Hotel Management Application Documentation

March 28th 2022

Version	Date	Description	Author
1.0	2/23/2022	Create initial vision skeleton with some sections missing that will be added in future iterations. Also outline what references will be included with the vision document.	Andrew Walker
1.1	2/26/2022	Updated with customer use cases/SSD/operation contracts	Jalen Gordon
1.2	2/26/2022	Updated with hotel manager use cases/SSD/operation contracts	Andrew Walker
1.3	2/28/2022	Updated with domain model and wireframes	Andrew Walker
1.4	2/28/2022	Updated with Gantt Diagram	Andrew Walker
2.1	3/28/2022	Updated with GRASP Patters, Sequence Diagrams and Class Diagram	Andrew Walker

GitHub Link

https://github.com/walker76-school/hotel-management-system

Hour Report

- Andrew Walker - 39 hours
- Jalen Gordon - 6 hours

1 Introduction

The purpose of this document is to collect, analyze, and define high-level needs and features of the Hotel Management Application (HMA). It focuses on the capabilities needed by the stakeholders, and the target users, and why they need to exist. The details of how the HMA fulfils these needs are detailed in the use-case and supplementary specifications.

References

Positioning

Problem Statement

Finding a concise system for managing creating, updating and deleting reservations for hotel systems is difficult. Currently most hotels utilize expensive solutions that aren't extendable to the specific hotel's needs. A better solution would be robust, extendable system which allows for a flexible backend and customizable front-end for the hotel managers and customers to use.

Product Position Statement

For	Hotel managers
Who	are in need of an easy system to manage hotel reservations
HMA	is a hotel reservation management system
That	is flexible and extendable for their needs
Unlike	other current industry products
Our product	provides a robust, extendable system which allows for a flexible backend and customizable front-end for the hotel managers

Alternatives and Competition

Currently, there are no enterprise applications available as competition.

Stakeholder and User Descriptions

Market Demographics

There are over 91,000 hotels and motels in the US, generating over $194 billion a year. Over 52,000 of those are hotels.

Stakeholder Summary

Name	Description	Responsibilities
HMA	We, the company, that is creating the application	- Creation of service - Application maintenance - Testing features - Providing demonstrations of product - Customer Validation
Customer	The customer of HMA	- Monitors project progress - Specifies requirements of deliverable

User Summary

Name	**Description**	**Responsibilities**	Stakeholder
Hotel Manager	The purchaser of the HMA	- Cancel a reservation - Modify a reservation - View hotel reservations	Customer
Hotel Customer	They initialize the contact with the service by requesting a booking	- Create account - Book a reservation - Cancel a reservation - Modify a reservation - View all reservations	Customer

User Level Goals

HMA will provide a hotel reservation management system for hotel customers that need a seamless and easy experience for booking hotels. They will have the ability to modify and cancel reservations. The hotel managers will be able to easily manage all reservations for their hotel, including modifying and cancelling existing reservations.

User Environment

Each hotel customer will create their own account before booking new reservations. The customers will search for a hotel in the location of their choosing with

dates available in their requested range. A hotel customer can view all their booked reservations. A hotel manager is limited to cancelling and modifying booking initiated by customers. They will also be able to view all reservations for their hotel.

Product Overview

Product Perspective

The HMA application will be built on the Spring framework, with Java serving out the backend. Data will be stored in the JPA database. For the front end, NodeJS and associated JavaScript will be used to render web pages. The application will be deployed on the cloud-based service Heroku. In order to avoid downtime, developers can work on the application code internally, and when they have a new change, they push the change to Heroku and Heroku will update the page whenever users refresh the page.

A user will be able to access HMA via the web from any web-connected device to begin services.

Assumptions and Dependencies

There are several assumptions and dependencies that are connected with the technologies being used. Primarily, in order to access the application users must have internet connection. The final product will be released as a web app and not as a desktop or Android/iOS app. A further assumption is that the user will be operating the application in English. At this time, no plans are made to release the application in any languages other than English. Testing for the application will occur primarily in the browsers Google Chrome, Firefox, and Safari, and support for the application in other browsers is not guaranteed.

Cost and Pricing

Pricing for the HMA will be calculated on a per-hotel basis and be based around a number of location factors, including number of hotel rooms and potential annual bookings.

Licensing and Installation

We offer no installation assistance and licensing will be agreed upon individually with each hotel that purchases our service.

Summary of Product Features

Features of the application include the ability to: create a user account, delete a user account, and list, edit, cancel, or start a booking.

Users should be able to create an account with a username and password, as well as an email. Optional additional information includes a cell phone number, and an additional email.

Users should be able to delete their account and be sure that their account data will not be compromised. Upon account deletion, user account data will be wiped from the database and not be available to anyone.

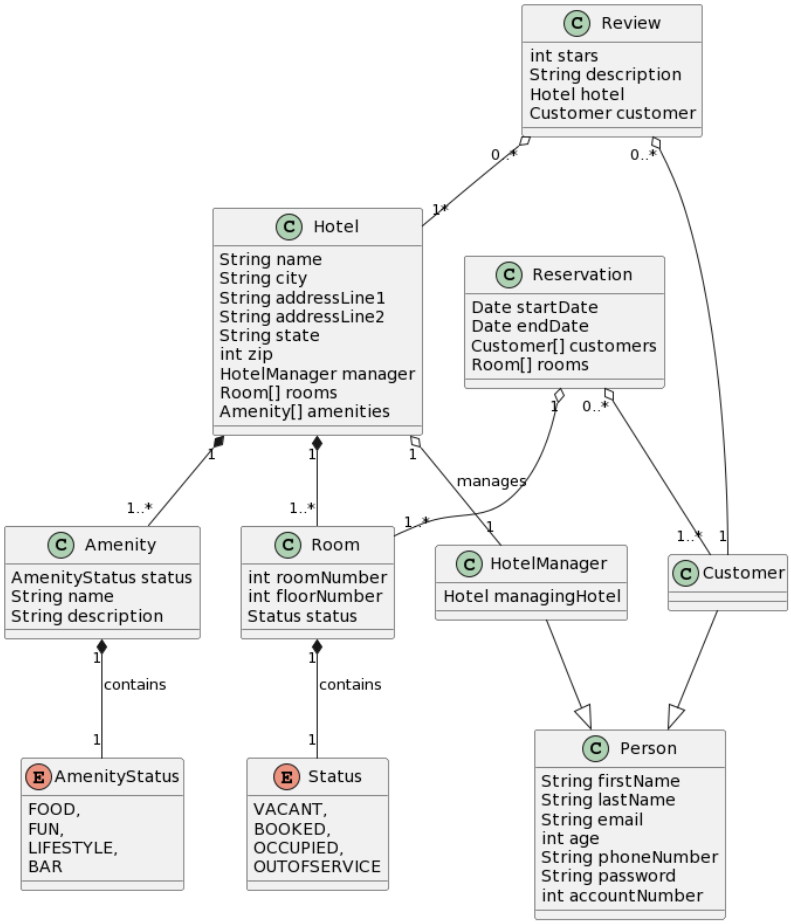
Once a hotel customer logs on to the application, they can search for hotels with availability. If the user decides that they like a hotel, then the user has the option to create a reservation. However, if the user changes their mind, the user may modify or cancel that reservation. If a user has multiple reservations, the user will be able to list and modify them.

Once a hotel manager logs on to the application, they can view, modify and cancel all reservations for their hotel.

Other Product Requirements

The user must have continual access to the internet for the application to work. If the user is disconnected from the application at any time, the application may cease to work properly.

Domain Model



Modify Reservation - Hotel Manager

Scope

Hotel Manager Portal

Level

user-goal

Primary Actor

Hotel Manager

Stakeholders and Interests

- Hotel - Wants to be able to easily see all reservations of the hotel to adequately understand current capacity and revenue

Preconditions

The hotel manager is a registered admin for the hotel.

Success Guarantee (or Postconditions)

The hotel management portal is populated with all current and upcoming reservations.

Main Success Scenario (or Basic Flow)

1. Hotel manager logs in with hotel manager account.
2. Hotel manager navigates to management portal
3. System presents hotel manager with all current and upcoming reservations

Extensions (or Alternative Flows)

*a User loses internet connection

1. System is unable to process user requests without internet
2. System presents error to user
3. User acknowledges error

3a. User regains internet connection with cache

1. System resumes using cached session

3b. User regains internet connection without cache

1. System starts new session

1a Hotel Manager is not registered as a manager account to hotel

1. Hotel manager navigates to manager registration screen
2. Hotel manager converts account to manager account
3. Hotel manager links account to hotel with hotel code

2a System is unable to find reservations

1. System is unable to locate any reservations
2. System presents error to user
3. User acknowledges error
4. System retries to find reservations

Special Requirements

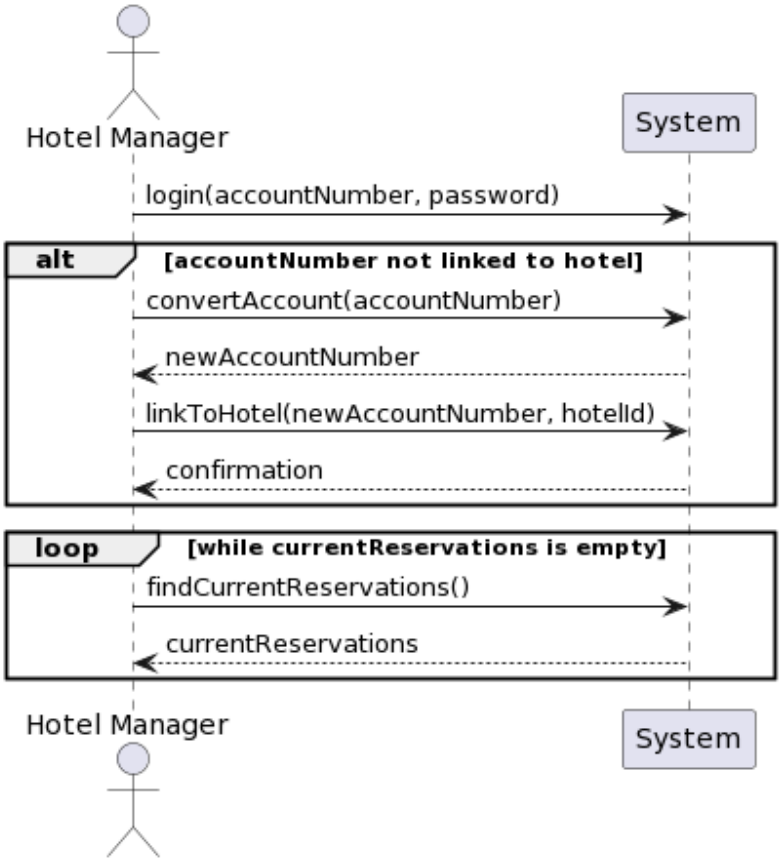
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Frequent


Hotel Manager
System
login(accountNumber, password)
alt
[accountNumber not linked to hotel]
convertAccount(accountNumber)
newAccountNumber
linkToHotel(newAccountNumber, hotelId)
confirmation
loop
[while currentReservations is empty]
findCurrentReservations()
currentReservations
Hotel Manager
System

Modify Reservation - Hotel Manager

Scope

Hotel Manager Portal

Level

user-goal

Primary Actor

Hotel Manager

Stakeholders and Interests

- Hotel - Wants to be able to easily modify reservations to potentially upsell customers
- Hotel Manager - Wants to have as seamless a process as possible for modifying so as not to waste time

Preconditions

The hotel manager is a registered admin for the hotel and has logged in to the admin portal. Also, at least one booking must be made for the hotel that the hotel manager will modify.

