Linnéuniversitetet
Kalmar Växjö

2DV603

REQUIREMENTS ENGINEERING

Authors: - Alaa Alwan
- Mohammed Basel Nasrini
Team: 13
Supervisor : Mauro Caporuscio
Semester: Autumn 2018
Subject: Re-engineering Legacy Software
Course Code: 2DV603

Content

1. Requirements Engineering

1.1 General knowledge about the domain

The hotel contains rooms in which guests can stay. The room's rate differs according to the room's quality. Usually, the hotel's clerk does the process of the reservation. When a guest wants to make a reservation, he should have a contact with the clerk or he can do that online. If the online alternative is not available, the guest should visit the hotel or he should have a contact with the clerk by a phone or E-mail or any another available way. The guest should provide specific personal information in addition to the reservation's details like the room's details, the number of guests, the check in and out date...etc. The guest can cancel the reservation at any time but some fees may be charged if the cancelation is done too late (usually the cancellation is free if the cancellation is done before more than 24 hours).

1.2 The World and the Machine

World phenomena:

- The guest needs a room.
- The discussion between the guest and the clerk.
- The calls between the clerk and the manager.
- Additional services provided to the guest.

Shared phenomena:

- Room allocation.
- Guest’s personal-information.
- Reservation's details
- Notes encoding.
- Additional services encoding and billing.

Machine phenomena:

- List the available rooms.
- Create a new reservation res=new reservation (Guest, Room, Period).
- Database queries.

1.3 Goal

Create a system to manage the front-desk activities of the "Linnaeus Hotel". For every guest reserving room, a room should be reserved and check guests in and out of the hotel.

1.4 Domain assumptions

- For every guest call, details about the check in and check out date are correctly registered.
- For every guest call, details about the quality level an properties (Size, number of the bed, smoking status, view, and number of guest) of the wanted room are correctly registered.
- For every guest call, details about the guest (First and last name, date of birth, passport number, telephone number, and may more details like E-mail, credit card, and specific description) are correctly registered.

1.5 Requirement

When a guest want to reserve a room, room is reserved. The program should list the available rooms within a 2 seconds according to information. The customer can request a specific room, this can be allocated in advance at the discretion of the manager. A reservation can be canceled at any time but some fees (a percentage of the room price) may be charged if the cancelation is done too late. When a guest want to check out, the whole procedure must take in average less than 60 seconds to complete.

1.6 Defining the Problem and the Scope

The current used system by the hotel is paper-based. This leads to bad service and west time and money. The hotel manager want to change the current system to automated system to save money and to serve the guests better.

The software is to be used in hotels by the hotel clerk. The clerk will be able to create, cancel, update and list reservations. He will be able to check guests in and out of the hotel.

1.7 Actors and Tasks

- The clerk: will be able to create, cancel, update and list reservations. He will be able to checks guests in and out of the hotel. Furthermore, the clerk will be able to manage the customers' derails.
- PDF reader: Will be used by the program as a third party to view the bill as PDF and print it.

1.8 Environment

A front-Desk in a hotel with PC running Windows. Later, the hotel manager wants to use another PC running Mac OS X. Therefore, the application must be compatible with both Windows and Mac OS X.

The programing language that will be used to implement the software is JAVA. Therefore, the program can work on all platforms needed.

There is no similar software in the place. The work in the hotel is completely paper-based. Therefore, the implementing will starts from the scratch.

1.9 Participants are involved in reservation a room

- The clerk
- The guest
- The manager

1.10 Interviewing

The interview was made with the hotel manager and the clerk in Växjö:

1 How many computers the software will be installed on? If more than one computer, do you need a shared database?
Ans: Currently, the hotel has two location (Växjö and Kalmar). At each location, the software will be installed on one computer (clerk's computer). The clerk can make reservation in both locations. Therefore, it is the database should be shared and the clerk in one location should know about rooms' status in another location.

2 What the operating system of the front-desk computer?
Ans: Currently, the operating system in Växjö is windows 10, while in Kalmar is Mac OS X. Therefore, the software must compatible with both.

3 Should the software saves each new change directly to the database, or the software will not save the new changes until the user presses "save"?
Ans: The information should be saved to the data base automatically after the end of each process (e.g. Create new reservation, Edit customer’s information, checking or check out…etc.)

4 If there is just one guest, should the system views in the search results the rooms that have more than one bed?
Ans: yes, the guest can freely book any type of room regardless the number of guests.

5 If the guest wants to book a specific room, should the program record the call between the clerk and the manager to save the manager's decision for the future?
Ans: No.

6 What the fees for the cancelation?
Ans:

- The cancelation is before 24 hours.
- If less than 24 hours, the guest will be billed for 1 day.
- If the cancelation during stay at the hotel (after the check in day but before the checkout day), the guest will be billed from the first day to the next day of the cancellation.

7 What the software will do if the guest did not come in the first day?
Ans: The software should cancel the reservation and the guest will be billed for the first two days.

8 How the program will decide the quality level and the rate?

Ans: the minimal quality level should be 3 with rate: 30 Dollar. Then each extra point costs 10 dollar. The software increases the quality level as the following extra points:

- Size:
    - Small: 0
    - Medium: 1
    - Large: 2
- Adjoined rooms:
    - 2 adjoined room: 2
    - 3 adjoined room 4

Generally (each extra adjoined room gives 2 points extra. Currently, the hotel has maximum 3 adjoined rooms).

- The view:
    - No view: 0 point
    - View: 1 point
- The balcony:
    - No balcony: 0
    - Balcony: 1
- Air conditioner:
    - No air conditioner: 0
    - Air conditioner: 1
- Max number of persons:
    - Each person: 1

9 Are there other services will affect the final bill?

Ans: Yes, the clerk should be able to add any extra service to the reservation during the guest's stay (e.g. buy service).

10 Should the smoking status affect the quality level?

Ans: No.

11 What the form of bill?

The clerk should be able to view the bill as PDF file.

