# 

2DV603 - Requirements Specification

By Johan Andersson (ja222um), David Larsson (dl222is) & Fredrik Norrman (fn222ep)

*Version 1*

###### 

# 

# 

# 

# 

[**1 Introduction**](#_hsihjvhwc4vi) **3**

[1.1 Stakeholders](#_ibz76j4f3mcp) 3

[1.2 Stakeholder needs](#_wpppt0wq2hx2) 3

[1.3 Purpose](#_252x7c7hn8fr) 3

[**2 Scope**](#_ybjlyk566pc4) **3**

[**3 Requirements**](#_z7a7xv76j1pf) **4**

[3.1 Functional Requirements](#_ndv6rnjshh9) 4

[3.2 Quality Requirements](#_lfg6xme2nlnc) 5

[3.3 System Requirements](#_xsun1qtuynqw) 5

[**4 Use Cases**](#_53xtrjk1sgs6) **6**

[4.1 Scenarios](#_90mbivya1yk8) 6

[4.1.1 Scenario #1 - Successfully books specific room.](#_gfd7ezsnj28i) 6

[4.1.2 Scenario #2 - Successfully books any room.](#_r3nn9xt9886u) 6

[4.1.3 Scenario #3 - Fully booked](#_94owndjmlb0k) 6

[4.1.4 Scenario #4 - Check in and out](#_6ndcfipvx2s3) 6

[4.2 Use Case Model](#_1pt2wm1ptzzt) 7

[4.3 Use Cases](#_n4mxv0lu1sr1) 8

[**5 Sequence Diagrams**](#_5fmd7v1ig6eu) **12**

[5.1 Book a specific room](#_cawx3plhpmnr) 12

[5.2 Book any room](#_fy0y32rlvh7n) 13

[5.3 No available rooms](#_q7s7hkuhy2go) 14

[5.4 Check in](#_ytwj4j97dpvw) 15

[5.5 Check out](#_skzf8pcjti4h) 16

[**6 State Diagrams**](#_6bq4vt2dnr24) **17**

[6.1 Book a room](#_n9m29i26azpj) 17

[6.2 Book a specific room](#_q8wxudlfaqbh) 18

[6.3 Check availability](#_ajjog67iwpil) 19

[6.4 Check specific room](#_lhkysk2cadj9) 20

[6.5 Cancel booking](#_xzseic3xxd8x) 21

[6.6 Check in](#_ljwh96np0ab3) 22

[6.7 Check out](#_7w48evp4ueow) 23

[6.8 Save customer info](#_9kk3p15ldjeb) 24

[6.9 List customers](#_5r0moxcb58zs) 25

[6.10 Search for customer](#_7ib4qwda5xbv) 26

[**7 Domain Model**](#_6zc8w7v3xh6c) **26**

[7.1 Domain Model](#_xpeoa1vrcsmo) 27

[**8 Non-functional models**](#_1qsw5mrsvpwm) **28**

[8.1 Check room](#_fmsfjtdx8yfp) 28

[8.1.1 Use Case](#_efjj5mg67mzj) 28

[8.1.2 Sequence Diagram](#_apul80tzdmkk) 28

[8.2 Checkout process](#_zchwd47n1rsu) 30

[8.2.1 Use Case](#_i05ia922rboz) 30

[8.2.2 Sequence Diagram](#_y2rlvj7keu1g) 31

[8.3 Check in process](#_jzqiiryii6tn) 32

[8.3.1 Use Case](#_u3qg81x4p8kq) 32

[8.3.2 Sequence Diagram](#_qut1rbzfes7x) 32

# 1 Introduction

## 1.1 Stakeholders

The stakeholders for this project is the Linnaeus Hotel. Which is a company that have two different buildings that are being used as the hotel, one in Kalmar and one in Växjö.

## 1.2 Stakeholder needs

The stakeholders require a software for helping them in their daily work. A software to manage their front-desk activities. They need an efficient system that will improve the quality of guest management.

## 1.3 Purpose

The main purpose of this application is to re-engineer the legacy software that already exists. The system being used right now is a outdated software with poor functionality and generally hard to use and very non-efficient. So this application is created to be able to have a more futuristic application that serves the hotel well and improves their guest management system by smoothing out the process of doing day-to-day front-desk activities.

The application is supposed to save them time and money, by providing a easy to use, time efficient way to enter reservations, check guests in and out etc.

# 2 Scope

This application will allow the people working at the front-desk of the Linnaeus Hotel to manage front-desk activities, such as enter reservations, manage reservations, check guests in and out, as well as keeping information about each guest. This application can be reused for other hotels as well, or if the Linnaeus Hotel wish to broaden their company and buy more buildings, it will be easy to do so.