Success Guarantee (or Postconditions)

The booking is modified.

Main Success Scenario (or Basic Flow)

1. Hotel manager views list of current hotel reservations from system
2. Hotel manager clicks to modify the chose reservation
3. Hotel manager updates information in the modify form
4. Hotel manager saves modifications of the chose reservation

Extensions (or Alternative Flows)

*a User loses internet connection

    1. System is unable to process user requests without internet
    2. System presents error to user

3. User acknowledges error
    3a. User regains internet connection with cache
        1. System resumes using cached session
    3b. User regains internet connection without cache
        1. System starts new session

1a System is unable to find reservations

1. System is unable to locate any reservations
2. System presents error to user
3. User acknowledges error
4. System retries to find reservations

2a System is unable to find specific reservation

1. System is unable to locate any specific reservation
2. System presents error to user
3. User acknowledges error
4. System retries to find specific reservation

Special Requirements

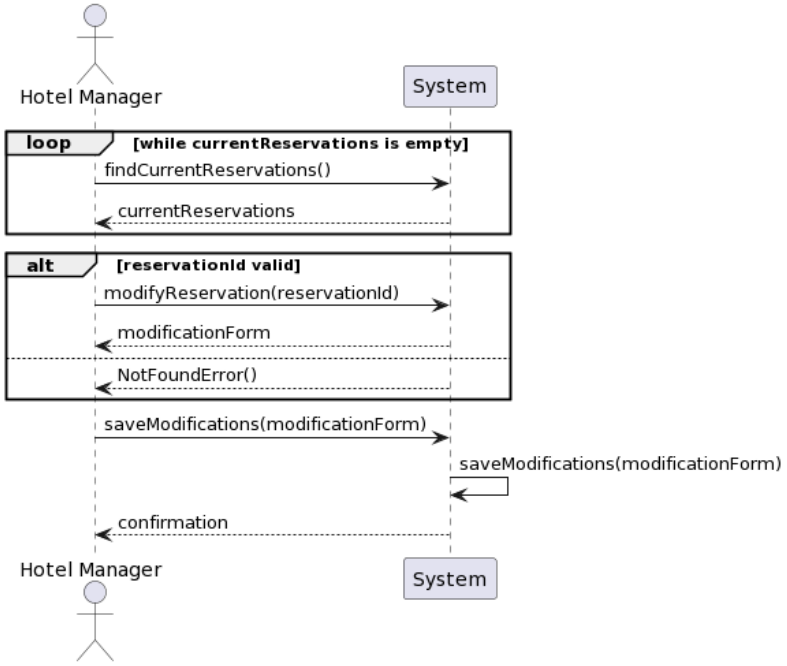
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Frequent


Hotel Manager
System
loop
[while currentReservations is empty]
findCurrentReservations()
currentReservations
alt
[reservationId valid]
modifyReservation(reservationId)
modificationForm
NotFoundError()
saveModifications(modificationForm)
saveModifications(modificationForm)
confirmation
Hotel Manager
System

Cancel Reservation - Hotel Manager

Scope

Hotel Manager Portal

Level

user-goal

Primary Actor

Hotel Manager

Stakeholders and Interests

- Hotel - Wants to be able to easily cancel unneeded reservations to free up as much availability for booking as possible
- Hotel Manager - Wants to have as seamless a process as possible for cancelling so as not to waste time

Preconditions

The hotel manager is a registered admin for the hotel and has logged in to the admin portal. Also, at least one booking must be made for the hotel that the hotel manager will cancel.

Success Guarantee (or Postconditions)

The booking is cancelled.

Main Success Scenario (or Basic Flow)

1. Hotel manager views list of current hotel reservations from system
2. Hotel manager clicks to modify the chose reservation
3. Hotel manager clicks to cancel the chosen reservation in the modification screen

Extensions (or Alternative Flows)

*a User loses internet connection

    1. System is unable to process user requests without internet
    2. System presents error to user
    3. User acknowledges error

3a. User regains internet connection with cache
    1. System resumes using cached session

3b. User regains internet connection without cache
    1. System starts new session

1a System is unable to find reservations

1. System is unable to locate any reservations
2. System presents error to user
3. User acknowledges error
4. System retries to find reservations

2a System is unable to find specific reservation

1. System is unable to locate any specific reservation
2. System presents error to user
3. User acknowledges error
4. System retries to find specific reservation

Special Requirements

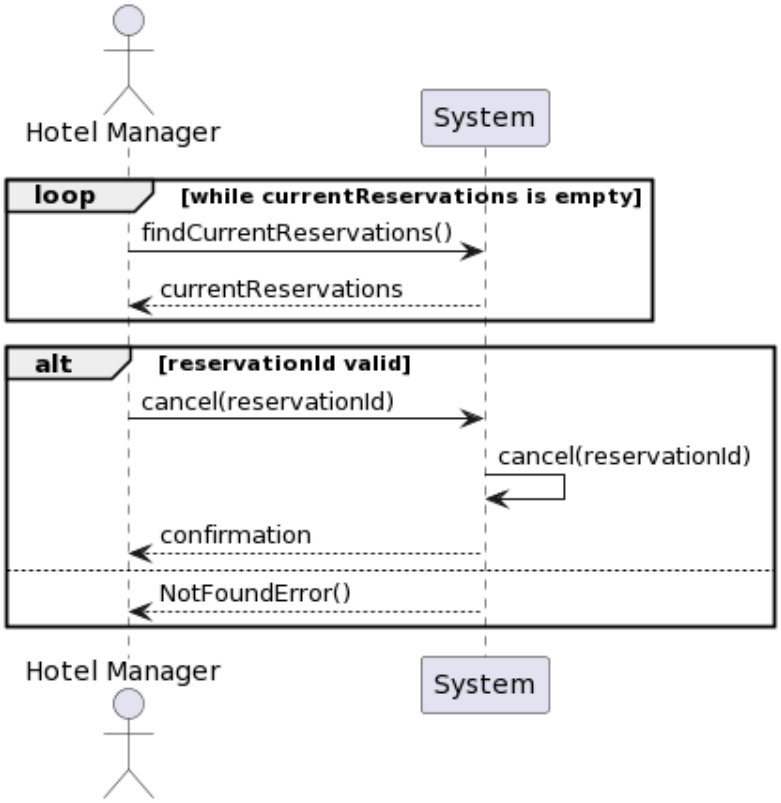
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Infrequent to Frequent - Cancellations are not nearly as frequent as modifications which will usually be done by the customer as opposed to the manager


Hotel Manager
System
loop
[while currentReservations is empty]
findCurrentReservations()
currentReservations
alt
[reservationId valid]
cancel(reservationId)
cancel(reservationId)
confirmation
NotFoundError()
Hotel Manager
System

Create Hotel Reservation - Customer

Scope

Hotel Reservation System

Level

user-goal

Primary Actor

Hotel Customer

Stakeholders and Interests

- Hotel - Wants to be able to provide customers the ability to create reservations without any errors or conflicts
- Customer - Wants to be able to create hotel reservation in an efficient and convenient manner

Preconditions

The customer should be able to login to his/her account

Success Guarantee (or Postconditions)

Customer can successfully create a reservation

Main Success Scenario (or Basic Flow)

1. Customer logs into hotel reservation system as a user
2. Customer navigates to booking screen and inputs requested information
3. Customer clicks on 'Book Now' to process reservation

Extensions (or Alternative Flows)

2a System is unable to find specific reservation

1. System is unable to locate any specific reservation
2. System presents error to user
3. User acknowledges error
4. System retries to find specific reservation

Special Requirements

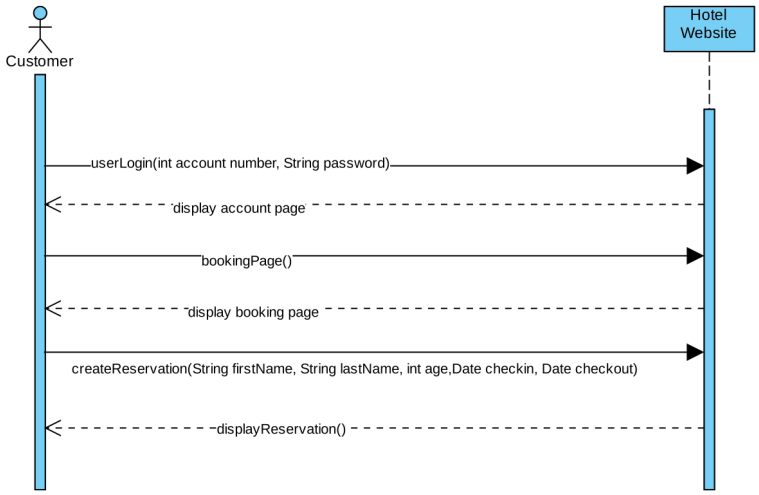
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Frequent


Customer
Hotel Website
userLogin(int account number, String password)
display account page
bookingPage()
display booking page
createReservation(String firstName, String lastName, int age,Date checkin, Date checkout)
displayReservation()

Modify Reservation - Customer

Scope

Hotel Reservation System

Level

user-goal

Primary Actor

Hotel Customer

Stakeholders and Interests

- Hotel - Wants to be able to accommodate any reservation modifications from the customer to enhance better booking experience
- Customer - Wants to be able modify/edit hotel reservation in an efficient and convenient manner

Preconditions

The customer should sign into his/her account, and already have a current reservation made.

