Software Requirements Specification

Hotel Reservation System

Team Information:

- ***Members:***
  - *Harshit Ghosh(Roll No.: 22052982)*
  - *Jyotika Jayani (Roll No.: 22052987)*
  - *Shreya Agarwal (Roll No.: 22051024)*
  - *Snigdha Pandey (Roll No.: 2205776)*
  - *Siddharth Panda (Roll No.: 22051026)*

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed Software Requirements Specification (SRS) for the Hotel Reservation System. This system is designed to manage hotel reservations, room availability, payments, and customer interactions. The SRS outlines the functionalities, interfaces, and performance criteria of the system.

1.2 Scope

The Hotel Reservation System is intended to be a comprehensive solution for managing hotel operations, including room management, customer reservations, payment processing, and reporting. The system will be used by administrators, hotel managers, and customers to facilitate the entire booking process from room search to payment.

1.3 Definitions, Acronyms, and Abbreviations

- **SRS**: Software Requirements Specification
- **UML**: Unified Modeling Language
- **Admin**: Administrator of the hotel system
- **Manager**: Hotel Manager responsible for daily operations
- **Customer**: End user who makes reservations

1.4 References

- UML Class Diagram for the Hotel Reservation System
- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

1.5 Overview

This SRS document is organized into the following sections:

- **Overall Description**: Describes the system's general functionality and user interactions.
- **System Features**: Details the specific functionalities of the system.
- **External Interface Requirements**: Describes the interfaces with which the system will interact.
- **System Attributes**: Discusses the non-functional requirements, such as performance and security.

2. Overall Description

2.1 Product Perspective

The Hotel Reservation System is an independent software solution designed to manage the operations of a hotel. It interfaces with users via a web-based application, providing functionalities such as hotel search, reservation creation, room management, payment processing, and reporting.

The system components, as shown in the UML class diagram, include classes for Hotel, Room, Customer, Reservation, Payment, Admin, Manager, SearchCriteria, DateRange, and Report.

2.2 Product Functions

The primary functions of the system include:

- **Hotel Management**: Admin and Managers can add or remove rooms, view all reservations, and manage the overall hotel operations.
- **Reservation Handling**: Customers can search for available hotels, make reservations, and view their booking history.
- **Payment Processing**: The system handles payment transactions, including initiation and processing.

- **Reporting**: Managers can generate reports based on various criteria to analyze hotel performance.

2.3 User Classes and Characteristics

- **Admin**: Has full control over the system, including user management, payment processing, and hotel management.
- **Manager**: Manages hotel operations, handles reservations, and generates reports.
- **Customer**: Can search for hotels, make reservations, and manage their bookings.

2.4 Operating Environment

The Hotel Reservation System will operate in a web-based environment, accessible via standard web browser. The system will be hosted on a cloud-based server with a relational database backend.

2.5 Design and Implementation Constraints

- The system must be scalable to handle multiple hotels and a large number of customers simultaneously.
- It must comply with data protection regulations, including GDPR for customer data.

2.6 Assumptions and Dependencies

- The system assumes a reliable internet connection for accessing the web application.
- Integration with third-party payment gateways will be required for processing payments.

3. System Features

3.1 Hotel Management

3.1.1 Description

This feature allows Admins and Managers to manage hotel information, including adding or removing rooms, viewing reservations, and updating hotel details.

3.1.2 Functional Requirements

- **FR1**: The system shall allow Admins to add new rooms to the hotel.
- **FR2**: The system shall allow Admins to remove existing rooms from the hotel.
- **FR3**: The system shall allow Managers to view all reservations associated with the hotel.

- **FR4**: The system shall allow Managers to update hotel information, including Name and Location.

3.2 Room Management

3.2.1 Description

This feature provides functionalities for checking room availability, managing room details, and updating room status.

3.2.2 Functional Requirements

- **FR1**: The system shall allow Managers to check the availability of rooms for specific dates.
- **FR2**: The system shall allow Admins to update room details, including Room Number and Price Per Night.

3.3 Customer Reservation

3.3.1 Description

Customers can search for hotels based on specific criteria, create reservations, and view their booking history.

3.3.2 Functional Requirements

- **FR1**: The system shall allow Customers to search for hotels based on location, check-in date, check-out date, and number of guests.
- **FR2**: The system shall allow Customers to create a reservation for an available room.
- **FR3**: The system shall allow Customers to view their past reservations and booking history.

3.4 Payment Processing

3.4.1 Description

This feature handles the initiation and processing of payments related to hotel reservations.

3.4.2 Functional Requirements

- **FR1**: The system shall allow Customers to initiate payment after reservation creation.
- **FR2**: The system shall process payments and update the payment status in the Reservation.

3.5 Reporting

3.5.1 Description

Managers can generate reports based on various criteria, such as reservation status, room availability, and financial transactions.

3.5.2 Functional Requirements

- **FR1**: The system shall allow Managers to generate reports detailing reservations within a specific date range.
- **FR2**: The system shall allow Managers to generate financial reports based on payments received.

4. External Interface Requirements

4.1 User Interfaces

- **Admin Interface**: A web-based dashboard allowing full control over hotel operations, user management, and payments.
- **Manager Interface**: A web-based interface for handling daily hotel operations, generating reports, and managing reservations.
- **Customer Interface**: A web-based interface for searching hotels, making reservations, and managing personal bookings.

4.2 Hardware Interfaces

The system does not require any specific hardware interfaces. It will be hosted on cloud servers and accessed via standard web browsers.

4.3 Software Interfaces

- **Payment Gateway**: The system will integrate with third-party payment gateways (e.g., Stripe, PayPal) for processing transactions.
- **Database**: The system will use a relational database (e.g., MySQL, PostgreSQL,MongoDB) for storing all data related to hotels, reservations, and payments.

4.4 Communication Interfaces

- **HTTP/HTTPS**: The system will communicate with clients and external services using HTTP/HTTPS protocols.
- **RESTful APIs**: The system will expose and consume RESTful APIs for interactions with external systems, such as payment gateways.

5. System Attributes

5.1 Performance

- The system shall be able to handle up to 10(as of now) concurrent users without performance degradation.
- The response time for searching hotels shall be less than 2 seconds under normal load conditions.