12 What is the stakeholders' vision for the future?

Ans:

- The ability to book a room online by the guest itself and without any help from the clerk.
- Online reservation can be done on the hotel's website or via a third party (like booking.com) by provides the needed API to the third party.
- All calls between the clerk and another person such as the guest or the manager should be done via the software and it should be recorded.
- The clerk needs to log in to the software before he starts to use it.
- The hotel's manager could become a stakeholder with additional functions like register, update, and delete a clerk and rooms.

1.11 List of few competing software

- execu/Suite PMS:
    - Online hotel reservation system. Therefore, hotel should not pay fees to a 3rd party service when customers make reservations on hotel’s website.
    - Real time availability, There is no chance of overbooking.
- ASI FrontDesk:
    - Powerful graphical user interface gives color and image coded status in Room.
- Littlehotelier:
    - Easily create phone and walk-in bookings.
    - In a few clicks, you can drag and drop reservations.
- Cloudbeds:
    - Automatically update the availability through the booking engine.
    - Check-in and check-out guests through a beautiful drag-and-drop interface.

2. Scenarios

The mains scenarios are the reservation scenario, checkout scenario, checkout scenario and register a new room. All use cases will be built about these scenarios.

2.1 Reservation Scenario

The **guest** wants to reserve a room in the hotel. The **guest** makes a contact with the hotel's **clerk** to reserve a room. The **clerk** asks the **guest** about the number of guests, desired room specifications and the check-in and check-out date. The **clerk** inputs the information to the software and presses search. The software will list the available rooms within 2 seconds according to the input details. The **clerk** selects the suitable room and presses reserve (The guest can ask to reserve a specific room, this can be allocated in advance at the discretion of the **manager**). The system will ask to the personal information to the **guest**. The clerk asks the **guest** about the necessary information and then the **clerk** inputs the information to the system and then presses confirm. At this point, the reservation is done.

2.2 Check-in Scenario

The **guest** wants to check in. The **clerk** asks the **guest** to the reservation's details. The **clerk** search in the system for the corresponding reservation. The system views the reservation. The **clerk** changes the status of the reservation to checked-in.

2.3 Check-out Scenario

The **guest** wants to check out. The **clerk** asks the **guest** to the room number. The **clerk** search in the system for the corresponding reservation. The system views the reservation. The clerk chooses to check out the reservation. The program views the bill. The clerk prints the bill. The **guest** pays the rate. The **clerk** changes the status of the bill to paid . The whole checkout procedure must take in average less than 60 seconds to complete.

3. Requirements

3.1 Functional requirements

- **The clerk:**

1. Choose the location (Växjö or Kalmar)
2. Rooms:
   2.1. List all rooms.
   2.2. View an existing room's details.
   2.3. Search for rooms with specific details.
3. Reservations Management:
   3.1. Create Reservation.
   3.2. Search for existing reservations and view reservation’s details in a table.
   3.3. Edit a pending reservation.
   3.4. Cancel a pending reservation (Changes the reservation’s status and view the bill).
   3.5. Delete a pending or checked out reservation (Delete the reservation completely without bill).
   3.6. Add service to a reservation.
4. Customers Management (*Creating new customer is result for creating new reservation*):
   4.1. Search for existing customers.
   4.2. Edit an existing customer’s information.
   4.3. Delete an existing customer (that does not has a pending or checked in reservation).
5. Bills Management:
   5.1. Search for existing bills.
   5.2. View Bill as PDF.
   5.3. Print bill.
   5.4. Mark as paid.
6. Check in.
7. Check out.

3.2 Non-Functional Requirements

1. The software must be compatible with both Windows and Mac OS X.
2. Search for rooms with specific details must take less than 2 seconds to complete.
3. The whole checkout procedure must take in average less than 60 seconds to complete.

4. Use Cases and Diagrams

- **Use case #1**

Use case #1
Actors: Clerk
Description: Choose the location
Pre-Conditions: Hotel client exists on the computer
Main flow: 1. The clerk open the hotel clien program 2. The program asks the clerk to choose the location. 3. The clerk choosee the location and confirms. 4. The program save the location and continues loading.
Post-Condition: The location defined.
Alternative flow:

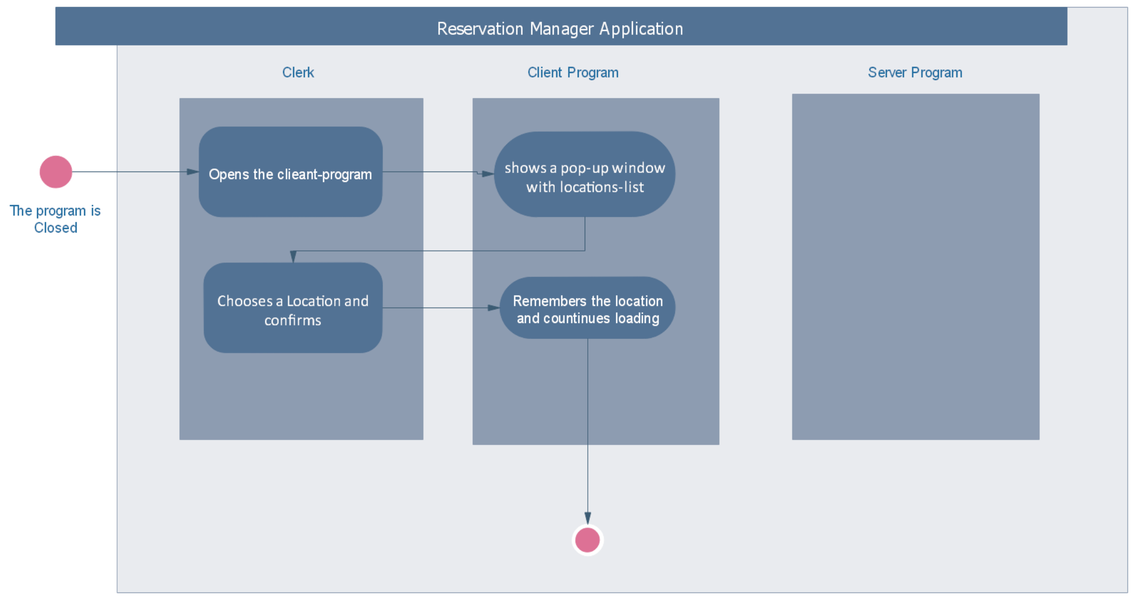
Reservation Manager Application
Clerk
Client Program
Server Program
The program is Closed
Opens the clieant-program
shows a pop-up window with locations-list
Chooses a Location and confirms
Remembers the location and countinues loading