Success Guarantee (or Postconditions)

Customer can successfully modify a reservation

Main Success Scenario (or Basic Flow)

1. Customer logs into hotel reservation system as a user
2. Customer selects a reservation to modify
3. Customer modifies the selected reservation on modification screen
4. Customer saves modifications to process the changes

Extensions (or Alternative Flows)

*a User loses internet connection

    1. System is unable to process user requests without internet
    2. System presents error to user
    3. User acknowledges error

3a. User regains internet connection with cache
    1. System resumes using cached session

3b. User regains internet connection without cache
    1. System starts new session

2a No reservation exists

1. System presents error to user
2. Customer books at least one hotel room booking
3. Customer returns to modification window

Special Requirements

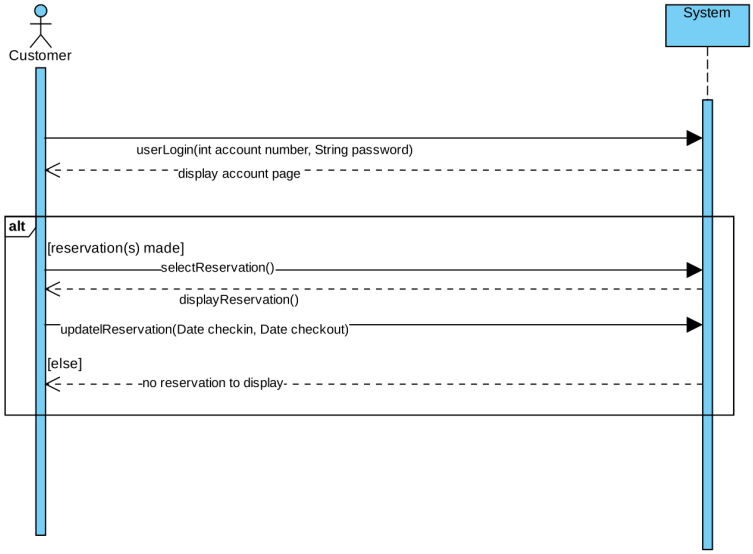
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Frequent


Customer
System
userLogin(int account number, String password)
display account page
alt
[reservation(s) made]
selectReservation()
displayReservation()
updatelReservation(Date checkin, Date checkout)
[else]
no reservation to display

Modify Reservation - Customer

Scope

Hotel Reservation System

Level

user-goal

Primary Actor

Hotel Customer

Stakeholders and Interests

- Hotel - Wants to be able to cancel unwanted reservations to better manage inventory, and increase the availability of room bookings as possible
- Customer - Wants to be able cancel hotel reservation in an efficient and convenient manner

Preconditions

The customer should sign into his/her account, and already have a current reservation made.

Success Guarantee (or Postconditions)

Customer can successfully cancel a reservation

Main Success Scenario (or Basic Flow)

1. Customer logs into hotel reservation system as a user
2. Customer selects reservation to modify
3. Customer cancels the selected reservation on modification screen

Extensions (or Alternative Flows)

*a User loses internet connection

1. System is unable to process user requests without internet
2. System presents error to user
3. User acknowledges error

3a. User regains internet connection with cache

1. System resumes using cached session

3b. User regains internet connection without cache

1. System starts new session

2a No reservation exists

1. System presents error to user
2. Customer books at least one hotel room booking
3. Customer returns to modification window

Special Requirements

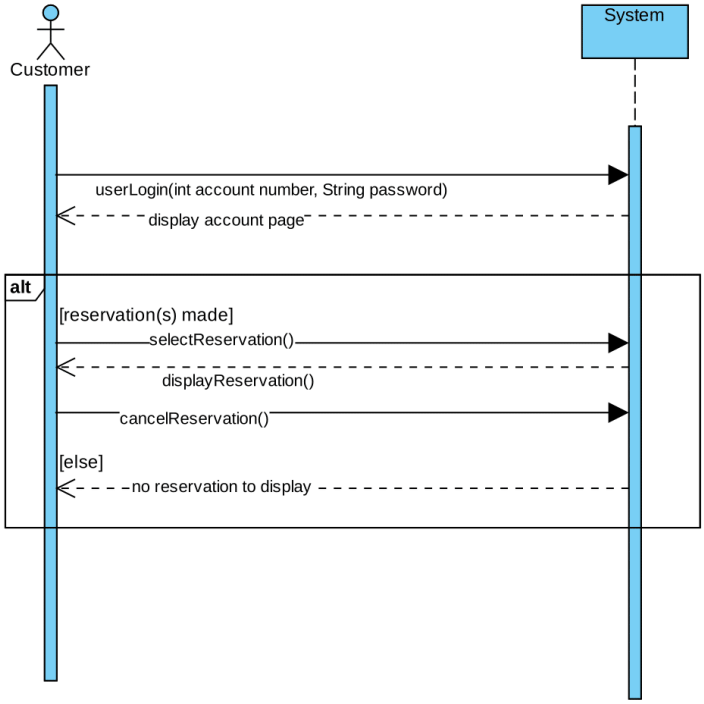
- N/A

Technology and Data Variations List

N/A

Frequency of Occurrence

Frequent


Customer
System
userLogin(int account number, String password)
display account page
alt
[reservation(s) made]
selectReservation()
displayReservation()
cancelReservation()
[else]
no reservation to display

Operation Contracts

login

- **Operation:** login(accountNumber, password)
- **Cross References:** Cancel Reservation [HM]
- **Preconditions** User is logged out
- **Postconditions:** User is logged in

convertAccount

- **Operation:** convertAccount(accountNumber)
- **Cross References:** Cancel Reservation [HM]
- **Preconditions** User is logged in
- **Postconditions:** User's account is converted to a manager account and a new accountNumber is returned

linkToHotel

- **Operation:** linkToHotel(accountNumber, hotelId)
- **Cross References:** Cancel Reservation [HM]
- **Preconditions** User is logged in
- **Postconditions:** User's account is linked as a manager to the hotel with ID hotelId

findCurrentReservations

- **Operation:** findCurrentReservations()
- **Cross References:** Cancel Reservation [HM], Modify Reservation [HM], Cancel Reservation [HM]
- **Preconditions** User is logged in as a manager account of a hotel
- **Postconditions:** User is presented with all current and upcoming reservations

modifyReservation

- **Operation:** modifyReservation(reservationId)
- **Cross References:** Modify Reservation [HM]
- **Preconditions** User is logged in as a manager account of a hotel
- **Postconditions:** User is presented with a modification form of the reservation with ID reservationId

saveModifications

- **Operation:** saveModifications(modificationForm)
- **Cross References:** Modify Reservation [HM]
- **Preconditions** User is logged in as a manager account of a hotel
- **Postconditions:** Changes to the reservation are persisted in the system

cancel

- **Operation:** cancel(reservationId)
- **Cross References:** Cancel Reservation [HM]
- **Preconditions** User is logged in as a manager account of a hotel
- **Postconditions:** Reservation with ID reservationId is cancelled in the system

userLogin

- **Operation:** userLogin(accountNumber, password)
- **Cross References:** Create Reservation [HC]
- **Preconditions** User is logged out
- **Postconditions:** User is logged in

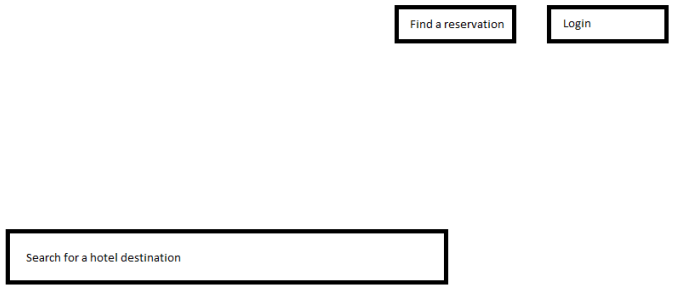
createReservation

- **Operation:** createReservation(firstName, lastName, age, checkin, checkout)
- **Cross References:** Create Reservation [HC]
- **Preconditions** User is logged in
- **Postconditions:** User has a new reservation populated in the system with dates checkin and checkout for the selected room

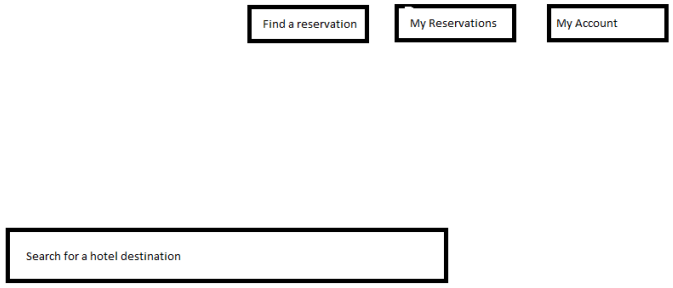
updateReservation

- **Operation:** updateReservation(checkin, checkout)
- **Cross References:** Create Reservation [HC]
- **Preconditions** User is logged in with a reservation selected
- **Postconditions:** User has an updated reservation populated in the system with dates checkin and checkout for the selected room

