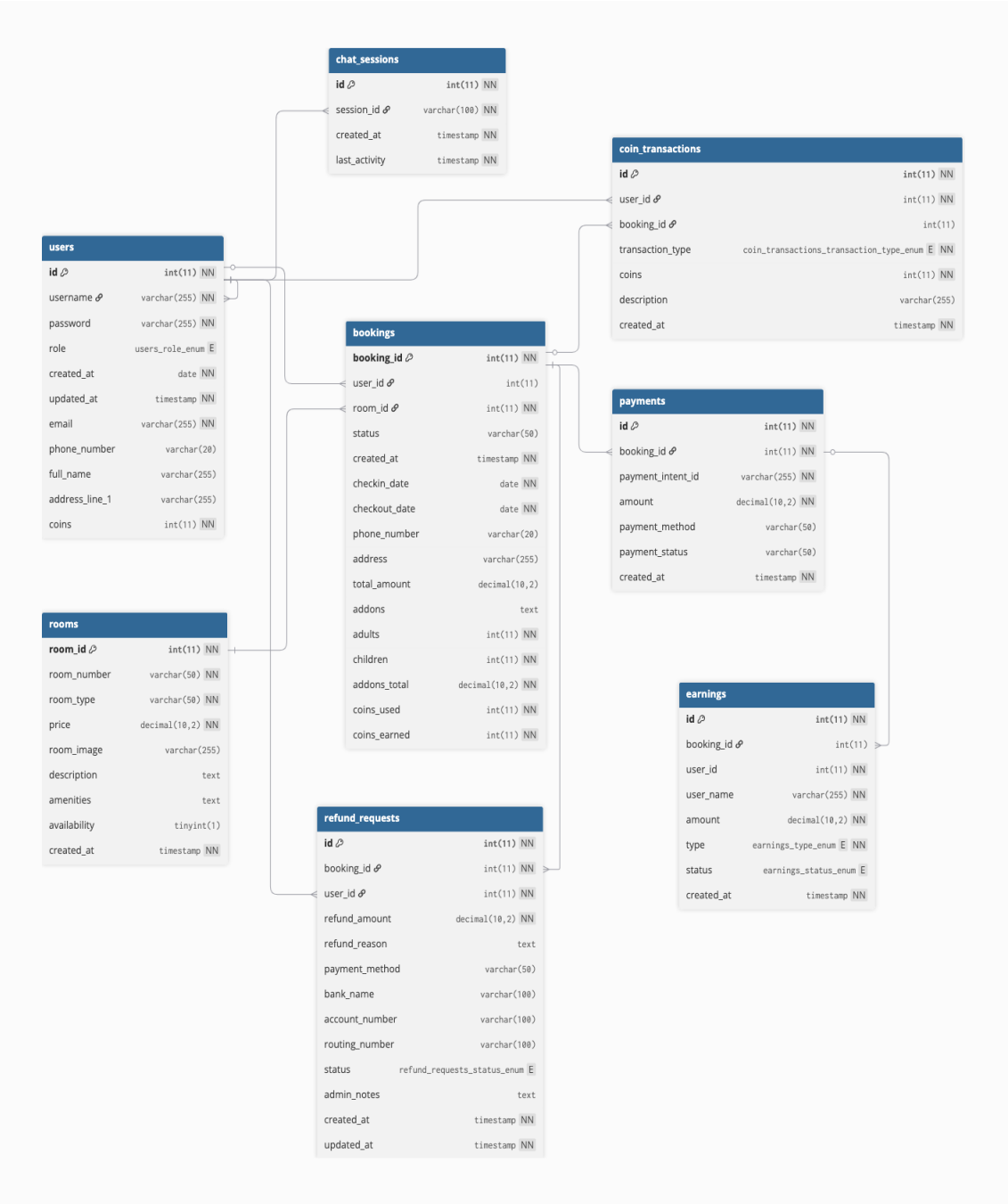


Figure 6: ER Diagram

2.5 Coding: Appendix A

This part contains the sample coding of the Hotel Management Project. It demonstrates the key functionalities, such as registration, login, booking, refund, loyalty reward, smart booking recommendation, and other system operations. The code is provided for reference purposes to demonstrate the login and structure.

A.1: Database Connection

```
<?php
// Error reporting for debugging
error_reporting(E_ALL);
ini_set('display_errors', 1);

// Database connection settings
$host = 'localhost';
$dbname = 'hotel_management';
$username = 'root';
$password = '';

// Establishing a connection to the database using PDO
try {
$pdo = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e) {
die("Connection failed: " . $e->getMessage());
}
?>
```

A.2: Registration

```
<?php
include('db.php');
include('font.php');
include('header.php');

$username = '';
$password = '';
$confirm_password = '';
$email = '';
$phone_number = '';
$full_name = '';
$address_line_1 = '';
$address_line_2 = '';
$city = '';
$state = '';
$country = '';
$pincode = '';

if (isset($_POST['register'])) {
$username = $_POST['username'];
$password = $_POST['password'];
$confirm_password = $_POST['confirm_password'];
```

```
$email = $_POST['email'];
$phone_number = $_POST['phone_number'] ?? '';
$full_name = $_POST['full_name'] ?? '';
$address_line_1 = $_POST['address_line_1'] ?? '';
$address_line_2 = $_POST['address_line_2'] ?? '';
$city = $_POST['city'] ?? '';
$state = $_POST['state'] ?? '';
$country = $_POST['country'] ?? '';
$pincode = $_POST['pincode'] ?? '';

// Validate passwords
if ($password !== $confirm_password) {
$error_message = "Passwords do not match!";
} else {
// Hash the password
$hashed_password = password_hash($password, PASSWORD_DEFAULT);

// Check if username or email already exists
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username OR email = :email");
$stmt->bindParam(':username', $username);
$stmt->bindParam(':email', $email);
$stmt->execute();

if ($stmt->rowCount() > 0) {
$error_message = "Username or email already taken!";
} else {
// Insert new user with all address fields
$stmt = $pdo->prepare("INSERT INTO users (username, password, email, phone_number, full_name,
address_line_1, address_line_2, city, state, country, pincode, role) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
$stmt->execute([$username, $hashed_password, $email, $phone_number, $full_name, $address_line_1,
$address_line_2, $city, $state, $country, $pincode, 'user']);
$success_message = "Registration successful!";
// Auto-login after registration
$user_id = $pdo->lastInsertId();
$_SESSION['user_id'] = $user_id;
$_SESSION['username'] = $username;
$_SESSION['role'] = 'user';
header('Location: user/dashboard.php');
exit();
}}}?>
```

A.3: Login

```
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
$username = $_POST['username'];
$password = $_POST['password'];

// Query the database to check if the username exists
$stmt = $pdo->prepare("SELECT * FROM users WHERE username = :username");
$stmt->bindParam(':username', $username);
$stmt->execute();
$user = $stmt->fetch(PDO::FETCH_ASSOC);

// If user exists and password matches
```

```
if ($user && password_verify($password, $user['password'])) {
$_SESSION['user_id'] = $user['id'];
$_SESSION['username'] = $user['username'];
$_SESSION['role'] = $user['role'];

if ($user['role'] == 'admin') {
header('Location: dashboard.php');
}elseif($user['role'] == 'receptionist'){
header('Location: receptionist/index.php');
}else {
header('Location: user/dashboard.php');
}
exit(); }}
```

A.4: Get Smart Room recommendation

```
<?php
session_start();
include('../db.php');
header('Content-Type: application/json');

// Enable error reporting for debugging (remove in production)
error_reporting(E_ALL);
ini_set('display_errors', 0);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
try {
// Get and validate preferences from POST
$group_size = intval($_POST['group_size'] ?? 1);
$room_types = $_POST['room_types'] ?? [];
$price_range = intval($_POST['price_range'] ?? 120);
// Validate inputs
if ($group_size < 1 || $group_size > 10) {
$group_size = 1;
}
if ($price_range < 20 || $price_range > 200) {
$price_range = 120;
}
// Validate room types (ensure they are valid)
$valid_room_types = ['Single Room', 'Double Room', 'Suite', 'Delux Room'];
$room_types = array_intersect($room_types, $valid_room_types);
// Update session preferences
$_SESSION['preferred_room_types'] = $room_types;
$_SESSION['preferred_price_range'] = [0, $price_range];
$_SESSION['group_size'] = $group_size;
// Get all available rooms with their first image from room_images table - USING THE SAME QUERY AS ADMIN
$stmt = $pdo->prepare("
SELECT r.*,
(SELECT ri.image_path FROM room_images ri WHERE ri.room_id = r.room_id LIMIT 1) as primary_image
FROM rooms r
WHERE r.availability = 1
ORDER BY r.created_at DESC
");
$stmt->execute();
$all_rooms = $stmt->fetchAll(PDO::FETCH_ASSOC);
// Calculate recommendations using the same algorithm
```

```
$recommended_rooms = getSmartRecommendations($all_rooms, $room_types, [0, $price_range],
$group_size);
echo json_encode([
'success' => true,
'recommendations' => $recommended_rooms,
'preferences' => [
'room_types' => $room_types,
'max_price' => $price_range,
'group_size' => $group_size
],
'stats' => [
'total_rooms' => count($all_rooms),
'recommended_count' => count($recommended_rooms)
]
]);
} catch (Exception $e) {
error_log("Error getting recommendations: " . $e->getMessage());
echo json_encode([
'success' => false,
'message' => 'Error getting recommendations: ' . $e->getMessage()
]);
}
} else {
echo json_encode([
'success' => false,
'message' => 'Invalid request method. Please use POST.'
]);
}

// Recommendation algorithm function (same as in index.php)
function getSmartRecommendations($rooms, $preferred_types, $preferred_price_range, $group_size) {
$recommendations = [];
if (empty($rooms)) {
return $recommendations;
}
foreach ($rooms as $room) {
$match_score = 0;
$match_reasons = [];
// Price match (40% weight)
$price_score = calculatePriceScore($room['price'], $preferred_price_range[1]);
$match_score += $price_score * 0.4;
if ($price_score > 0.7) {
$match_reasons[] = 'Good Price';
}
// Room type match (30% weight) - Now checks against multiple preferred types
$type_score = calculateTypeScore($room['room_type'], $preferred_types, $group_size);
$match_score += $type_score * 0.3;
if ($type_score > 0.8) {
$match_reasons[] = 'Perfect Room Type';
} elseif ($type_score > 0.5) {
$match_reasons[] = 'Suitable Type';
}
// Group size compatibility (30% weight)
$size_score = calculateSizeScore($room['room_type'], $group_size);
$match_score += $size_score * 0.3;
if ($size_score > 0.8) {
$match_reasons[] = 'Ideal for Group';
} elseif ($size_score > 0.6) {
```

```
$match_reasons[] = 'Good Capacity';
}
// Add to recommendations if score is above threshold
if ($match_score > 0.3) {
$room['match_score'] = round($match_score, 2);
$room['match_reasons'] = $match_reasons;
$recommendations[] = $room;
}
}
// Sort by match score (highest first)
usort($recommendations, function($a, $b) {
return $b['match_score'] <=> $a['match_score'];
});
// LIMIT TO 3 RECOMMENDATIONS
return array_slice($recommendations, 0, 3);
}

function calculatePriceScore($room_price, $max_preferred_price) {
$room_price = floatval($room_price);
$max_preferred_price = floatval($max_preferred_price);
if ($room_price <= $max_preferred_price) {
return max(0.1, 1 - ($room_price / $max_preferred_price) * 0.3);
} else {
$excess = $room_price - $max_preferred_price;
return max(0, 1 - ($excess / $max_preferred_price));
}}

function calculateTypeScore($room_type, $preferred_types, $group_size) {
if (empty($preferred_types)) {
return 0.7;
}
// Exact match with any preferred type
if (in_array($room_type, $preferred_types)) {
return 1.0;
}
$similar_types = [
'Single Room' => ['Double Room'],
'Double Room' => ['Single Room', 'Suite'],
'Suite' => ['Double Room', 'Delux Room'],
'Delux Room' => ['Suite']
];
// Check if room type is similar to any preferred type
foreach ($preferred_types as $preferred_type) {
if (isset($similar_types[$preferred_type]) && in_array($room_type, $similar_types[$preferred_type])) {
return 0.6;}}
return 0.3;}

function calculateSizeScore($room_type, $group_size) {
$capacity_map = [
'Single Room' => 1,
'Double Room' => 2,
'Suite' => 3,
'Delux Room' => 4
];
$capacity = $capacity_map[$room_type] ?? 2;
$group_size = intval($group_size);
if ($capacity >= $group_size) {
```

```
if ($capacity == $group_size) return 1.0;
if ($capacity == $group_size + 1) return 0.8;
return 0.6;
} else {
return max(0.1, 1 - ($group_size - $capacity) * 0.3);}}?>
```

A.5: Chatbot

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

define('DB_HOST', 'localhost');
define('DB_NAME', 'hotel_management');
define('DB_USER', 'root');
define('DB_PASS', '');

try {
$pdo = new PDO("mysql:host=" . DB_HOST . ";dbname=" . DB_NAME . ";charset=utf8", DB_USER, DB_PASS);
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
} catch(PDOException $e) {
die("Database connection failed: " . $e->getMessage());
}

// FAQ responses for Roberto Hotel
$faq_responses = [
'greeting' => [
'text' => "Hello! Welcome to Roberto 👋 I'm your assistant. How can I help you today?",
'options' => ['Rooms', 'Pricing', 'Support', 'Technical Help']
],
'rooms' => [
'text' => "We offer various types of rooms. You can check our rooms by clicking Rooms from the top menu.",
'options' => ['Double', 'Delux', 'Super Delux', 'All Rooms', 'Back to Main Menu']
],
'double' => [
'text' => "Our Double Rooms:\n\n• Comfortable double bed\n• Free WiFi\n• Air conditioning\n• Private
bathroom\n• Starting from $80/night",
'options' => ['Book Now', 'View Photos', 'Compare Rooms', 'Back to Rooms']
],
'delux' => [
'text' => "Our Delux Rooms:\n\n• King size bed\n• Free WiFi + Breakfast\n• Air conditioning\n• Luxury
bathroom\n• Mini fridge\n• Starting from $120/night",
'options' => ['Book Now', 'View Photos', 'Compare Rooms', 'Back to Rooms']
],
'superdelux' => [
'text' => "Our Super Delux Rooms:\n\n• King size bed + Sofa\n• Free WiFi + Breakfast\n• Air conditioning\n•
Premium bathroom\n• Mini bar + TV\n• Balcony view\n• Starting from $180/night",
'options' => ['Book Now', 'View Photos', 'Compare Rooms', 'Back to Rooms']
],
'allrooms' => [
'text' => "All our room types:\n\n• Double Room: $80-100/night\n• Delux Room: $120-150/night\n• Super
Delux: $180-250/night\n• Suite: $300-400/night\n\nCheck our website for current availability!",
'options' => ['Double Room', 'Delux Room', 'Super Delux', 'Main Menu']
],
'support' => [
```

```
'text' => "Our support team is here to help!\n\n📞 Call: +8801741835718\n📧 Email:
admin@roberto.com\n💬 Live Chat Available",
'options' => ['Technical Issues', 'Billing', 'Account Help', 'Back']
],
'pricing' => [
'text' => "You can easily check our pricing from the website. Mostly pricing starts from $80 to $400. Check our
Pricing page for detailed rates!",
'options' => ['Double Room Price', 'Delux Room Price', 'Super Delux Price', 'All Prices', 'Back']
],
'technical' => [
'text' => "For technical assistance:\n\n🔧 Check our documentation\n🛠 Submit a support ticket\n📋
Browse our knowledge base",
'options' => ['WiFi Issues', 'Booking Problems', 'Payment Issues', 'Back']
],
'booking' => [
'text' => "To make a booking:\n\n1. Visit our website\n2. Select dates and room type\n3. Fill guest
information\n4. Complete payment\n\nNeed help? Call +8801741835718",
'options' => ['Check Availability', 'Special Offers', 'Contact Support', 'Back']
],
'location' => [
'text' => "Roberto Hotel Location:\n\n📍 City Center\n📍 Near shopping district\n📍 Easy access to
attractions\n\nVisit our website for exact address and directions!",
'options' => ['Get Directions', 'Transportation', 'Nearby Attractions', 'Back']
]
];
?>
```

A.6: Add Room

```
<?php
// add_room_ajax.php
session_start();
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Unauthorized']);
exit();
}

include('db.php');

// Enable error reporting
error_reporting(E_ALL);
ini_set('display_errors', 1);

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Invalid request method']);
exit();
}

// Get form data
$room_number = trim($_POST['room_number'] ?? '');
$room_type = $_POST['room_type'] ?? '';
$price = floatval($_POST['price'] ?? 0);
$description = trim($_POST['description'] ?? '');
$amenities = trim($_POST['amenities'] ?? '');
```

```
// Validate required fields
if (empty($room_number) || empty($room_type) || empty($price)) {
echo json_encode(['success' => false, 'message' => 'All required fields must be filled']);
exit();
}

// Validate room type
$allowed_types = ['single', 'double', 'suite', 'delux', 'family'];
if (!in_array($room_type, $allowed_types)) {
echo json_encode(['success' => false, 'message' => 'Invalid room type selected']);
exit();
}

// Check if files were uploaded
if (!isset($_FILES['room_images']) || empty($_FILES['room_images']['name'][0])) {
echo json_encode(['success' => false, 'message' => 'Please select at least one room image']);
exit();
}

try {
// Check if room number already exists
$check_stmt = $pdo->prepare("SELECT room_id FROM rooms WHERE room_number = ?");
$check_stmt->execute([$room_number]);
if ($check_stmt->fetch()) {
echo json_encode(['success' => false, 'message' => 'Room number already exists']);
exit();
}

// Insert room into database
$stmt = $pdo->prepare("INSERT INTO rooms (room_number, room_type, price, description, amenities)
VALUES (?, ?, ?, ?, ?)");
$stmt->execute([$room_number, $room_type, $price, $description, $amenities]);
$room_id = $pdo->lastInsertId();

// Handle multiple image uploads
$uploaded_images = [];
$images = $_FILES['room_images'];
$upload_dir = "uploads/rooms/";
// Create uploads directory if it doesn't exist
if (!is_dir($upload_dir)) {
mkdir($upload_dir, 0755, true);
}

for ($i = 0; $i < count($images['name']); $i++) {
if ($images['error'][$i] === UPLOAD_ERR_OK) {
$image_name = $images['name'][$i];
$image_tmp = $images['tmp_name'][$i];
// Generate unique filename
$file_extension = pathinfo($image_name, PATHINFO_EXTENSION);
$unique_filename = uniqid() . '_' . time() . '.' . $file_extension;
$target_path = $upload_dir . $unique_filename;
// Validate image type
$allowed_mime_types = ['image/jpeg', 'image/jpg', 'image/png', 'image/gif', 'image/webp'];
$file_type = $images['type'][$i];
if (!in_array($file_type, $allowed_mime_types)) {
```

```
throw new Exception('Invalid file type for: ' . $image_name);
}
// Move uploaded file
if (move_uploaded_file($image_tmp, $target_path)) {
// Insert image record
$image_stmt = $pdo->prepare("INSERT INTO room_images (room_id, image_path) VALUES (?, ?)");
$image_stmt->execute([$room_id, $unique_filename]);
$uploaded_images[] = $unique_filename;
}}}

// Set first image as primary room image
if (!empty($uploaded_images)) {
$update_stmt = $pdo->prepare("UPDATE rooms SET room_image = ? WHERE room_id = ?");
$update_stmt->execute([$uploaded_images[0], $room_id]);
}

// Get the newly created room
$room_stmt = $pdo->prepare("
SELECT r.*, GROUP_CONCAT(ri.image_path) as images
FROM rooms r
LEFT JOIN room_images ri ON r.room_id = ri.room_id
WHERE r.room_id = ?
GROUP BY r.room_id
");
$room_stmt->execute([$room_id]);
$new_room = $room_stmt->fetch(PDO::FETCH_ASSOC);

echo json_encode([
'success' => true,
'message' => 'Room added successfully with ' . count($uploaded_images) . ' images',
'room' => $new_room
]);

