Carti Hotel Management System

Use-Case-Realization Specification: Manage Services

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/11/2024 | 1.0 | Final version | Dang Gia Nguyen |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 2

1.1 Purpose 2

1.2 Scope 2

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 References 2

1.5 Overview 2

2. Flow of Events 2

3. Derived Requirements 2

Use-Case-Realization Specification: Manage Services

# Introduction

## Purpose

This document describes how the Manage Services use-case is realized within the design model of the Carti Hotel Management System projects, in terms of collaborating objects.

## Scope

This document applies to the Carti Hotel Management System.

## Definitions, Acronyms, and Abbreviations

None

## References

None

## Overview

The subsequent section outlines the design of the use case and its associated requirements. In particular, the flow of events-design section incorporates the implementation of the class and sequence diagram for the use case, accompanied by concise textual explanations.

The following section presents the derived requirements essential for the implementation of the use case. These requirements serve as imperative guidelines to ensure the effective realization and functionality of the use case.

# Flow of Events—Design

### Flow of events

The following diagrams depict the essential classes and the sequence of their collaboration to bring about the realization of the Manage Services use case. After the user successfully logs in with the role of a receptionist, they are directed to the Receptionist Page. Subsequently, the receptionist can manage all services for a specific customer, including adding and deleting services

### Add service

On the Receptionist Page, the receptionist enters the customer’s name in the search bar and clicks the search button. The Search Controller then identifies all customers matching the input and displays a list of results on the screen. The receptionist can click on a customer, leading the system to navigate to the Reservation Detail Page. Here, the Reservation Controller retrieves the reservation history of the customer and displays it on the screen.

Subsequently, the receptionist selects the Service Listing button and is directed to the Service Page. The Service Controller then will retrieve the services from the Services database, and the services will be shown in table form on the screen.

After choosing the services reserved by the customer, the receptionist clicks the Save Button. This action prompts the Service Controller to obtain service information and add the services to the Service database. Additionally, the services are added to the customer’s bill. If all processes occur successfully, the Service Controller sends back a success message.

### Delete service

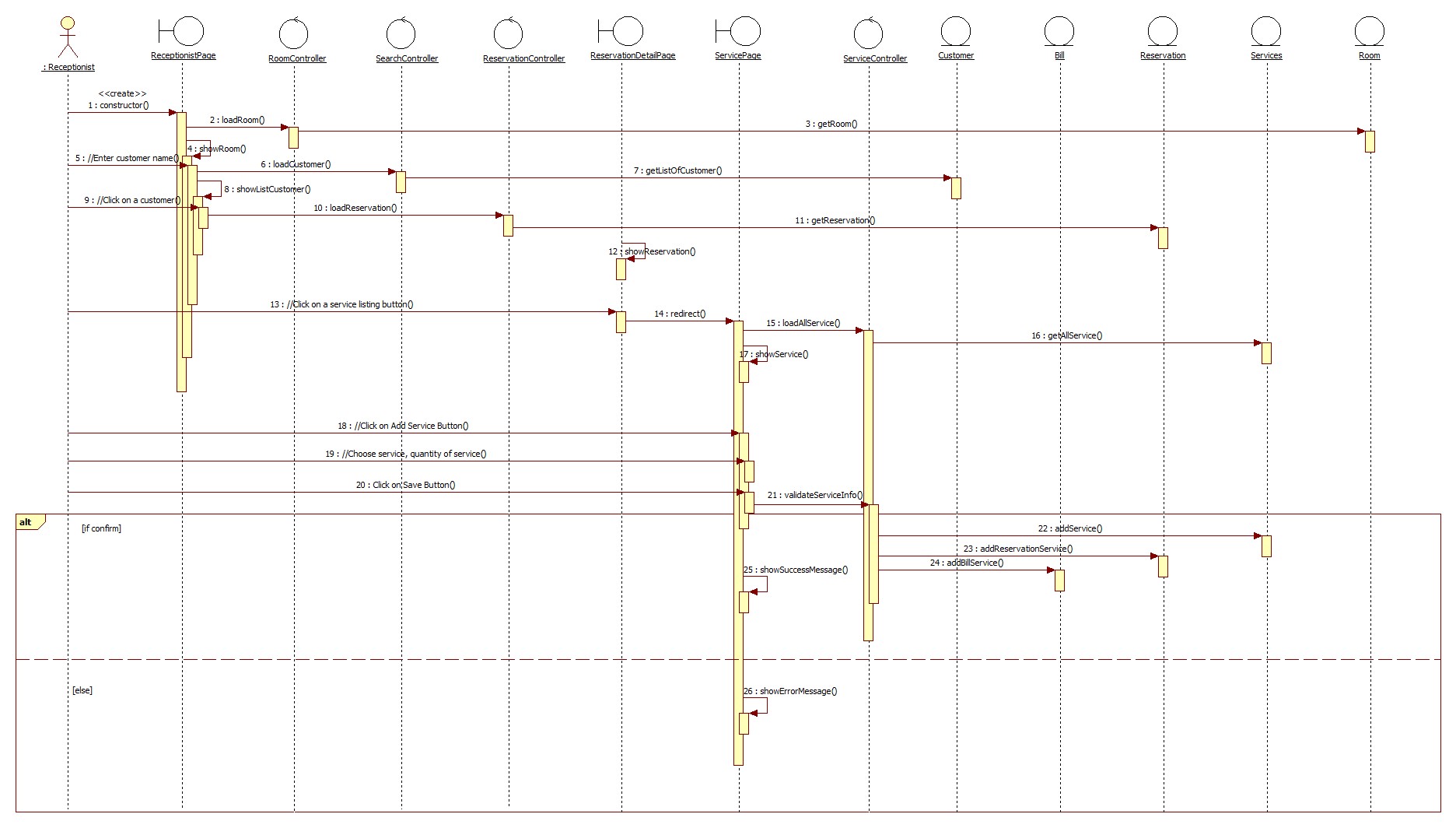
In the Receptionist Page, the receptionist inputs the customer's name into the search bar and clicks the search button. The Search Controller then identifies all customers that match the input and presents a list of results on the screen. The receptionist can select a customer, prompting the system to navigate to the Reservation Detail Page. On this page, the Reservation Controller retrieves the reservation history of the customer and exhibits it on the screen.