5.2 Security

- The system shall require user authentication for Admins, Managers, and Customers.
- All sensitive data, including payment information, shall be encrypted in transit and at rest.
- The system shall log all user activities for auditing purposes.

5.3 Maintainability

- The system shall be designed with modular components to facilitate easy updates and maintenance.
- The codebase shall follow industry-standard practices, including the use of version control and automated testing.

5.4 Portability

- The system shall be deployable on any cloud platform supporting standard web technologies (e.g., AWS, Azure, Google Cloud).
- The system shall be compatible with all major web browsers.

6. Other Requirements

6.1 Legal and Regulatory Requirements

- The system must comply with GDPR regulations for handling customer data.
- Payment processing must adhere to PCI DSS standards.

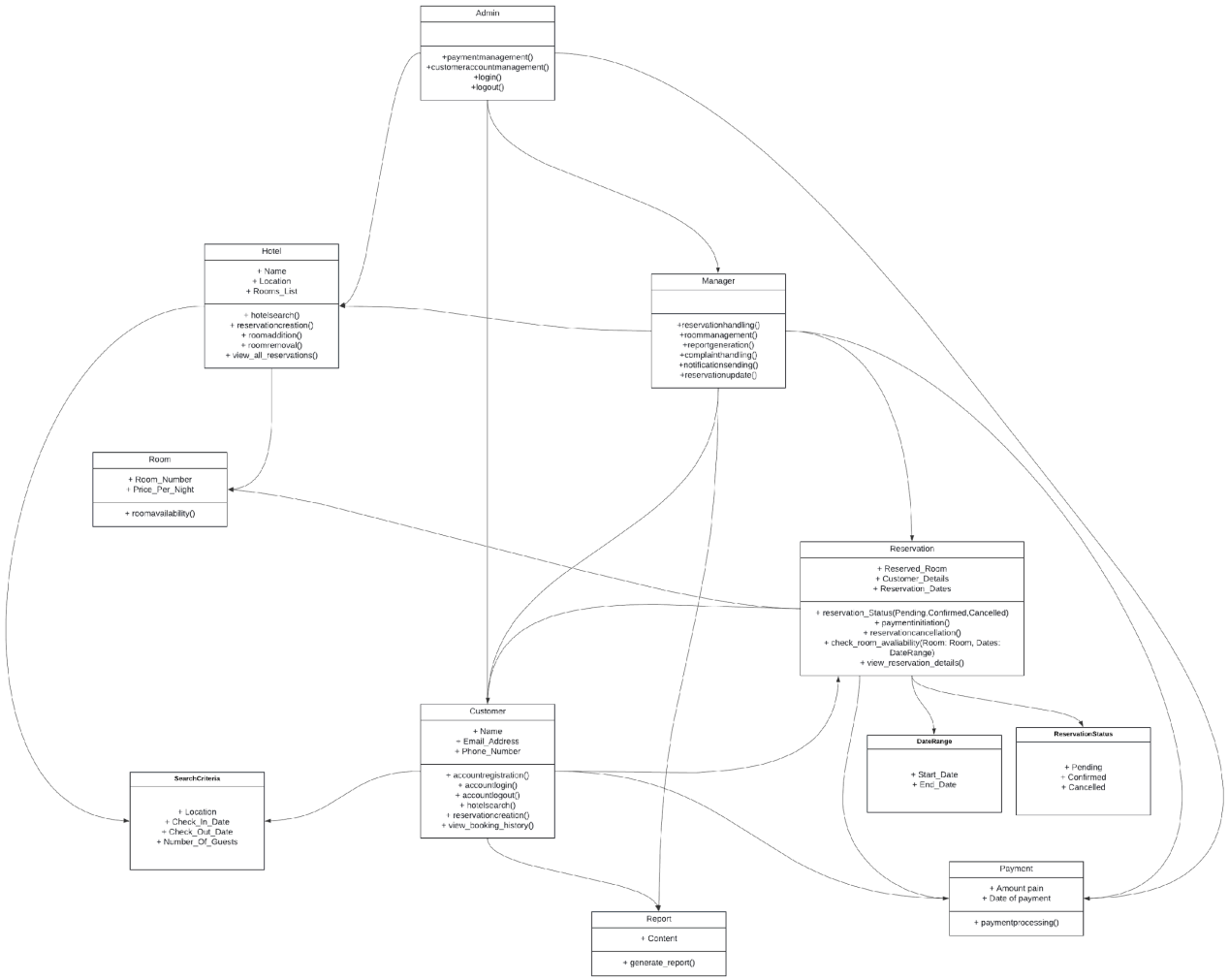
6.2 Safety Requirements

- There are no specific safety requirements for this system.