} catch (Exception $e) {
// Rollback room insertion if error occurs
if (isset($room_id)) {
try {
$pdo->prepare("DELETE FROM rooms WHERE room_id = ?")->execute([$room_id]);
} catch (Exception $rollback_error) {
// Ignore rollback errors
}}
echo json_encode(['success' => false, 'message' => 'Error: ' . $e->getMessage()]);
}
?>
```

A.7: Update Room

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Invalid request method']);
exit();
}

if (!isset($_POST['room_id'])) {
```

```
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Room ID required']);
exit();
}

$room_id = $_POST['room_id'];
$room_number = trim($_POST['room_number'] ?? '');
$room_type = $_POST['room_type'] ?? '';
$price = floatval($_POST['price'] ?? 0);
$description = trim($_POST['description'] ?? '');
$amenities = trim($_POST['amenities'] ?? '');

// Validate required fields
if (empty($room_number) || empty($room_type) || empty($price)) {
echo json_encode(['success' => false, 'message' => 'All required fields must be filled']);
exit();
}

// Validate room type
$allowed_types = ['single', 'double', 'suite', 'delux', 'family'];
if (!in_array($room_type, $allowed_types)) {
echo json_encode(['success' => false, 'message' => 'Invalid room type']);
exit();
}

try {
// Check if room number already exists (excluding current room)
$check_stmt = $pdo->prepare("SELECT room_id FROM rooms WHERE room_number = ? AND room_id != ?");
$check_stmt->execute([$room_number, $room_id]);
if ($check_stmt->fetch()) {
echo json_encode(['success' => false, 'message' => 'Room number already exists']);
exit();
}

// Update room in database
$stmt = $pdo->prepare("UPDATE rooms SET room_number = ?, room_type = ?, price = ?, description = ?,
amenities = ? WHERE room_id = ?");
$stmt->execute([$room_number, $room_type, $price, $description, $amenities, $room_id]);

// Handle new image uploads if any
if (isset($_FILES['room_images']) && !empty($_FILES['room_images']['name'][0])) {
$images = $_FILES['room_images'];
$upload_dir = "uploads/rooms/";
// Create uploads directory if it doesn't exist
if (!is_dir($upload_dir)) {
mkdir($upload_dir, 0755, true);
}

for ($i = 0; $i < count($images['name']); $i++) {
if ($images['error'][$i] === UPLOAD_ERR_OK) {
$image_name = $images['name'][$i];
$image_tmp = $images['tmp_name'][$i];
// Generate unique filename
$file_extension = pathinfo($image_name, PATHINFO_EXTENSION);
$unique_filename = uniqid() . '_' . time() . '.' . $file_extension;
$target_path = $upload_dir . $unique_filename;
```

```
// Validate image type
$allowed_mime_types = ['image/jpeg', 'image/jpg', 'image/png', 'image/gif', 'image/webp'];
$file_type = $images['type'][$i];
if (!in_array($file_type, $allowed_mime_types)) {
throw new Exception('Invalid file type: ' . $image_name);
}
// Move uploaded file
if (move_uploaded_file($image_tmp, $target_path)) {
// Insert image record
$image_stmt = $pdo->prepare("INSERT INTO room_images (room_id, image_path) VALUES (?, ?)");
$image_stmt->execute([$room_id, $unique_filename]);
}}}}

// Get the updated room
$room_stmt = $pdo->prepare("
SELECT r.*, GROUP_CONCAT(ri.image_path) as images
FROM rooms r
LEFT JOIN room_images ri ON r.room_id = ri.room_id
WHERE r.room_id = ?
GROUP BY r.room_id
");
$room_stmt->execute([$room_id]);
$updated_room = $room_stmt->fetch(PDO::FETCH_ASSOC);

echo json_encode([
'success' => true,
'message' => 'Room updated successfully',
'room' => $updated_room
]);

} catch (Exception $e) {
echo json_encode(['success' => false, 'message' => 'Error: ' . $e->getMessage()]);
}
```

A.8: Delete Room

```
if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Invalid request method']);
exit();
}

if (!isset($_POST['room_id']) || empty($_POST['room_id'])) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Room ID is required']);
exit();
}

$room_id = $_POST['room_id'];

try {
// Get room images first
$image_stmt = $pdo->prepare("SELECT image_path FROM room_images WHERE room_id = ?");
$image_stmt->execute([$room_id]);
$images = $image_stmt->fetchAll(PDO::FETCH_COLUMN);
// Delete room images from database
$delete_images_stmt = $pdo->prepare("DELETE FROM room_images WHERE room_id = ?");
$delete_images_stmt->execute([$room_id]);
// Delete the room from database
$delete_room_stmt = $pdo->prepare("DELETE FROM rooms WHERE room_id = ?");
$delete_room_stmt->execute([$room_id]);
if ($delete_room_stmt->rowCount() > 0) {
// Delete the associated image files
foreach ($images as $image) {
$image_path = 'uploads/rooms/' . $image;
if (file_exists($image_path)) {
unlink($image_path);
}
}
header('Content-Type: application/json');
echo json_encode(['success' => true, 'message' => 'Room and associated images deleted successfully']);
} else {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'No room was deleted. It may have been already deleted.']);
}
} catch (PDOException $e) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Database error: ' . $e->getMessage()]);
}
```

A.9: Update Profile

```
$user_id = $_SESSION['user_id'];

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
echo json_encode(['success' => false, 'message' => 'Invalid request method']);
exit();
}
```

```
// Get form data
$email = trim($_POST['email'] ?? '');
$phone_number = trim($_POST['phone_number'] ?? '');
$full_name = trim($_POST['full_name'] ?? '');
$address_line_1 = trim($_POST['address_line_1'] ?? '');
$address_line_2 = trim($_POST['address_line_2'] ?? '');
$city = trim($_POST['city'] ?? '');
$state = trim($_POST['state'] ?? '');
$country = trim($_POST['country'] ?? '');
$pincode = trim($_POST['pincode'] ?? '');

// Validate required fields
if (empty($email) || empty($address_line_1) || empty($city) || empty($state) || empty($country) ||
empty($pincode)) {
echo json_encode(['success' => false, 'message' => 'All required fields must be filled']);
exit();
}

// Validate email format
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
echo json_encode(['success' => false, 'message' => 'Invalid email format']);
exit();
}

try {
// First, get current user data to check if email is actually changing
$current_user_stmt = $pdo->prepare("SELECT email FROM users WHERE id = ?");
$current_user_stmt->execute([$user_id]);
$current_user = $current_user_stmt->fetch(PDO::FETCH_ASSOC);
if (!$current_user) {
echo json_encode(['success' => false, 'message' => 'User not found']);
exit();
}
$current_email = $current_user['email'];
// Only check for duplicate email if the email is actually changing
if ($email !== $current_email) {
// Check if new email already exists in other users
$check_stmt = $pdo->prepare("SELECT id FROM users WHERE email = ? AND id != ?");
$check_stmt->execute([$email, $user_id]);
if ($check_stmt->fetch()) {
echo json_encode(['success' => false, 'message' => 'Email already exists. Please use a different email.']);
exit();
}
}

// Update user profile with new address fields
$stmt = $pdo->prepare("UPDATE users SET email = ?, phone_number = ?, full_name = ?, address_line_1 = ?,
address_line_2 = ?, city = ?, state = ?, country = ?, pincode = ?, updated_at = CURRENT_TIMESTAMP WHERE id =
?");
$stmt->execute([$email, $phone_number, $full_name, $address_line_1, $address_line_2, $city, $state, $country,
$pincode, $user_id]);
// Get updated user data
$stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");
$stmt->execute([$user_id]);
$updated_user = $stmt->fetch(PDO::FETCH_ASSOC);
// Remove password from response
```

```
unset($updated_user['password']);
echo json_encode([
'success' => true,
'message' => 'Profile updated successfully',
'user' => $updated_user
]);

} catch (Exception $e) {
echo json_encode(['success' => false, 'message' => 'Error: ' . $e->getMessage()]);
}
```

A.10: Cart Function

```
if (session_status() == PHP_SESSION_NONE) {
session_start();
}

class CartManager {
private $cart_key = 'booking_cart';
private $cart_expiry = 1800; // 30 minutes in seconds
public function __construct() {
$this->initializeCart();
}
private function initializeCart() {
if (!isset($_SESSION[$this->cart_key])) {
$_SESSION[$this->cart_key] = [
'items' => [],
'created_at' => time(),
'user_id' => $_SESSION['user_id'] ?? null
];
}
}
public function addToCart($room_id, $checkin_date, $checkout_date, $room_data) {
// Check if cart is expired
if ($this->isCartExpired()) {
$this->clearCart();
}
$item_key = $this->generateItemKey($room_id, $checkin_date, $checkout_date);
$_SESSION[$this->cart_key]['items'][$item_key] = [
'room_id' => $room_id,
'checkin_date' => $checkin_date,
'checkout_date' => $checkout_date,
'room_number' => $room_data['room_number'],
'room_type' => $room_data['room_type'],
'price' => floatval($room_data['price']),
'nights' => $this->calculateNights($checkin_date, $checkout_date),
'total' => $this->calculateTotal(floatval($room_data['price']), $checkin_date, $checkout_date),
'added_at' => time()
];
return true;
}
public function removeFromCart($item_key) {
if (isset($_SESSION[$this->cart_key]['items'][$item_key])) {
unset($_SESSION[$this->cart_key]['items'][$item_key]);
return true;
}
return false;
```

```
}
public function getCart() {
if ($this->isCartExpired()) {
$this->clearCart();
return ['items' => []];
}
return $_SESSION[$this->cart_key];
}
public function clearCart() {
$_SESSION[$this->cart_key] = [
'items' => [],
'created_at' => time(),
'user_id' => $_SESSION['user_id'] ?? null
];
}
public function getCartCount() {
$cart = $this->getCart();
return count($cart['items']);
}
public function getCartTotal() {
$cart = $this->getCart();
$total = 0;
foreach ($cart['items'] as $item) {
$total += floatval($item['total']);
}
return $total;
}
public function getCartSummary() {
$cart = $this->getCart();
$subtotal = $this->getCartTotal();
$tax_rate = 0.10; // 10% tax
$tax_amount = $subtotal * $tax_rate;
$grand_total = $subtotal + $tax_amount;
return [
'subtotal' => $subtotal,
'tax_amount' => $tax_amount,
'grand_total' => $grand_total,
'item_count' => count($cart['items'])
];
}
private function generateItemKey($room_id, $checkin_date, $checkout_date) {
return md5($room_id . $checkin_date . $checkout_date);
}
private function calculateNights($checkin, $checkout) {
$checkin = new DateTime($checkin);
$checkout = new DateTime($checkout);
$interval = $checkout->diff($checkin);
return $interval->days;
}
private function calculateTotal($price, $checkin, $checkout) {
$nights = $this->calculateNights($checkin, $checkout);
return $price * $nights;
}
private function isCartExpired() {
if (!isset($_SESSION[$this->cart_key]['created_at'])) {
return false;
}
return (time() - $_SESSION[$this->cart_key]['created_at']) > $this->cart_expiry;
```

```
}
public function getTimeRemaining() {
if (!isset($_SESSION[$this->cart_key]['created_at'])) {
return 0;
}
$expiry_time = $_SESSION[$this->cart_key]['created_at'] + $this->cart_expiry;
$time_left = $expiry_time - time();
return max(0, $time_left);
}
}
```

A.11 : Booking

```
<?php
// Check if session is already started
if (session_status() == PHP_SESSION_NONE) {
session_start();
}

if (!isset($_SESSION['user_id'])) {
header('Location: ../login.php');
exit();
}

$cart_data = $cart->getCart();
if (empty($cart_data['items'])) {
header('Location: cart.php');
exit();
}

$cart_summary = $cart->getCartSummary();
$time_left = $cart->getTimeRemaining();

// Calculate total nights from cart items
$total_nights = 0;
foreach ($cart_data['items'] as $item) {
$total_nights = max($total_nights, $item['nights']);
}

// Get user details for auto-fill
$user_stmt = $pdo->prepare("SELECT * FROM users WHERE id = ?");
$user_stmt->execute([$_SESSION['user_id']]);
$user = $user_stmt->fetch(PDO::FETCH_ASSOC);

// Check for payment messages
$payment_success = isset($_GET['payment_success']);
$payment_error = isset($_GET['payment_error']);
$payment_message = $_GET['message'] ?? '';

// Get base URL for localhost
$base_url = (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] === 'on' ? "https" : "http") .
"://$_SERVER[HTTP_HOST]";
$return_url = $base_url . '/checkout/payment/process_payment_return.php';
?>
```

```
<script>
function calculateCoinDiscount() {
const useCoinsInput = getElement('use_coins');
const useCoins = parseInt(useCoinsInput?.value || 0);
const coinBalance = parseInt(getElement('coin-balance')?.textContent || 0);
const coinValue = 1; // $1 per coin
// Validate coins input
const maxCoins = Math.min(coinBalance, Math.floor(calculateTotalWithoutCoinDiscount()));
const actualCoins = Math.min(useCoins, maxCoins);
if (useCoinsInput && useCoins !== actualCoins) {
useCoinsInput.value = actualCoins;
}
const coinDiscount = actualCoins * coinValue;
// Update coin discount display
updateElementText('coin-discount', coinDiscount.toFixed(2));
// Recalculate final total
calculateFinalTotal();
return coinDiscount;
}

// Calculate total without coin discount for validation
function calculateTotalWithoutCoinDiscount() {
const roomSubtotal = <?php echo $cart_summary['subtotal']; ?>;
const tax = roomSubtotal * 0.1;
const addonsTotal = calculateAddonsTotal();
return roomSubtotal + tax + addonsTotal;
}

// Total calculation to include coin discount
function calculateFinalTotal() {
const roomSubtotal = <?php echo $cart_summary['subtotal']; ?>;
const tax = roomSubtotal * 0.1;
const addonsTotal = calculateAddonsTotal();
const coinDiscountElement = getElement('coin-discount');
const coinDiscount = coinDiscountElement ? parseFloat(coinDiscountElement.textContent) || 0 : 0;
const finalTotal = Math.max(0, roomSubtotal + tax + addonsTotal - coinDiscount);
// Update all total displays
updateElementText('summary-addons', '$' + addonsTotal.toFixed(2));
updateElementText('summary-final-total', '$' + finalTotal.toFixed(2));
updateElementText('final-amount', finalTotal.toFixed(2));
return finalTotal;
}

// Use maximum coins
function useMaxCoins() {
const coinBalance = parseInt(getElement('coin-balance')?.textContent || 0);
const maxCoins = Math.min(coinBalance, Math.floor(calculateTotalWithoutCoinDiscount()));
const useCoinsInput = getElement('use_coins');
if (useCoinsInput) {
useCoinsInput.value = maxCoins;
}
calculateCoinDiscount();
}

async function handleFormSubmit(e) {
e.preventDefault();
const submitBtn = getElement('submit-btn');
```

```
const loadingDiv = getElement('payment-loading');
const errorsDiv = getElement('payment-errors');
const messageDiv = getElement('message');
// Validate Stripe initialization
if (!isStripeInitialized || !elements) {
showMessage('Payment system is not ready. Please wait a moment and try again.', 'error');
return;
}
// Disable form and show loading
if (submitBtn) {
submitBtn.disabled = true;
submitBtn.innerHTML = 'Processing Payment...';
}
if (loadingDiv) {
loadingDiv.style.display = 'block';}
if (errorsDiv) {
errorsDiv.style.display = 'none';
}
if (messageDiv) {
messageDiv.style.display = 'none';
}
try {
console.log('Confirming payment...');
// Confirm payment with Stripe
const { error: stripeError } = await stripe.confirmPayment({
elements: elements,
confirmParams: {
return_url: '<?php echo $return_url; ?>',
},
redirect: 'if_required'
});
if (stripeError) {
console.error('Stripe error:', stripeError);
throw new Error(stripeError.message);
}
console.log('Payment successful, submitting form data...');
// If we reach here, payment was successful
// Now submit the form data to process_checkout.php
const formData = new FormData(getElement('checkoutForm'));
formData.append('payment_intent_id', '<?php echo $_SESSION["payment_intent_id"] ?? ""; ?>');
const response = await fetch('process_checkout.php', {
method: 'POST',
body: formData
});
const result = await response.json();
console.log('Process checkout response:', result);
if (result.success) {
showMessage(result.message, 'success');
setTimeout(() => {
window.location.href = result.redirect;
}, 1500);
} else {
throw new Error(result.message);
}
} catch (error) {
console.error('Payment processing error:', error);
if (errorsDiv) {
errorsDiv.textContent = error.message;
```

```
errorsDiv.style.display = 'block';
}
if (submitBtn) {
submitBtn.disabled = false;
updateSubmitButtonState(true);
}
if (loadingDiv) {
loadingDiv.style.display = 'none';
}}}
</script>
```

A.12 Initial and config Stripe

```
// stripe_config
<?php

// Manual include for Stripe PHP library
$stripe_path = __DIR__ . '/../../stripe-php/init.php';
if (file_exists($stripe_path)) {
require_once $stripe_path;
} else {
die("Stripe PHP library not found. Please download from: https://github.com/stripe/stripe-php");
}

class StripePayment {
private $secretKey;
private $publishableKey;
public function __construct() {
// Test keys - replace with your actual Stripe keys
$this->secretKey =
'sk_test_51SRmMDEZhzQrywIMBV8cAJqFnwcGZgWgt9qDsmCC3QXjxmxXkoi4Meo4Kpvbk2lA7ZNtavcyYDnVC
JmC7TktpCZY00i5gbXhJj'; // Your test secret key
$this->publishableKey =
'pk_test_51SRmMDEZhzQrywIMjKqJYjVuJqQIRdNxAAoL6TgB33WtoSTahwDOL9hSvk9hbiaDvBYcCX9sUSMCJu
FEZglMa4hK00aBfovH3w'; // Your test publishable key
\Stripe\Stripe::setApiKey($this->secretKey);
\Stripe\Stripe::setAppInfo("Roberto Hotel", "1.0", "https://yourdomain.com");
}
public function getPublishableKey() {
return $this->publishableKey;
}
public function createPaymentIntent($amount, $currency = 'usd', $metadata = []) {
try {
$paymentIntent = \Stripe\PaymentIntent::create([
'amount' => $this->formatAmount($amount, $currency),
'currency' => $currency,
'metadata' => $metadata,
'automatic_payment_methods' => [
'enabled' => true,
],
]);
return [
'success' => true,
'client_secret' => $paymentIntent->client_secret,
'payment_intent_id' => $paymentIntent->id
];
} catch (\Stripe\Exception\CardException $e) {
return [
'success' => false,
```

```
'error' => $e->getError()->message
];
} catch (\Stripe\Exception\RateLimitException $e) {
return [
'success' => false,
'error' => 'Too many requests. Please try again.'
];
} catch (\Stripe\Exception\InvalidRequestException $e) {
return [
'success' => false,
'error' => 'Invalid request. Please check your payment details.'
];
} catch (\Stripe\Exception\AuthenticationException $e) {
return [
'success' => false,
'error' => 'Authentication error. Please contact support.'
];
} catch (\Stripe\Exception\ApiConnectionException $e) {
return [
'success' => false,
'error' => 'Network error. Please check your connection.'
];
} catch (\Stripe\Exception\ApiErrorException $e) {
return [
'success' => false,
'error' => 'Stripe API error: ' . $e->getMessage()
];
} catch (Exception $e) {
return [
'success' => false,
'error' => 'Unexpected error: ' . $e->getMessage()
];
}
}
public function confirmPayment($paymentIntentId) {
try {
$paymentIntent = \Stripe\PaymentIntent::retrieve($paymentIntentId);
return [
'success' => true,
'payment_intent' => $paymentIntent
];
} catch (Exception $e) {
return [
'success' => false,
'error' => $e->getMessage()
];
}
}
private function formatAmount($amount, $currency) {
// Convert dollars to cents
$amount = floatval($amount);
// Handle zero-decimal currencies
$zeroDecimalCurrencies = ['BIF', 'CLP', 'DJF', 'GNF', 'JPY', 'KMF', 'KRW',
'MGA', 'PYG', 'RWF', 'UGX', 'VND', 'VUV', 'XAF',
'XOF', 'XPF'];
if (in_array(strtoupper($currency), $zeroDecimalCurrencies)) {
return intval($amount);
}
```

```
return intval($amount * 100);
}}

// Global instance
$stripePayment = new StripePayment();
?>



//Initial Strip in checkout process
let stripe;
let elements;
let paymentIntentClientSecret;
let isStripeInitialized = false;

// Update submit button state
function updateSubmitButtonState(enabled = true) {
const submitBtn = getElement('submit-btn');
if (!submitBtn) return;
if (enabled && isStripeInitialized) {
const currentTotal = calculateFinalTotal();
submitBtn.innerHTML = `Pay $<span id="final-amount">${currentTotal.toFixed(2)}</span>`;
submitBtn.disabled = false;
} else {
submitBtn.innerHTML = 'Loading Payment System...';
submitBtn.disabled = true;
}
}

// Wait for DOM to be fully loaded
document.addEventListener('DOMContentLoaded', function() {
console.log('DOM loaded, initializing Stripe...');
initializeStripe();
initializeEventListeners();
calculateAddons(); // Initial calculation
});

async function initializeStripe() {
try {
console.log('Initializing Stripe...');
// Initialize Stripe
stripe = Stripe('<?php echo $stripePayment->getPublishableKey(); ?>');
const response = await fetch('payment/create_payment_intent.php', {
method: 'POST',
headers: {
'Content-Type': 'application/json',
}
});
const result = await response.json();
console.log('Payment intent response:', result);
if (!result.success) {
throw new Error(result.error);
}
paymentIntentClientSecret = result.clientSecret;
const appearance = {
theme: 'stripe',
```

```
variables: {
colorPrimary: '#42c1b4',
colorBackground: '#f6f9fc',
colorText: '#32325d',
colorDanger: '#df1b41',
fontFamily: 'Inter, system-ui, sans-serif',
spacingUnit: '4px',
borderRadius: '4px'
}
};
elements = stripe.elements({
clientSecret: paymentIntentClientSecret,
appearance: appearance
});
// Create and mount the Payment Element
const paymentElement = elements.create('payment', {
layout: {
type: 'tabs',
defaultCollapsed: false
}
});
const paymentElementContainer = getElement('payment-element');
if (paymentElementContainer) {
paymentElement.mount('#payment-element');
isStripeInitialized = true;
console.log('Stripe Elements mounted successfully');
updateSubmitButtonState(true);
} else {
throw new Error('Payment element container not found');
}
} catch (error) {
console.error('Error initializing Stripe:', error);
showMessage('Error initializing payment system: ' + error.message, 'error');
const paymentElementContainer = getElement('payment-element');
if (paymentElementContainer) {
paymentElementContainer.innerHTML =
'<div style="color: #dc3545; padding: 20px; text-align: center; border: 1px solid #dc3545; border-radius:
4px;">' +
'Payment system temporarily unavailable. Please try again later.' +
'</div>';
}
updateSubmitButtonState(false);
}}
```