Afterward, the receptionist clicks on the Service Listing button, leading to redirection to the Service Page. Subsequently, the Service Controller retrieves the services from the Services database, and the displayed services appear in table form on the screen.

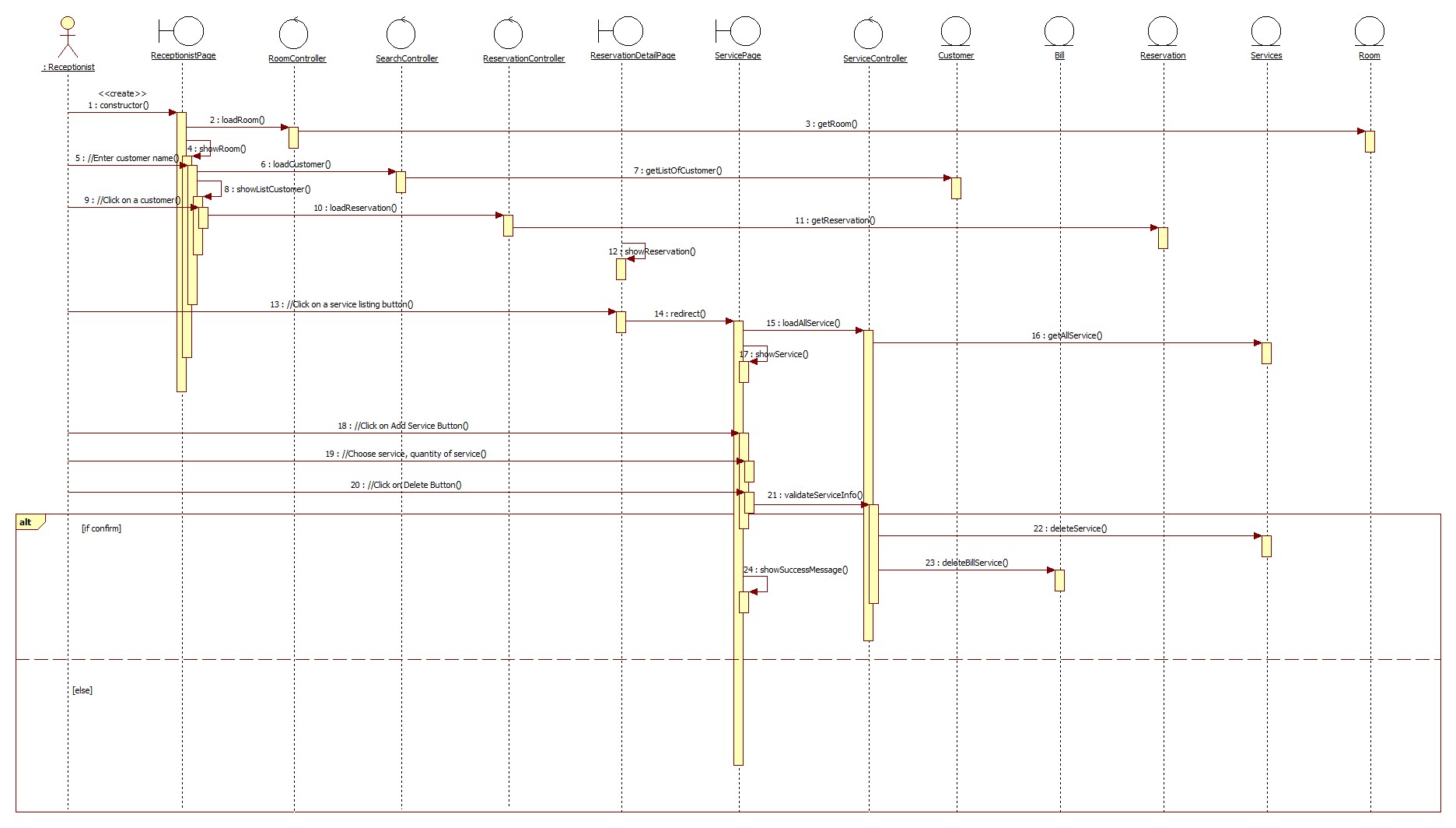
The receptionist selects a service and clicks the Delete Service button. Subsequently, a confirmation box appears. If