7. Appendices

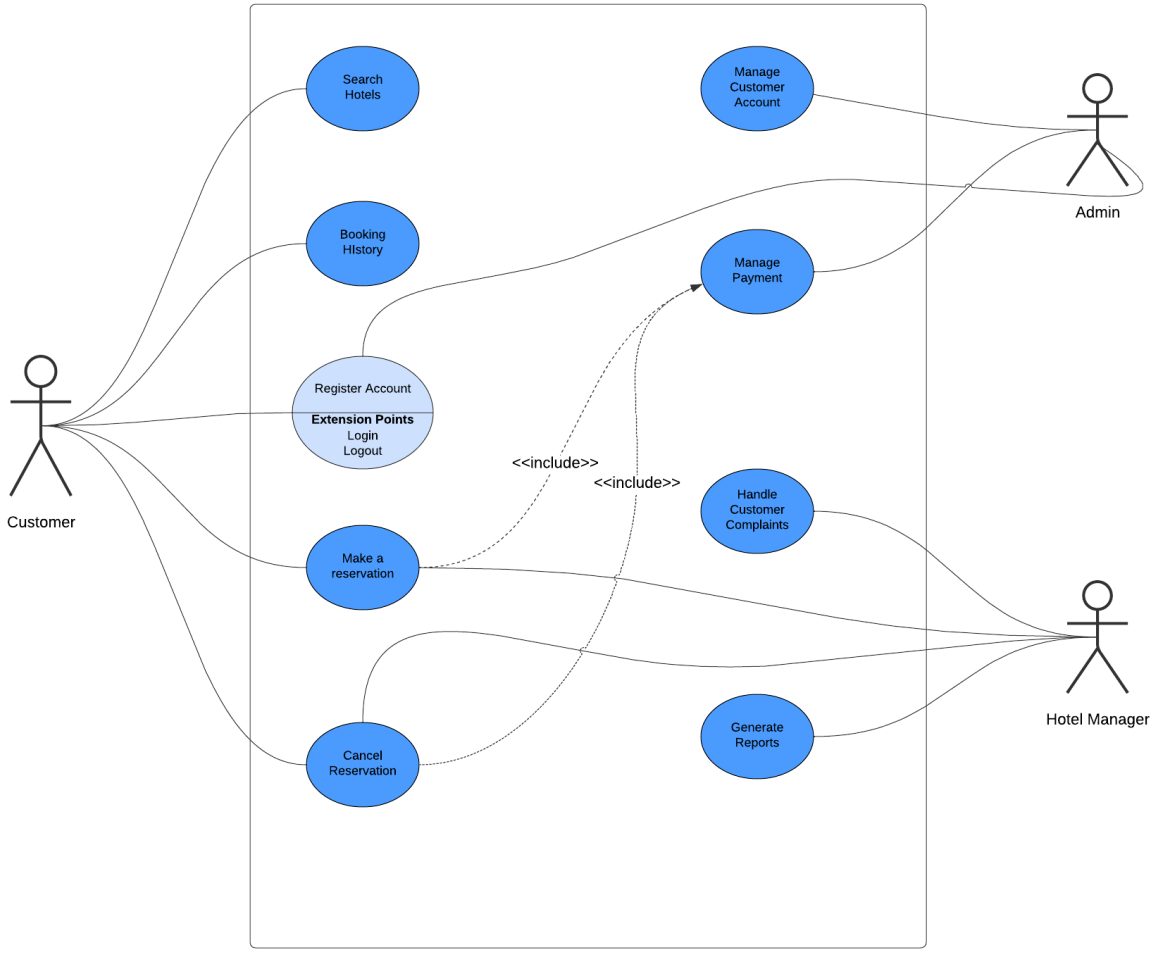
7.1 Class Diagram

The following class diagram from the UML model illustrates the relationships and attributes of the system's components. This diagram provides a structural view of the Hotel Reservation System, showing the classes involved, their properties, methods, and how they interact with one another.



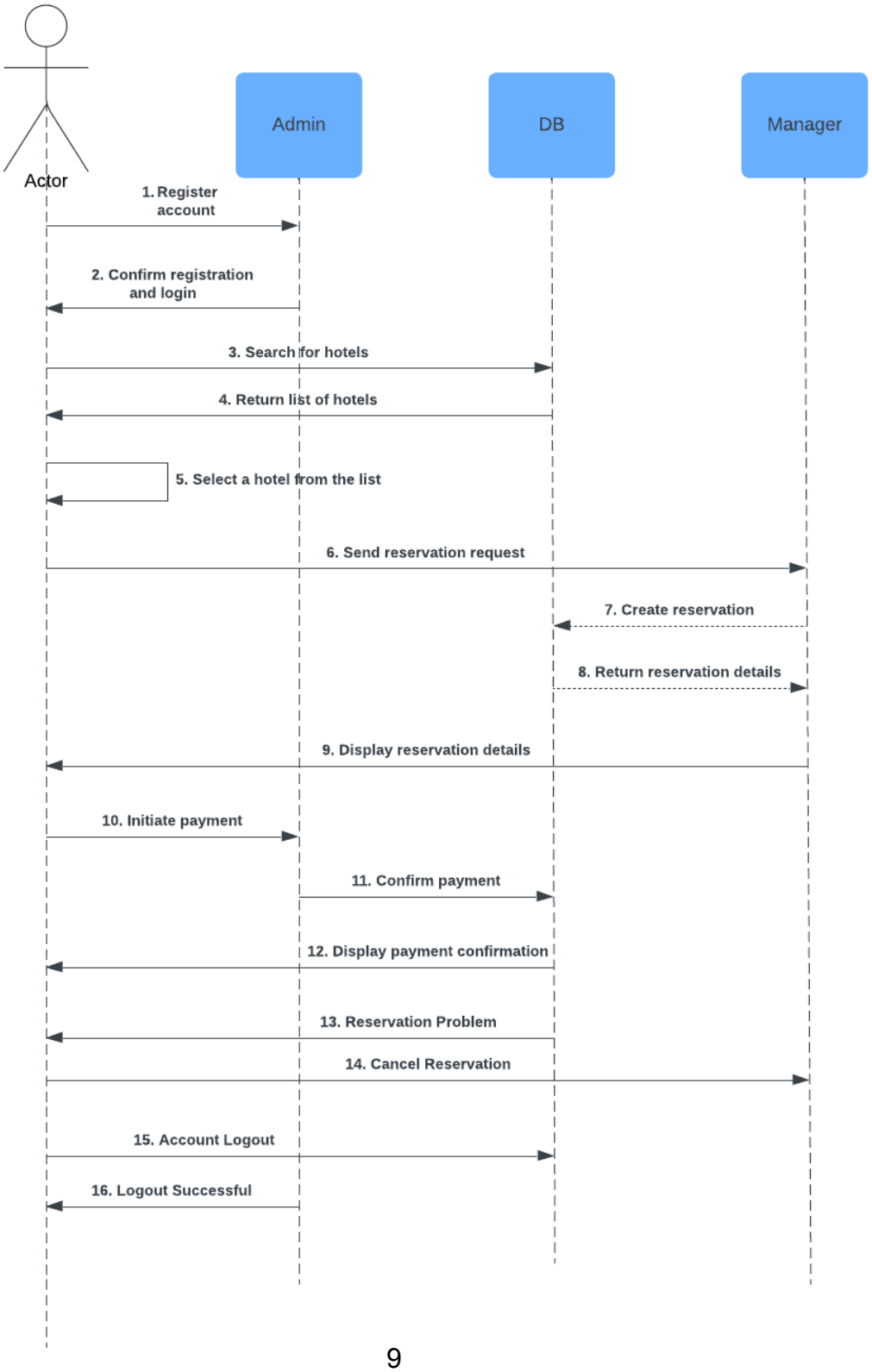
7.2 Use Case Diagram

The use case diagram below illustrates the interactions between users (actors) and the system. It identifies the primary use cases, such as searching for hotels, making reservations, and managing accounts, and shows how different actors, like customers, managers, and administrators, interact with these use cases.