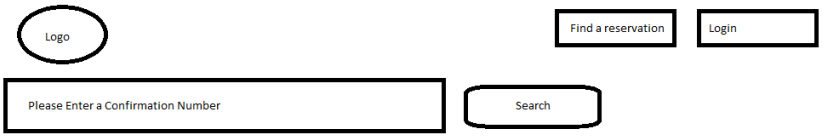
Wireframe - Homepage Logged Out



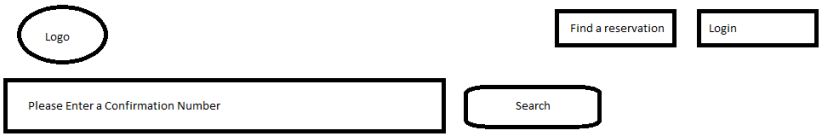
Wireframe - Homepage Logged In



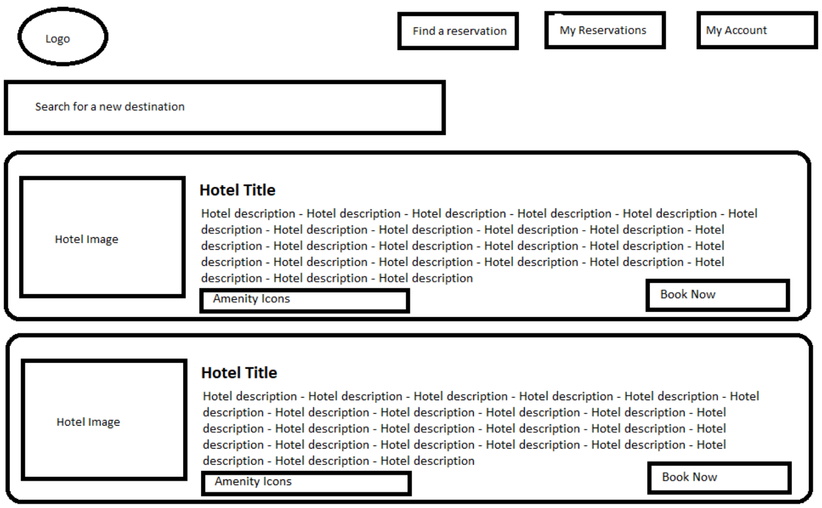
Wireframe - Find Reservation Search



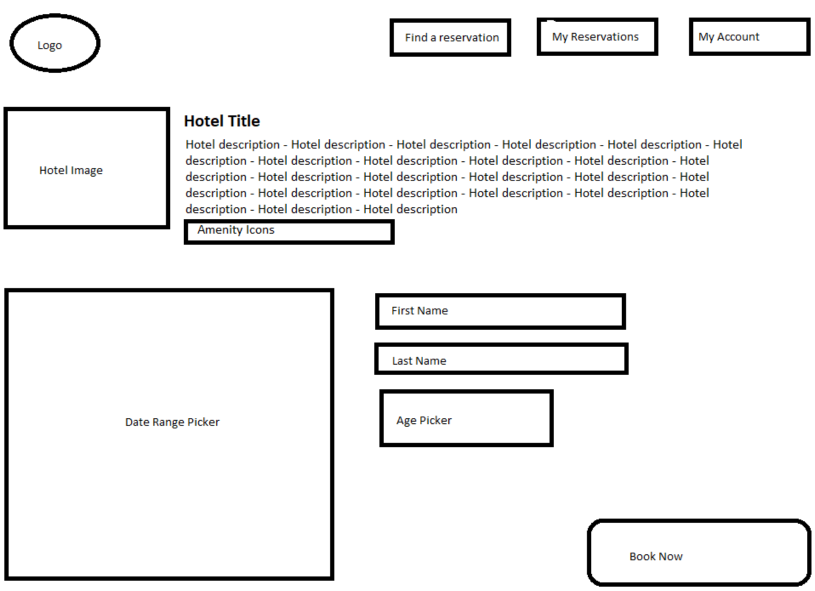
Wireframe - Find Reservation Results



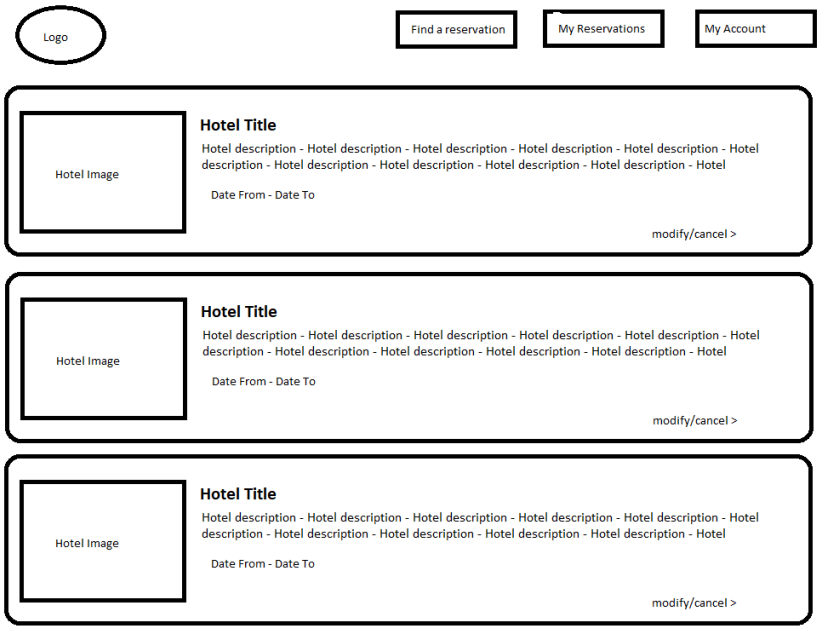
Wireframe - Destination Search Results



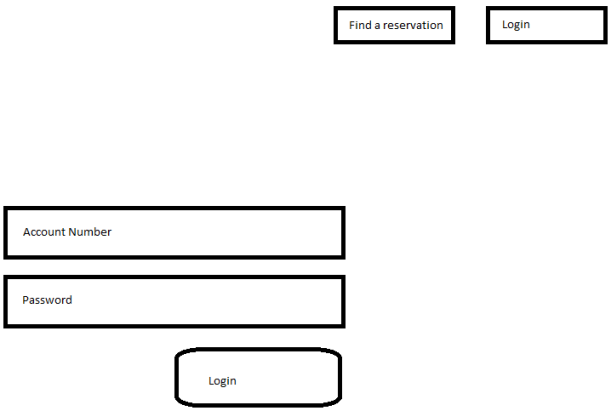
Wireframe - Book a Reservation



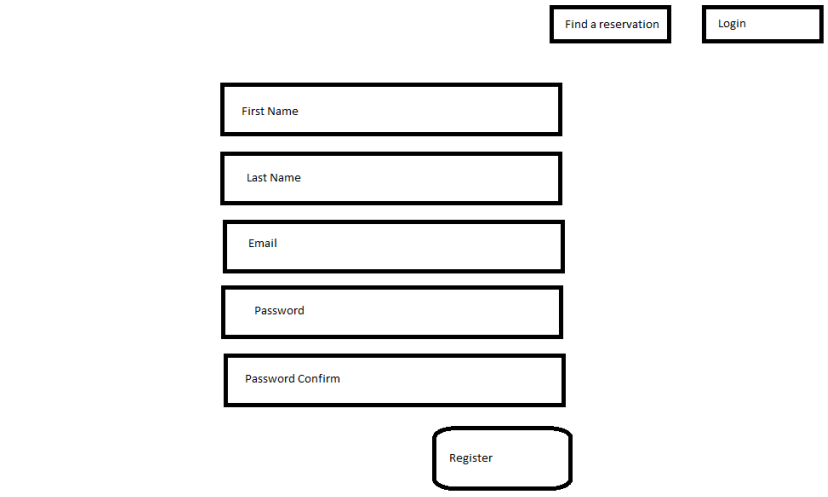
Wireframe - My Reservations



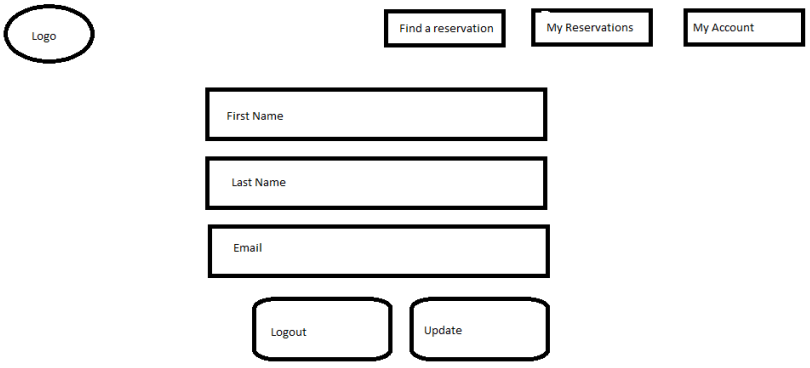
Wireframe - Login



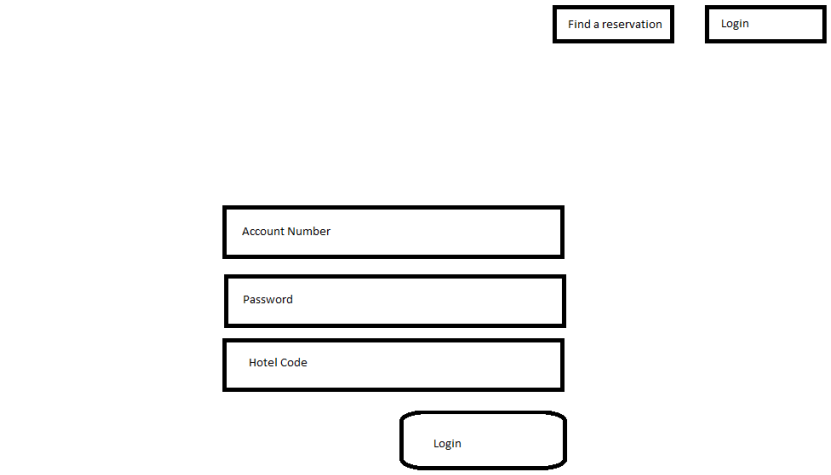
Wireframe - Registration



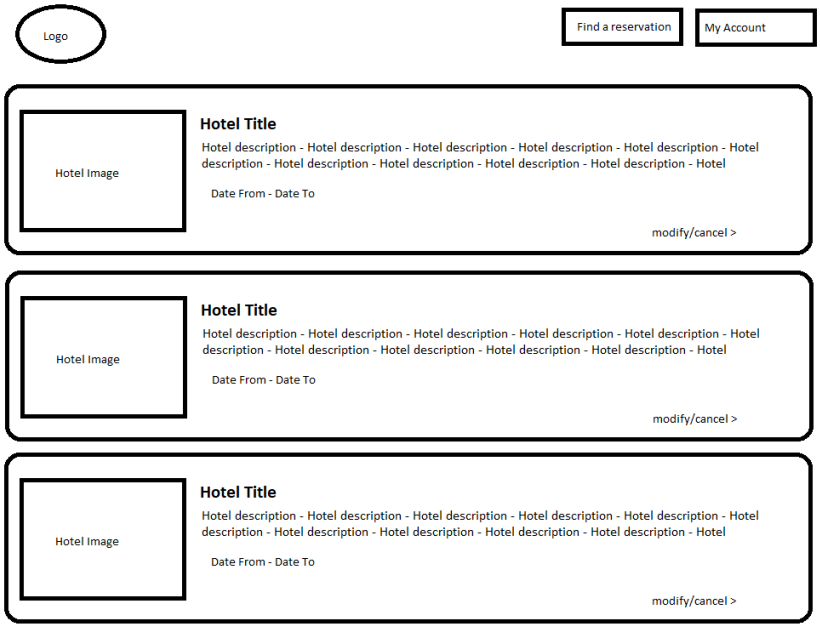
Wireframe - My Account



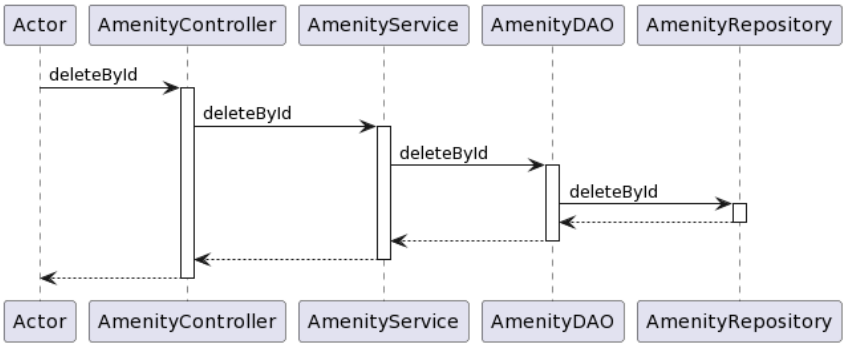