# 

# 3 Requirements

## 3.1 Functional Requirements

Here is the functional requirements for the software, functional meaning actual functionality requirements. Features that the application should have, such as “Book a room”.

These requirements will be briefly explained - but for a further understanding of them, please see the Use Cases (section 4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | | | |
| F1.1 | Book a room | Rooms should be able to be booked for customers in the system. The rooms have unique numbers. Information should be recorded about the customer. | | | |
| F1.2 | Boom specific room | A specific room should be able to be booked in the system for a customer. Criterias can be number of beds, location, room size, smoking/non-smoking, access to the next room | | | |
| F2.1 | Check availability | The system should be able to check if there are rooms available at the given date and time | | | |
| F2.2 | Check specific room | The system should be able to check if a specific room is available at given date/time | | | |
| F3.1 | Cancel booking | The system should be able to cancel bookings | | | |
| F4.1 | Check in | The system should be able to check in customers | | | |
| F4.2 | Check out | The system should be able to check out customers | | | |
| F5.1 | Save customer information | The system should be able to save information about each customer | | | |
| F6.1 | List customers | The system should be able to list all known customers | | | |
| F6.2 | Search customers | The system should be able to search for customers | | | |
| F7.1 | Print bill | The system should be able to print a bill for the customer | | | |

## 

## 3.2 Quality Requirements

Quality Requirements are also important to have an efficient and smooth running system. We do not have many quality requirements documented, but these are the requested ones. More quality requirements will be improvised and implemented during that stage, to ensure the quality of the software.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Name** | **Description** | | | |
| Q1.1 | Check room | The check for room availability must take less than 2 second | | | |
| Q2.1 | Checkout process | The checkout process must take in average less than 60 seconds to complete | | | |
| Q2.2 | Checkin process | The check in process must take in average less than 60 seconds to complete | | | |

## 3.3 System Requirements

Since this software is developed in Java, the system requirements are not going to contain anything vital. The software will be available through a .jar file, which can be run everywhere.

The software will not have any system dependencies other than having Java installed on the system you’re running the application on.

# 4 Use Cases

## 4.1 Scenarios

### 4.1.1 Scenario #1 - Successfully books specific room.

A guest arrives at the Linnaeus Hotel in Växjö and wishes to book a room for the duration of the 29th of May to 31st of May. The guest walks up to the front desk and informs the clerk working about this and the clerk asks the guest for his/her personal information and if the guest wants any specific rooms. The guest provides his/her information and that he/she wants a single bed room. So the clerk searches the system if there are any available single bed rooms. The clerk is prompted with that room 102 is available and proceeds to input the guests information and offers the guest with the cost. The guest accepts the offer and the clerk books the room for the specific date.

### 4.1.2 Scenario #2 - Successfully books any room.

A guest arrives at the Linnaeus Hotel in Kalmar and wishes to book a room for the night only. The guest tells the clerk that he/she does not have any specific demands for what kind of rooms, only that he/she needs a room for the night. The clerk searches to see if there are any available rooms for the night. The clerk is then prompted with a list of all rooms and offers the guest the cheapest one, due to no quality demands by the guest. The clerk then books the room and proceeds to input the guests information into the system.

### 4.1.3 Scenario #3 - Fully booked

A guest arrives at the Linnaeus Hotel in Kalmar and wishes to book a room for the night only. The guest tells the clerk that he/she does not have any specific demands for what kind of rooms, only that he/she needs a room for the night. The clerk searches to see if there are any available rooms for the night. The clerk is then prompted with an empty list of available rooms. The hotel is fully booked. The clerk informs the guest of this and the guest leaves the hotel.

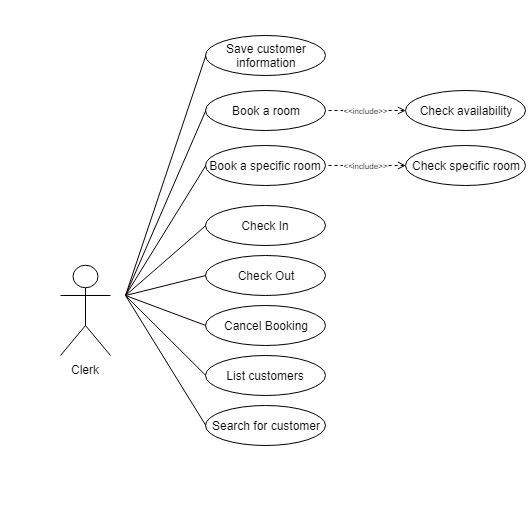
### 4.1.4 Scenario #4 - Check in and out

A guest that have already booked a room arrives at the Linnaeus Hotel in Kalmar. The guest walks up to the front desk and tells the clerk that he/she wishes to check in and then presents the clerk with his/her reservation id. The clerk inputs the reservation id into the system and checks the guest in. The guest is given his/her key and informed of where his/her room is.