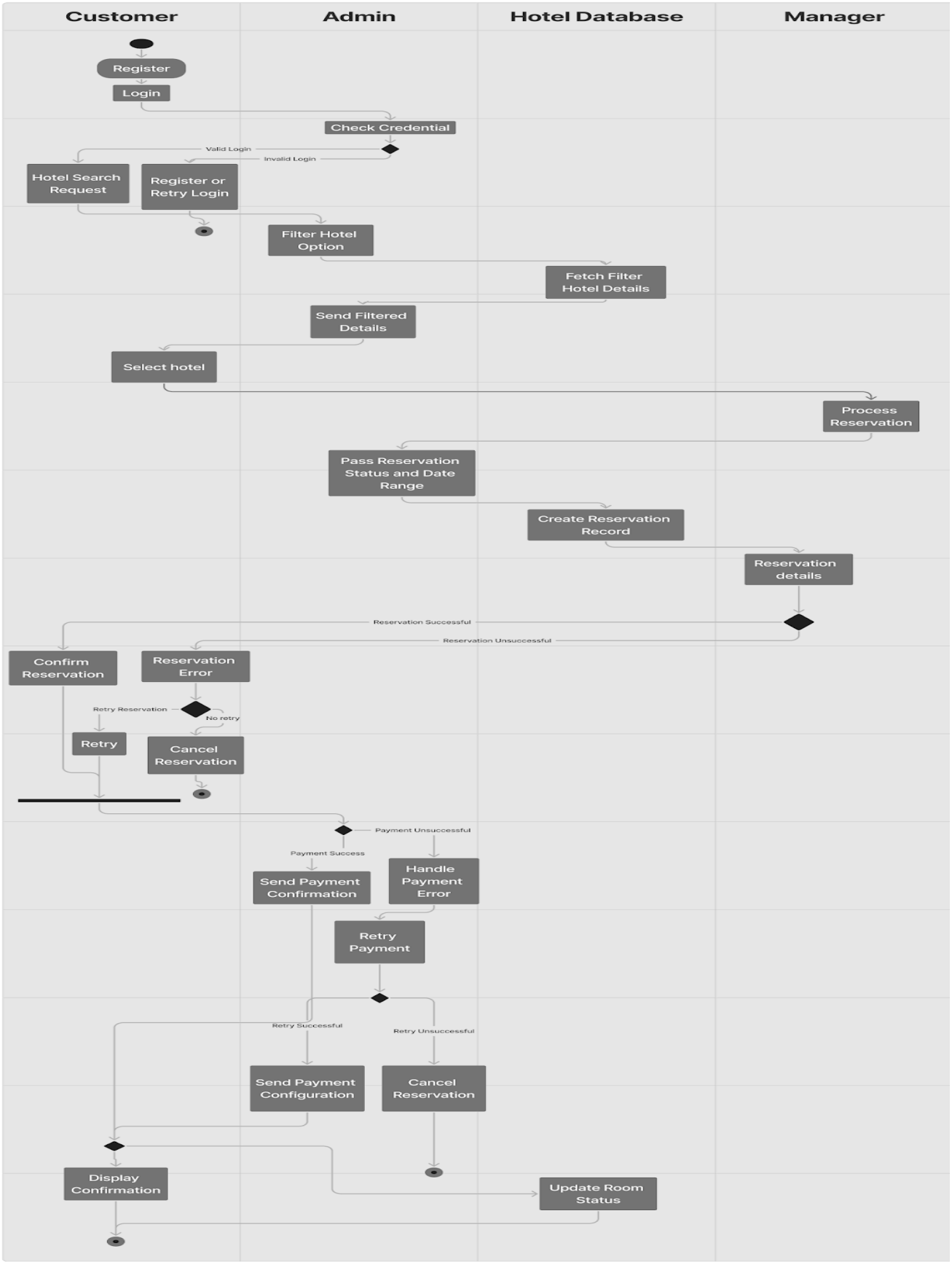
7.3 Sequence Diagram

The sequence diagram presents a dynamic view of the system by showing how objects interact in a particular scenario of a use case. It emphasizes the order of messages exchanged between objects during the execution of the process, such as during reservation creation or payment processing.