Wireframe - Admin Login



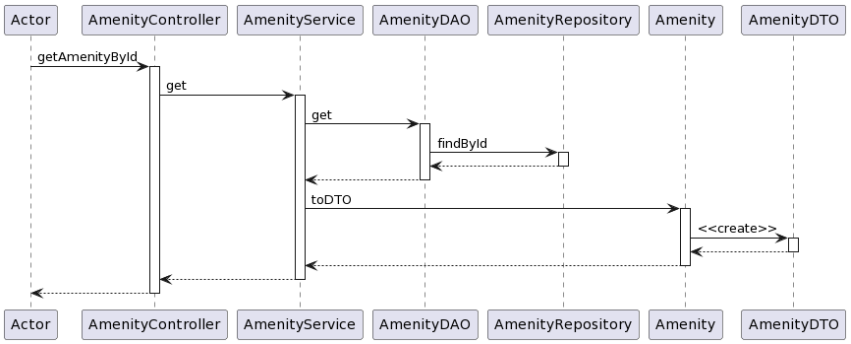
Wireframe - Admin Hotel Reservations



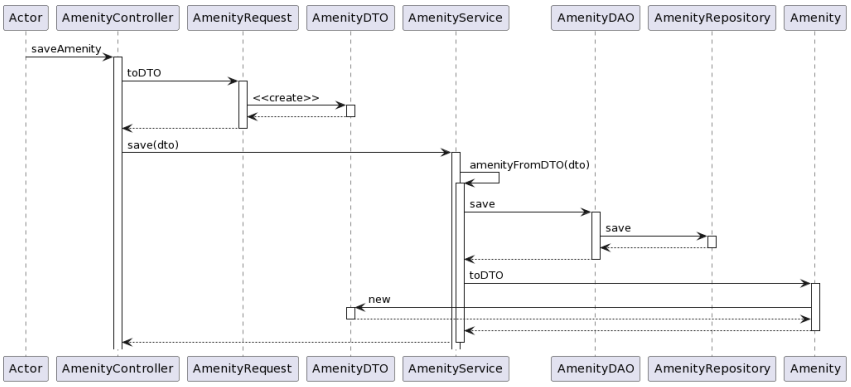
Sequence Diagram - AmenityController#deleteById



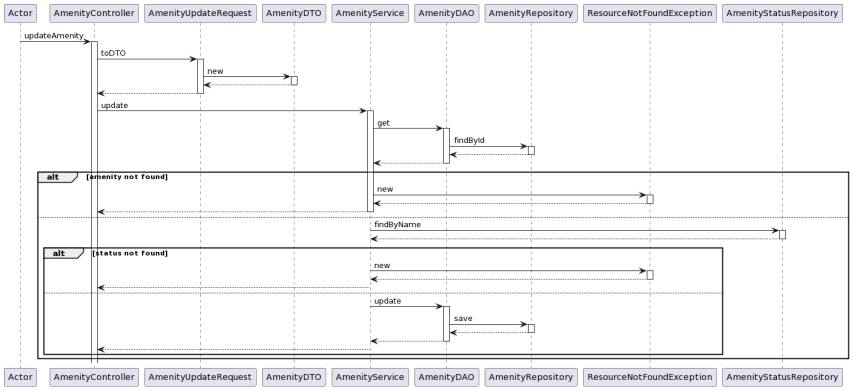
Sequence Diagram - AmenityController#getAmenityById



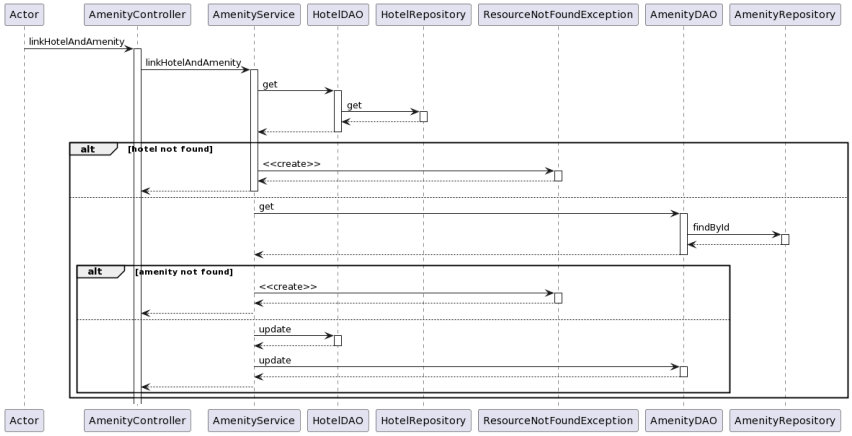
Sequence Diagram - AmenityController#saveAmenity



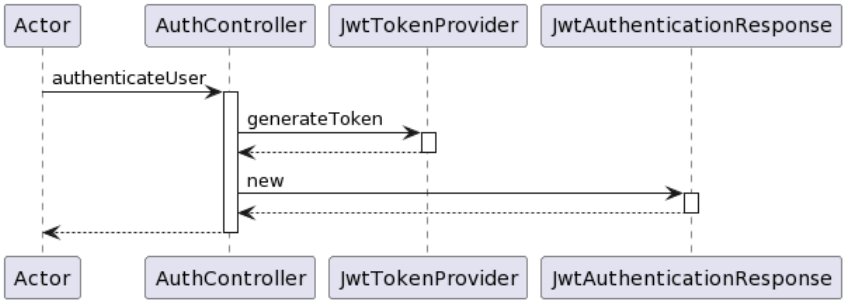
Sequence Diagram - AmenityController#updateAmenity



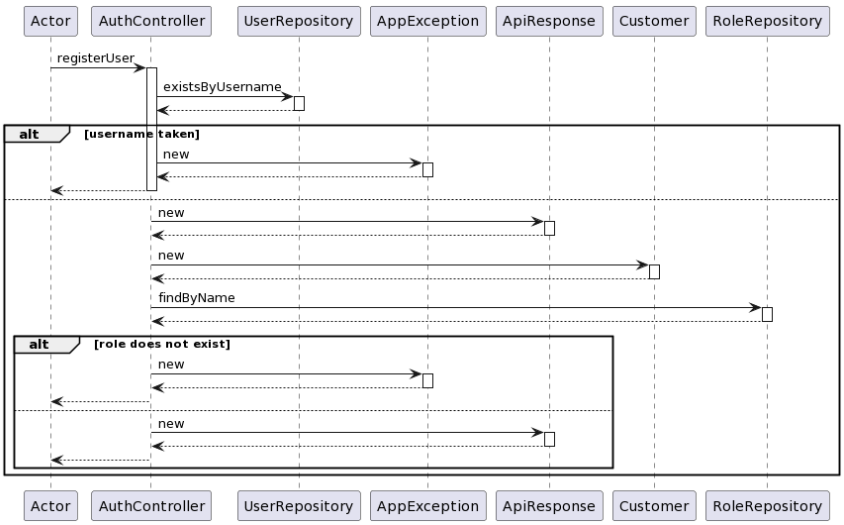
Sequence Diagram - AmenityController#linkHotelAndAmenity



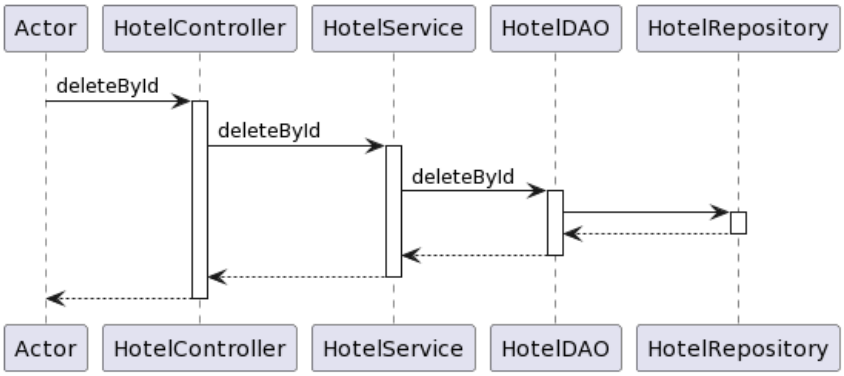
Sequence Diagram - AuthController#authenticateUser



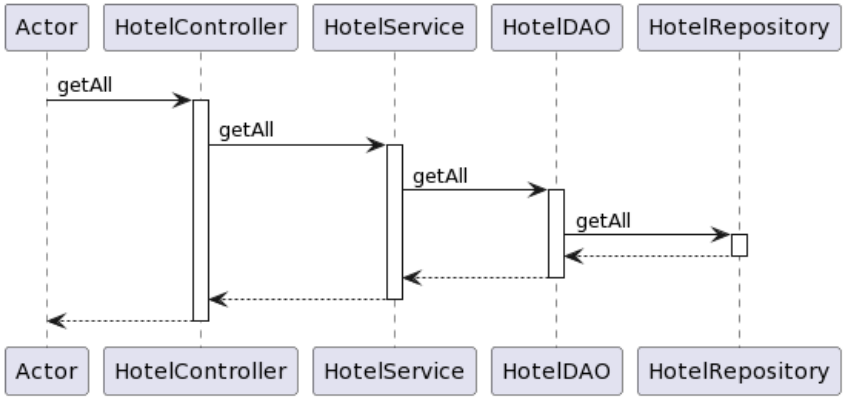
Sequence Diagram - AuthController#registerUser