the receptionist confirms the deletion, the Service Controller initiates a request to remove the service from the Service database. On the other hand, if the receptionist decides to cancel the deletion process, the Service Controller terminates the operation.

## sequence diagram

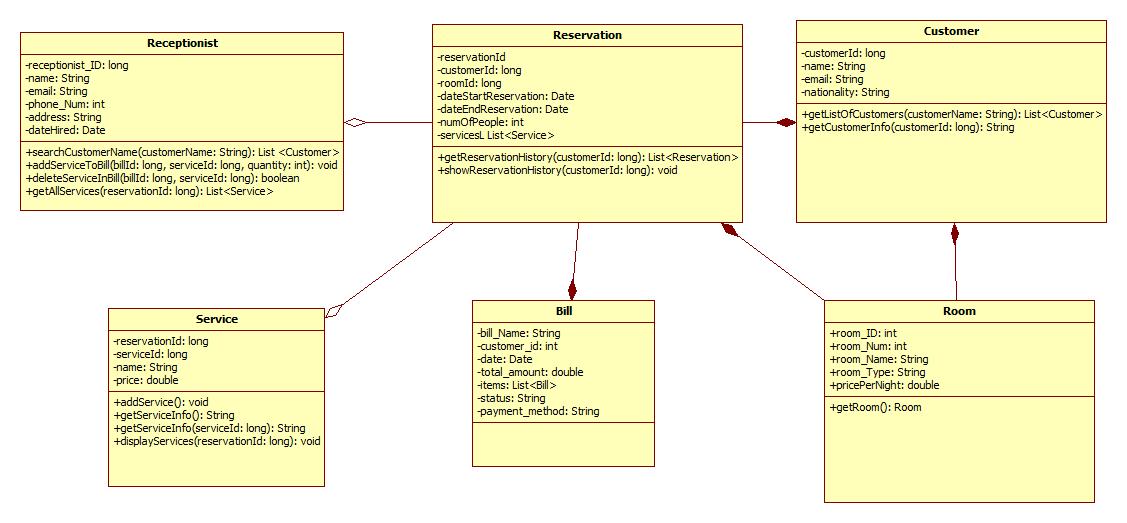
* + 1. **Add Service**



### **Delete Service**



## class diagram



# Derived Requirements

none