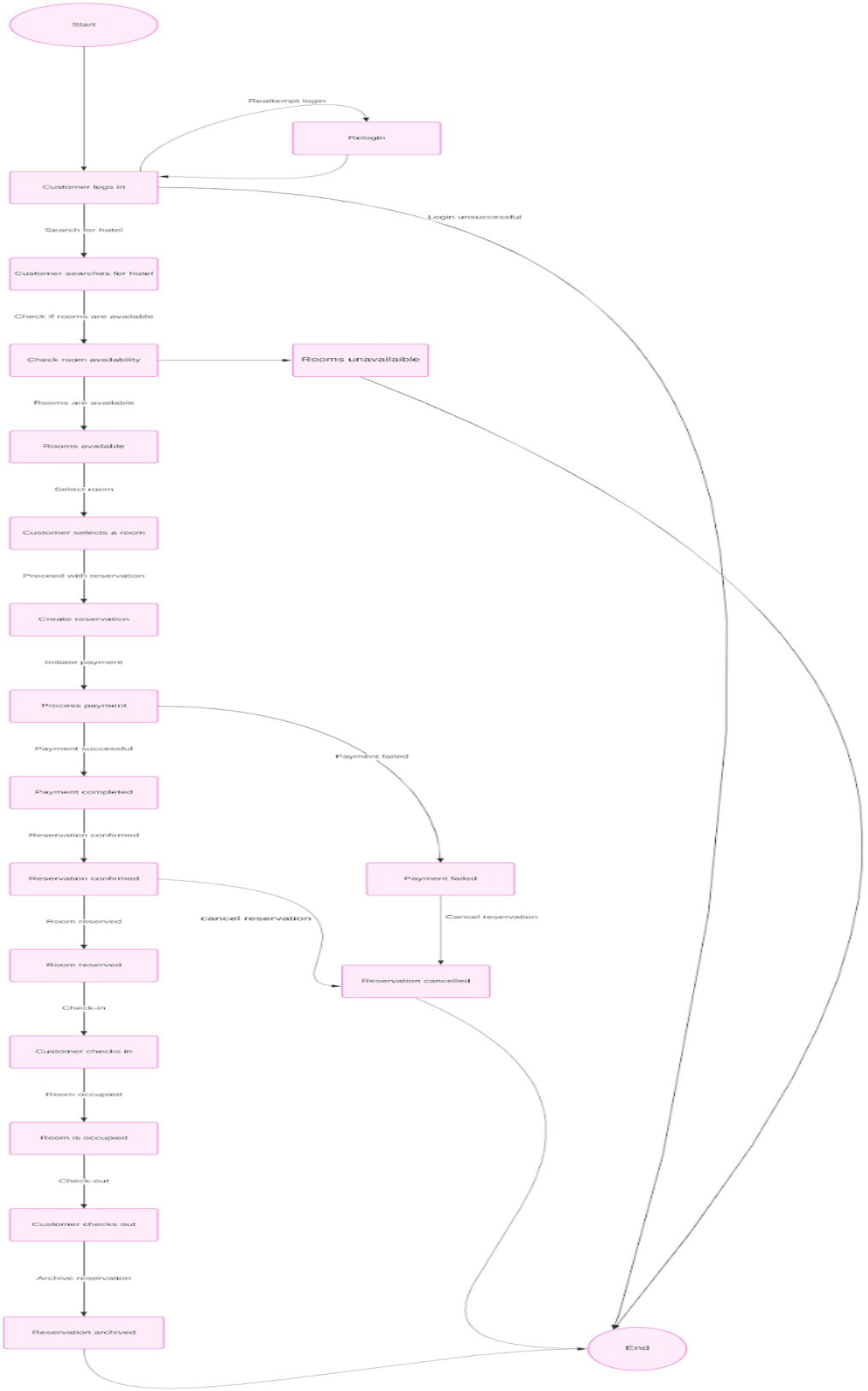
7.4 Activity Diagram

This diagram outlines the sequence of activities involved in the hotel reservation process. It starts from user login, searches for available rooms, confirms booking, processes payment, and ends with a reservation confirmation or cancellation.



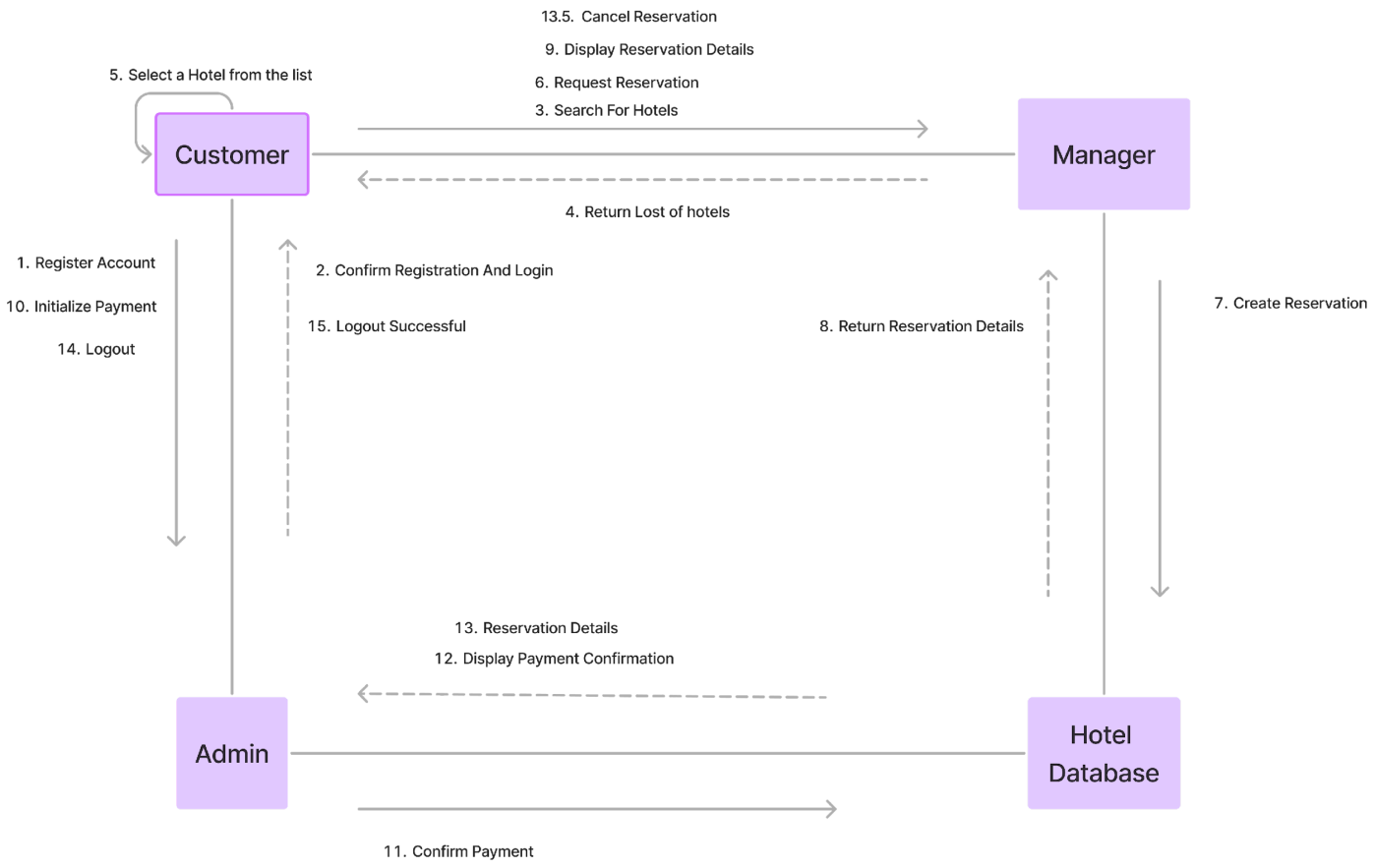
7.5 Statechart Diagram

This diagram shows the different states of a reservation, from Initiated to Confirmed, Pending Payment, Cancelled, or Completed. Each transition is triggered by an event like user actions or system processes.