FIGURE 1 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF THE CHOOSE LOCATION USE CASE

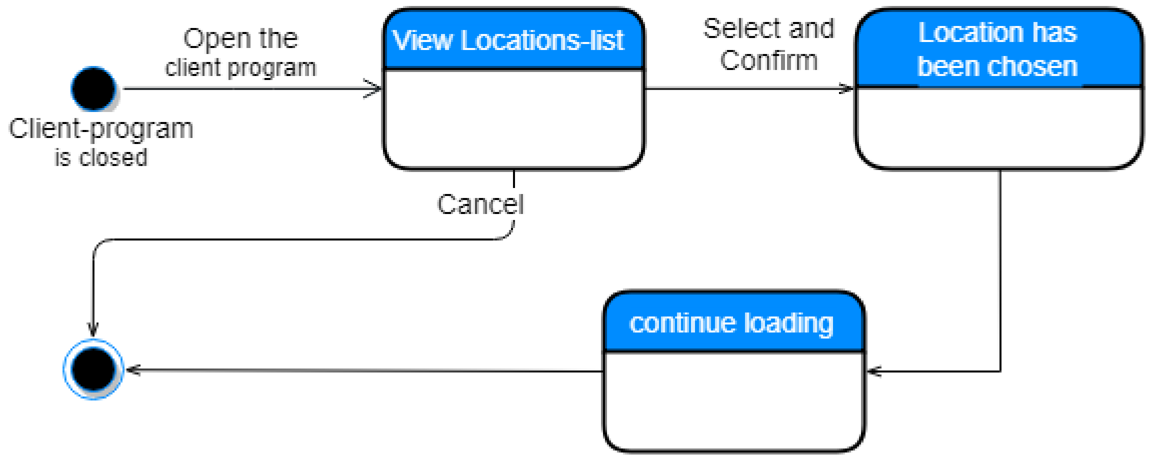

FIGURE 2 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO CHOOSE LOCATION USE CASE

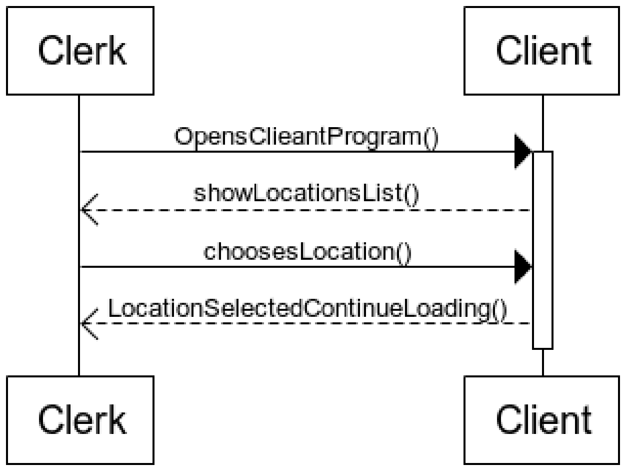

FIGURE 3 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM CHOOSE LOCATION USE CASE

- **Use case #2.1**

Use case #2.1
Actors: Clerk
Description: List all rooms
Pre-Conditions: - Use case 1
Main flow: 1. The clerk chooses to view the room list (Deafalt when the program starts)   2. The program shows all rooms in the main window.
Post-Condition: The program displays all rooms.
Alternative flow:

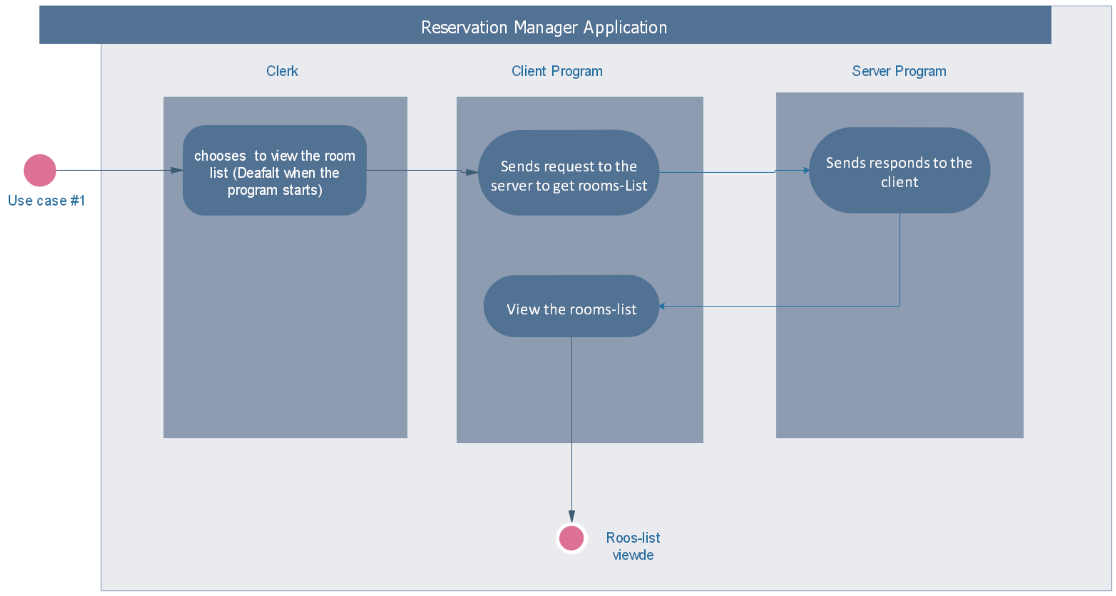
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #1
chooses to view the room list (Deafalt when the program starts)
Sends request to the server to get rooms-List
Sends responds to the client
View the rooms-list
Roos-list viewde

Figure 4 - UML Activity Diagram Represents The Actions Steps Of The List all Rooms Use Case