A.13 : Booking Success

```
$booking_ids = $_SESSION['booking_ids'] ?? [];
$booking_details = $_SESSION['booking_details'] ?? [];

// Clear session data
unset($_SESSION['booking_success']);
unset($_SESSION['booking_ids']);
unset($_SESSION['booking_details']);

include('header.php');
include('../font.php');
?>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Booking Successful - Robetro Hotel</title>
<link rel="stylesheet" href="./style.css">
</head>
<body>

<section class="success-section">
<div class="container success-container">
<div class="success-icon"></div>
<h2 style="color: green; font-weight: 500; text-transform: uppercase; margin-bottom: 20px;">Booking
Successful!</h2>
<p style="font-size: 1.1rem; margin-bottom: 1.5rem;">
Thank you for your booking, <strong><?php echo htmlspecialchars($booking_details['full_name'] ?? 'Guest');
?></strong>!
</p>
<p style="margin-bottom: 1.5rem;">
A confirmation email has been sent to <strong><?php echo htmlspecialchars($booking_details['email'] ?? '');
?></strong>
with all your booking details.
</p>
<div class="booking-details">
<h4 style="margin-top: 0; color: #333;">Booking Details</h4>
<p><strong>Booking Reference Numbers:</strong><br>
<?php
foreach ($booking_ids as $id) {
echo "<span style='background: #42c1b4; color: white; padding: 4px 8px; border-radius: 4px; margin: 0 10px;
margin-top:-5px;'>#{$id}</span>";
}
?>
</p>
<?php if (!empty($booking_details['addons_details'])): ?>
<p><strong>Additional Services:</strong></p>
<ul class="addons-list">
<?php foreach ($booking_details['addons_details'] as $addon): ?>
<li><?php echo $addon; ?></li>
<?php endforeach; ?>
</ul>
<p><strong>Additional Services Total:</strong> $<?php echo
number_format($booking_details['addons_total'] ?? 0, 2); ?></p>
<?php endif; ?>
<!-- Add this to your booking_success.php after the booking details -->
<?php if (!empty($booking_details['coins_used']) && $booking_details['coins_used'] > 0): ?>
<div class="successUsedCoin" style="background: #fff3cd">
<h4 style="margin: 0 0 10px 0; color: #856404;">Coins Used</h4>
<p style="margin: 0; font-size: 1.2rem;">
<strong>-<?php echo $booking_details['coins_used']; ?> </strong>coins were deducted from your balance
</p>
<p style="margin: 5px 0 0 0; color: #856404; font-size: 0.9rem;">
You saved $<?php echo number_format($booking_details['coin_discount'], 2); ?> using your coins
</p>
</div>
<?php endif; ?>

<?php if (!empty($booking_details['coins_earned']) && $booking_details['coins_earned'] > 0): ?>
```

```
<div class="successEarnedCoins">
<h4 style="margin: 0 0 10px 0; color: #155724;">New Coins Earned!</h4>
<p style="margin: 0; font-size: 1.2rem;">
<strong>+<?php echo $booking_details['coins_earned']; ?> </strong>coins added to your balance
</p>
<p style="margin: 5px 0 0 0; color: #0c5460; font-size: 0.9rem;">
You earned 1 coin for every $15 spent on this booking
</p>
</div>
<?php endif; ?>

<!-- Show net coin change -->
<?php
$net_coins = ($booking_details['coins_earned'] ?? 0) - ($booking_details['coins_used'] ?? 0);
if ($net_coins != 0): ?>
<div class="successNetCoins">
<h4 style="margin: 0 0 10px 0; color: #383d41;">Net Coin Change</h4>
<p style="margin: 0; font-size: 1.2rem;">
<strong><?php echo $net_coins > 0 ? '+' : ''; ?><?php echo $net_coins; ?> </strong>coins net change to your
balance
</p>
</div>
<?php endif; ?>

</div>
<p style="color: #666; margin-bottom: 2rem;">Your booking details are now available in your account
dashboard. We look forward to welcoming you!</p>
<div class="btn-group bs_btn">
<a href="../user/dashboard.php" class="btn btn-primary">View Dashboard</a>
<a href="../rooms.php" class="btn btn-secondary">Book Another Room</a>
<a href="../index.php" class="btn btn-secondary">Back to Home</a>
</div>
</div>
</section>
```

A.13 : Cancel Booking

```
<?php
// user/cancel_booking.php
session_start();
header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
echo json_encode(['success' => false, 'message' => 'Unauthorized']);
exit();
}

include('../db.php');

$user_id = $_SESSION['user_id'];
$booking_id = $_POST['booking_id'] ?? '';

if (empty($booking_id)) {
echo json_encode(['success' => false, 'message' => 'Booking ID required']);
exit();
}
```

```
try {
// Verify the booking belongs to the user and is not already cancelled
$stmt = $pdo->prepare("SELECT * FROM bookings WHERE booking_id = ? AND user_id = ?");
$stmt->execute([$booking_id, $user_id]);
$booking = $stmt->fetch(PDO::FETCH_ASSOC);
if (!$booking) {
echo json_encode(['success' => false, 'message' => 'Booking not found']);
exit();
}

// Check if booking is already cancelled
if ($booking['status'] === 'cancelled') {
echo json_encode(['success' => false, 'message' => 'Booking is already cancelled']);
exit();
}

// Check if check-in date is in the future (can't cancel past bookings)
$checkin_date = new DateTime($booking['checkin_date']);
$today = new DateTime();
if ($checkin_date <= $today) {
echo json_encode(['success' => false, 'message' => 'Cannot cancel booking after check-in date']);
exit();
}

// Start transaction
$pdo->beginTransaction();

// Update booking status to cancelled
$update_stmt = $pdo->prepare("UPDATE bookings SET status = 'cancelled' WHERE booking_id = ?");
$update_stmt->execute([$booking_id]);

// Update room availability back to available
$room_stmt = $pdo->prepare("UPDATE rooms SET availability = 1 WHERE room_id = ?");
$room_stmt->execute([$booking['room_id']]);

// Check if coins were earned from this booking
if ($booking['coins_earned'] > 0) {
// Subtract the earned coins from user's balance
$subtract_coins_stmt = $pdo->prepare("UPDATE users SET coins = coins - ? WHERE id = ?");
$subtract_coins_stmt->execute([$booking['coins_earned'], $user_id]);
// Record the coin deduction transaction
$coin_transaction_stmt = $pdo->prepare("
INSERT INTO coin_transactions
(user_id, booking_id, transaction_type, coins, description)
VALUES (?, ?, 'used', ?, 'Coins deducted due to booking cancellation')
");
$coin_transaction_stmt->execute([
$user_id,
$booking_id,
$booking['coins_earned']
]);
}

// ========== ADD THIS: Get updated user coin balance ==========
$user_coins_stmt = $pdo->prepare("SELECT coins FROM users WHERE id = ?");
```

```
$user_coins_stmt->execute([$user_id]);
$user_data = $user_coins_stmt->fetch(PDO::FETCH_ASSOC);
$updated_coins = $user_data['coins'];
// ========== END ADDITION ==========

$pdo->commit();

echo json_encode([
'success' => true,
'message' => 'Booking cancelled successfully' .
($booking['coins_earned'] > 0 ? ' and coins deducted' : ''),
'booking_id' => $booking_id,
'coins_deducted' => $booking['coins_earned'] > 0 ? $booking['coins_earned'] : 0,
'updated_coins' => $updated_coins // Add this line
]);
} catch (Exception $e) {
// Rollback transaction on error
if (isset($pdo) && $pdo->inTransaction()) {
$pdo->rollBack();
}
error_log("Cancel booking error: " . $e->getMessage());
echo json_encode([
'success' => false,
'message' => 'Error cancelling booking: ' . $e->getMessage()
]);
}?>
```

A.14: Royalty Coin

```
<?php
class CoinSystem {
private $pdo;
private $coin_rate = 15; // $15 = 1 coin
private $coin_value = 1; // 1 coin = $1 discount

public function __construct($pdo) {
$this->pdo = $pdo;
}

// Calculate coins earned from a booking amount
public function calculateCoinsEarned($amount) {
return floor($amount / $this->coin_rate);
}

// Award coins for a booking
public function awardCoins($user_id, $booking_id, $amount) {
$coins_earned = $this->calculateCoinsEarned($amount);
if ($coins_earned > 0) {
// Update user's coin balance (ADD coins)
$stmt = $this->pdo->prepare("UPDATE users SET coins = coins + ? WHERE id = ?");
$stmt->execute([$coins_earned, $user_id]);
// Record transaction
$this->recordTransaction($user_id, $booking_id, 'earned', $coins_earned, "Coins earned from booking
#$booking_id");
// Update booking with coins earned
$stmt = $this->pdo->prepare("UPDATE bookings SET coins_earned = ? WHERE booking_id = ?");
```

```
$stmt->execute([$coins_earned, $booking_id]);
return $coins_earned;
}
return 0;
}

// Use coins for discount (SUBTRACT coins)
public function useCoins($user_id, $booking_id, $coins_to_use, $max_discount) {
// Get user's current coin balance
$stmt = $this->pdo->prepare("SELECT coins FROM users WHERE id = ?");
$stmt->execute([$user_id]);
$user = $stmt->fetch();
$available_coins = $user['coins'];
// Validate coins to use - cannot use more than available or more than max discount
$coins_to_use = min($coins_to_use, $available_coins, $max_discount);
if ($coins_to_use > 0) {
// Update user's coin balance (SUBTRACT coins)
$stmt = $this->pdo->prepare("UPDATE users SET coins = coins - ? WHERE id = ?");
$stmt->execute([$coins_to_use, $user_id]);
// Record transaction (negative value for used coins)
$this->recordTransaction($user_id, $booking_id, 'used', -$coins_to_use, "Coins used for booking
#$booking_id");
return $coins_to_use;
}
return 0;
}

// Record coin transaction
private function recordTransaction($user_id, $booking_id, $type, $coins, $description) {
$stmt = $this->pdo->prepare("
INSERT INTO coin_transactions (user_id, booking_id, transaction_type, coins, description)
VALUES (?, ?, ?, ?, ?)
");
$stmt->execute([$user_id, $booking_id, $type, $coins, $description]);
}

// Get user's coin balance
public function getCoinBalance($user_id) {
$stmt = $this->pdo->prepare("SELECT coins FROM users WHERE id = ?");
$stmt->execute([$user_id]);
$user = $stmt->fetch();
return $user['coins'] ?? 0;
}

// Get coin transaction history
public function getTransactionHistory($user_id, $limit = 10) {
$stmt = $this->pdo->prepare("
SELECT ct.*, b.booking_id
FROM coin_transactions ct
LEFT JOIN bookings b ON ct.booking_id = b.booking_id
WHERE ct.user_id = ?
ORDER BY ct.created_at DESC
LIMIT ?
");
$stmt->execute([$user_id, $limit]);
return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

```
// Get coin value for display
public function getCoinValue() {
return $this->coin_value;
}

// Get coin rate for display
public function getCoinRate() {
return $this->coin_rate;
}}

// Initialize coin system
if (isset($pdo)) {
$coinSystem = new CoinSystem($pdo);
}
?>
```

A.15 : Refund Request

```
<?php
// user/refund_request.php
session_start();
header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
echo json_encode(['success' => false, 'message' => 'Unauthorized']);
exit();
}

include('../db.php');

$user_id = $_SESSION['user_id'];
$booking_id = $_POST['booking_id'] ?? '';
$refund_amount = $_POST['refund_amount'] ?? '';
$refund_reason = $_POST['refund_reason'] ?? '';
$payment_method = $_POST['payment_method'] ?? '';
$bank_name = $_POST['bank_name'] ?? '';
$account_number = $_POST['account_number'] ?? '';
$routing_number = $_POST['routing_number'] ?? '';

if (empty($booking_id) || empty($refund_amount)) {
echo json_encode(['success' => false, 'message' => 'Booking ID and refund amount are required']);
exit();
}

try {
// Verify the booking belongs to the user and is cancelled
$stmt = $pdo->prepare("
SELECT b.*, r.room_number, r.room_type
FROM bookings b
JOIN rooms r ON b.room_id = r.room_id
WHERE b.booking_id = ? AND b.user_id = ? AND b.status = 'cancelled'
");
$stmt->execute([$booking_id, $user_id]);
$booking = $stmt->fetch(PDO::FETCH_ASSOC);
```

```
if (!$booking) {
echo json_encode(['success' => false, 'message' => 'Cancelled booking not found or does not belong to you']);
exit();
}

// Verify refund amount doesn't exceed booking amount
if ($refund_amount > $booking['total_amount']) {
echo json_encode(['success' => false, 'message' => 'Refund amount cannot exceed booking amount']);
exit();
}

// Check if refund request already exists
$check_stmt = $pdo->prepare("SELECT id FROM refund_requests WHERE booking_id = ? AND user_id = ?");
$check_stmt->execute([$booking_id, $user_id]);
if ($check_stmt->fetch()) {
echo json_encode(['success' => false, 'message' => 'Refund request already submitted for this booking']);
exit();
}

// Insert refund request
$stmt = $pdo->prepare("
INSERT INTO refund_requests
(booking_id, user_id, refund_amount, refund_reason, payment_method, bank_name, account_number,
routing_number, status)
VALUES (?, ?, ?, ?, ?, ?, ?, ?, 'pending')
");
$stmt->execute([
$booking_id,
$user_id,
$refund_amount,
$refund_reason,
$payment_method,
$bank_name,
$account_number,
$routing_number
]);

echo json_encode([
'success' => true,
'message' => 'Refund request submitted successfully. We will process it within 3-5 business days.',
'refund_id' => $pdo->lastInsertId()
]);
} catch (Exception $e) {
error_log("Refund request error: " . $e->getMessage());
echo json_encode([
'success' => false,
'message' => 'Error submitting refund request: ' . $e->getMessage()
]);
}
?>
```

A.16 : Earning

```
<?php
```

```
session_start();
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Unauthorized']);
exit();
}

include('db.php');

header('Content-Type: application/json');

try {
// Get earnings summary - using COALESCE to handle NULL values
$summary_stmt = $pdo->prepare("
SELECT
COUNT(*) as total_bookings,
COUNT(CASE WHEN status = 'confirmed' THEN 1 END) as confirmed_bookings,
COUNT(CASE WHEN status = 'cancelled' THEN 1 END) as cancelled_bookings,
COALESCE(SUM(CASE WHEN status IN ('confirmed', 'checked_in', 'checked_out') THEN
COALESCE(total_amount, 0)
ELSE 0 END), 0) as total_earnings,
COALESCE(SUM(CASE WHEN status = 'cancelled' THEN COALESCE(total_amount, 0) ELSE 0 END), 0) as
total_cancellations,
COALESCE(SUM(CASE WHEN status IN ('confirmed', 'checked_in', 'checked_out') THEN
COALESCE(total_amount, 0)
ELSE 0 END), 0) -
COALESCE(SUM(CASE WHEN status = 'cancelled' THEN COALESCE(total_amount, 0) ELSE 0 END), 0) as
net_earnings
FROM bookings
");
$summary_stmt->execute();
$summary = $summary_stmt->fetch(PDO::FETCH_ASSOC);

// Get monthly earnings for graph
$monthly_stmt = $pdo->prepare("
SELECT
DATE_FORMAT(created_at, '%Y-%m') as month,
COALESCE(SUM(CASE WHEN status IN ('confirmed', 'checked_in', 'checked_out') THEN
COALESCE(total_amount, 0)
ELSE 0 END), 0) as monthly_earnings,
COALESCE(SUM(CASE WHEN status = 'cancelled' THEN COALESCE(total_amount, 0) ELSE 0 END), 0) as
monthly_cancellations
FROM bookings
WHERE created_at >= DATE_SUB(NOW(), INTERVAL 12 MONTH)
GROUP BY DATE_FORMAT(created_at, '%Y-%m')
ORDER BY month
");
$monthly_stmt->execute();
$monthly_data = $monthly_stmt->fetchAll(PDO::FETCH_ASSOC);

// Get recent bookings for the table
$recent_stmt = $pdo->prepare("
SELECT
b.booking_id,
b.user_id,
b.user_name,
b.user_email,
```

```
COALESCE(b.total_amount, 0) as total_amount,
b.status,
b.created_at,
b.checkin_date,
b.checkout_date,
r.room_number,
r.room_type
FROM bookings b
LEFT JOIN rooms r ON b.room_id = r.room_id
ORDER BY b.created_at DESC
LIMIT 50
");
$recent_stmt->execute();
$recent_bookings = $recent_stmt->fetchAll(PDO::FETCH_ASSOC);

echo json_encode([
'success' => true,
'summary' => $summary,
'monthly_data' => $monthly_data,
'recent_bookings' => $recent_bookings
]);

} catch (Exception $e) {
echo json_encode(['success' => false, 'message' => 'Error: ' . $e->getMessage()]);}
?>
```

A.17: Process Check-in

```
/ Check if user is logged in
if (!isset($_SESSION['user_id'])) {
http_response_code(401);
echo json_encode(['success' => false, 'message' => 'Unauthorized access']);
exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
// Get booking_id from POST data
$booking_id = isset($_POST['booking_id']) ? intval($_POST['booking_id']) : 0;
if ($booking_id <= 0) {
echo json_encode(['success' => false, 'message' => 'Invalid booking ID']);
exit;
}
try {
// First, verify the booking exists and get its current status
$booking_stmt = $pdo->prepare("SELECT status, room_id FROM bookings WHERE booking_id = ?");
$booking_stmt->execute([$booking_id]);
$booking = $booking_stmt->fetch(PDO::FETCH_ASSOC);
if (!$booking) {
echo json_encode(['success' => false, 'message' => 'Booking not found']);
exit;
}
// In your database, 'booked' is the confirmed status
if ($booking['status'] !== 'confirmed') {
echo json_encode(['success' => false, 'message' => 'Only confirmed bookings (status: confirmed) can be
checked in. Current status: ' . $booking['status']]);
exit;
```

```
}
// Start transaction
$pdo->beginTransaction();
// Update booking status to checked_in
$update_booking = $pdo->prepare("UPDATE bookings SET status = 'checked_in' WHERE booking_id = ?");
$update_booking->execute([$booking_id]);
// Update room availability
$update_room = $pdo->prepare("UPDATE rooms SET availability = 0 WHERE room_id = ?");
$update_room->execute([$booking['room_id']]);
// Commit transaction
$pdo->commit();
echo json_encode(['success' => true, 'message' => 'Guest checked in successfully!']);
} catch (Exception $e) {
// Rollback transaction on error
if ($pdo->inTransaction()) {
$pdo->rollBack();
}
error_log("Check-in error: " . $e->getMessage());
echo json_encode(['success' => false, 'message' => 'Database error: ' . $e->getMessage()]);
}
} else {
http_response_code(405);
echo json_encode(['success' => false, 'message' => 'Method not allowed']);
}

<script>
function processCheckin(bookingId) {
if (confirm('Are you sure you want to check in this guest?')) {
showLoading(bookingId, 'checkin');
// Use relative path from current page
fetch('../receptionist/ajax/process_checkin.php', {
method: 'POST',
headers: {
'Content-Type': 'application/x-www-form-urlencoded',
},
body: 'booking_id=' + bookingId
})
.then(response => {
if (!response.ok) {
throw new Error('Server error: ' + response.status);
}
return response.json();
})
.then(data => {
hideLoading(bookingId, 'checkin');
if (data.success) {
showAlert('Guest checked in successfully!', 'success');
// Reload after 1.5 seconds
setTimeout(() => {
location.reload();
}, 1500);
} else {
showAlert('Error: ' + data.message, 'error');
}
})
.catch(error => {
hideLoading(bookingId, 'checkin');
console.error('Check-in error:', error);
```

```
showAlert('Failed to process check-in: ' + error.message, 'error');
});
}}

</script>
```

A.18: Process Check-out

```
<?php
session_start();