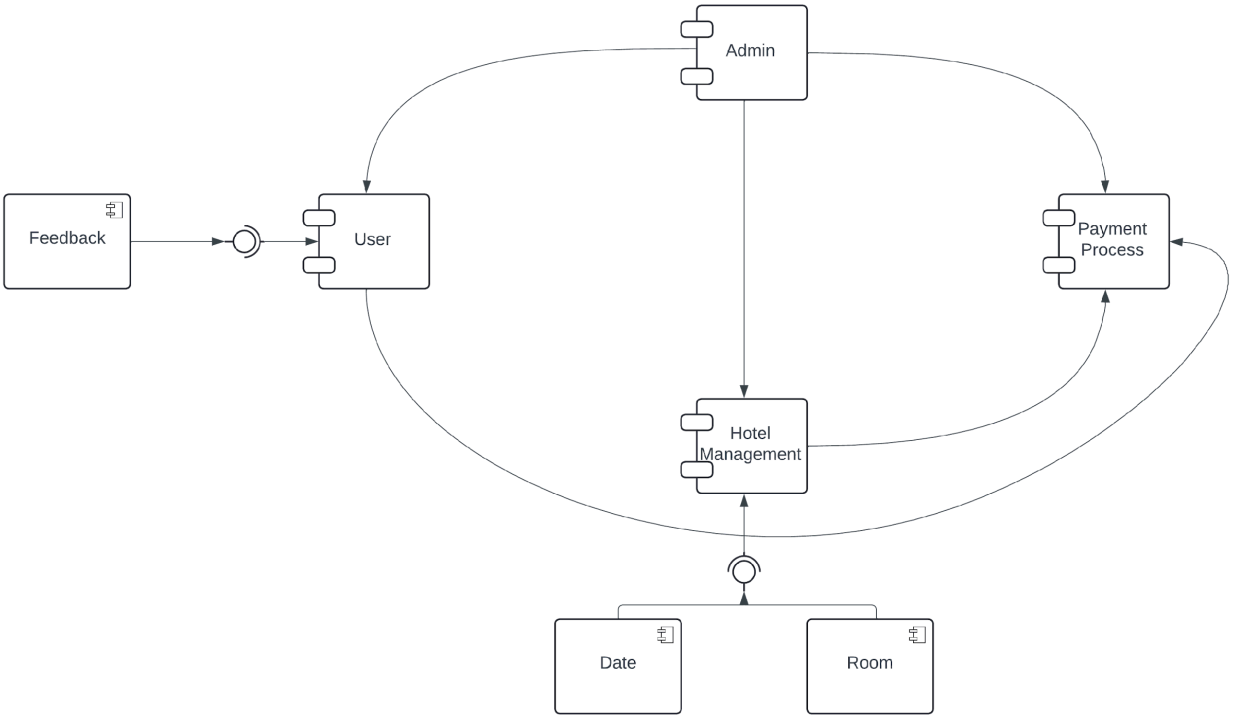
7.6 Collaboration Diagram

This diagram illustrates the interactions between key objects such as User, Admin, Database, and Manager during a hotel reservation. It shows the messages exchanged and the order of interactions to complete the reservation workflow.



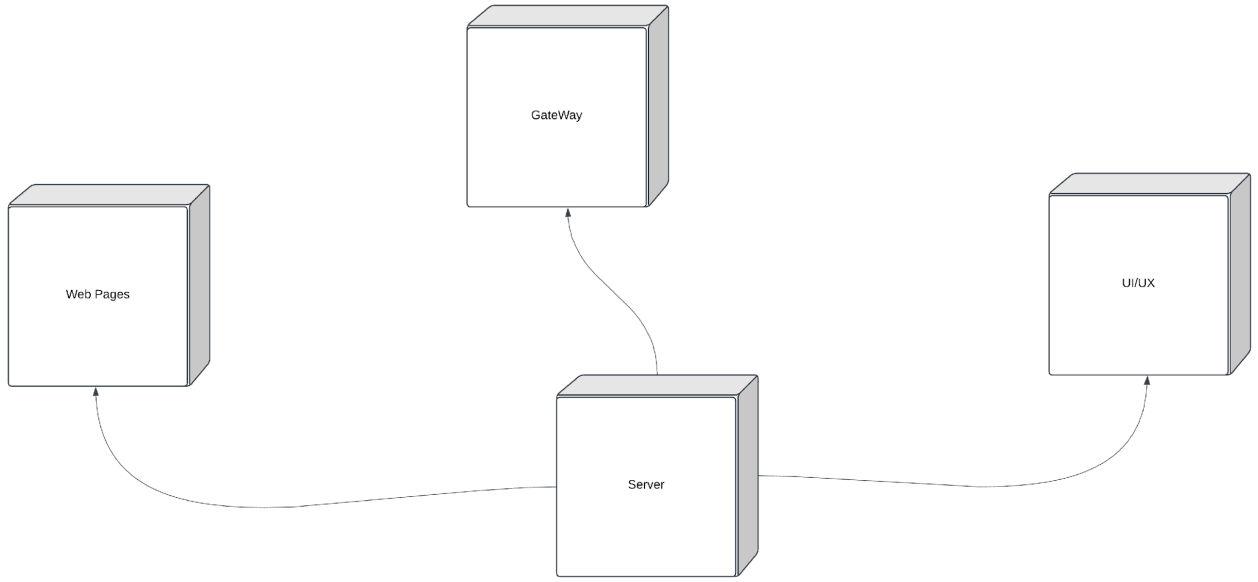
7.7 Component Diagram

This diagram represents the various components of the Hotel Reservation System, including User Interface, Business Logic, Database, and Payment Gateway. It details how these components connect and depend on each other.



7.7 Deployment Diagram

This diagram illustrates the physical architecture of the system, with components deployed on servers, databases hosted on cloud infrastructure, and client interactions through web browsers. It details the network connections and data flow.