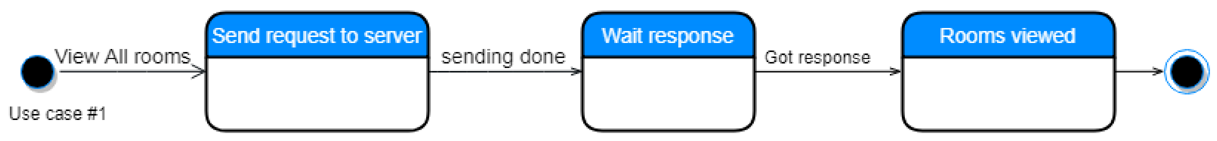

Figure 5 - UML State Machine Diagram Represents the Changes In the Status Of the System Over Time In Response To List All Rooms Use Case

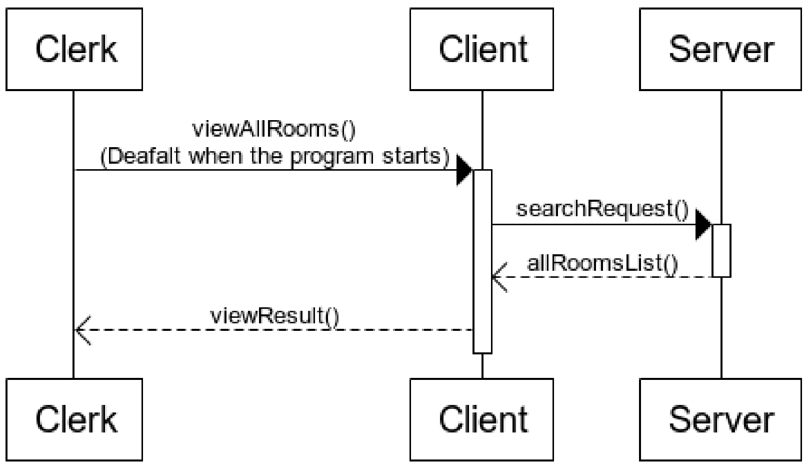

Figure 6 - UML Sequence Diagram Represents The Actors and the Interactions Exchanged Between Them to Perform List All Rooms Use Case

- **Use case #2.2**

Use case #2.2
Actors: Clerk
Description: View room's details
Pre-Conditions: - Use case 2.1
Main flow: 1. The clerk selects a room and chooses to view the details.   2. The program shows the selected room's details.
Post-Condition: Room's details are viewed.
Alternative flow:
Special Requirements:

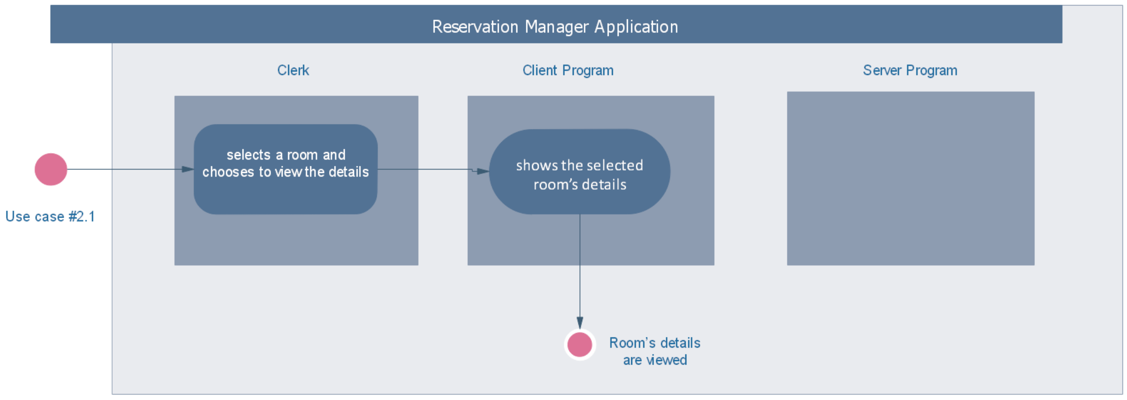
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #2.1
selects a room and chooses to view the details
shows the selected room's details
Room's details are viewed

FIGURE 7 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF THE VIEW ROOM'S DETAILS USE CASE

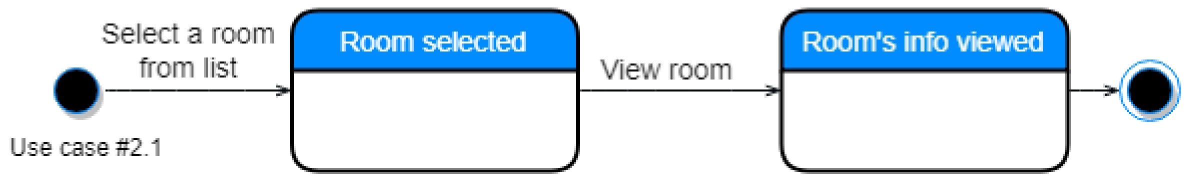

FIGURE 8 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO VIEW ROOM'S DETAILS USE CASE

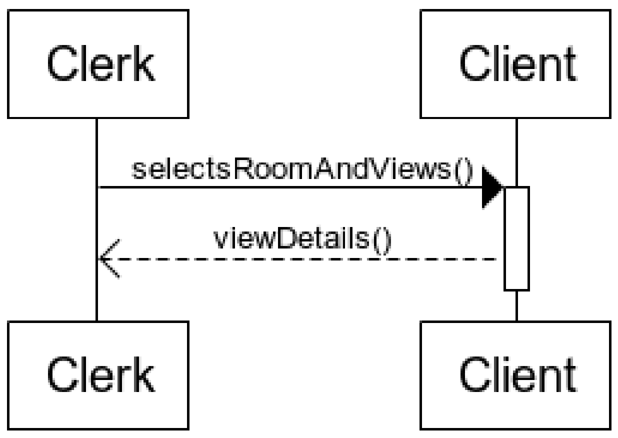

FIGURE 9 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM VIEW ROOM'S DETAILS USE CASE