// Include Main Database
$db_path = __DIR__ . '/../../db.php';
if (!file_exists($db_path)) {
$db_path = __DIR__ . '/../../../db.php';
}

require_once $db_path;

header('Content-Type: application/json');

if (!isset($_SESSION['user_id'])) {
http_response_code(401);
echo json_encode(['success' => false, 'message' => 'Unauthorized access']);
exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
$booking_id = isset($_POST['booking_id']) ? intval($_POST['booking_id']) : 0;
if ($booking_id <= 0) {
echo json_encode(['success' => false, 'message' => 'Invalid booking ID']);
exit;
}
try {
// Verify booking exists and is checked in
$booking_stmt = $pdo->prepare("SELECT status, room_id FROM bookings WHERE booking_id = ?");
$booking_stmt->execute([$booking_id]);
$booking = $booking_stmt->fetch(PDO::FETCH_ASSOC);
if (!$booking) {
echo json_encode(['success' => false, 'message' => 'Booking not found']);
exit;
}
if ($booking['status'] !== 'checked_in') {
echo json_encode(['success' => false, 'message' => 'Only checked-in bookings can be checked out. Current
status: ' . $booking['status']]);
exit;
}
$pdo->beginTransaction();
// Update booking status to checked_out
$update_booking = $pdo->prepare("UPDATE bookings SET status = 'checked_out' WHERE booking_id = ?");
$update_booking->execute([$booking_id]);
// Update room availability
$update_room = $pdo->prepare("UPDATE rooms SET availability = 1 WHERE room_id = ?");
$update_room->execute([$booking['room_id']]);
```

```
$pdo->commit();
echo json_encode(['success' => true, 'message' => 'Guest checked out successfully!']);
} catch (Exception $e) {
if ($pdo->inTransaction()) {
$pdo->rollBack();
}
error_log("Check-out error: " . $e->getMessage());
echo json_encode(['success' => false, 'message' => 'Database error: ' . $e->getMessage()]);
}
} else {
http_response_code(405);
echo json_encode(['success' => false, 'message' => 'Method not allowed']);
}
?>
```

A.19: Dashboard (to get data)

```
// Query to get all user data
$user_stmt = $pdo->prepare("SELECT * FROM users");
$user_stmt->execute();
$users = $user_stmt->fetchAll(PDO::FETCH_ASSOC);

// Query to get all subscriber data
$subscriber_stmt = $pdo->prepare("SELECT * FROM newsletter_subscribers");
$subscriber_stmt->execute();
$subscribers = $subscriber_stmt->fetchAll(PDO::FETCH_ASSOC);

// Query to get all contact data
$contact_stmt = $pdo->prepare("SELECT * FROM contact_submissions ORDER BY created_at DESC");
$contact_stmt->execute();
$contacts = $contact_stmt->fetchAll(PDO::FETCH_ASSOC);

// Query to get all blogs data
$blog_stmt = $pdo->prepare("SELECT * FROM blogs ORDER BY created_at DESC");
$blog_stmt->execute();
$blogs = $blog_stmt->fetchAll(PDO::FETCH_ASSOC);
```

A.20 Showing Graph / chart with chat.js

```
createCharts: function(monthlyData, summary) {
try {
this.createEarningsChart(monthlyData);
this.createDistributionChart(summary);
} catch (error) {
console.error('Error creating charts:', error);
}
},

createEarningsChart: function(monthlyData) {
const ctx = document.getElementById('earningsChart');
if (!ctx) {
console.error('Earnings chart canvas not found');
return;
}
// Destroy existing chart if it exists
```

```
if (this.charts.earnings) {
this.charts.earnings.destroy();
}

const labels = monthlyData.map(item => {
const date = new Date(item.month + '-01');
return date.toLocaleDateString('en-US', { month: 'short', year: 'numeric' });
});

const earningsData = monthlyData.map(item => parseFloat(item.monthly_earnings) || 0);
const cancellationData = monthlyData.map(item => parseFloat(item.monthly_cancellations) || 0);

// Check if any data to display
const hasData = earningsData.some(val => val > 0) || cancellationData.some(val => val > 0);
if (!hasData) {
ctx.getContext('2d').font = '16px Arial';
ctx.getContext('2d').fillStyle = '#6c757d';
ctx.getContext('2d').textAlign = 'center';
ctx.getContext('2d').fillText('No monthly data available', ctx.width / 2, ctx.height / 2);
return;
}

try {
this.charts.earnings = new Chart(ctx.getContext('2d'), {
type: 'bar',
data: {
labels: labels,
datasets: [
{
label: 'Earnings',
data: earningsData,
backgroundColor: 'rgba(40, 167, 69, 0.8)',
borderColor: 'rgba(40, 167, 69, 1)',
borderWidth: 1
},
{
label: 'Cancellations',
data: cancellationData,
backgroundColor: 'rgba(220, 53, 69, 0.8)',
borderColor: 'rgba(220, 53, 69, 1)',
borderWidth: 1
}]},
options: {
responsive: true,
scales: {
y: {
beginAtZero: true,
ticks: {
callback: function(value) {
return '$' + value;}}}}}});} catch (error) {
console.error('Error creating earnings chart:', error);
}
},

createDistributionChart: function(summary) {
const ctx = document.getElementById('distributionChart');
if (!ctx) {
```

```
console.error('Distribution chart canvas not found');
return;
}
// Destroy existing chart if it exists
if (this.charts.distribution) {
this.charts.distribution.destroy();
}

const totalEarnings = parseFloat(summary.total_earnings) || 0;
const totalCancellations = parseFloat(summary.total_cancellations) || 0;
const netEarnings = parseFloat(summary.net_earnings) || 0;

// Check if we have data to display
if (totalEarnings === 0 && totalCancellations === 0) {
ctx.getContext('2d').font = '16px Arial';
ctx.getContext('2d').fillStyle = '#6c757d';
ctx.getContext('2d').textAlign = 'center';
ctx.getContext('2d').fillText('No financial data available', ctx.width / 2, ctx.height / 2);
return;
}

try {
this.charts.distribution = new Chart(ctx.getContext('2d'), {
type: 'doughnut',
data: {
labels: ['Confirmed Bookings', 'Cancellations', 'Net Earnings'],
datasets: [{
data: [totalEarnings, totalCancellations, netEarnings],
backgroundColor: [
'rgba(40, 167, 69, 0.8)',
'rgba(220, 53, 69, 0.8)',
'rgba(0, 123, 255, 0.8)'
],
borderColor: [
'rgba(40, 167, 69, 1)',
'rgba(220, 53, 69, 1)',
'rgba(0, 123, 255, 1)'
],
borderWidth: 1
}]
},
options: {
responsive: true,
plugins: {
tooltip: {
callbacks: {
label: function(context) {
const label = context.label || '';
const value = context.raw || 0;
return label + ': $' + value.toFixed(2);
}}}}});
} catch (error) {
console.error('Error creating distribution chart:', error);
}},
```

A.21: Send Email

```php
<?php
session_start();
if (!isset($_SESSION['user_id'])) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Unauthorized']);
exit();
}

include('db.php');

if ($_SERVER['REQUEST_METHOD'] !== 'POST') {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Invalid request method']);
exit();
}

if (!isset($_POST['contact_id']) || !isset($_POST['email']) || !isset($_POST['name'])) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Missing required data']);
exit();
}

$contact_id = $_POST['contact_id'];
$email = $_POST['email'];
$name = $_POST['name'];

try {
// Manual PHPMailer includes
require 'PHPMailer/src/Exception.php';
require 'PHPMailer/src/PHPMailer.php';
require 'PHPMailer/src/SMTP.php';

// Import PHPMailer classes
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;
use PHPMailer\PHPMailer\SMTP;

// Create a new PHPMailer instance
$mail = new PHPMailer(true);

// Server settings
$mail->isSMTP();
$mail->Host = 'smtp.gmail.com'; // Gmail SMTP server
$mail->SMTPAuth = true;
$mail->Username = 'hotelrobertoinfo@gmail.com'; // Gmail address
$mail->Password = 'waph wlqa jkmz aovr'; // Gmail app password
$mail->SMTPSecure = PHPMailer::ENCRYPTION_STARTTLS;
$mail->Port = 587;

// Recipients
$mail->setFrom('hotelrobertoinfo@gmail.com', 'Roberto Hotel & Resort');
$mail->addAddress($email, $name);
$mail->addReplyTo('hotelrobertoinfo@gmail.com', 'Roberto Hotel & Resort');

// Content
$mail->isHTML(true);
```

```
$mail->Subject = 'Thank you for contacting Roberto - Hotel & Resort';
$message = "
<html>
<head>
<title>Thank you for contacting Roberto</title>
<style>
body { font-family: Arial, sans-serif; line-height: 1.6; color: #333; }
.container { max-width: 600px; margin: 0 auto; padding: 20px; }
.header { background: #f8f9fa; padding: 20px; text-align: center; border-radius: 5px; }
.content { padding: 20px; background: white; }
.footer { text-align: center; padding: 20px; color: #666; font-size: 14px; }
</style>
</head>
<body>
<div class='container'>
<div class='header'>
<h2>Roberto - Hotel & Resort</h2>
</div>
<div class='content'>
<p>Hello <strong>$name</strong>,</p>
<p>We received your information. We will contact you shortly.</p>
<p>Thank you. Have a good day!</p>
<br>
<p>Best regards,<br><strong>Roberto Team</strong></p>
</div>
<div class='footer'>
<p>&copy; " . date('Y') . " Roberto - Hotel & Resort. All rights reserved.</p>
</div>
</div></body></html>
";
$mail->Body = $message;
// Alternative plain text version
$mail->AltBody = "Hello $name,\n\nWe received your information. We will contact you shortly.\n\nThank
you. Have a good day!\n\nBest regards,\nRoberto Team";

// Send email
$mail->send();
// Log the email sending in database
$stmt = $pdo->prepare("UPDATE contact_submissions SET email_resent = 1, email_resent_at = NOW() WHERE
id = ?");
$stmt->execute([$contact_id]);
header('Content-Type: application/json');
echo json_encode(['success' => true, 'message' => 'Email sent successfully']);
} catch (Exception $e) {
header('Content-Type: application/json');
echo json_encode(['success' => false, 'message' => 'Error sending email: ' . $mail->ErrorInfo]);
}
?>
```

A.22 Logout