The guests duration of stay passes and it is time to check out. The guest walks to the front desk to hand the key back and tells the clerk what room he/she was staying at. The clerk searches for the reservation for that room and then checks it out.

## 4.2 Use Case Model

Below is a figure over the Use Cases, before presenting them. This is to give a broad view of the logic behind them and a way to observe and connect their relationships for better understanding of the flow of the software. All use cases only include one actor, and that is the hotel clerk working at the front-desk. And that is simply because that is the only person who will be interacting with the software itself. The other participants are participating from the outside, interacting with the clerk.



## 

## 4.3 Use Cases

|  |  |
| --- | --- |
| **Requirement** | 1.1 Book a room |
| **Description** | The system should allow for booking of a room. |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. The clerk searches for rooms available for a specific date. 2. The clerk selects any of the available rooms. 3. The clerk inputs information about the person staying at the room, such as guest information and duration of stay. 4. The clerk marks the room as booked. |
| **Post Conditions** | The hotel room should be reserved, meaning no other person should be able to book it for the guests duration of stay. |

|  |  |
| --- | --- |
| **Requirement** | 1.2 Book a specific room |
| **Description** | The system should allow for booking of a specific room. |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. The clerk searches for a room using the specified criteria. 2. The clerk selects any of the room that meets the criteria. 3. The clerk marks the room as booked and inputs information about the guest staying there as well as the duration of stay. |
| **Post Conditions** | The specific hotel room should be reserved, meaning no other person should be able to book it for the guests duration of stay. |

|  |  |
| --- | --- |
| **Requirement** | 2.1 Check availability |
| **Description** | The system should allow for checking if there are any available rooms |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. The clerk inputs the duration of which to check for available rooms. 2. The clerk is presented with a list of rooms available. |
| **Post Conditions** | Rooms available should be presented in a list-view. |

|  |  |
| --- | --- |
| **Requirement** | 2.2 Check specific room |
| **Description** | The system should allow for checking if specific room is available. |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. The clerk inputs the specific room info the search field(s) 2. The clerk is presented with the specific room and its current reservations. (if any) |
| **Post Conditions** | Room information should be presented with a list of all current reservations for the room. |

|  |  |
| --- | --- |
| **Requirement** | 3.1 Cancel Booking |
| **Description** | The system should be able to cancel any current booking. |
| **Actors** | Clerk |
| **Preconditions** | The booking must exist. |
| **Flow** | 1. The clerk searches for the reservation, via either reservation-id or guests personal information. 2. The clerk opens the reservation and marks it as available and removes the current booking. |
| **Post Conditions** | The hotel room must now have the status of available and the guest must not have any current reservation active for that room. |

|  |  |
| --- | --- |
| **Requirement** | 4.1 Check In |
| **Description** | The system should be able to check in customers. |
| **Actors** | Clerk |
| **Preconditions** | The guest must have a room already booked at the hotel. |
| **Flow** | 1. The clerk inputs the reservation id into the search fields. 2. The clerk clicks the button to mark the guest as checked in. |
| **Post Conditions** | The guest must be marked as checked in. |

|  |  |
| --- | --- |
| **Requirement** | 4.1 Check Out & 7.1 Print Bill |
| **Description** | The system should be able to check out customers. |
| **Actors** | Clerk |
| **Preconditions** | The guest must have a room already booked at the hotel and the guest must already have checked in. |
| **Flow** | 1. The clerk inputs the personal information of the guest into search fields and finds the booking and the guests info. 2. The clerk clicks the button to mark the guest as checked out. 3. The clerk is prompted with a PDF Bill containing information about the reservation & guest. |
| **Post Conditions** | The guest must be marked as checked out and the room must be set to available again. As well as having the bill presented for the clerk to print. |

|  |  |
| --- | --- |
| **Requirement** | 5.1 Save customer information |
| **Description** | The system should be able to save information about each customer. |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. Clerk is given a guests information from the guest 2. The clerk inputs the personal information of the customer and saves it. |
| **Post Conditions** | The guest must have a “profile” with information about him/her that shows current reservation and relevant info. |

|  |  |
| --- | --- |
| **Requirement** | 6.1 List customers |
| **Description** | The system should be able to list all known customers. |
| **Actors** | Clerk |
| **Preconditions** | No preconditions |
| **Flow** | 1. The clerk clicks the view all customers window. 2. The clerk should be presented with a list of all known customers. |
| **Post Conditions** | The list should be displayed to the clerk and it should contain all customers in the system. |