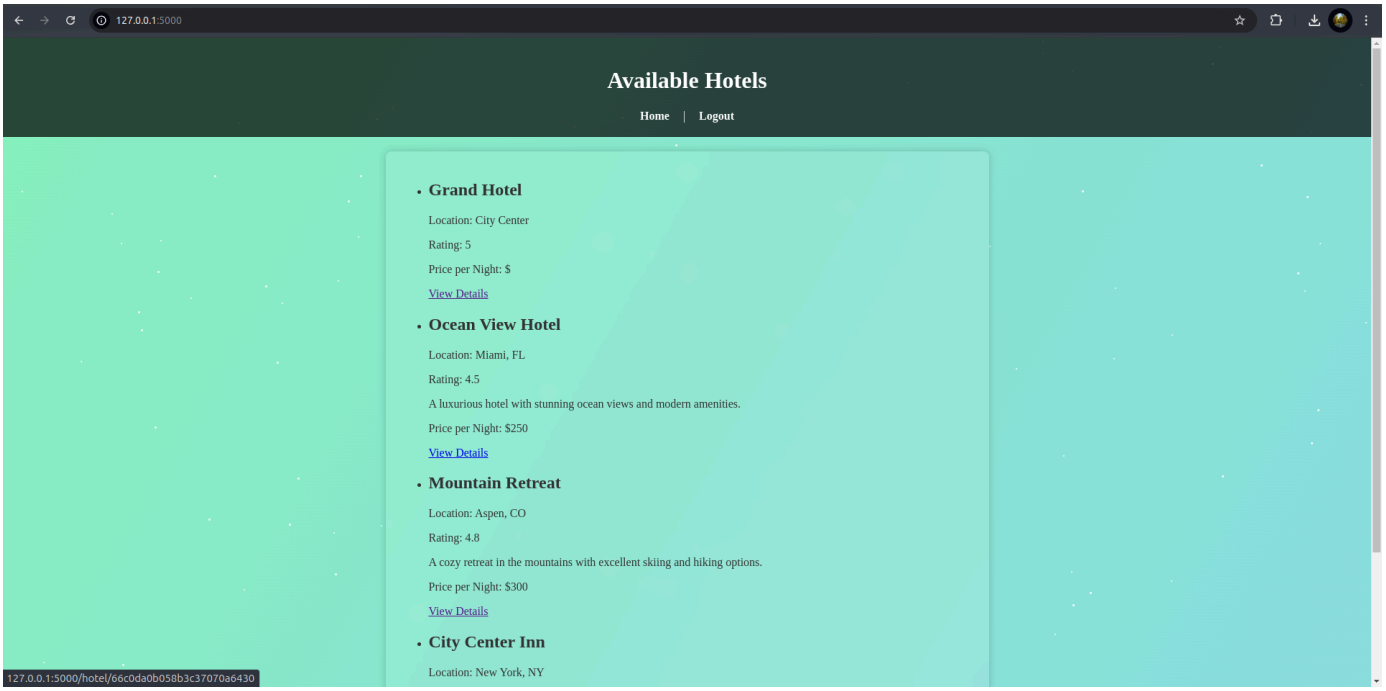
8. UI

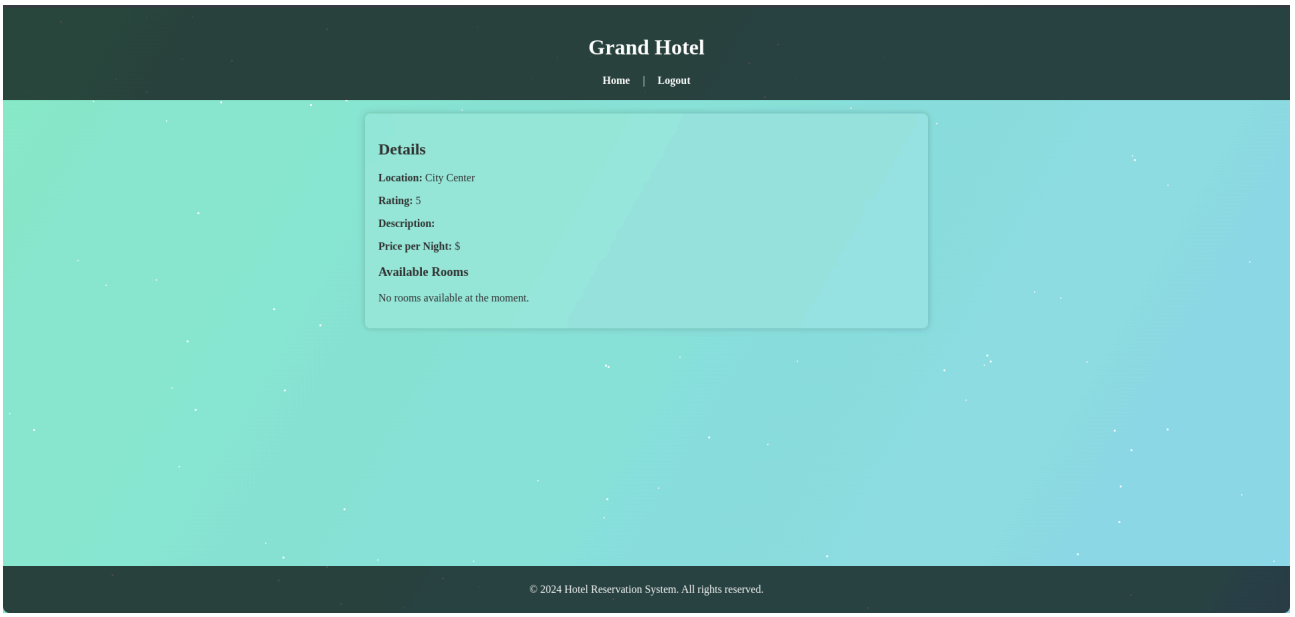
Description of the UI

The user interface of the hotel reservation system is designed to be clean, intuitive, and visually appealing. The layout emphasizes ease of navigation and efficient booking processes. The following are the main components of the UI:

1. **Home Page:**
    - Features a welcoming gradient background animation for a modern appearance.
    - A navigation bar at the top provides links to the Hotel List, Login, and other key sections.
2. **Hotel List Page:**
    - Displays hotels in a grid or list view, each with an image, name, location, price, and rating.
    - Users can click "View Details" to learn more about a specific hotel.
3. **Hotel Details Page:**
    - Contains a detailed view of the selected hotel, showcasing multiple images, a description, amenities, and room options.
    - Users can initiate the booking process or return to the hotel list.