```
<?php
include('db.php');
session_start();
session_destroy();
header('Location: login.php');
exit();
```

?>

2.6 Summary

This chapter represents the complete design and implementation of the Roberto Hotel & Resort system. I explained here how the system is planned, developed, and moved with functional and non-functional requirements. The chapter mapped the function requirements such as registration, login, cart system, booking, payment, and loyalty program. Also, performance, security, usability, etc. include no functional requirements.

Various diagrams are included in this chapter, such as use case diagram, activity diagram, sequence diagram, class diagram, and ER diagram to understand the workflow, dataflow and user interaction with system structure.

Sample coding is also introduced in this chapter with implementation. So, we can say chapter 2 is a blueprint of the entire system.

Chapter 3 Software Testing

3.1 Introduction

Software testing is a very important part of SDLC that ensures all modules are working correctly (individually and collectively). The chapter represents the testing conducted to verify and validate all the functional and non-functional requirements. The main objective is to find and identify system errors, how system behaves, and interconnected modules are working as per requirements.

3.2 Testing Features

The Roberto Hotel & Resort System has many functionalities and is tested to ensure its stability. All the features below are tested with condition, valid input, invalid input, and error case scenario.

3.2.1 Feature to Be Tested

a. User Registration
b. Login
c. Update User Information
d. Change Password
e. Smart Booking Recommendation
f. Cart system
g. Room Booking
h. Online payment system
i. Booking cancellation
j. Refund
k. Add Room
l. Room Management
m. Earning
n. User Management
o. Contact Management
p. Loyalty Reward System
q. Add Blog
r. Blog Management
s. Chatbot
t. Logout

3.3 Testing Strategies

3.3.1 Test Approach

The text approaches I adopted in this this project in below:

3.3.1.1 Black Box Testing: Bax box testing is software testing method which mainly focus on the input and output of a functionality. Features like registration, login, booking are tested based on the expected result

3.3.1.2 Functional Testing: Every individual module is tested to verify its behaviour. Like booking, payment, loyalty reward are tested to ensure the workflow

3.31.3 Integration Testing: The interconnected modules are tested in this approach.

Such as:

- Booking --> Payment --> Reward Coin
- Reward Coin --> Discount
- Cancellation Booking --> Refund Request

3.3.1.4 Usability Testing: System navigations, form validation, system interface and responsive design are tested to ensure good usability experience.

3.31.5 Security Testing: The security criteria are tested to prevent unauthorized access to the system.

Such as:

- Invalid login attempts
- Payment redirection securely
- Access limitation for users

3.3.2 Pass/Fail Criteria

Test case marked **Pass** if:

- Providing expected output
- Functions work fine under valid input
- System handles invalid input by showing correct error detection message
- Integrated service like payment returns expected result
- No conflict between collective modules

Test case marked **Faill** if:

- Does not match with expected output
- Functions do not validate correct input
- Server crash during interconnected modules processing
- User interface behaves incorrectly

3.4 System Testing (Test Cases with Report)

Table: 3.4.1: **Test case 01: User Registration**

Test case:		**Test case name:** User Registration				
System: Roberto Hotel System		**Sub-system:** User Authentication				
Designed by: Sowrav Mondal		**Design Date:** 02/07/2025				
Executed by: Sowrav Mondal		**Execution Date:** 02/07/2025				
	Description: The user register on the system by using the valid information					
	Precondition: Users visit the registration page					
Step	**Name**	**Username**	**Email**	**Password**	**Pass /Fail**	**Comments**
1	Sowrav Mondal	sowrav	sowrav17sep@gmail.com	992288	Pass	Valid data and registration successful
2		sowrav	Sowrav17sep@gmail.com	992288	Fail	User must enter name
3	Sowrav Mondal	sowrav		992288	Fail	Email required for registration
4	Akash Sarker	akash	akashsarker@gmail.com	657565	Pass	Valid data and registration success
5	Akash Sarker	akash	akashsarker@gmail.com		Fail	Password is required
Post Condition: User needs to put all valid information to register on the system						

Table: 3.4.2: **Test case 02: Login**

Test case:		**Test case name:** Login		
System: Roberto Hotel System		**Sub-system:** User Authentication		
Designed by: Sowrav Mondal		**Design Date:** 02/07/2025		
Executed by: Sowrav Mondal		**Execution Date:** 02/07/2025		
	Description: User logs into the system with valid credentials and they will be automatically redirected to their dashboard			
	Precondition: User must be registered on the system			
Step	**Username**	**Password**	**Pass /Fail**	**Comments**
1	sowrav	992288	Pass	Login successful and redirect to user dashboard
2	sowrav		Fail	User must enter password
3	admin	656567	Pass	Login successful and redirect to admin dashboard
4		656567	Fail	Username is required
5	akash	656567	Fail	Login successful and redirect to receptionist dashboard
Post Condition: User needs to put all valid login credentials				

Table: 3.4.3: **Test case 03: Update User Information**

Test case:			**Test case name:** Update User Information			
System: Roberto Hotel System			**Sub-system:** Profile Management			
Designed by: Sowrav Mondal			**Design Date:** 13/10/2025			
Executed by: Sowrav Mondal			**Execution Date:** 14/10/2025			
		Description: The user register on the system by using the valid information				
		Precondition: Users visit the registration page				
Step	**Name**	**Phone**	**Address Line**	**City**	**Pass /Fail**	**Comments**
1	Sowrav Mondal	01741835718	Khagan, Akrain	Dhaka	Pass	Update successful
2	Sowrav Mondal	01741835718		Dhaka	Fail	Address Line required
3	Sowrav Mondal	01741835718	Khagan, Akrain		Fail	City is required
4		01741835718	Khagan, Akrain	Dhaka	Pass	Update successful
Post Condition: User needs to put all valid information to update their information in system						

Table: 3.4.4: **Test case 04: Change Password**

Test case:		**Test case name:** Change Password			
System: Roberto Hotel System		**Sub-system:** Profile Management			
Designed by: Sowrav Mondal		**Design Date:** 13/10/2025			
Executed by: Sowrav Mondal		**Execution Date:** 13/10/2025			
	Description: The user wants to change current password to new one				
	Precondition: Users visit the registration page				
Step	**Current Password**	**New Password**	**Confirm Password**	**Pass /Fail**	**Comments**
1	992288	676767	676767	Pass	Password changed successful
2	992288	6767	9898	Fail	Password does not match
3	656567	45455	45455	Fail	Password must be 6 character long
Post Condition: Password update only if validation is passed					

Table: 3.4.5: **Test case 05: Smart Booking Recommendation**

Test case:		**Test case name:** Smart Booking Recommendation		
System: Roberto Hotel System		**Sub-system:** Recommendation Engine		
Designed by: Sowrav Mondal		**Design Date:** 16/09/2025		
Executed by: Sowrav Mondal		**Execution Date:** 16/09/2025		
	Description: System will recommend room by user preference			
	Precondition: Users visit the smart booking page			
Step	**Preference**	**Response**	**Pass /Fail**	**Comments**
1	Room Type: Couple	Couples' rooms recommended	Pass	Matching room recommended
2	Price Range: 80 to 100	Deluxe room recommended	Pass	Matching room recommended
3		No recommendation	Fail	Need input / preferences
Post Condition: Recommended rooms only when user input preference and their needs				

Table: 3.4.6: **Test case 06: Cart System**

Test case:				**Test case name:** Cart System		
System: Roberto Hotel System				**Sub-system:** Cart Management		
Designed by: Sowrav Mondal				**Design Date:** 13/10/2025		
Executed by: Sowrav Mondal				**Execution Date:** 13/10/2025		
		Description: Customer can add & remove rooms and a process checkout option on cart system				
		Precondition: Users must be logged in into the system				
Step	**Action**	**Room**	**Check-in and Checkout**	**Response**	**Pass /Fail**	**Comments**
1	Add	Suite	9/12/25 to 12/12/25	Room added successfully to the cart	Pass	Room Added
2	Remove	Family	11/12/25 to 12/12/25	Room removed successfully from the cart	Pass	Room Removed
3	Process Checkout	Suite	9/12/25 to 12/12/25	Go to checkout page to make a booking	Pass	Works
4	Add	Suite	9/12/25 to 12/12/25	Room not available	Fail	Room not added
Post Condition: Rooms appears in the cart with proper date						

Table: 3.4.7: **Test case 07: Room Booking**

Test case:		**Test case name:** Room Booking					
System: Roberto Hotel System		**Sub-system:** Booking Management					
Designed by: Sowrav Mondal		**Design Date:** 06/08/2025					
Executed by: Sowrav Mondal		**Execution Date:** 06/08/2025					
	Description: Customer confirms bookings						
	Precondition: Customer must be logged in into the system and room need to be added on cart						
Step	**Cart Status**	**Name**	**Phone**	**Address**	**Payment Method**	**Pass /Fail**	**Comments**
1	Room added	Sowrav Mondal	01741835718	Khagan,Savar	Online Payment	Pass	Booking Confirmed
2	Cart Expire	Sowrav Mondal	01741835718	Khagan,Savar		Fail	No room found
3	Room added	Sowrav Mondal	01741835718	Khagan,Savar		Fail	Payment must be completed
4	Room added	Akash Sarker	01936493976	Akrain,Savar	Online Payment	Pass	Booking Confirmed
Post Condition: Payment must be completed							

Table: 3.4.8: **Test case 08: Online Payment System**

Test case:							
Test case name: Online Payment System							
System: Roberto Hotel System							
Sub-system: Stripe Payment Gateway							
Designed by: Sowrav Mondal							
Design Date: 25/09/2025							
Executed by: Sowrav Mondal							
Execution Date: 29/09/2025							
		Description: Customer confirms bookings					
		Precondition: Customer must be logged in into the system and room need to be added on cart					
Step	**Method**	**Account / card number**	**Expiration date**	**Security code**	**Amount**	**Pass /Fail**	**Comments**
1	Online Payment	4242 4242 4242 4242	12/26	847	26	Pass	Payment Success - > Booking Confirmed
2	Online Payment		12/26	847	26	Fail	Card Number Required
3	Online Payment	4242 4242 4242 4242	12/26		26	Fail	Security code required
4	Online Payment	4242 4242 4242 4242	12/26	847	26	Fail	Payment gateway error
Post Condition: Booking status booked from confirmed and transaction details saved in database							

Table: 3.4.9: **Test case 09: Booking Cancellation**

Test case:				**Test case name:** Booking Cancellation	
System: Roberto Hotel System				**Sub-system:** Booking Management	
Designed by: Sowrav Mondal				**Design Date:** 15/10/2025	
Executed by: Sowrav Mondal				**Execution Date:** 15/10/2025	
		Description: Customer cancels an existing booking due to some reason			
		Precondition: Customer must have a booking on the system and check-in date should not be expired			
Step	**Booking Status**	**Action**	**Alert Action**	**Pass / Fail**	**Comments**
1	Confirm	Cancel Booking	Okay	Pass	Booking Cancelled
2	Confirm	Cancel Booking	No or cancel	Fail	Customer don't want to cancel the booking