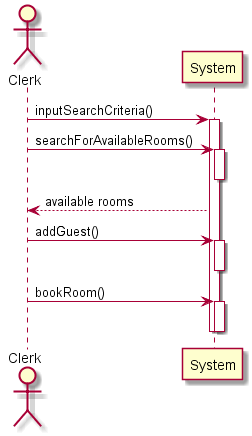
|  |  |
| --- | --- |
| **Requirement** | 6.2 Search for customer |
| **Description** | The system should be able to search for a specific customer. |
| **Actors** | Clerk |
| **Preconditions** | Customer is in the system. |
| **Flow** | 1. The clerk navigates to the view customers list. 2. The clerk inputs a customer's name and/or personal number into a search field. 3. The clerk should be presented with information about the customer. |
| **Post Conditions** | The information should be accurate and about the specified customer. |

# 5 Sequence Diagrams

Here we’ll present the sequence diagrams derived from the scenarios. These are to show the functionality of the system on a high-level.

## 5.1 Book a specific room

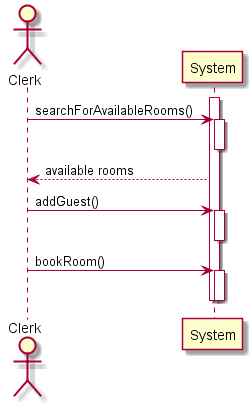
This shows the how the functionality of booking a specific room is supposed to happen. It starts with the clerk putting in the search criteria for the specific room into the system, and then telling the system to do the search. The system searches for any available rooms with that criteria and returns a list with them.  
The clerk then adds the guest into the system and books the specific room.



## 

## 5.2 Book any room

This shows what is supposed to happen when booking any room. The clerk asks the system to perform a search without criteria. And the system performs the search and then returns a list of all available rooms to the clerk. The clerk then adds the guest that wishes to book the room to the system then books the room itself.

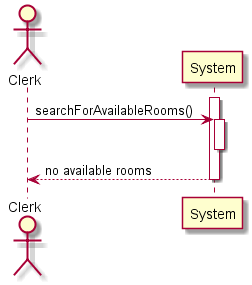


## 

## 5.3 No available rooms

This shows what happens when the system can’t find any available rooms. If it is fully booked or if there are no rooms matching the criteria.

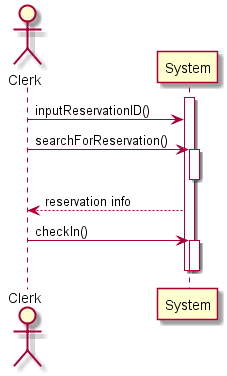
The clerk asks the system for a list of all available rooms and the system performs the search and then returns an empty list.



## 

## 5.4 Check in

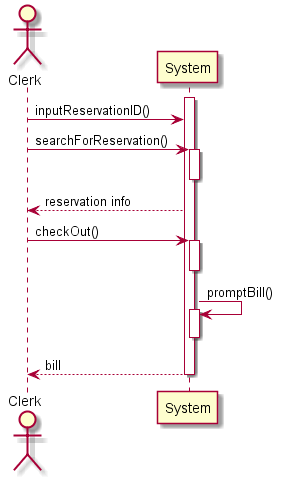
This shows what happens when the clerk checks a guest into the system. It starts by the clerk receiving the reservation ID from the guest and then inputs it into the system and tells it to search for the reservation. The reservation info is then presented by the system and the clerk receives the option to check the guest in.



## 

## 5.5 Check out

This shows what goes on in the system when a clerk checks a guest out. The functionality is very similar to what goes on when checking in. The clerk receives the reservation ID and then inputs into the system and the system performs a search and presents the reservation info to the clerk. The clerk then gets the option to check the guest out. If the clerk checks the guest out, the system generates a bill for the clerk and then presents it.