- **Use case #2.3**

Use case #2.3
Actors: Clerk
Description: Search for rooms with specific details
Pre-Conditions: - Use case 1
Main flow: 1. The clerk chooses to search rooms 2. The program asks the clerk to input the room‘s details. 3. The clerk inputs the room’s details and confirms. 4. The program shows the rooms that match the search inputs.
Post-Condition: The program displays the rooms that match the search inputs.
Alternative flow: **4.a The system did not find a room matchs the details.** 4.a.1 The system show message that there is no room matches the search inputs. 4.a.2 Continue to 2
Special Requirements: The software views the available rooms within 2 seconds

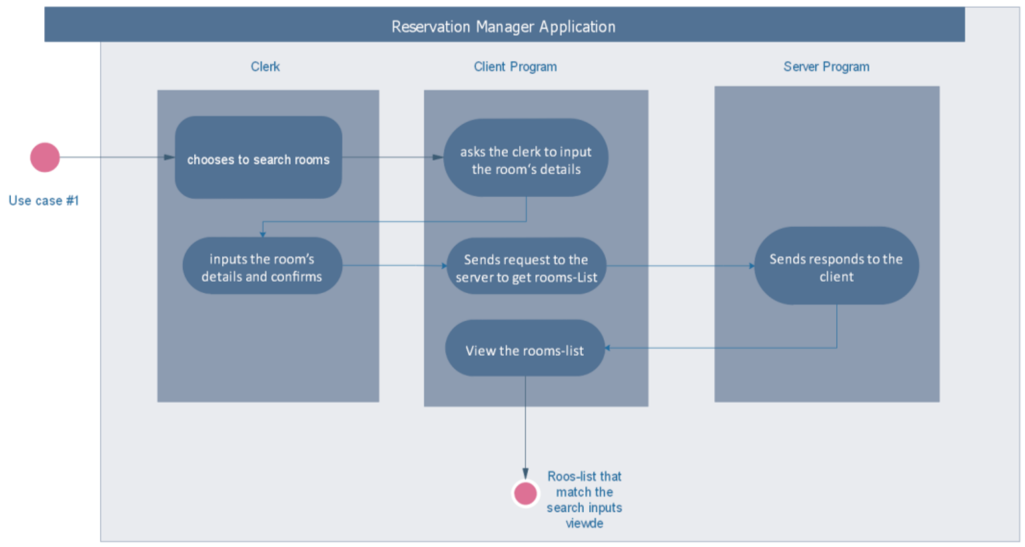
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #1
chooses to search rooms
asks the clerk to input the room's details
inputs the room's details and confirms
Sends request to the server to get rooms-List
Sends responds to the client
View the rooms-list
Roos-list that match the search inputs viewde

FIGURE 10 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF THE SEARCH FOR ROOMS USE CASE

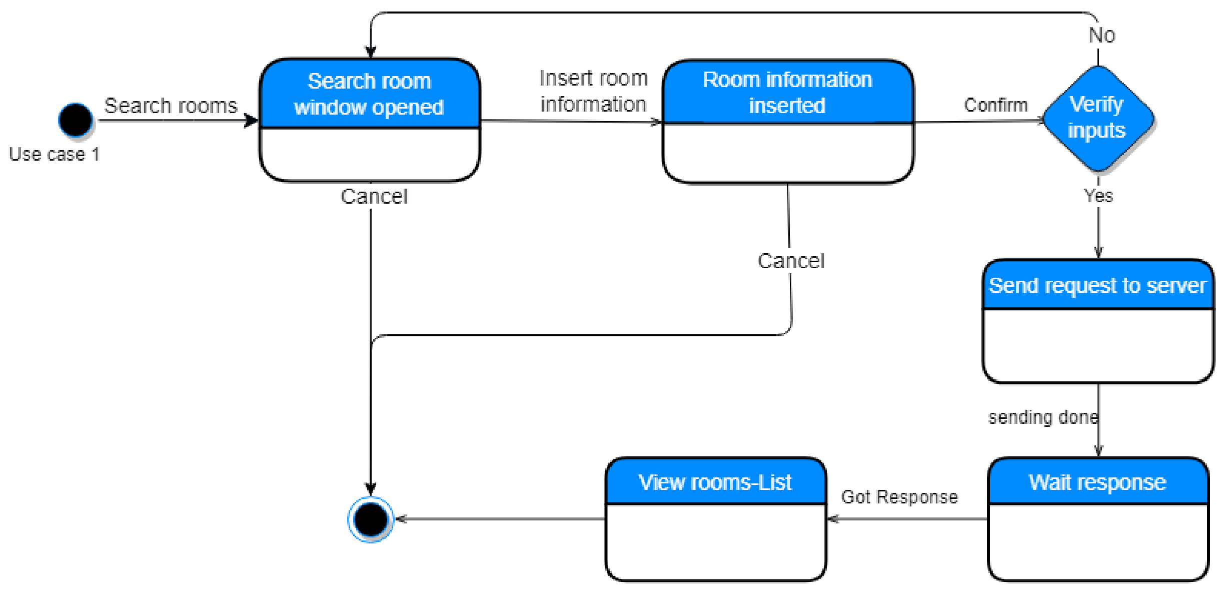

FIGURE 11 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO SEARCH FOR ROOMS USE CASE