3	Checked-in			Fail	Booking cannot be cancelled after check-in date
4	Cancelled	Cancel Booking	okay	Fail	Booking Already cancelled
Post Condition: Booking status changed to cancelled and customer cannot use cancel booking					

Table: 3.4.10: **Test case 10: Refund**

Test case:		**Test case name:** Booking Cancellation				
System: Roberto Hotel System		**Sub-system:** Booking Management				
Designed by: Sowrav Mondal		**Design Date:** 15/10/2025				
Executed by: Sowrav Mondal		**Execution Date:** 16/10/2025				
		Description: Customers receive refund immediately after submitting refund request and a 2% deduction will be applied from total paid amount				
		Precondition: Customer must have a valid cancelled booking				
Step	**Booking Status**	**Refund request**	**Paid Amount**	**Expected Amount**	**Pass / Fail**	**Comments**
1	Confirm	Not Possible	100		Fail	Booking needs to be cancelled to submit refund request
2	Cancelled	Submitted	180	176.4	Pass	Refund process successful
3	Cancelled	Submitted	180	176.4	Fail	Refund already processed
4	Cancelled	Submitted	180	180	Fail	Invalid amount
Post Condition: Refund process immediately and transaction save in database. Also Booking status changed to Refunded						

Table: 3.4.11: **Test case 11: Add Room**

Test case:				**Test case name:** Add Room			
System: Roberto Hotel System				**Sub-system:** Room Management (Admin)			
Designed by: Sowrav Mondal				**Design Date:** 08/08/2025			
Executed by: Sowrav Mondal				**Execution Date:** 08/08/2025			
			Description: Admin adds new room to the system by providing valid room details				
			Precondition: Admin must be logged-in and navigate to dashboard Add Room tab				
Step	**Room Number**	**Type**	**Price**	**Images**	**Amenities**	**Pass /Fail**	**Comments**
1	R 101	Single	25	single1.jpg single2.jpg	Wi-Fi	Pass	Room added successfully
2	R 102	Single		single3.jpg single2.jpg	Wi-Fi	Fail	Room price required
3	R 201	Deluxe	60	deluxe.jpg	Wi-Fi, AC	Pass	Room added successfully
4	R 202	Deluxe	60		Wi-Fi, AC	Fail	Room image required
5		Suite	90	suite1.jpg	Wi-Fi, AC, TV	Fail	Room number required
Post Condition: Room appear in the Room management section and room listing page							

Table: 3.4.12: **Test case 12: Room Management**

Test case:		**Test case name:** Room Management		
System: Roberto Hotel System		**Sub-system:** Room Management (Admin)		
Designed by: Sowrav Mondal		**Design Date:** 08/08/2025		
Executed by: Sowrav Mondal		**Execution Date:** 08/08/2025		
		Description: Admin manages existing rooms including editing and deleting rooms		
		Precondition: Admin must be logged-in and navigate to dashboard Room Management tab		
Step	**Action**	**Input Data / Room Status**	**Pass /Fail**	**Comments**
1	Edit	Change price from: 120 to 80 Change room type: Deluxe to Double	Pass	Room updated successfully
2	Edit	Remove Images	Fail	Room image required
3	Edit	Amenities: Wi-Fi, Ac, TV, Balcony, Mini bar, large bed, Sitting Area	Pass	Room updated successfully
4	Delete	Room status: Booked	Fail	Booked room cannot be delectable
5	Delete	Room status: Available	Pass	Room deleted successfully
Post Condition: Room updates immediately appear in the Room management section and room listing page with update information. Room is permanently removed from the system				

Table: 3.4.13: **Test case 13: Earning Management**

Test case:				**Test case name:** Earning Management	
System: Roberto Hotel System				**Sub-system:** Earning Management Module	
Designed by: Sowrav Mondal				**Design Date:** 02/11/2025	
Executed by: Sowrav Mondal				**Execution Date:** 02/11/2025	
			Description: Admin can see total earning, booking, cancellation with clickable graphs		
			Precondition: Admin must be logged-in and navigate to dashboard Earning tab		
Step	**Action**	**Response**	**Expected Result**	**Pass /Fail**	**Comments**
1	Click Earning	Earning bar hide on graph	Show only cancellation bar on graph	Pass	Data visible
2	Click Cancellation	Cancellation bars hide on graph	Show only earning bar on graph	Pass	Data visible
3	Click Earning	Earning bar hide on graph	Graphs failed to load	Fail	Data not displaying
4	Click Confirmed Booking	Confirmed booking hide on graph	Show only cancellation booking on donut chart	Pass	Data showing
5	Click or hover on graph	Tooltip comes	Tooltip shows those portion value	Pass	Individual model data showing
Post Condition: Admin can view total booking, total earning, profit from on graphs and clicking them showing their data in tooltip					

Table: 3.4.14: **Test case 14: User Management**

Test case:			**Test case name:** User Management		
System: Roberto Hotel System			**Sub-system:** User Management Module		
Designed by: Sowrav Mondal			**Design Date:** 13/10/2025		
Executed by: Sowrav Mondal			**Execution Date:** 13/10/2025		
			Description: Admin can see total user including their information and coins they have and can be able to delete a user from system		
			Precondition: Admin must be logged-in and navigate to User sections		
Step	**Action**	**Response**	**Expected Result**	**Pass /Fail**	**Comments**
1	View user	All user list loaded	Show all user data including their information	Pass	Data visible
2	Delete	Showing a confirmation alert	User deleted successfully from the system	Pass	Deleted
3	Delete	User has a valid booking	User cannot be deleted at this time	Fail	User stays on the system
Post Condition: Showing user information update to date and on deleted user removes from database					

Table: 3.4.15: **Test case 15: Contact Management**

Test case:			**Test case name:** Contact Management		
System: Roberto Hotel System			**Sub-system:** Contact Message		
Designed by: Sowrav Mondal			**Design Date:** 13/10/2025		
Executed by: Sowrav Mondal			**Execution Date:** 13/10/2025		
	Description: Admin can view, send email, export and delete the contact message or support request.				
	Precondition: Admin must be logged-in and navigate to dashboard Contact Detail section				
Step	**Action**	**Condition**	**Expected Result**	**Pass /Fail**	**Comments**
1	View Contact List		All contact lists appear	Pass	Admin view all contact list
2	Send Email	Valid email address	Email sent to the customer	Pass	Customers receive email
3	Send Email	Invalid email address	Show error	Fail	Email cannot be sent
4	Delete	Valid contact id	Contact deleted	Pass	Contact information removed
5	Delete	Already deleted	Server error	Fail	Contact already deleted
6	Export	Click export button	File export as excel	Pass	All contact information exported
Post Condition: Email resent to the customer. Contact list exported and after delete, the contact list re-appears correctly					

Table: 3.4.16: **Test case 16: Loyalty System Reward**

Test case:			**Test case name:** Loyalty System Reward		
System: Roberto Hotel System			**Sub-system:** Loyalty Reward, Coin Transaction		
Designed by: Sowrav Mondal			**Design Date:** 02/10/2025		
Executed by: Sowrav Mondal			**Execution Date:** 05/10/2025		
	Description: Loyalty system gives customer loyalty coins after completing the booking. Also, customer can use them as a discount on next booking				
	Precondition: User must be login to the system as a customer				
Step	**Action**	**Input / Condition**	**Expected Result**	**Pass /Fail**	**Comments**
1	Complete Booking	Paid amount: 200	Coin Earned: 13	Pass	Coin added to the customer account
2	Complete another booking	Paid amount: 150	Coin Earned: 10	Pass	Coin added to the customer account
3	View Coin	Go to my coin section	Total coin: 23	Pass	Show all coin
4	Redeem coin	Use max	Get discount: 23$	Pass	Customers get discount by coin
5	Redeem Coin	Use 10 coin	Insufficient coin	Fail	Customer has no coin on his account
Post Condition: Coin add and subtract immediately after booing. Dashboard will show exact coin amount					

Table: 3.4.17: **Test case 17: Add Blog**

Test case:			**Test case name:** Add Blog			
System: Roberto Hotel System			**Sub-system:** Blog Management (Admin)			
Designed by: Sowrav Mondal			**Design Date:** 07/08/2025			
Executed by: Sowrav Mondal			**Execution Date:** 07/08/2025			
	Description: Admin adds new to the system by providing valid blog details					
	Precondition: Admin must be logged-in and navigate to dashboard Add Blog tab					
Step	**Title**	**Content**	**Author**	**Image**	**Pass /Fail**	**Comments**
1	One photo, a deluge of threats	VAIL, Ariz. — Cienega High School Principal Kim Middleton woke up early last Saturday to urgent me ..	Tyler Kingkade	blog1.jpg	Pass	Blog added successfully
2		President Donald Trump has pardoned Rudy Giuliani and scores of others..	Patrick Smith	blog2.jpg	Fail	Title required
3	Trump pardons Rudy Giuliani	President Donald Trump has pardoned Rudy Giuliani and scores of others..	Patrik Smith		Fail	Please upload a image
Post Condition: Blog appear in the Blog management section and blog listing page						

Table: 3.4.18: **Test case 18: Blog Management**

Test case:		**Test case name:** Blog Management		
System: Roberto Hotel System		**Sub-system:** Blog Management (Admin)		
Designed by: Sowrav Mondal		**Design Date:** 07/08/2025		
Executed by: Sowrav Mondal		**Execution Date:** 07/08/2025		
		Description: Admin manages existing blogs including editing and deleting blogs		
		Precondition: Admin must be logged-in and navigate to dashboard Blog Management tab		
Step	**Action**	**Input Data / Response**	**Pass /Fail**	**Comments**
1	Edit	Change title to: Supreme Court weighs Rastafarian man	Pass	Blog updated successfully
2	Edit	Remove Images	Fail	Blog image required
3	Edit	Change Author to: Sowrav Mondal	Pass	Blog updated successfully
4	Delete	Confirm delete on alert	Pass	Blog deleted successfully
Post Condition: Blogs appear on the blog listing and blog management. Deleted blogs removed from the system				

Table: 3.4.19: **Test case 19: Chatbot**

Test case:			**Test case name:** Chatbot		
System: Roberto Hotel System			**Sub-system:** Chatbot System		
Designed by: Sowrav Mondal			**Design Date:** 14/10/2025		
Executed by: Sowrav Mondal			**Execution Date:** 16/10/2025		
	Description: The user communicates with chatbot to ask question, get information and receive support				
	Precondition: Users must visit the website				
Step	**Action**	**Input**	**Expected result**	**Pass /Fail**	**Comments**
1	Open Chatbot		Hello! I'm your assistant. How can I help you today?	Pass	Reply with greetings
2	Send message	Support	Our support team is here to help! Call: +8801741835718 Email: admin@roberto.com	Pass	Provides contact information
3	Select option	Deluxe room	Our Deluxe Rooms: • King size bed • Free Wi-Fi + • Air conditioning • Luxury bathroom • Starting from $120/night	Pass	Password must be 6 character long
4	Send	(empty)	(nothing)	Fail	Message does not send
Post Condition: Chatbot working fine and handling message smoothly					

Table: 3.4.20: **Test case 20: Logout**

Test case:		**Test case name:** Logout		
System: Roberto Hotel System		**Sub-system:** User Authentication		
Designed by: Sowrav Mondal		**Design Date:** 02/07/2025		
Executed by: Sowrav Mondal		**Execution Date:** 02/07/2025		
	Description: User logout from the system and sessions destroyed to prevent unauthorized access			
	Precondition: User must be logged in the system			
Step	**Action**	**Condition**	**Pass /Fail**	**Comments**
1	Logout	Valid Session	Pass	Logout successful and redirected to login page