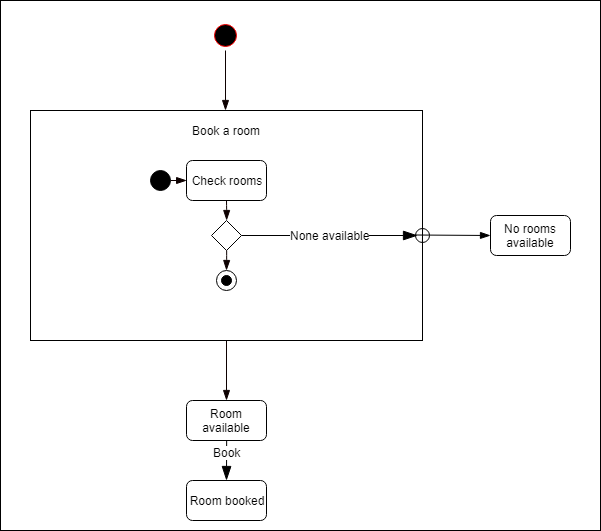
# 

# 6 State Diagrams

Here below we’ll present some state diagrams. These are there to show the behaviour of the application if certain conditions are or aren’t met.

## 6.1 Book a room

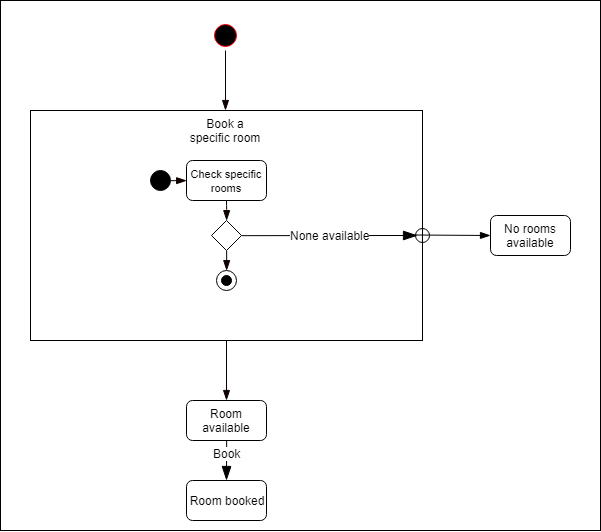
The first diagram is book a room, which shows how the application behaves when booking a room. It starts by the user requesting to book a room using the UI. And then the application checks whether or not there are any rooms available. If rooms are available, the user is shown the available rooms and have the option to book one. If no rooms are available the user is informed.



## 

## 6.2 Book a specific room

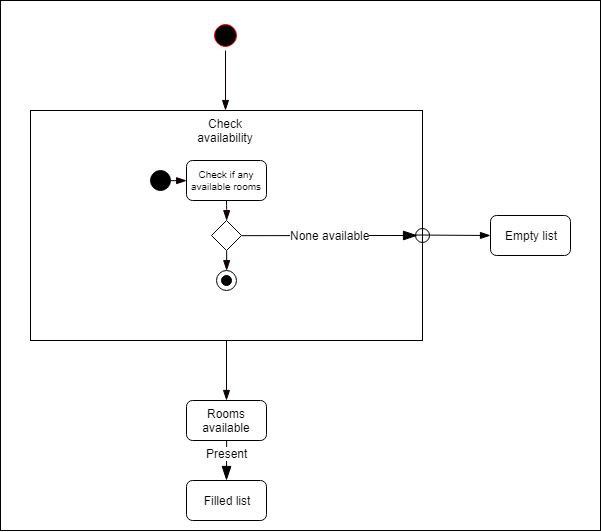
This diagram shows almost the same as 5.1 Book a room. The key difference here is that the user requests to book a specific room, e.g Room 101 or a Suite. And then the flow is similar. The application checks if there are any rooms available that meets the criteria. If there is, give the user the option to book the room. Else, inform the user that no rooms are available.



## 

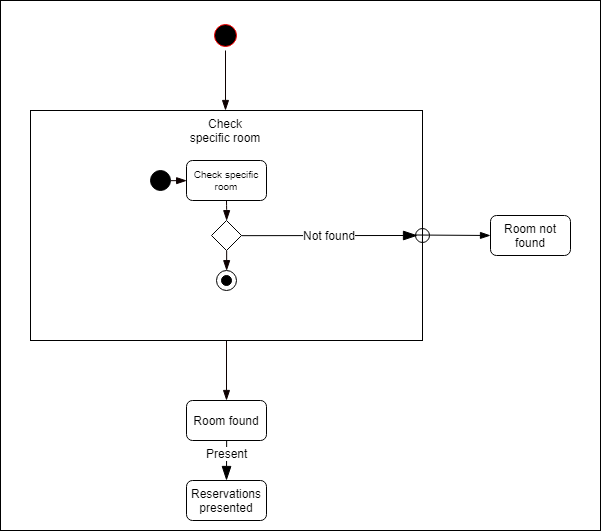
## 6.3 Check availability

Here the behaviour of checking for availability is shown. It starts by the user checking if there is any available rooms and then depending on whether or not there is, the application returns either an empty list or a list filled with the available rooms.