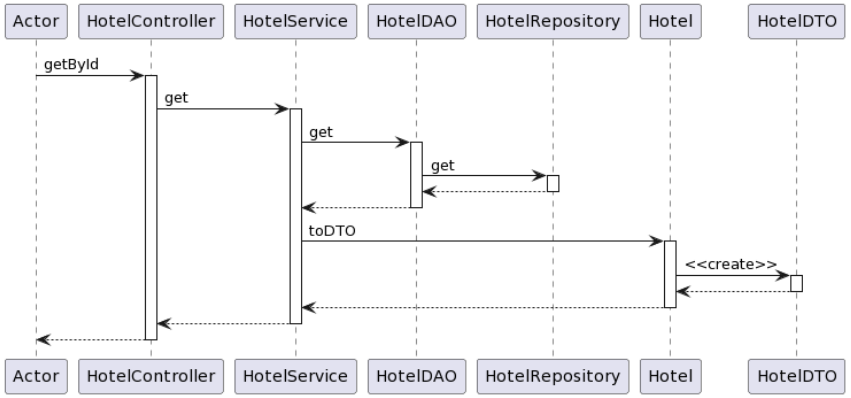
Sequence Diagram - HotelController#deleteById



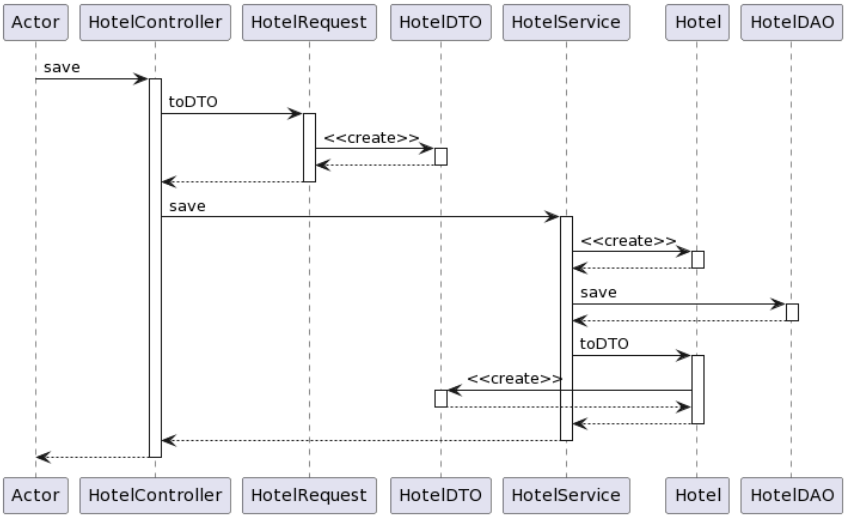
Sequence Diagram - HotelController#getAll



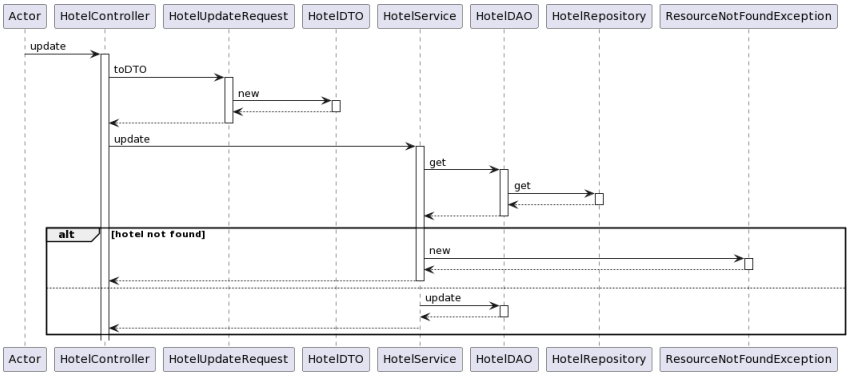
Sequence Diagram - HotelController#getById



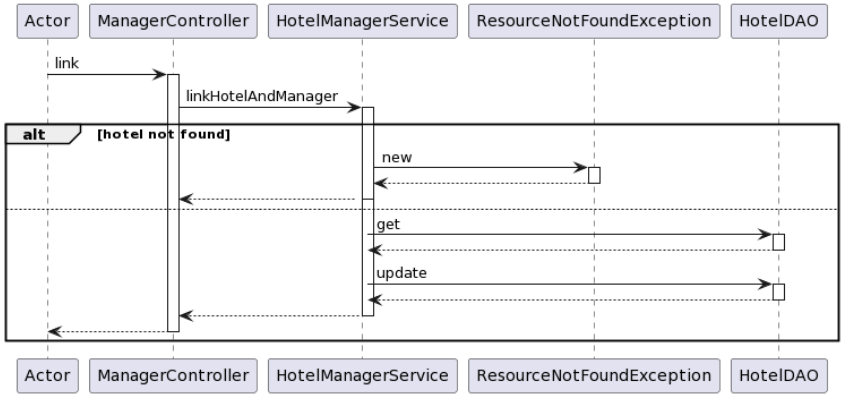
Sequence Diagram - HotelController#save



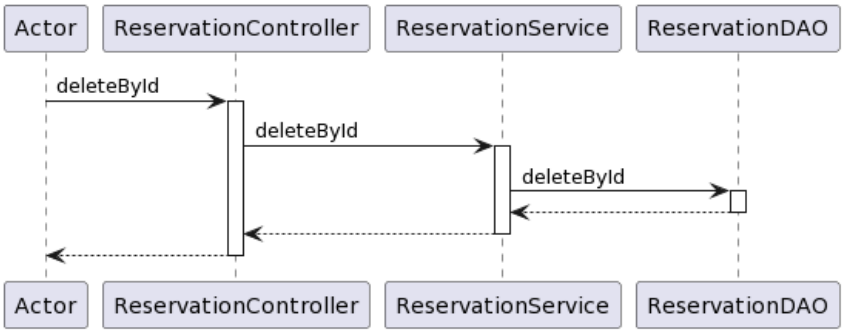
Sequence Diagram - HotelController#update



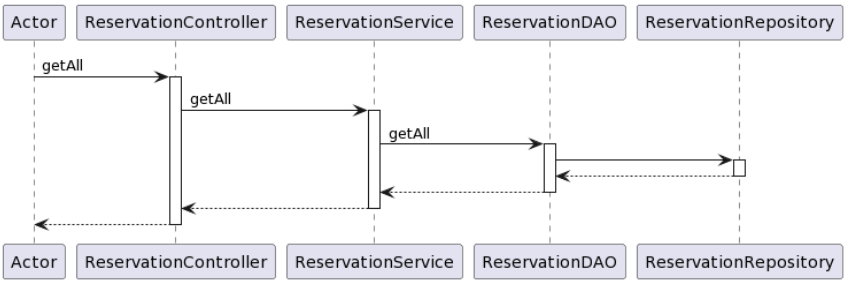
Sequence Diagram - ManagerController#link



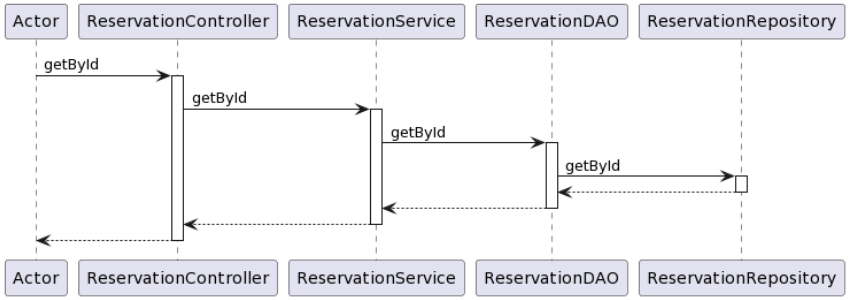
Sequence Diagram - ReservationController#deleteById



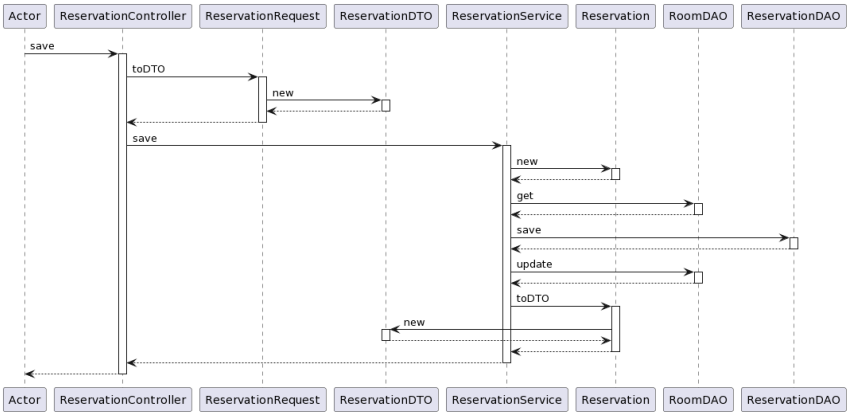
Sequence Diagram - ReservationController#getAll



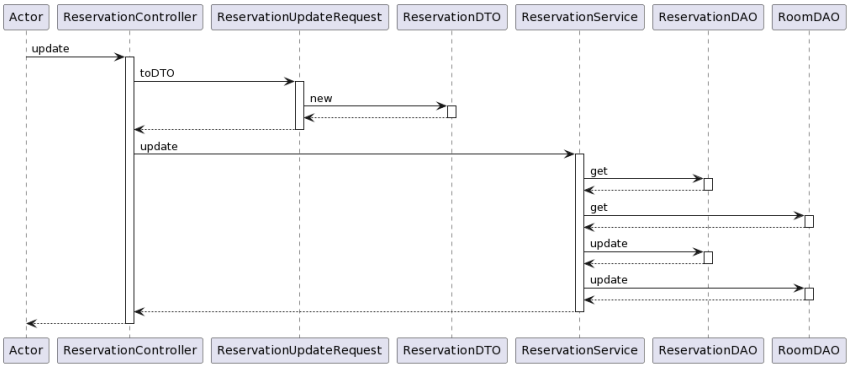
Sequence Diagram - ReservationController#getById