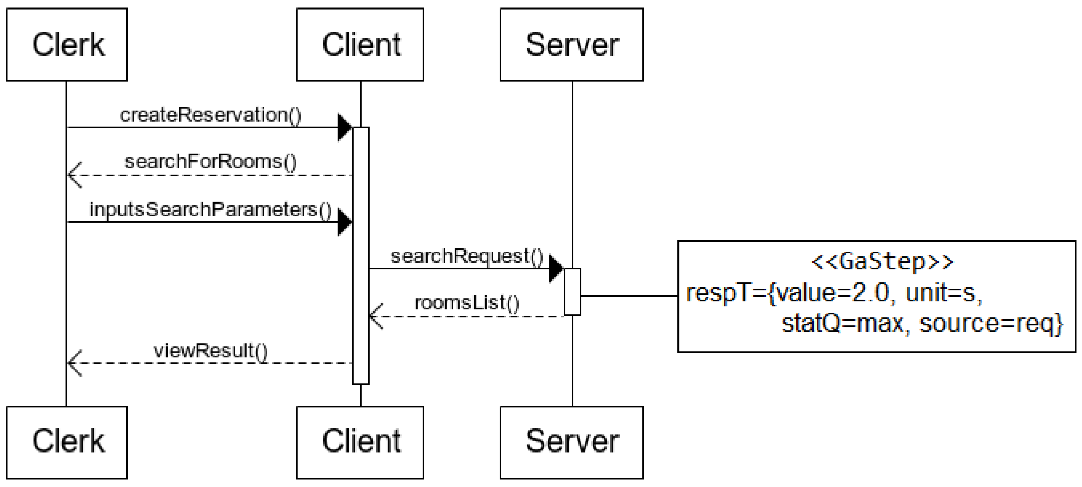

FIGURE 12 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM SEARCH FOR ROOMS USE CASE

- **Use case #3.1**

Use case #3.1
Actors: Clerk
Description: Create new reservation
Pre-Conditions: Use case 2.3
Main flow: 1. The clerk selects a room and chooses to reserve it. 2. The program asks the clerk to input customer's details. 3. The clerk inputs the customer's details and confirms. 4. The program create new customer 5. The system reserves the room.
Post-Condition: New reservation is created.
Alternative flow: **3.a The customer selects an existing customer.** 3.a.1 The program view the customer's details. 3.a.2 The clerk ensure and edit the customer's details. 3.a.3 The the program update the customer's details. 3.a.4 Continue to 5

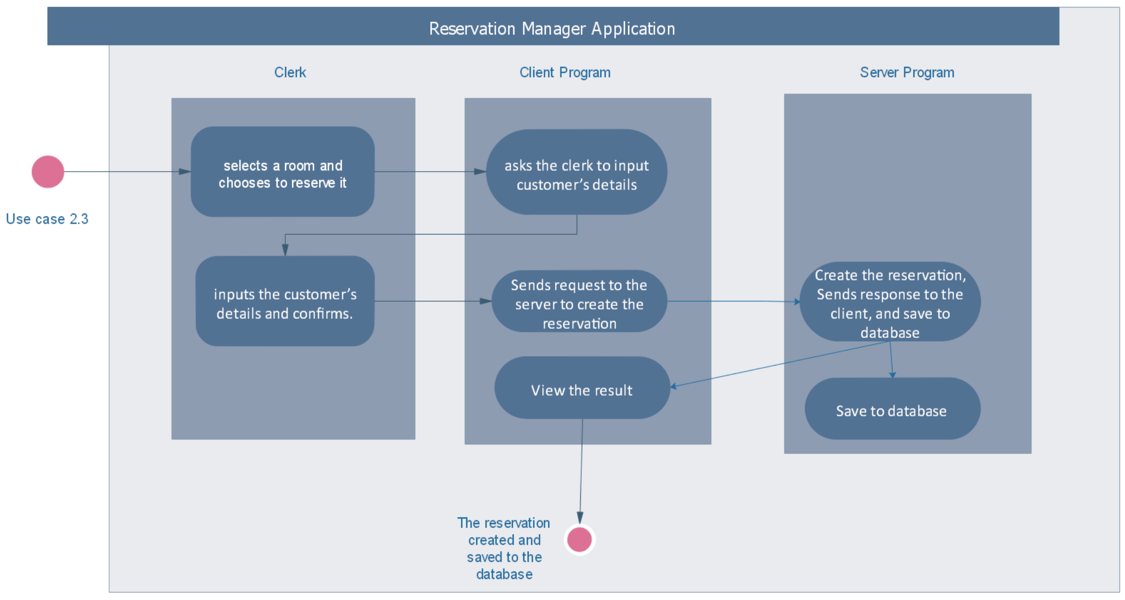
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 2.3
selects a room and chooses to reserve it
asks the clerk to input customer's details
inputs the customer's details and confirms.
Sends request to the server to create the reservation
Create the reservation, Sends response to the client, and save to database
View the result
Save to database
The reservation created and saved to the database

FIGURE 13 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF THE CREATE NEW RESERVATION USE CASE

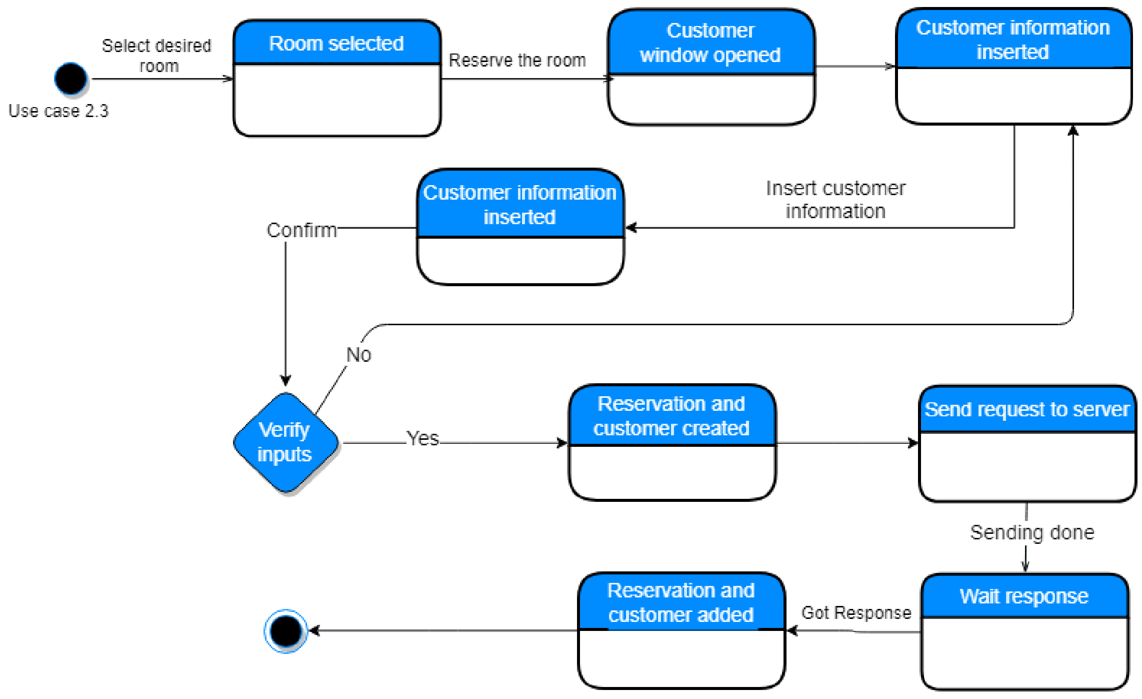

FIGURE 14 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO CREATE NEW RESERVATION USE CASE