## 

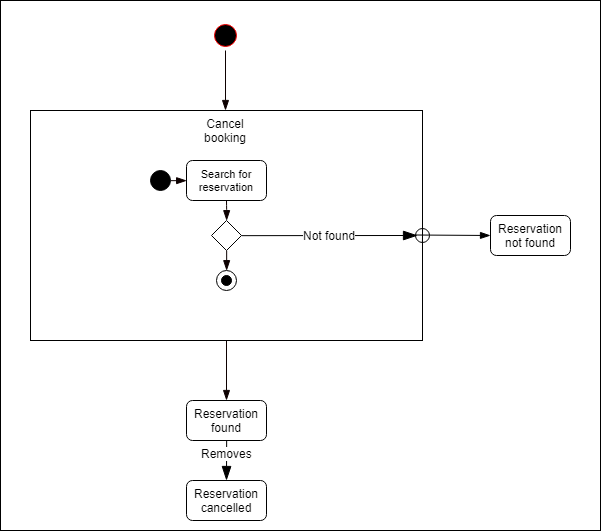
## 6.4 Check specific room

This one is similar to the previous one, 5.3. The difference is that this checks the availability of a specific room, e.g Room 101. If the specific room is available, the application presents the room and the reservations for that room. Else it informs that the room was not found. 

## 

## 6.5 Cancel booking

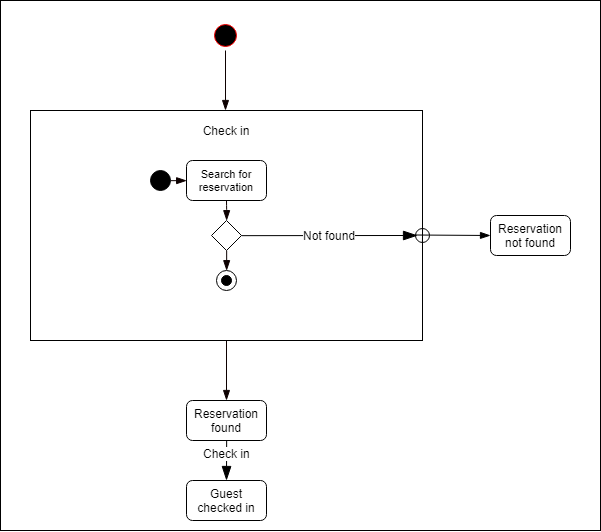
This diagram shows the behaviour when you’re cancelling a booking, meaning you delete it. It starts by the user searching for a specific reservation via using its reservation ID. If the reservation isn’t found, the user is shown that it wasn’t found. Else if it is found, the reservation and its information is presented and the user has the option to cancel it.



## 

## 6.6 Check in

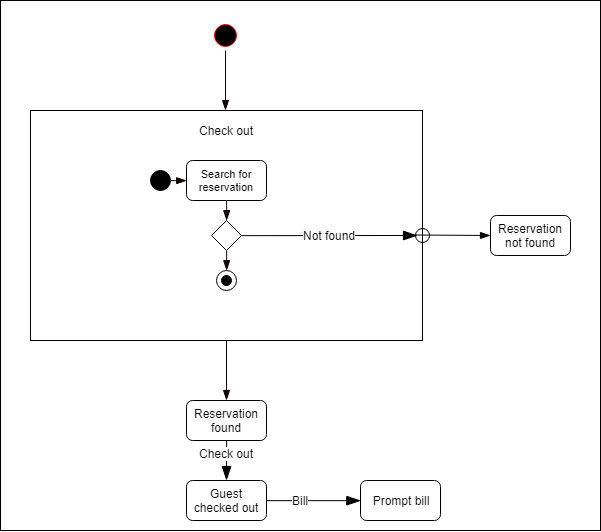
This diagram shows the behaviour when you’re checking a guest in. The user is given the reservation id by the guest and searches for the specific reservation with the use of it. If the reservation isn’t found, the user is indicated that this is the case. Else if it is found, the user is presented with the information about the reservation and given the option to check the guest in.