2	Logout	Session expired	Pass	Session already ended and system logout the user automatically
3	Multiple tab open, Logout from one	Valid Session	Pass	Session destroyed and logout from all tabs
4	Logout	Server does not respond	Fail	Try again to logout
5	Logout link broken	Invalid route or address	Fail	Logout function not processing
Post Condition: User fully logout from the system and it destroyed all sessions				

3.5 Summary

The complete testing process of each module of Roberto Hotel System is represented in this chapter to ensure every module functions are working correctly. I discussed testing approaches and strategies such as black box testing, usability testing, and security testing to verify its components and interaction.

Features were tested in detail, and each feature test case was executed using valid id, input, expected output and boundary inputs. The results of testing showed that how systems are behaving is different condition with different inputs including their workflow, success condition, validation error.

Overall testing confirms that the system is stable and reliable according to requirements. That test ensures the. system quality for real world use.

Chapter 4 Deployment and Maintenance

4.1 Introduction

The main purpose of this chapter is to explain the deployment and maintenance of the Roberto Hotel & Resort System. Deployment ensures that the system is configured correctly, installed properly, and accessible for end users. Maintenance ensures that the system is running smoothly after performing updates, fixing bugs, enhancing security, and improving performance.

The chapter mapped the steps taken to deploy the system on a live server and followed the SRLC model to ensure the system is reliable, stable, and scalable. The deployment environment, liver hosting requirements, and routine maintenance are also discussed in this chapter.

4.2 Try to follow the SRLC (software release life cycle)

The Software Release Life Cycle (SRLC) model describes the software that goes from initial development to deployment with further updates. The Roberto Hotel & Resort System follows a structured SRLC model or process to ensure quality, efficiency, and usability.

4.2.1 Pre-Alpha Stage

The Pre-Alpha stage describes initial stage of development:

1. Requirements gathering and feasibility studies
2. Preparing the UI/UX design
3. Creating database structure
4. Building core model like registration, login, booking

4..2.2 Alpha Phase

The alpha phase describes the partial completed modules and first stage testing:

1. Core functionality implemented
2. Early testing of booking and cart system
3. Early-stage testing and initial bug fixes
4. Testing without real user

4.2.3 Beta Release

The beta release a is mostly completed system:

1. The booking workflow integrated (Cart -- > Checkout --> Payment --> Booking -- > Loyalty reward)
2. Room management and user management modules are integrated for admin
3. Refund, booking cancellation, print booking details are introduced
4. Admin and receptionist's dashboard feature are implemented
5. Smart booking recommendation and chatbot functions are introduced
6. Perform testing with sample user

4.2.4 Release Candidate (RC)

This version is almost complete for deployment

1. Bug fixing and system optimization
2. Reviewing security (session, hash password, input validation)
3. Database optimization and cleanup
4. Integrating Stripe payment API completely
5. Adding SSL certificate to secure data transfer
6. Preparing documentation and user manual

4.2.5 Product Release

This is the final version for public use:

1. Deploy the system on the live server (Apache on cpanel)
2. Migrate the database
3. Enable domain and SSL certificate
4. Available the system for staff and customer
5. Monitoring the error

4.2.6 Maintenance Phase

1. Fixing bugs reported from user
2. Improving security and performance by update
3. Updating smart room recommendation algorithm
4. Adding news feature on customer demand
5. Backup scheduling timely

4.3 Summary

On this chapter I discussed, how the system is deployed in the live hosting server for real world environment. The SRLC model was followed in the deployment and maintenance phase to ensure high quality release. This chapter also discussed further improvement, bug fixes, performance optimization, and enhanced security by ongoing maintenance.

Chapter 5 User Manual

5.1 Introduction

This User manual guides a user to use the **Roberto Hotel & Resort Management** system step by step for both Administrator and Customer. The manual has proper instructions to using the system, such as registration, login, profile management, payment, loyalty system etc. The main objective of user manual is to make the system easy to understand for people who does not have proper technological knowledge.

5.2 Project Functionalities

User Dashboard :

1. Visit the website
2. Click the Login button from the navigation menu
3. Fill in the username and password information for the user or customer
4. Click the Login button

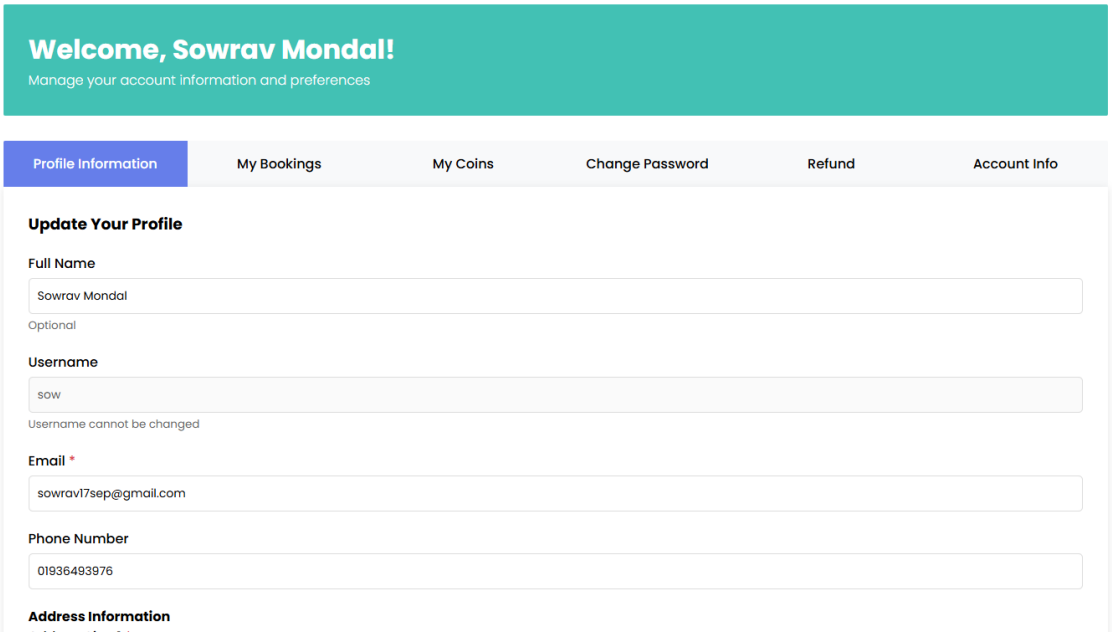


Figure 7.1: User Dashboard

Admin Dashboard :

1. Visit the website
2. Click the Login button from the navigation menu
3. Fill in the username and password information for the user or admin
4. Click the Login button

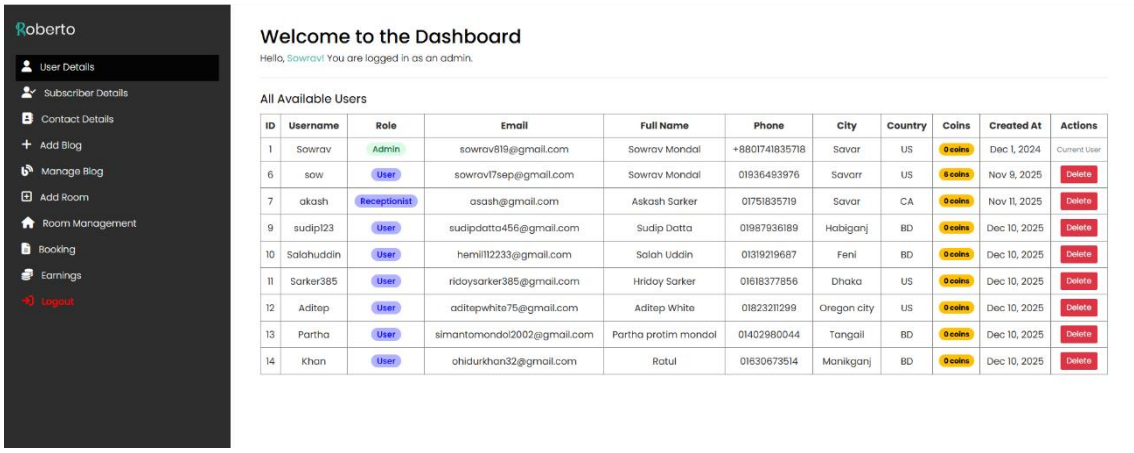


Figure 7.2: Admin Dashboard

Receptionist Dashboard :

1. Visit the website
2. Click the Login button from the navigation menu
3. Fill in the username and password information for the user or receptionist
4. Click the Login button

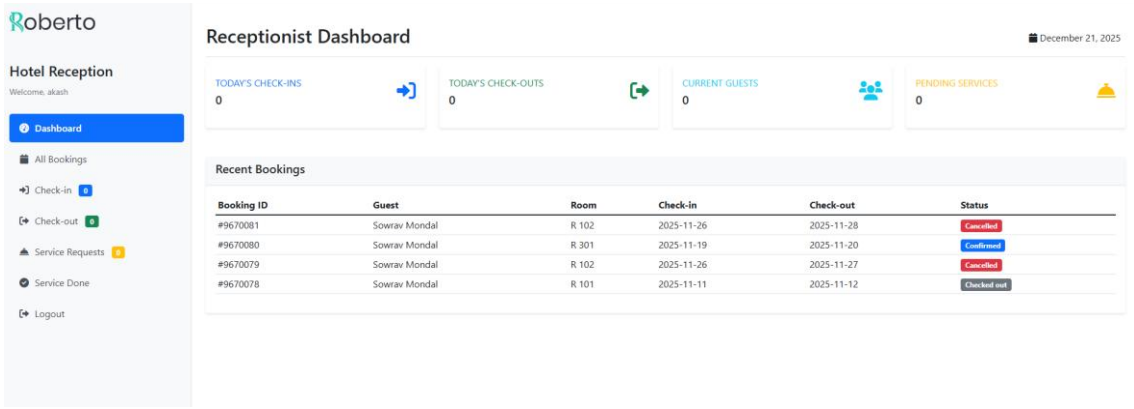


Figure 7.3: Receptionist Dashboard

User Registration:

5. Visit website
6. Click Register button from navigation menu
7. Fill-up the required information
8. Click Register button

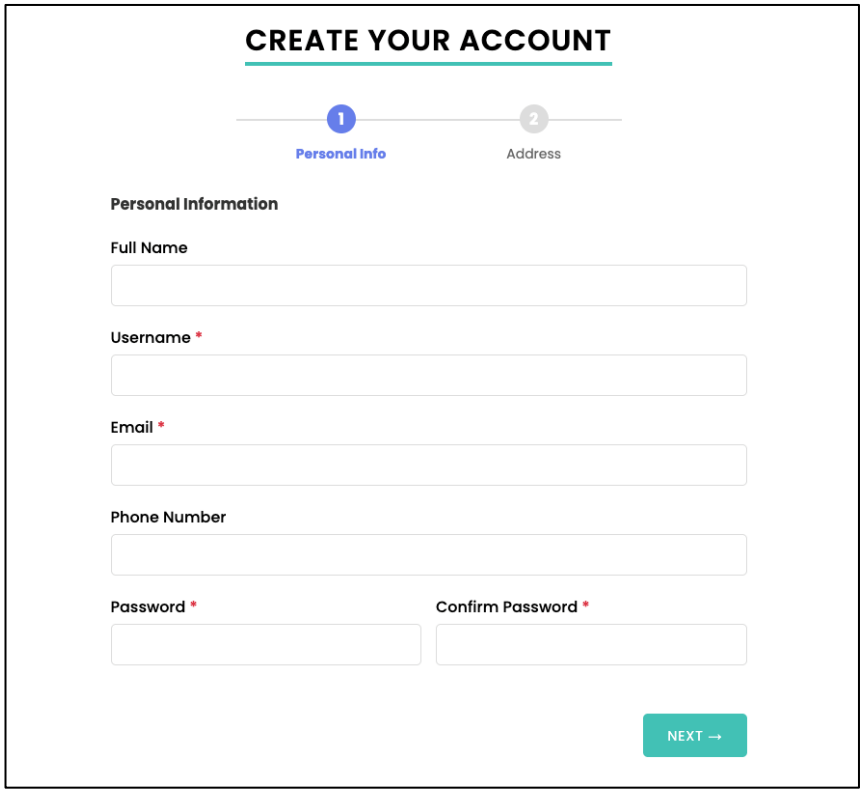


Figure 7.4: User Registration

Login (Customer, Admin, Receptionist):

1. Click Login button or option from menu
2. Enter valid username and password
3. Login successful and redirect to dashboard

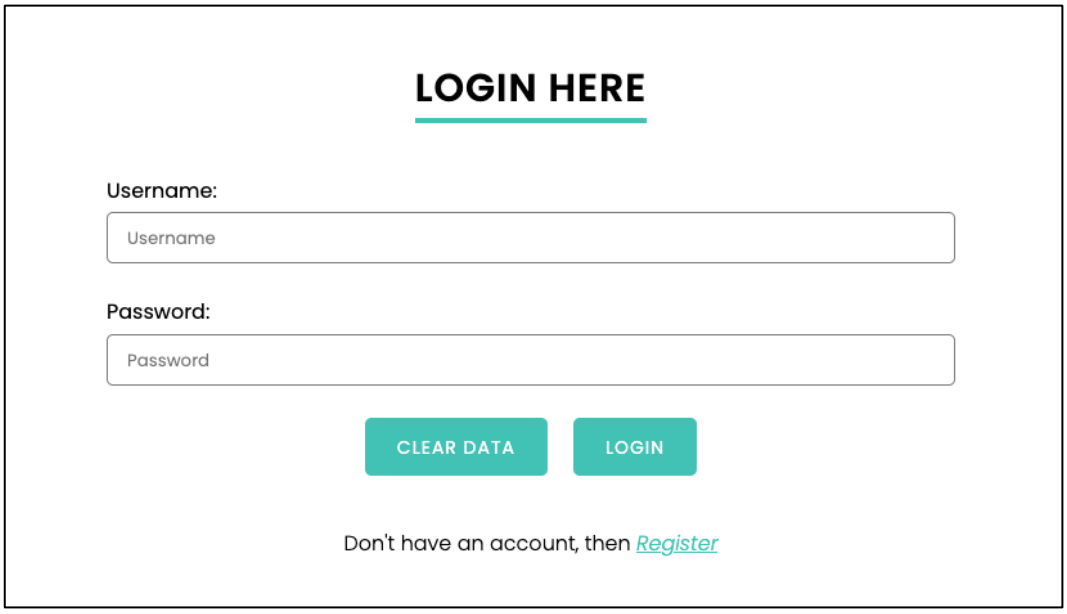


Figure 7.5: Login for customer, admin & receptionist

Update Profile Information (Customer):

1. Login to the system
2. Click My Account from the navigation menu
3. Redirect to My account page and Profile Information tab
4. Edit the information with valid input
5. Click update Profile Button
6. Massage shows “Profile successfully updated”

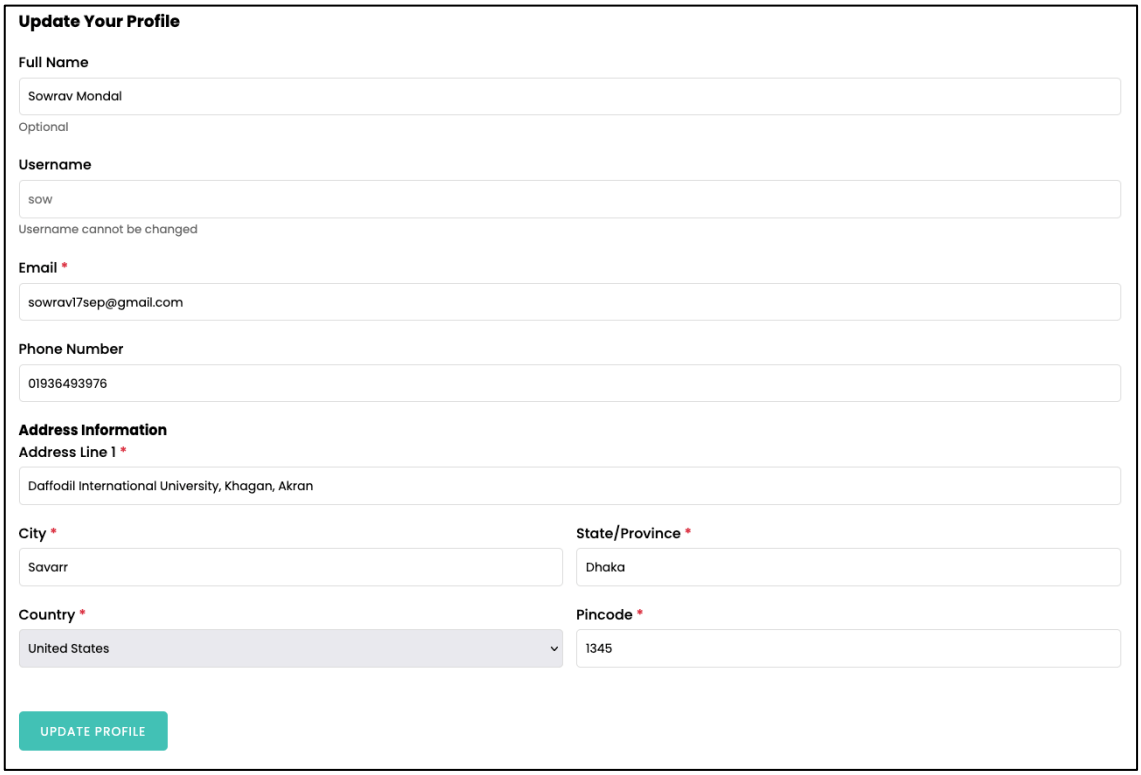


Figure 7.6: Update Profile Information

Change Password (Customer):

1. Login to the system
2. Click My Account from the navigation menu
3. Redirect to My account page
4. Click Changed Password tab
5. Input current password and New Password
6. Click Change Password button

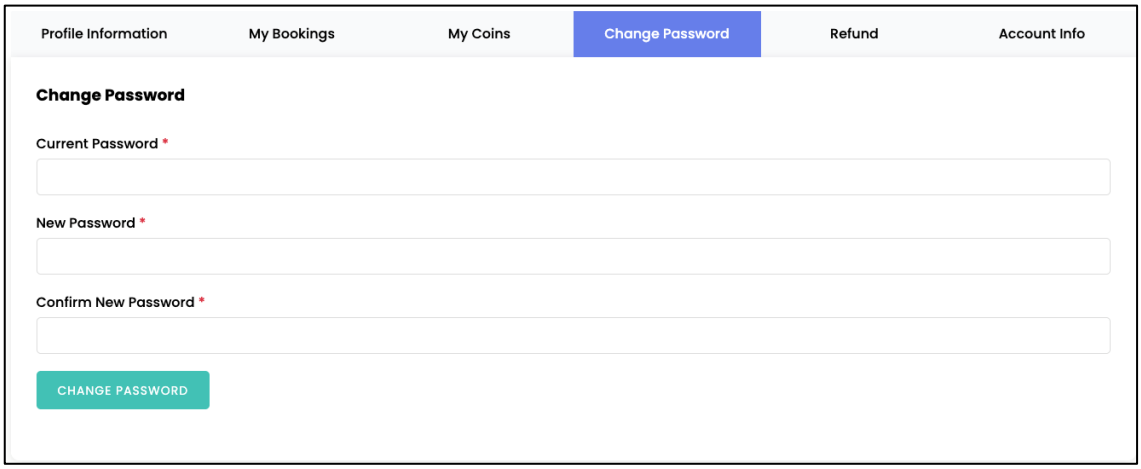


Figure 7.7: Changed Password

Smart Booking Recommendation (Customer):

1. Visit website
2. Click Smart Room from navigation menu and go to smart room page
3. Enter the preferences
4. Click Get Smart Recommendation button
5. Recommended room appear in below

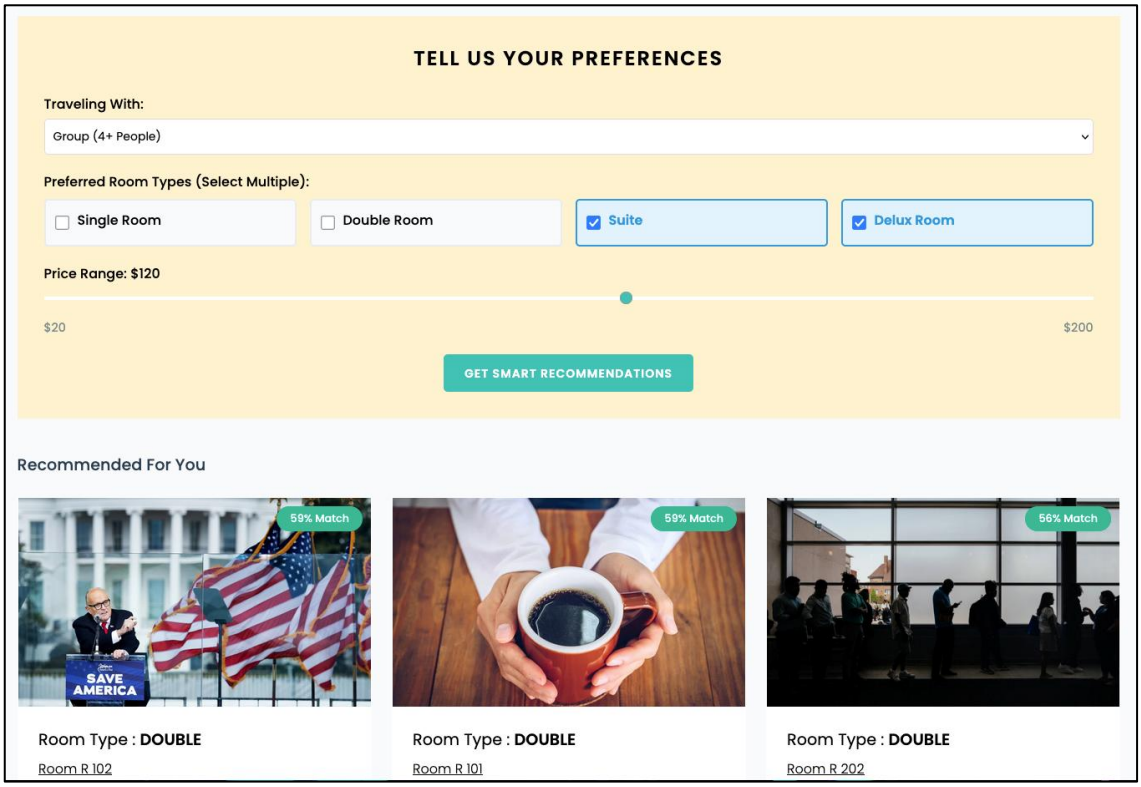


Figure 7.8: Smart Booking Recommendation

Cart System (Customer):

1. Login to the website
2. Select a from room from rooms page
3. Put check-in and check-out date
4. Click Add to Cart button
5. A successful message will show with Cart page link
6. Click Cart button
7. Show the room on cart

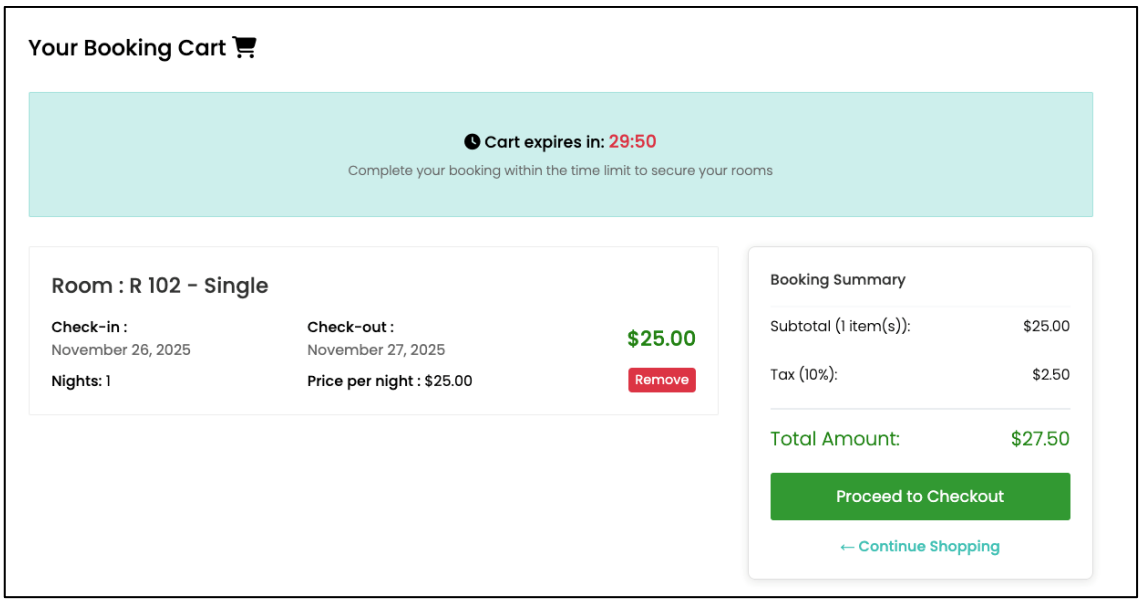


Figure 7.9: Cart System

Room Booking (Customer):

1. Login to the system
2. Click Cart from navigation menu
3. Got to cart page and show all booking on cart
4. Click Process to Checkout button and redirect to checkout page
5. Enter all valid information
6. Select additional Services
7. Make a payment
8. Redirect to booking successful page

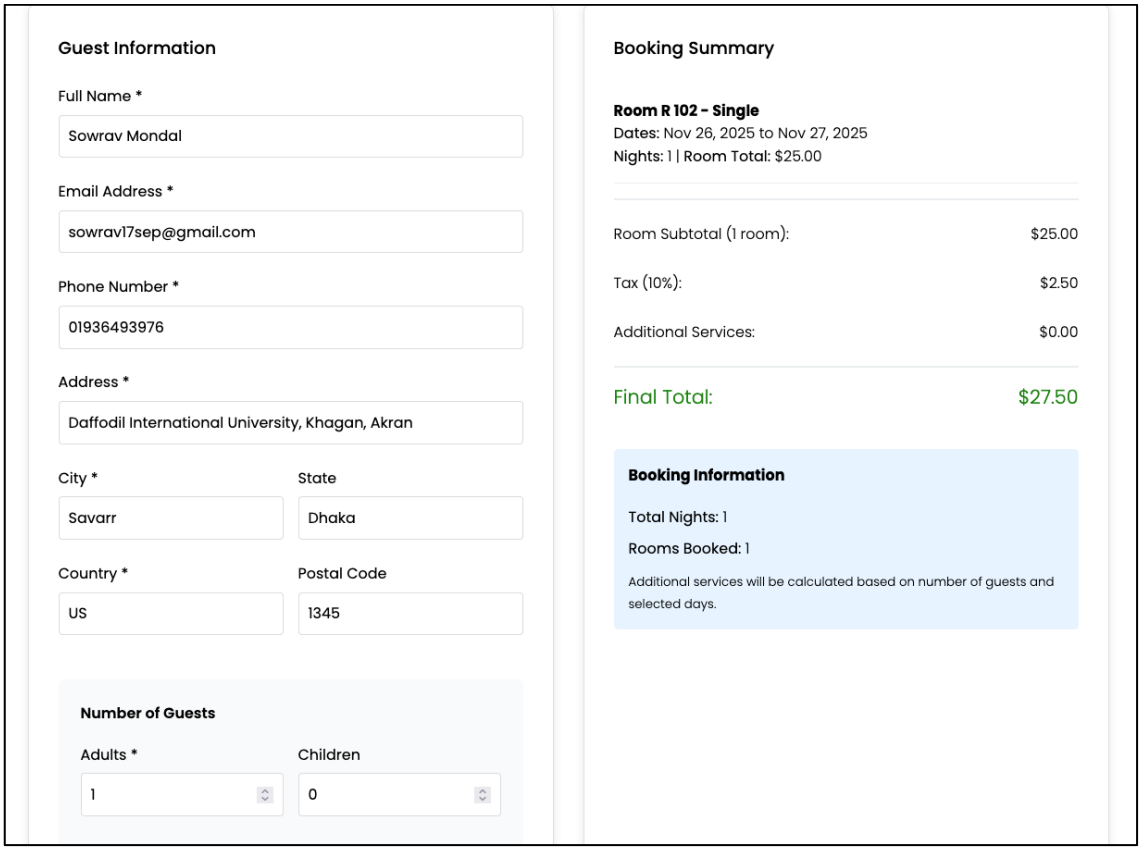


Figure 7.10: Room Booking

Online Payment System (Customer):

1. Go to checkout page
2. In the bottom of the page select payment method
3. Input valid card details
4. Click Pay to make a payment

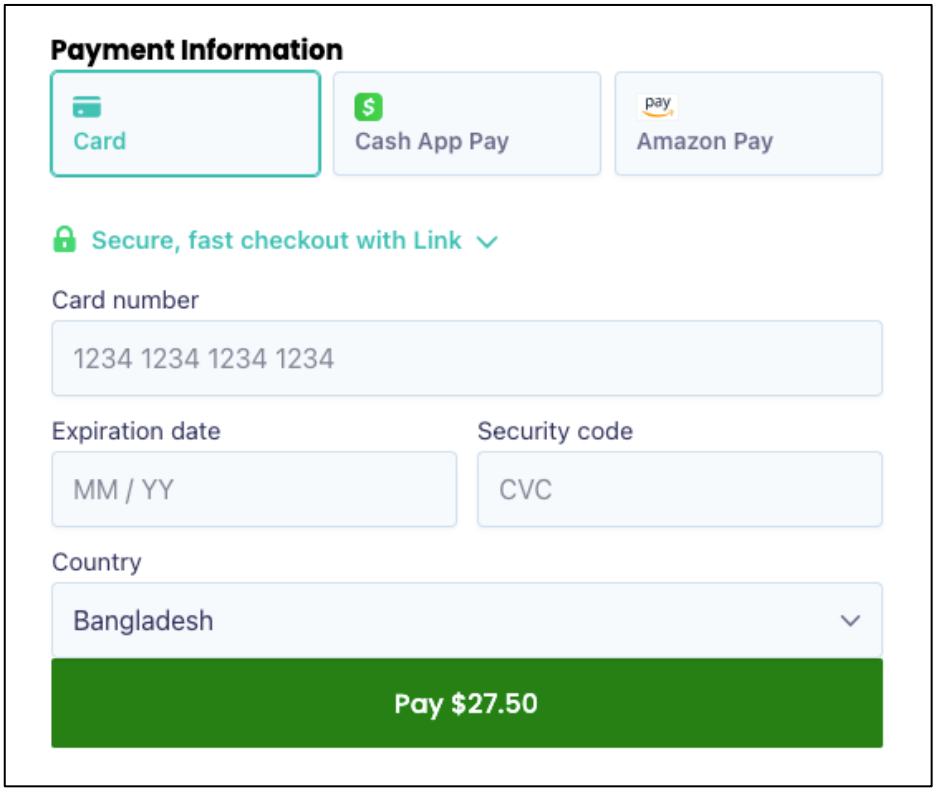


Figure 7.11: Online Payment System

Loyalty Reward (Customer):

1. Confirm a booking by making payment
2. Redirect to successful page
3. Coin adds and substrates information will show
4. Click Visit Dashboard button and redirect to my account page
5. Click My Coin tab
6. My coin section will show all available coin

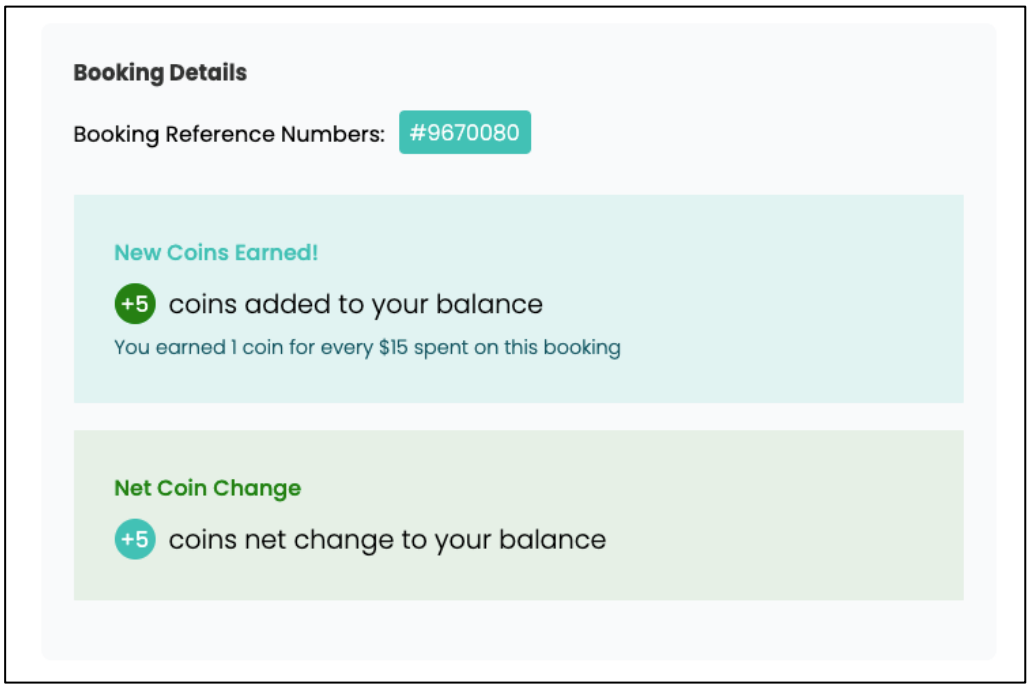


Figure 7.12: Loyalty Reward

Loyalty Discount (Customer):

Go to checkout page
Use coin for discount (1 coin = 1$)
System applies the discount on total amount

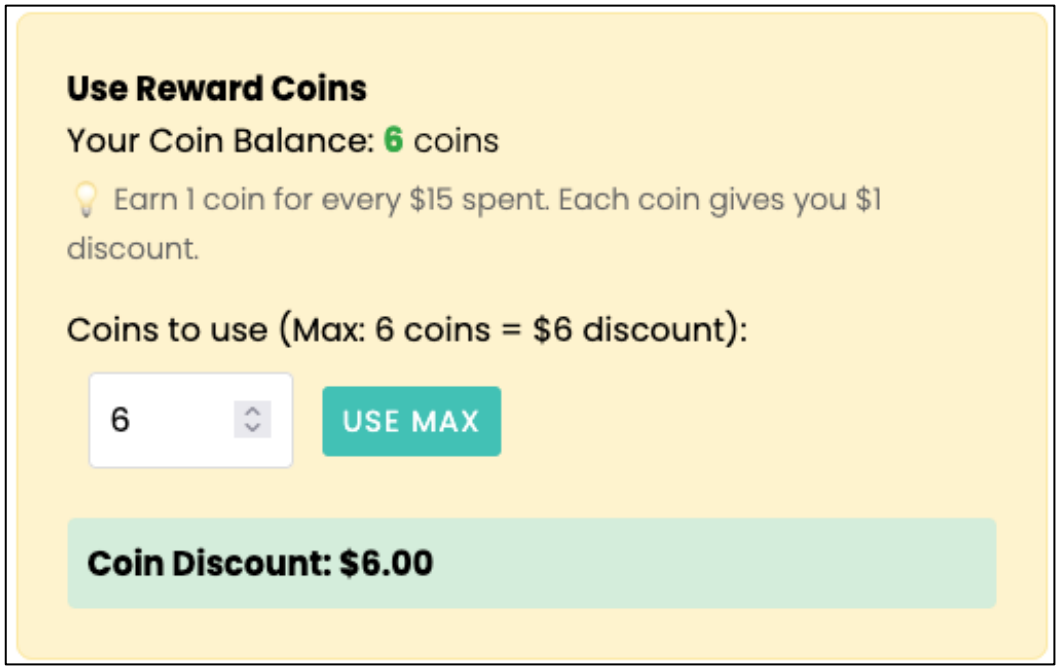


Figure 7.13: Loyalty Discount

Booking Cancellation (Customer):

1. Click Booking tab from My Account page
2. Booking section show all available booking
3. Click Cancel Booking
4. An Alert appear and confirm the cancellation
5. Massage will show about cancellation and updated the booking status Cancelled

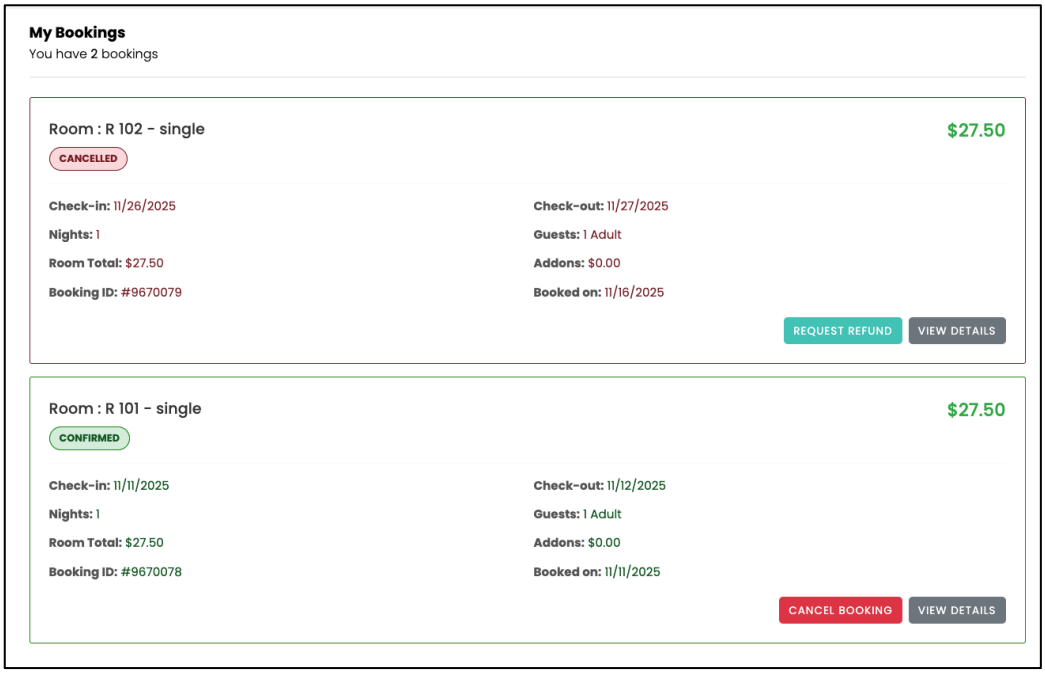


Figure 7.14: Booking Cancellation

Refund Request (Customer):

1. Click Booking tab from My Account page
2. Booking section show all available booking
3. Click Refund Request button
4. A popup appears and submits the form
5. Massage will show about refund request
6. Click Refund tab to show the requested refund and it's status

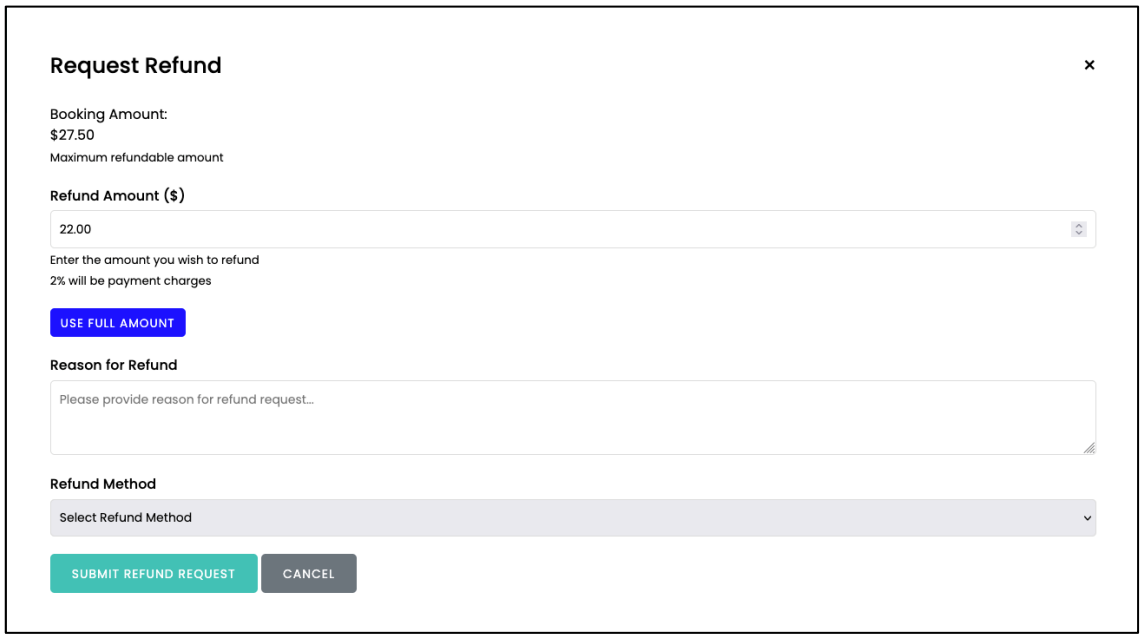


Figure 7.15: Refund Request

Customer Check-in and Checkout (Receptionist)

1. Login as receptionist with valid credential
2. Click Checking option from receptionist dashboard
3. All check-in and checkout information will be shown

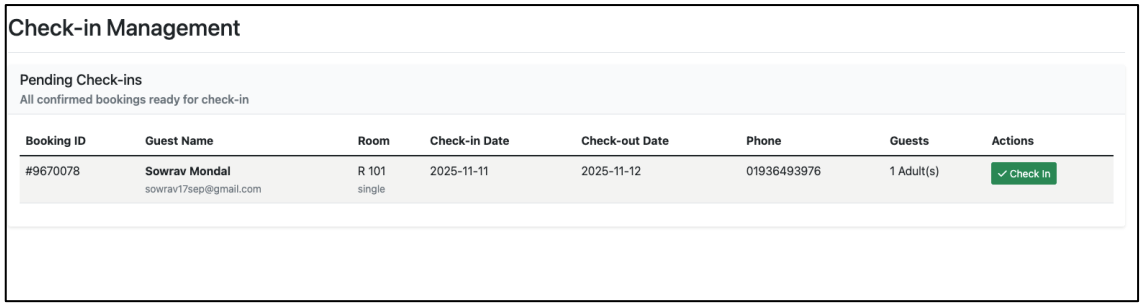


Figure 7.16: check-in and Check-out

Service Request (Receptionist):

1. Login as receptionist with valid credential
2. Click Service Request option from receptionist dashboard
3. All Service Request will be shown

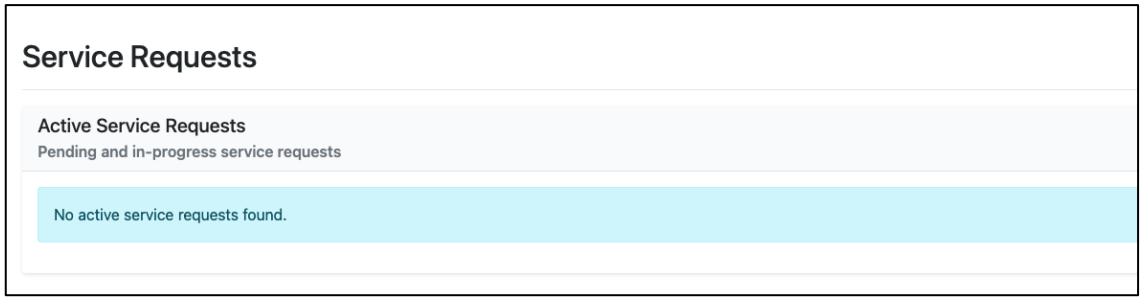


Figure 7.17: Service Request

Add Room (Admin):

1. Login as Admin and redirect to admin dashboard
2. Click Add Room from dashboard
3. Enter all information about room
4. Click Add Room button
5. Room will be added and redirected to Room management section

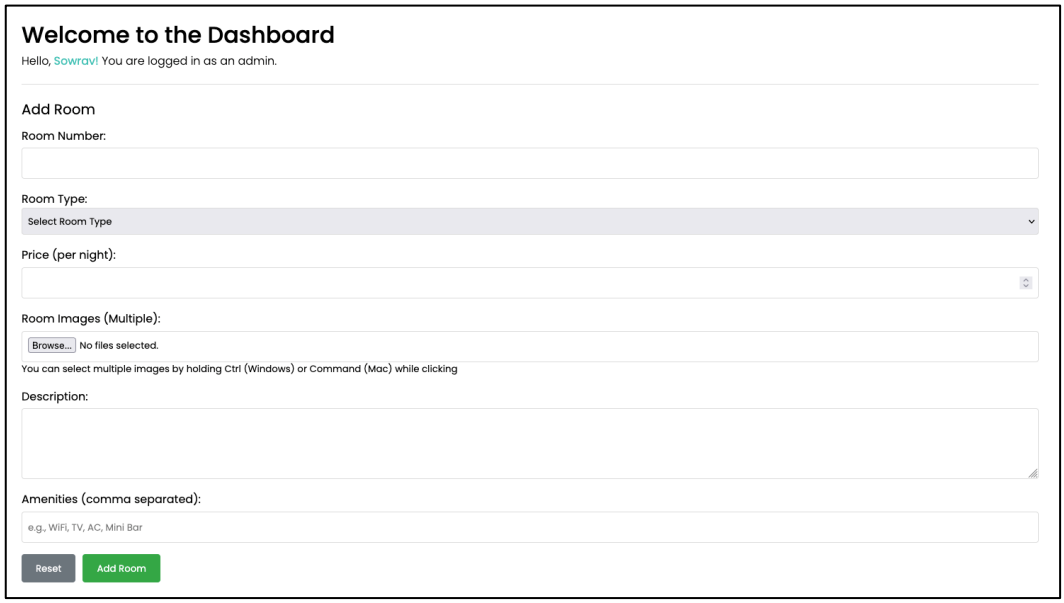


Figure 7.18: Add Room

Room Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Room Management section to view all room
3. To delete a room, click Delete and room will be deleted immediately
4. To edit a room, click Edit and a popup will appear.
5. Then edit the existing input and Click Update button
6. Room will be updated and appear in list

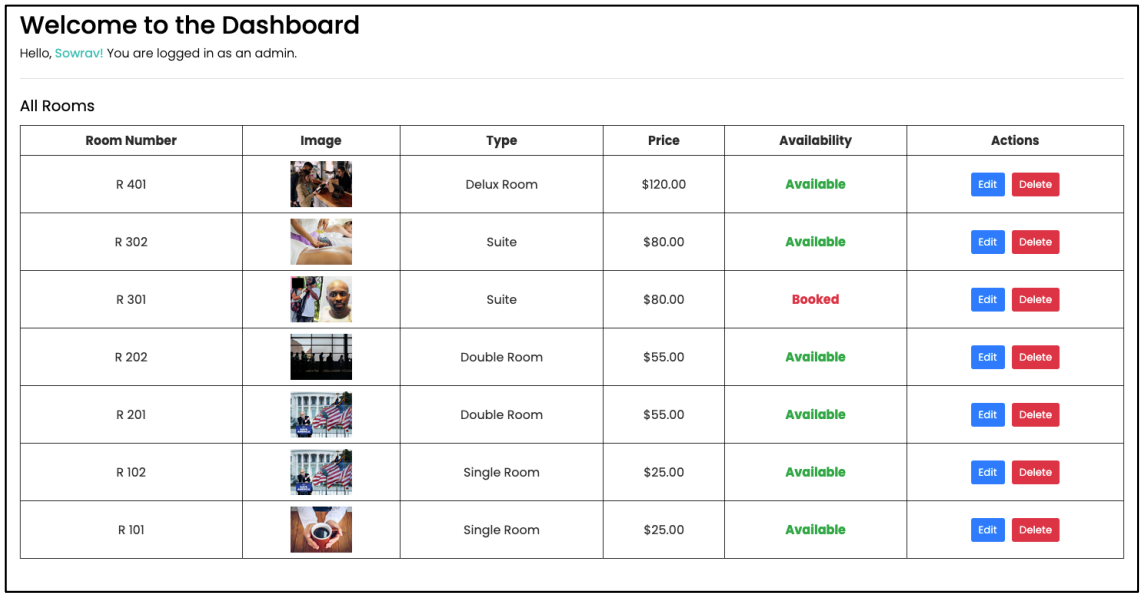


Figure 7.19: Room Management

Booking Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Booking section to view all booking

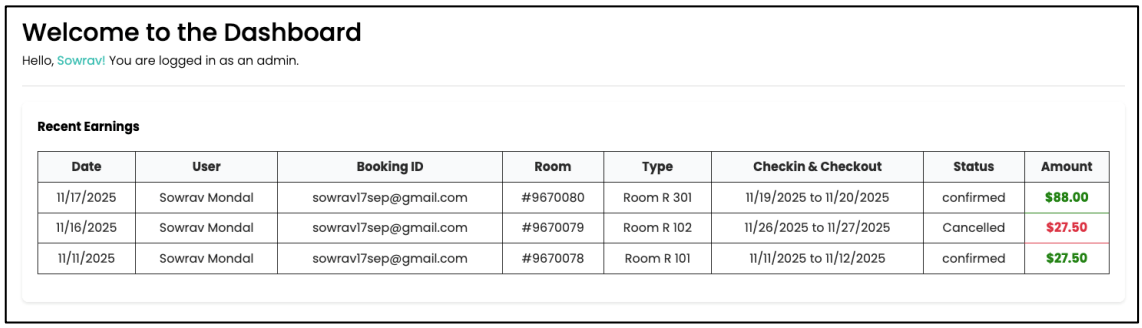


Figure 7.20: Booking Management

Earning Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Earning tab to view all earning details
3. Hover on graph any portion to get information on tooltip

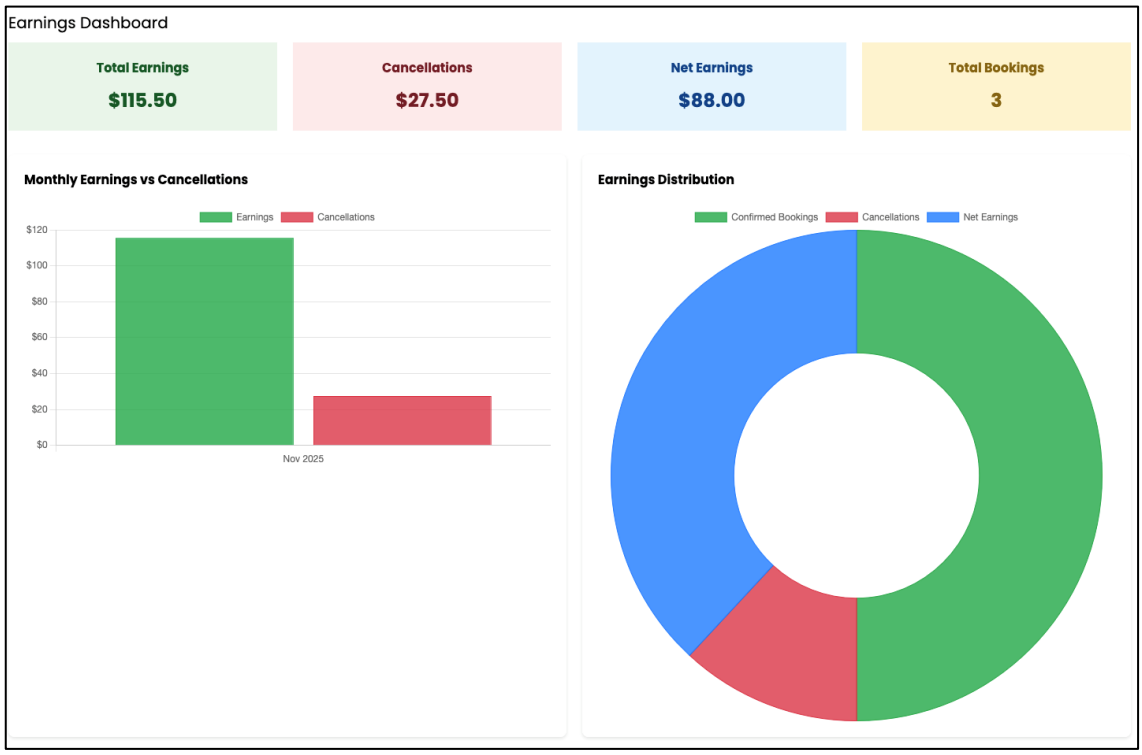


Figure 7.21: Earning Management

User Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Booking section to view all booking

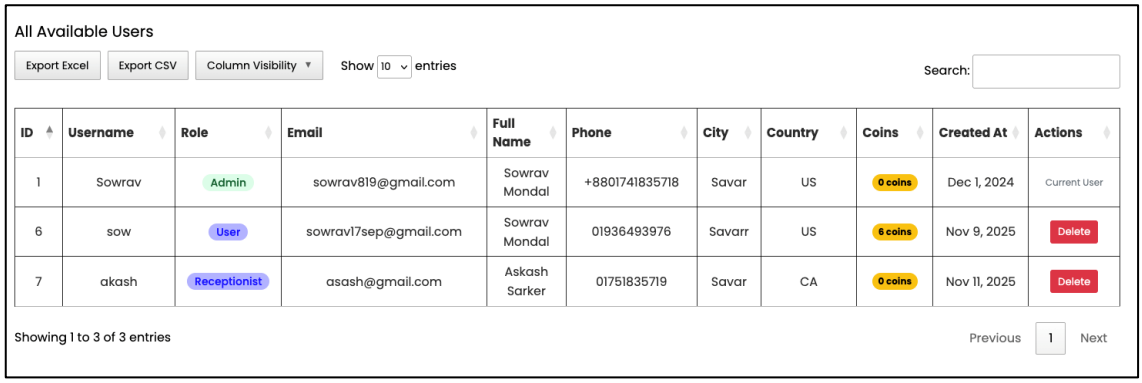


Figure 7.22: User Management

Contact Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Contact Management section to view all contact information