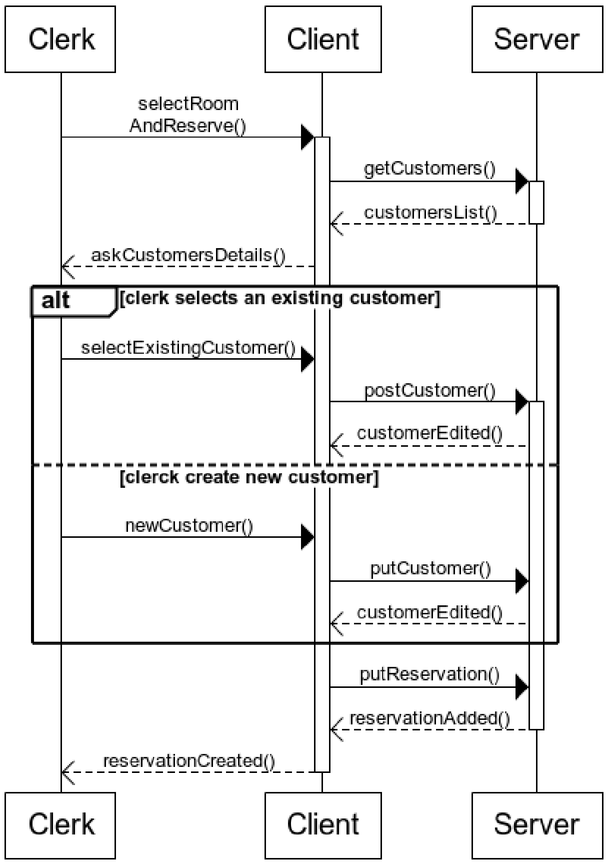

FIGURE 15 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM CREATE NEW RESERVATION USE CASE

- **Use case #3.2**

Use case #3.2
Actors: Clerk
Description: Search for existing reservations and view details
Pre-Conditions: - Use case 1. - Some reservation are already registered.
Main flow: 1. The clerk chooses "Search reservation " 2. The program asks the clerk to input the reservation's details. 3. The clerk inputs the reservation's details and confirms. 4. The program shows the reservations that match the search inputs (Table with their details).
Post-Condition: The program displays the reservation that match the search inputs.
Alternative flow: **4.a The system did not find a reservation matchs the details.** 4.a.1 The system show message that there is no reservation matches the search inputs. 4.a.2 Continue to 2

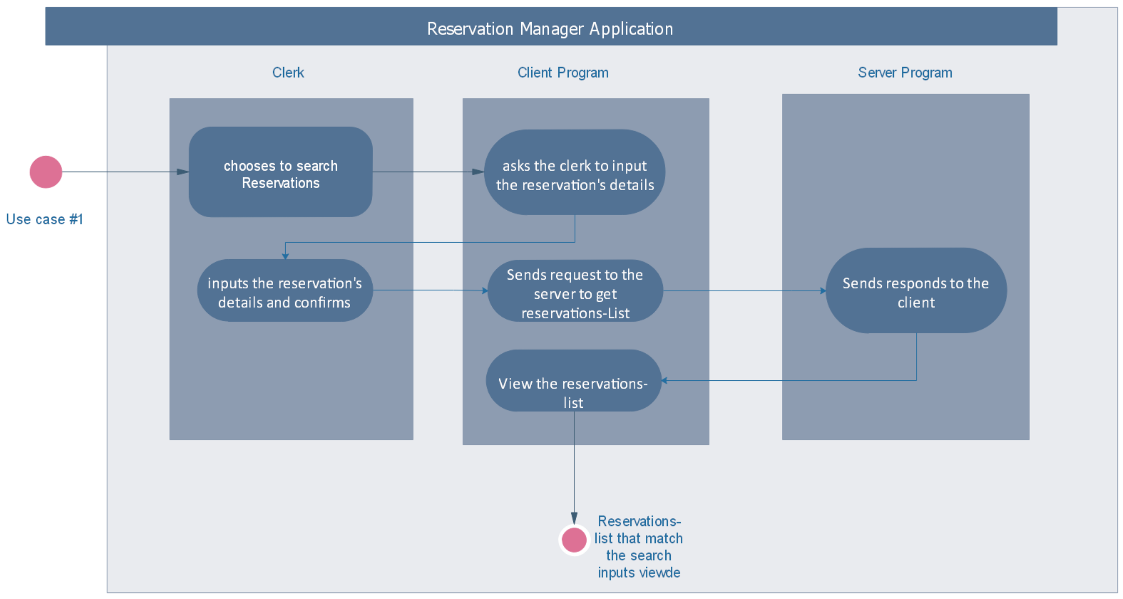
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #1
chooses to search Reservations
asks the clerk to input the reservation's details
inputs the reservation's details and confirms
Sends request to the server to get reservations-List
Sends responds to the client
View the reservations-list
Reservations-list that match the search inputs viewde

FIGURE 16 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF SEARCH FOR EXISTING RESERVATION AND VIEW DETAILS USE CASE