## 

## 6.7 Check out

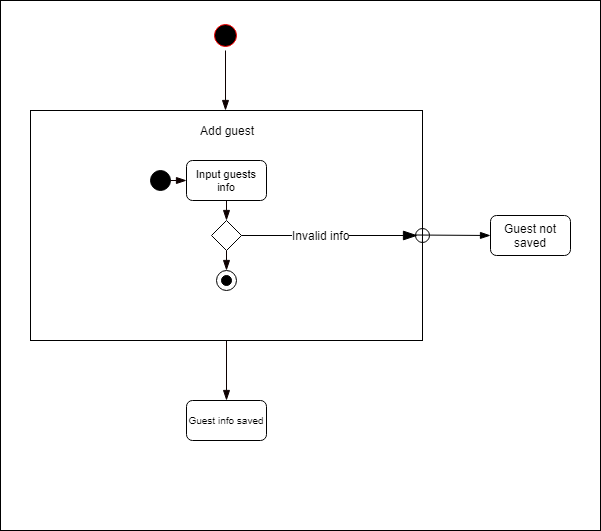
This one is similar to the check in diagram. It starts the same. The user is given a reservation id and uses it to find the reservation, if there is any. And if found the user is given the option to check the guest out. If the guest is successfully checked out, the user is prompted with the guests bill, in-case the user wishes to print the bill and hand it to the guest.



## 

## 6.8 Save customer info

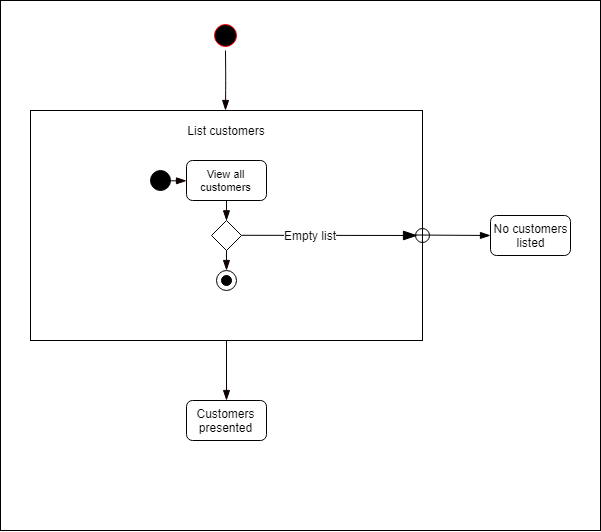
This diagram shows the behaviour of adding a guest and saving his/her information. It starts by the user receiving the guests personal information and inputting it into the application. And then if the information is valid the guest is created in the system and saved for future use. If it is invalid, the user is informed and the guest is not saved.



## 

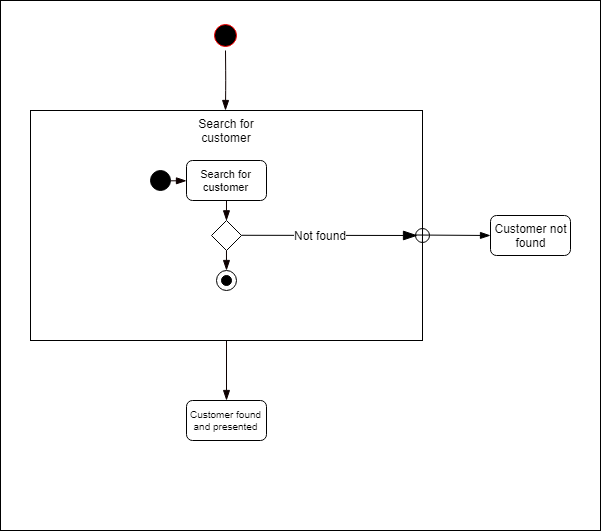
## 6.9 List customers

This shows the behaviour of the application when the user requests to see a list of all customers. It starts by the user sending the request to view all current customers. If there are any customers the customers are shown in a list to the user. If the list is empty, it means that there are no customers.



## 

## 6.10 Search for customer

This shows the behaviour of the application when it searches for a specific customer. It starts with the user typing in the search criteria, e.g a first name, last name etc. And then whether or not if the customer is found, show the customer in a list or don’t. 

# 7 Domain Model

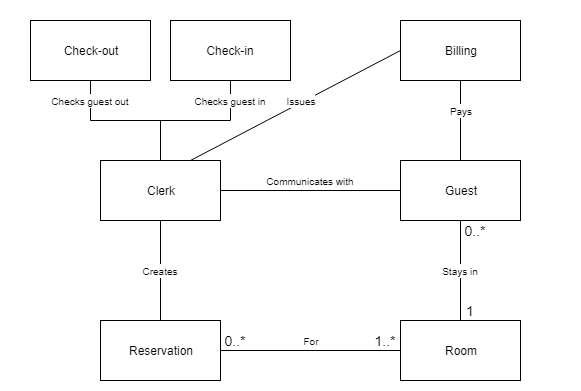
Here is a visual model of the system. This model shows a abstract way of describing the application. It is more or less showing the real world how it would work. The purpose of this model is to get a better understanding of the software being built without going into too much detail and complexity.