UI Screenshot





Code Snippets

Below are code snippets that provide an overview of how the application works, including the setup of routes, rendering templates, and managing data.

app.py

This file is the entry point of the Flask application. It defines and configures the app, connects to the MongoDB database, and sets up error handling.

```
from flask import Flask, render_template
from pymongo import MongoClient
from config import Config
from hotel_reservation_system.routes import bp
from flask_mail import Mail
from dotenv import load_dotenv
import os

def create_app():
    load_dotenv()  # Load environment variables from .env file
    app = Flask(__name__)
    app.secret_key = os.getenv('SECRET_KEY', '59a4ecd754c57833ffe61c94a568a390')  # Set a
unique and secret key

    # Register the blueprint
    app.register_blueprint(bp)
    app.config.from_object(Config)

    # Initialize MongoDB client and assign to the app context
    client = MongoClient(app.config['MONGO_URI'])
    app.db = client[app.config['MONGO_DB']]

    # Setup Flask-Mail
    mail = Mail(app)

    # Error handling pages
    @app.errorhandler(404)
    def page_not_found(e):
        return render_template('404.html'), 404

    @app.errorhandler(500)
    def internal_server_error(e):
        return render_template('500.html'), 500

    return app

if __name__ == "__main__":
    app = create_app()
    app.run(debug=True)
```

- **Key Components:**
  - **`create_app()` Function**: Initializes and configures the Flask app, including loading environment variables, setting the secret key, and connecting to MongoDB.
  - **Blueprint Registration**: Registers the routes defined in `hotel_reservation_system.routes` using a blueprint.
  - **MongoDB Connection**: Establishes a connection to the MongoDB database using `pymongo.MongoClient`.
  - **Flask-Mail Setup**: Configures email services for functionalities like user notifications or password reset.
  - **Error Handlers**: Defines custom 404 and 500 error pages using `render_template()` to display user-friendly error messages.

index.html

This is the homepage template that extends `base.html` and displays a list of available hotels.

```
{% extends 'base.html' %}

{% block title %}Hotel Reservation System{% endblock %}

{% block header %}Welcome to the Hotel Reservation System{% endblock %}

{% block content %}
<h2>Available Hotels</h2>
<div id="hotel-list">
   {% if hotels %}
       {% for hotel in hotels %}
           <div class="hotel-item">
               <h2>{{ hotel.name }}</h2>
               <p>Location: {{ hotel.location }}</p>
               <p>Rating: {{ hotel.rating }}</p>
               <p>Price per Night: ${{ hotel.price_per_night }}</p>
               <a href="{{ url_for('routes.get_hotel', hotel_id=hotel._id) }}">View
Details</a>
           </div>
       {% endfor %}
   {% else %}
       <p>No hotels available at the moment. Please check back later.</p>
   {% endif %}
</div>
{% endblock %}
```

- **Blocks:**
  - **`{% block title %}`: Sets the page title to "Hotel Reservation System".**
  - **`{% block header %}`: Adds a header message for the homepage.**
  - **`{% block content %}`: Dynamically displays hotels using a loop. If there are hotels available, it shows their details; otherwise, it displays a fallback message.**

base.html

This is the base template that provides the common structure for all pages, including a header, navigation bar, and footer.

```
<!--base.html-->

<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

   <link rel="icon" href="{{ url_for('static', filename='favicon.ico') }}">

   <title>{% block title %}Hotel Reservation System{% endblock %}</title>

</head>

<body>

   <header>

       <h1>{% block header %}Welcome to the Hotel Reservation System{% endblock %}</h1>

       <nav>

           <a href="{{ url_for('routes.index') }}">Home</a> |

           {% if 'user_id' in session %}

               <a href="{{ url_for('routes.logout') }}">Logout</a>

           {% else %}

               <a href="{{ url_for('routes.login') }}">Login</a> |

               <a href="{{ url_for('routes.register') }}">Register</a>

           {% endif %}

       </nav>

   </header>

   <main class="container">

       {% block content %}{% endblock %}

   </main>

   <footer>

       <p>© 2024 Hotel Reservation System. All rights reserved.</p>

   </footer>
```

```
    <script src="{{ url_for('static', filename='js/background-effects.js') }}"></script>

</body>

</html>
```

- **HTML Structure**:
  - **Header and Navigation**: Contains links for "Home," "Login," "Register," and "Logout" that adapt based on the user's session status.
  - **Blocks for Inheritance**: Defines `{% block title %}`, `{% block header %}`, and `{% block content %}` to be overridden in child templates.
  - **Static Files**: Links to external CSS (`style.css`) and JavaScript (`background-effects.js`) for styling and interactivity.

hotellist.html

```
{% extends 'base.html' %}

{% block title %}Available Hotels{% endblock %}

{% block header %}Available Hotels{% endblock %}

{% block content %}
{% if hotels %}
    <ul>
        {% for hotel in hotels %}
            <li>
                <h2>{{ hotel.name }}</h2>
                <p>Location: {{ hotel.location }}</p>
                <p>Rating: {{ hotel.rating }}</p>
                <p>{{ hotel.description }}</p>
                <p>Price per Night: ${{ hotel.price_per_night }}</p>
                <a href="{{ url_for('routes.get_hotel', hotel_id=hotel._id) }}">View
Details</a>
            </li>
        {% endfor %}
    </ul>
{% else %}
    <p>No hotels available at the moment.</p>
{% endif %}
{% endblock %}
```

This template extends `base.html` and is used to display a detailed list of hotels.

- **Blocks**:
  - **`{% block title %}` and `{% block header %}`**: Customize the page title and header for the hotel listing page.
  - **`{% block content %}`**: Lists all hotels with their details. Each hotel entry has its name, location, rating, description, and price, along with a link to view more details.

9. Glossary

This glossary provides definitions for terms used throughout the Software Requirements Specification (SRS) document:

- **Actor:** A user or system that interacts with the Hotel Reservation System.
- **Admin:** The system administrator responsible for managing user accounts, payments, and overall system settings.
- **Class Diagram:** A UML diagram that represents the static structure of the system by showing its classes, their attributes, and relationships.
- **Customer:** A user of the system who can search for hotels, make reservations, and manage their account.
- **Manager:** A user responsible for handling reservations, managing rooms, and generating reports.
- **Reservation:** The process of booking a room in a hotel, including details like room number, customer details, and reservation dates.
- **Use Case Diagram:** A UML diagram that depicts the functional requirements of the system by showing interactions between users and use cases.
- **Sequence Diagram:** A UML diagram that shows how objects interact in a particular scenario, highlighting the sequence of messages between them.
- **Payment:** The process of handling financial transactions within the system, including payment initiation and processing.