3. To Delete, click Delete button and confirm the delete on alert
4. To Send email, Click Send Email button

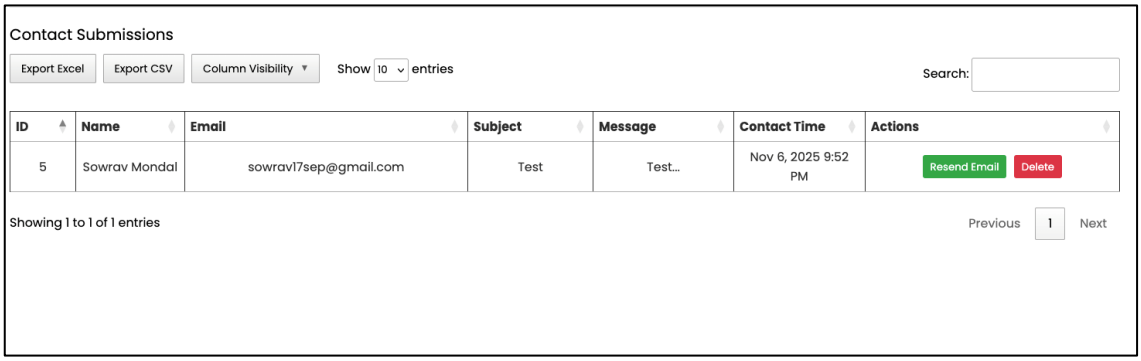


Figure 7.23: Contact Management

Chatbot System (Customer):

1. Visit website's any page
2. Click the chat icon which is on left-bottom side
3. Chatbot open with a greeting message
4. Select option to get message
5. Type custom message and press enter to send it

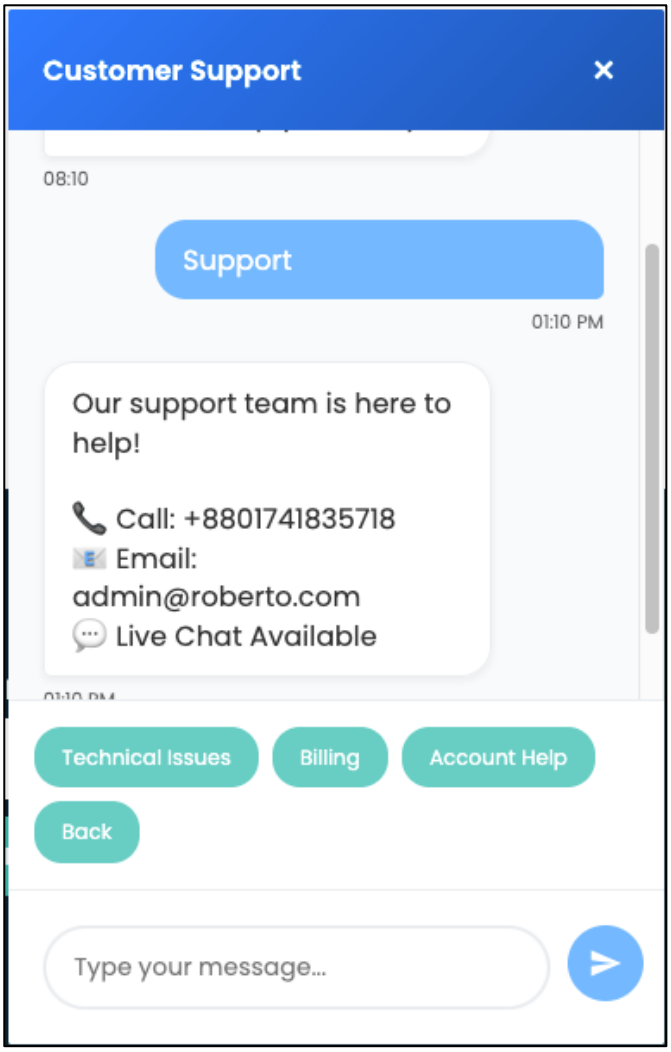


Figure 7.24: Chatbot System

Add Blog (Admin):

1. Login as Admin and redirect to admin dashboard
2. Click Add Blog from dashboard
3. Enter all information about blog
4. Click Add Blog button
5. Blog will be added and redirected to Blog management section

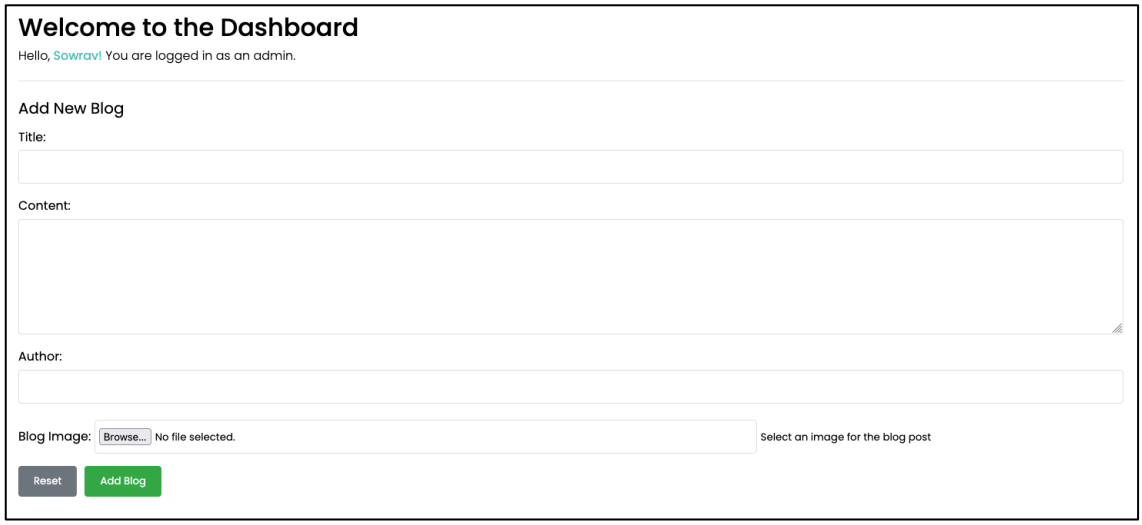


Figure 7.25: Add Blog

Blog Management (Admin):

1. Login as Admin and redirected to the admin dashboard
2. Click Blog Management section to view all blog
3. To delete a blog, click Delete and blog will be deleted immediately
4. To edit a blog, click Edit and a popup will appear.
5. Then edit the existing input and Click Update button
6. Blog will be updated and appear in list

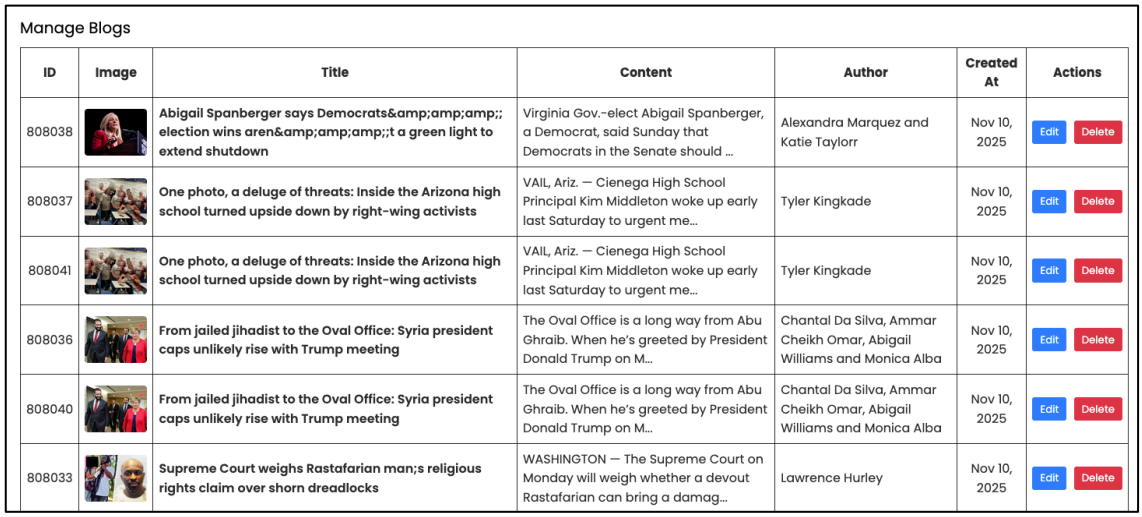


Figure 7.26: Blog Management

Logout (Admin, Receptionist, Customer)

1. For Customer, click Logout option from top navigation menu
2. For Admin, Click Logout from dashboard option which is the last option in dashboard
3. For Receptionist, Click Logout from dashboard option which is the last option in dashboard

5.3 Summary

The above user manual is a complete guild for using the **Roberto Hotel and Resort System.** Admin, Receptionist, and Customer can perform their corresponding task by following the user manual in this chapter. The system is designed user-friendly for all types of users to navigate easily on the system.

Chapter 6 Project Summary

6.1 Introduction

This chapter provides an overall summary of Roberto Hotel & Resort system. It highlights the project objective, features developed, challenges faced and overall result. The chapter all talks about project limitation, scope and future work to improve the usability and efficiency.

6.2 Project Limitation

All though Roberto Hotel & Resort System successfully delivered the core functionality such as booking, online payment, loyalty reward, discount, but several limitations appeared during the development of the system.

6.2.1 Time Constrains

Based a fixed academic timeline the project was developed, which limited the curtains features.

Because of a limited time frame:

- Some features are compromised
- Advance feature like multilanguage support is not implemented
- Comprehensive security and a big scale testing cannot be performed

6.2.2 Budget Constrains

The project is developed using free tools and deployed in low-cost web hosting due to budget limitation:

- CDN optimization and could hosting are not used, instead of those a low-cost shared hosting is used, which may affect performance of the system
- No paid UI or framework are used in this project
- Only test payment api are used because real API service charges cost

6.2.3 Technological Constrains

- No mobile application is developed, the focus on web-based service
- Chatbot are running on rules, because AI Driven chatbot requires a technical complexity.
- The system does not support a hotel chain system, due to development time framework.

6.2.4 Requirement Constrains

Some features are not implemented in this version:

Multi-language support was planned
An audit logs system for tracking activities left implemented
SMS notification was planned but not implemented

6.3 Scope

The Roberto Hotel & Resort system covers the following functionalities and modules:

Secure user registration and login
Customer profile management and password change
Advance cart and checkout system
Online payment using Stripe
Room booking module
Loyalty reward coin and discount
Booking cancellation and refund system
Admin dashboard for managing booking, earning, room, user
Receptionist dashboard for managing checking and checkout process
Basic reporting system to print or export data
Chatbot for assistance

The system focuses on small and medium-sized hotels and resorts who usually manage their hotel system in the traditional process. The project comes with a digital solution of affordability and usability to automate the hotel system with accuracy and good customer satisfaction.

6.4 Future Work

To improve performance, usability and more real-world adaptation, I planned some future work:

Mobile application development for Android and ISO for easy customer access
Multi-language system for better usability and customer satisfaction
Integrated with review system like Google Reviews, Trustpilot
Hotel chain system to include multiple branches support under on management
Sharing an unmetered amount of reward coin among the users

6.5 Conclusion

The Roberto Hotel & Resort system will successfully modernize the traditional hotel management process with a modern solution. Through smart room recommendation, online payment method, easy cancellation and refund, loyalty reward and discount system, centralized administrator management will attract both hotels and customers.

Even though the system has some limitations, it is scalable for a large scale of future development. The project demonstrates that, how a software can reduce manual workload and increase productivity in small and medium sized hotel. Overall, the project achieves the goal and provide a strong foundation for further development.

REFERENCES

1. Sommerville, Ian. *Software Engineering*. Pearson Education. https://www.pearson.com

2. Pressman, Roger S. *Software Engineering: A Practitioner's Approach*. https://highered.mheducation.com

3. W3Schools. "PHP, MySQL, HTML, CSS, JavaScript Tutorials." https://www.w3schools.com

4. Stripe. "Stripe Payment API Documentation." https://stripe.com/docs

5. Booking.com. "Hotel Booking System Features." https://www.booking.com

6. OYO Rooms. "Hotel Management and Automation Features." https://www.oyorooms.com

7. Mozilla Developer Network (MDN). "Web Development Tutorials." https://developer.mozilla.org

8. MySQL Official Documentation. https://dev.mysql.com/doc

9. UML Documentation. "Unified Modeling Language Guide." https://www.uml.org

10. GeeksforGeeks. "PHP and SQL Tutorials." https://www.geeksforgeeks.org

APPENDICES

Appendix A: Database Structure:

Table 7.1: booking

Field	Type	Description
booking_id	INT (PK)	Booking ID
user_id	INT (FK)	User
room_id	INT (FK)	Room
user_name	VARCHAR(100)	Customer name
user_email	VARCHAR(100)	Email
status	VARCHAR(50)	booking status
created_at	TIMESTAMP	Created date
checkin_date	DATE	Check-in
checkout_date	DATE	Check-out
phone_number	VARCHAR(20)	Contact
address_line_1	VARCHAR(255)	Address
address_line_2	VARCHAR(255)	Address
city	VARCHAR(100)	City
state	VARCHAR(100)	State
country	VARCHAR(100)	Country
pincode	VARCHAR(20)	Postal code
total_amount	DECIMAL(10,2)	Total payment
addons	TEXT	Extra services
adults	INT	No. of adults
children	INT	No. of children
breakfast	TINYINT	Breakfast addon
airport_pickup	TINYINT	Airport pickup
spa	TINYINT	Spa addon
gym_access	TINYINT	Gym addon
addons_total	DECIMAL(10,2)	Addons total
coins_used	INT	Loyalty coins used
coins_earned	INT	Loyalty coins earned

Table 7.2: User

Field	Type	Description
id	INT (PK)	User ID
username	VARCHAR(255)	Username
password	VARCHAR(255)	Password
role	ENUM(admin, user, receptionist)	Role
created_at	DATE	Created date
updated_at	TIMESTAMP	Updated
email	VARCHAR(255)	Email
phone_number	VARCHAR(20)	Phone
full_name	VARCHAR(255)	Full name
address_line_1	VARCHAR(255)	Address
address_line_2	VARCHAR(255)	Address
city	VARCHAR(100)	City
state	VARCHAR(100)	State
country	VARCHAR(100)	Country
pincode	VARCHAR(20)	Postal code
coins	INT	Loyalty coins

Table 7.3: rooms

Field	Type	Description
room_id	INT (PK)	Room ID
room_number	VARCHAR(50)	Room number
room_type	VARCHAR(50)	Type
price	DECIMAL(10,2)	Price
room_image	VARCHAR(255)	Main image
description	TEXT	Description
amenities	TEXT	Amenities
availability	TINYINT	Available? (0/1)
created_at	TIMESTAMP	Created date

Table 7.4: room_images

Field	Type	Description
id	INT (PK)	Image ID
room_id	INT (FK)	Room
image_path	VARCHAR(255)	File path
uploaded_at	TIMESTAMP	Uploaded date

Table 7.5: user_preferences

Field	Type	Description
id	INT (PK)	Preference ID
user_id	INT (FK)	User
preferred_room_type	VARCHAR(50)	Room type preference
max_price	DECIMAL(10,2)	Budget
group_size	INT	Group size
created_at	TIMESTAMP	Created
updated_at	TIMESTAMP	Updated

Table 7.6: payments

Field	Type	Description
id	INT (PK)	Payment ID
booking_id	INT (FK)	Booking
payment_intent_id	VARCHAR(255)	Stripe/Razor ID
amount	DECIMAL(10,2)	Payment amount
payment_method	VARCHAR(50)	Method
payment_status	VARCHAR(50)	Status
created_at	TIMESTAMP	Created date

Table 7.7: earnings

Field	Type	Description
id	INT (PK)	Earnings ID
booking_id	INT (FK)	Booking reference
user_id	INT (FK)	User
user_name	VARCHAR(255)	Customer name
amount	DECIMAL(10,2)	Amount
type	ENUM(booking, cancellation, refund)	Entry type
status	ENUM(completed, pending, cancelled)	Status
created_at	TIMESTAMP	Date

Table 7.8: refund_requests

Field	Type	Description
id	INT (PK)	Refund ID
booking_id	INT (FK)	Related booking
user_id	INT (FK)	User
refund_amount	DECIMAL(10,2)	Refund amount
refund_reason	TEXT	Reason
payment_method	VARCHAR(50)	Payment type
bank_name	VARCHAR(100)	Bank name
account_number	VARCHAR(100)	Account number
routing_number	VARCHAR(100)	Routing number
status	ENUM(pending, approved, rejected, processed)	Status
admin_notes	TEXT	Notes
created_at	TIMESTAMP	Created
updated_at	TIMESTAMP	Updated

Table 7.9: coin_transactions

Field	Type	Description
id	INT (PK)	Transaction ID
user_id	INT (FK)	User
booking_id	INT (FK)	Booking
transaction_type	ENUM(earned, used, bonus)	Type
coins	INT	Number of coins
description	VARCHAR(255)	Details
created_at	TIMESTAMP	Created date

Table 7.10: chat_messages

Field	Type	Description
id	INT (PK)	Message ID
session_id	VARCHAR(100)	Chat session
message_type	ENUM(user, bot)	Sender
message_text	TEXT	Message body
timestamp	TIMESTAMP	Sent time

Table 7.11: chat_sessions

Field	Type	Description
id	INT (PK)	Unique chat session ID
session_id	VARCHAR(100)	Unique session identifier
created_at	TIMESTAMP	When the session was created
last_activity	TIMESTAMP	Last time user/bot interacted

Table 7.12: contact_submissions

Field	Type	Description
id	INT (PK)	Submission ID
name	VARCHAR(255)	Sender name
email	VARCHAR(255)	Sender email
subject	VARCHAR(255)	Subject
message	TEXT	Message body
created_at	TIMESTAMP	Created time
email_resent	TINYINT	Resent flag
email_resent_at	TIMESTAMP	Resent date

221-35-819

ORIGINALITY REPORT

18%	12%	3%	16%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Northampton Student Paper	3%
2	Submitted to Daffodil International University Student Paper	2%
3	Submitted to Nanyang Technological University Student Paper	1%
4	Submitted to University of Gloucestershire Student Paper	1%
5	Submitted to RDI Distance Learning Student Paper	1%
6	dspace.daffodilvarsity.edu.bd:8080 Internet Source	1%
7	Submitted to NCC Education Student Paper	1%
8	Submitted to The International University of Management Student Paper	1%
9	Submitted to Midlands State University Student Paper	1%
10	Submitted to Chester College of Higher Education	<1%