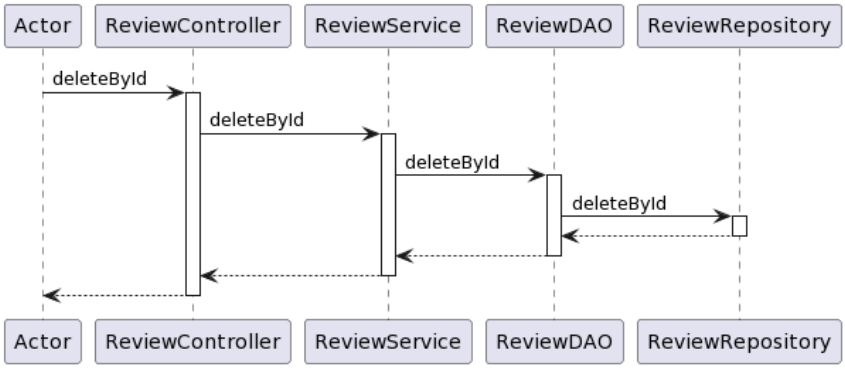
Sequence Diagram - ReservationController#save



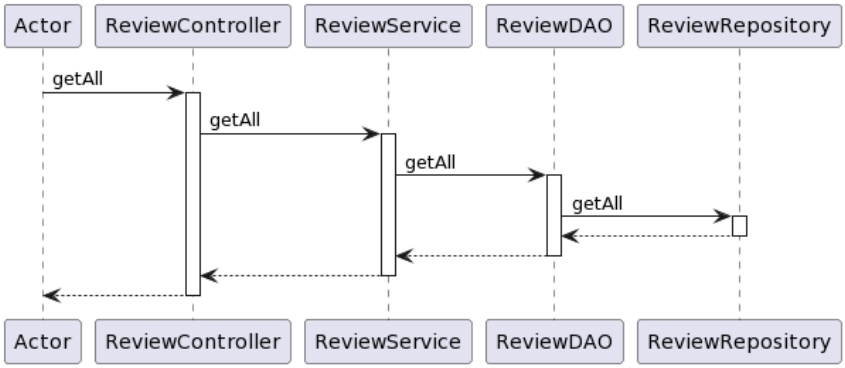
Sequence Diagram - ReservationController#update



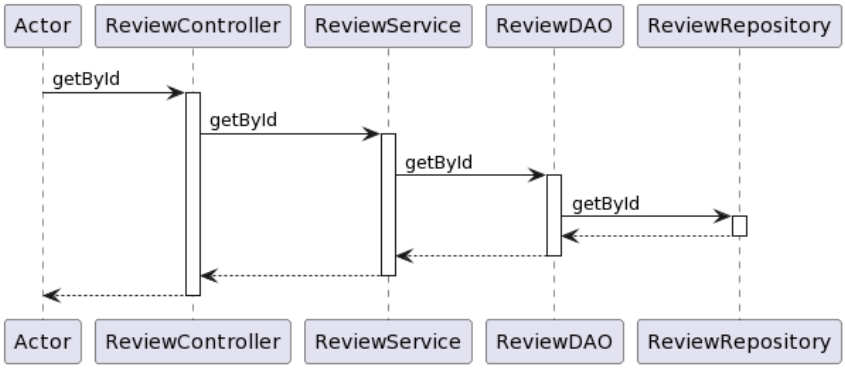
Sequence Diagram - ReviewController#deleteById



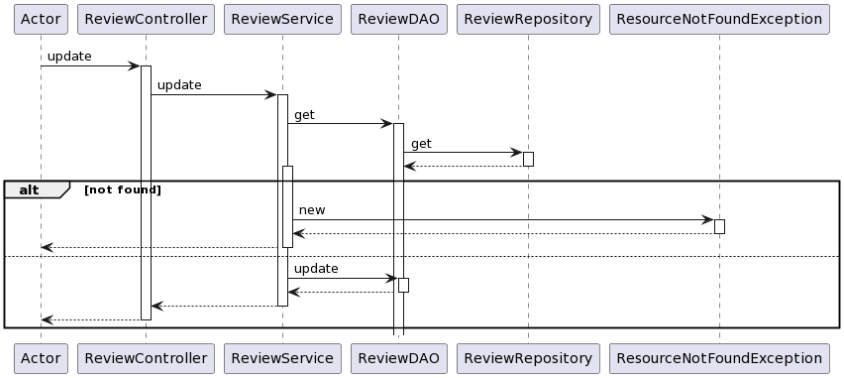
Sequence Diagram - ReviewController#getAll



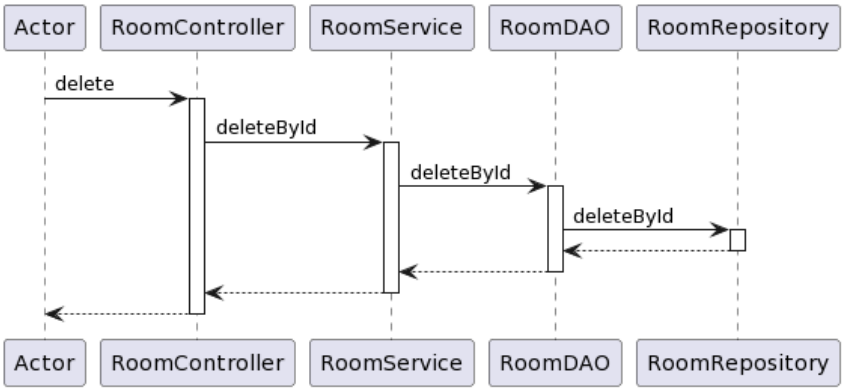
Sequence Diagram - ReviewController#getById



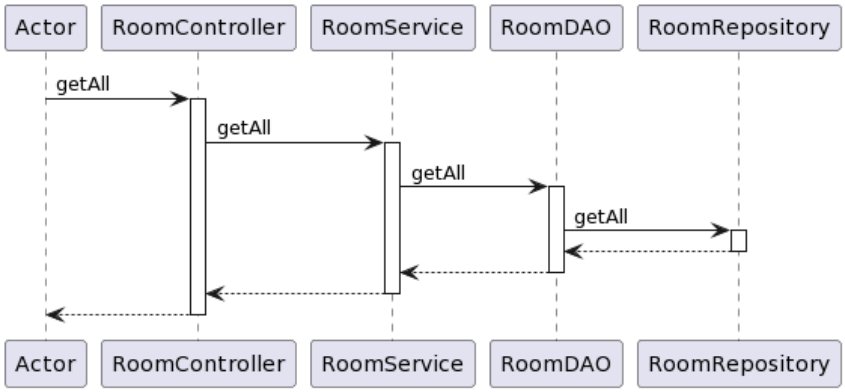
Sequence Diagram - ReviewController#update



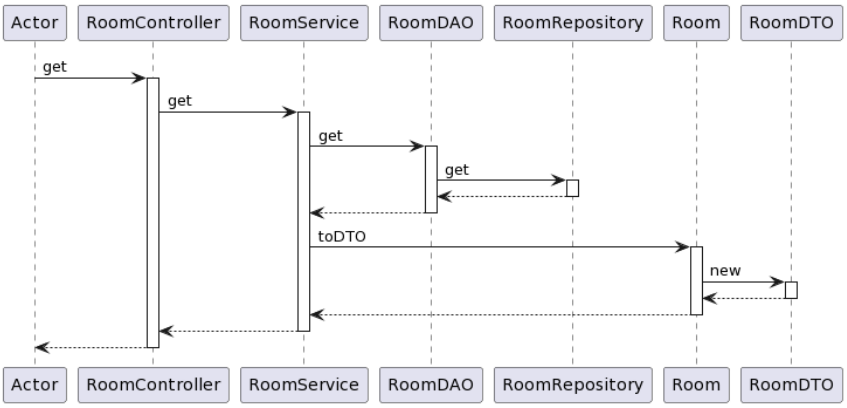
Sequence Diagram - RoomController#delete



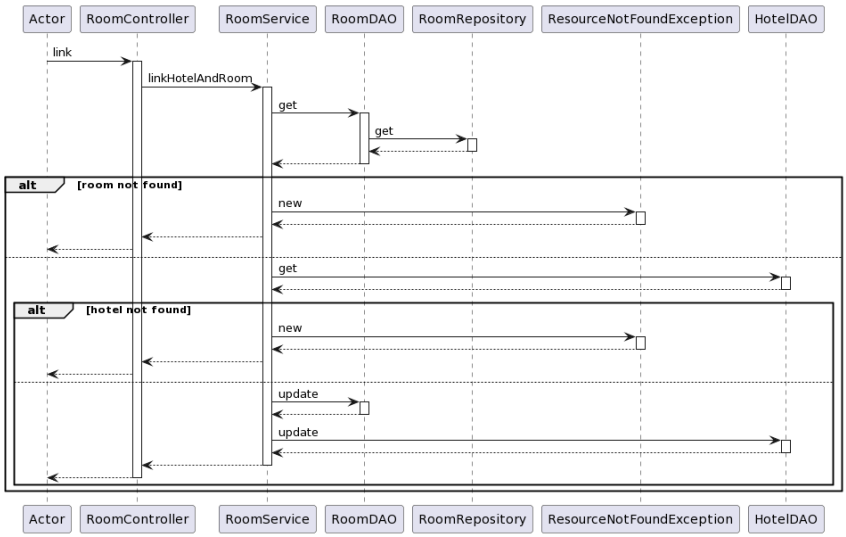
Sequence Diagram - RoomController#getAll