It shows the flow of how the hotel should work - the clerk handles all the management and the guest handles paying for his/her stay.

## 7.1 Domain Model

This shows how the application works from a real life perspective, sort of.

You can see who does what in the part of the application. And how each of the part is related to each other. This is not exactly how the application looks and works, but it gives an overall view of how it is supposed to work.



# 

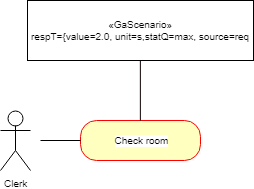
# 8 Non-functional models

## 8.1 Check room

Below will be the models for the non-functional requirement Q1.1, check room.  
The requirement states “The check for room availability must take less than 2 seconds.”

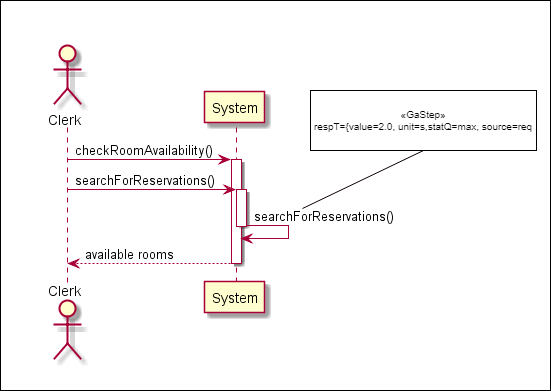
### 8.1.1 Use Case

Here is the use case-model for the check room. This shows that the whole check for rooms max duration must be 2.0 seconds for the requirement to be fulfilled.



### 8.1.2 Sequence Diagram

Here is the sequence diagram for the check room requirement. It shows that the step to search for the reservation and return it must take at max 2.0 seconds, else the requirement is not fulfilled.



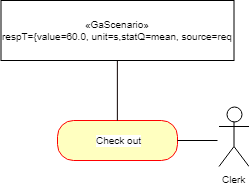
## 

## 8.2 Checkout process

Below will be the models for the non-functional requirement Q2.1, checkout process..  
The requirement states “The checkout process must take in average less than 60 seconds to complete.”

### 8.2.1 Use Case

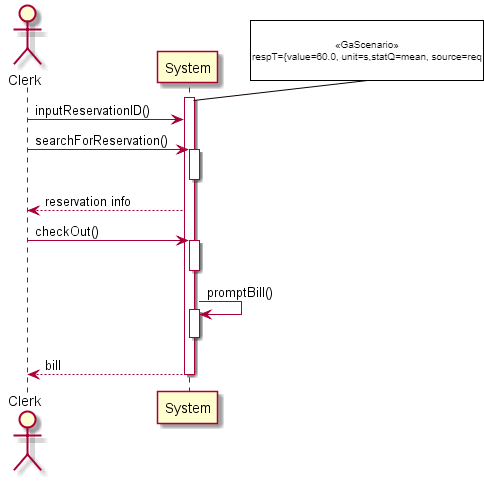
Here you can see the use case-model for the requirement. It shows that the whole scenario must have the average of 60 seconds else it isn’t fulfilled.



### 

### 8.2.2 Sequence Diagram

Below here is the sequence diagram showing the non-functional requirement. The whole process must take at average less than 60 seconds for the requirement to be fulfilled, which means that the clerk has to receive the id, input it, search for the reservation, confirm that it is the correct one, then check the guest out and have the bill ready. If it takes longer than 60 seconds, the requirement is not fulfilled.



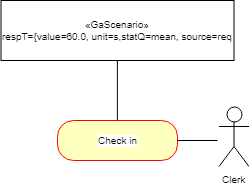
## 

## 8.3 Check in process

Below will be the models for the non-functional requirement Q3.1, check in process..  
The requirement states “The check in process must take in average less than 60 seconds to complete.”

### 8.3.1 Use Case

Here is the use case-model showing the non functional requirement 3.1. It shows that the whole check in process must take less than 60 seconds on average. If it doesn’t the requirement is not fulfilled.



### 8.3.2 Sequence Diagram

Below is the sequence diagram showing the non-functional requirement. The whole process must take less than 60 seconds on average. Meaning that the clerk has to receive the guests id, input it and search for the reservation. Confirm that it is the correct reservation, and then check the guest in. If it takes longer, the requirement is not fulfilled.