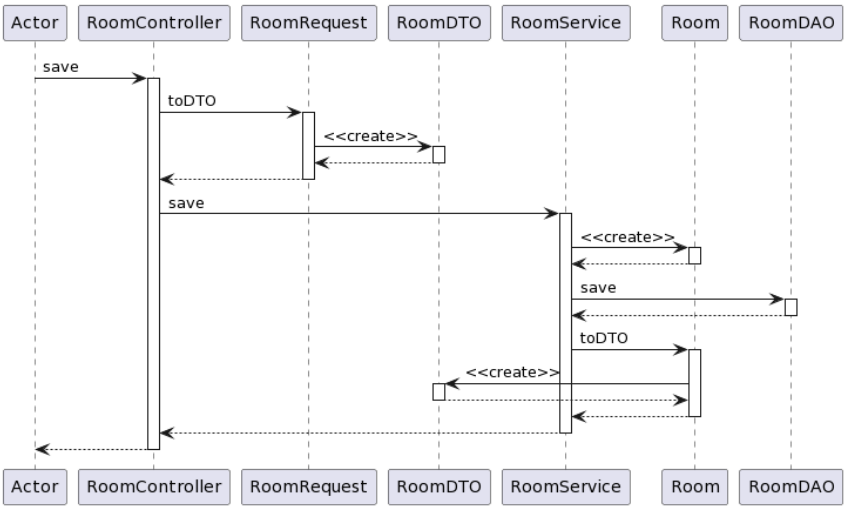
Sequence Diagram - RoomController#get



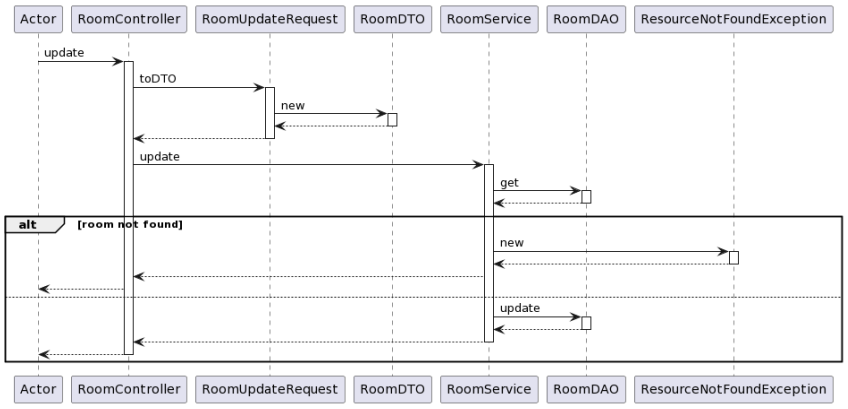
Sequence Diagram - RoomController#link



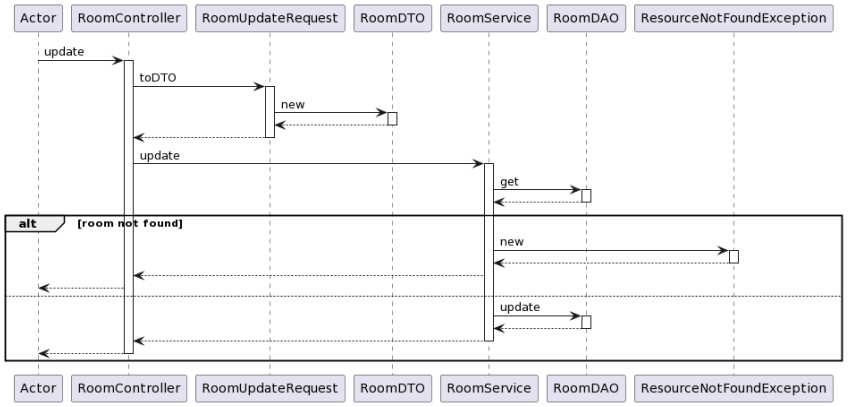
Sequence Diagram - RoomController#save



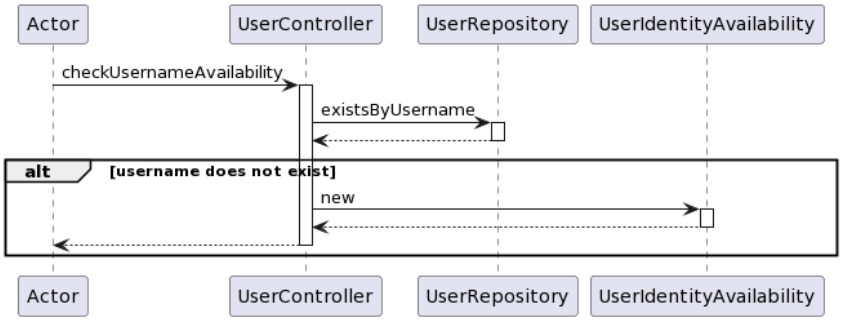
Sequence Diagram - RoomController#update



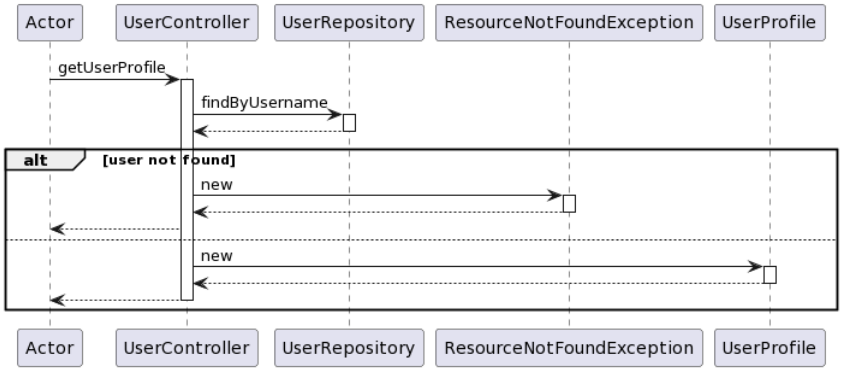
Sequence Diagram - RoomController#update



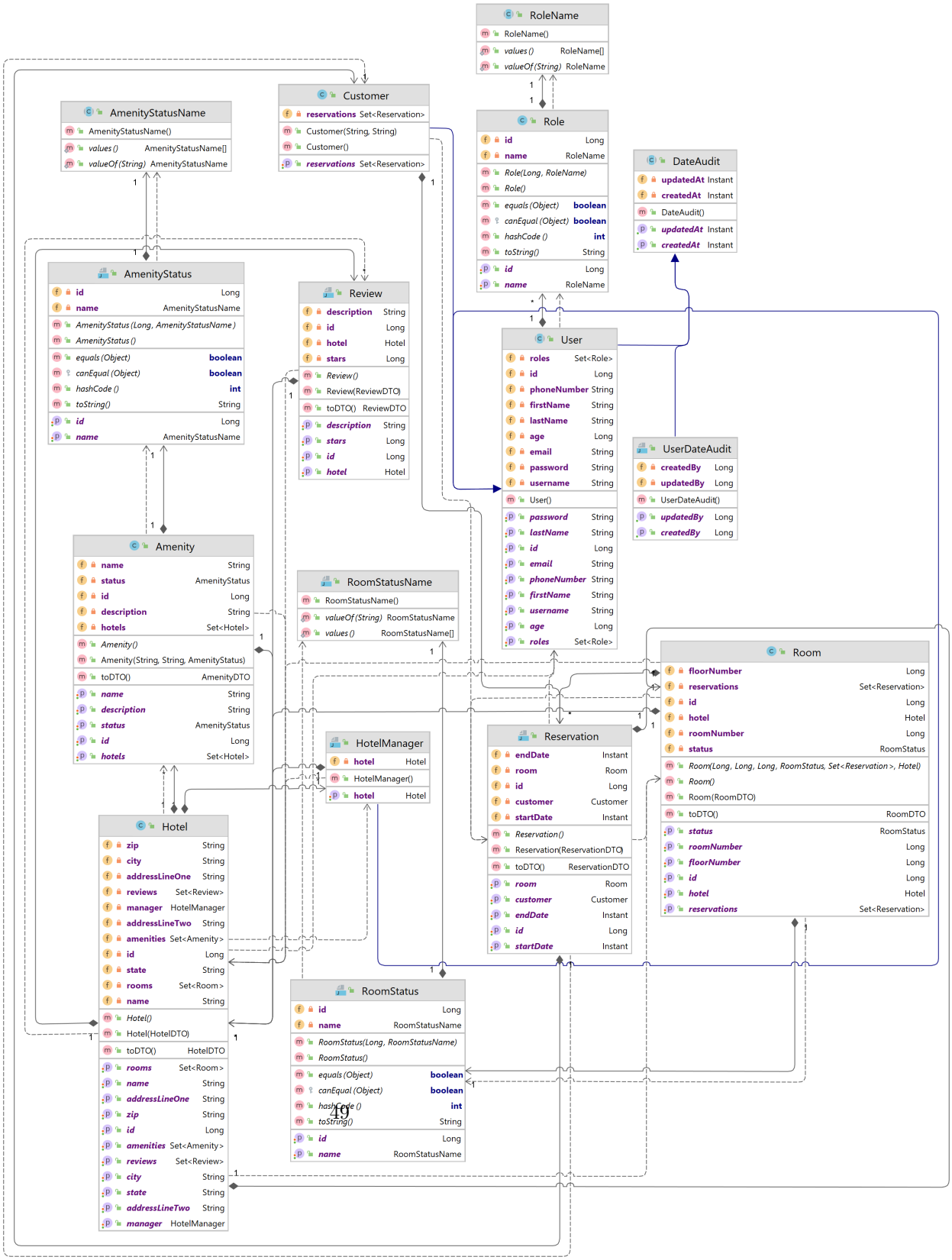
Sequence Diagram - UserController#checkUsernameAvailable



Sequence Diagram - UserController#getUserProfile



Class Diagram -



GRASP Patterns

Information expert

The responsibility for creating DTO objects for transfer is delegated to the entity objects since they have the information to be able to create them.

Creator

We delegate creation of entity objects to our DTO's since they have the initializing data that is required to construct an entity for persistence.

Controller

The system uses REST API Controllers for each of the entities in the system.

Indirection

Indirection is captured through our use of Pure Fabrication to delegate through DAO and Service objects.

Low coupling

We practice low coupling by making sure that whenever possible we use a common iterface, like the IService for services or DAO for the DAO objects. This lets us leverage the API without worrying about the underlying implementation being changed or different, which reduces the coupling between the layers of our application.

High cohesion

All of our project practices high cohesion has the controllers, DAO, service and repository objects all deal with a single entity/domain. For example, to manage a Hotel, there is a HotelController, HotelService, HotelDAO and finally HotelRepository which all only deal with Hotel data. The same could be said for our other entities.

Polymorphism

We use polymorphism with our User object which is extended by both HotelManager and Customer so we can capture the common data and functionality and avoid having to branch based on type. The distiguisher between a customer and hotel manager is the role that it has.

Protected Variations

To protect, we use interfaces for our DAO and Service objects to reduce the impact of internal changes on the other systems.

Pure fabrication

We use Pure Fabrication to reduce coupling by introducing DAO and Service objects to remove functionality away from the repository elements.