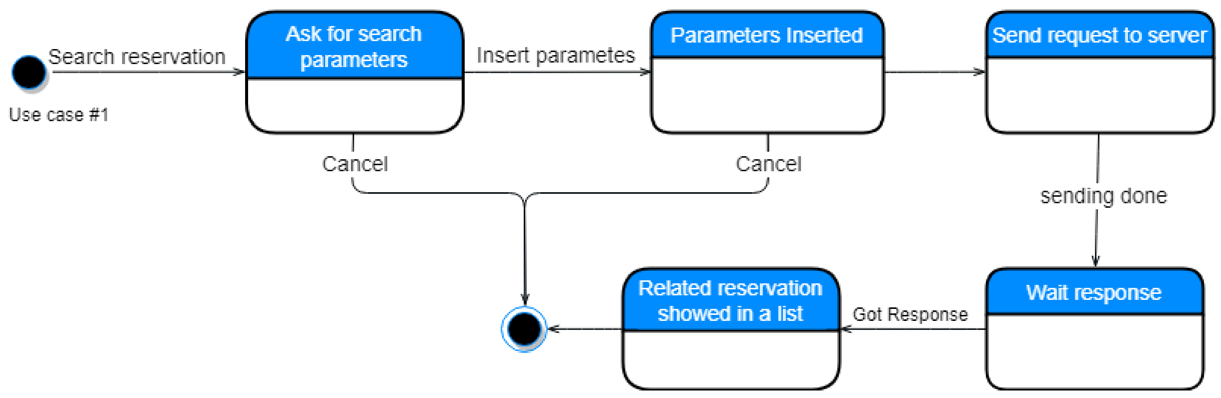

FIGURE 17 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO SEARCH FOR EXISTING RESERVATION AND VIEW DETAILS USE CASE

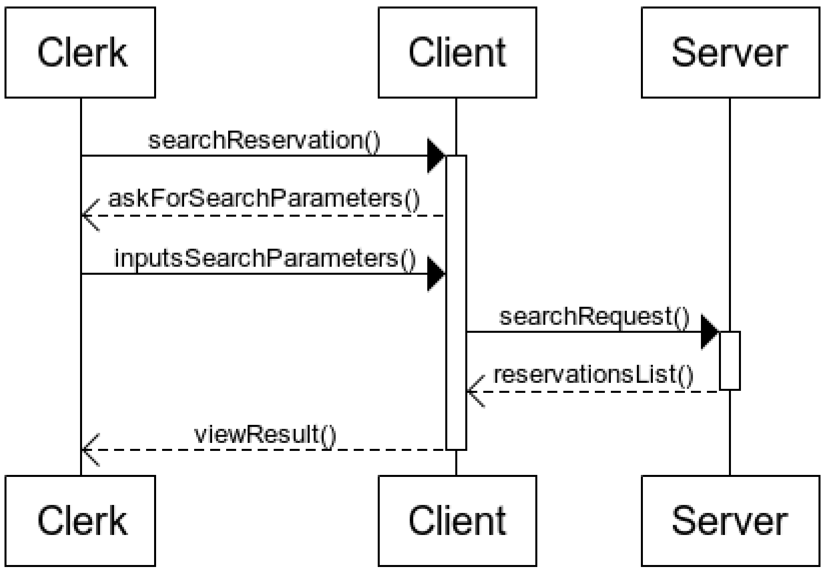

FIGURE 18 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM SEARCH FOR EXISTING RESERVATION AND VIEW DETAILS USE CASE

- **Use case #3.3**

Use case #3.3
Actors: Clerk
Description: Edit an existing pending-reservation's details
Pre-Conditions: Use case 3.2
Main flow: 1. The clerk selects a pending reservation and chooses to edit it. 2. The program asks the clerk to new reservation details. 3. The clerk changes the reservation's details and confirms. 4. The program update the reservation's details.
Post-Condition: The reservation's details are updated.
Alternative flow:

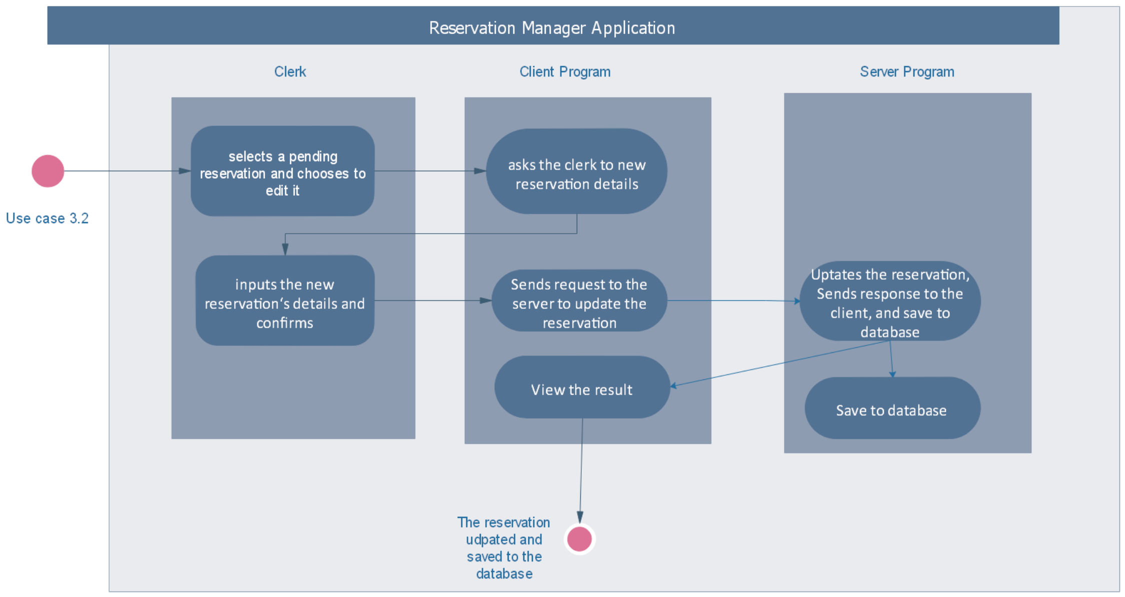
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a pending reservation and chooses to edit it
asks the clerk to new reservation details
inputs the new reservation's details and confirms
Sends request to the server to update the reservation
Uptates the reservation, Sends response to the client, and save to database
View the result
Save to database
The reservation udpated and saved to the database

FIGURE 19 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF EDIT AN EXISTING PENDING-RESERVATION USE CASE

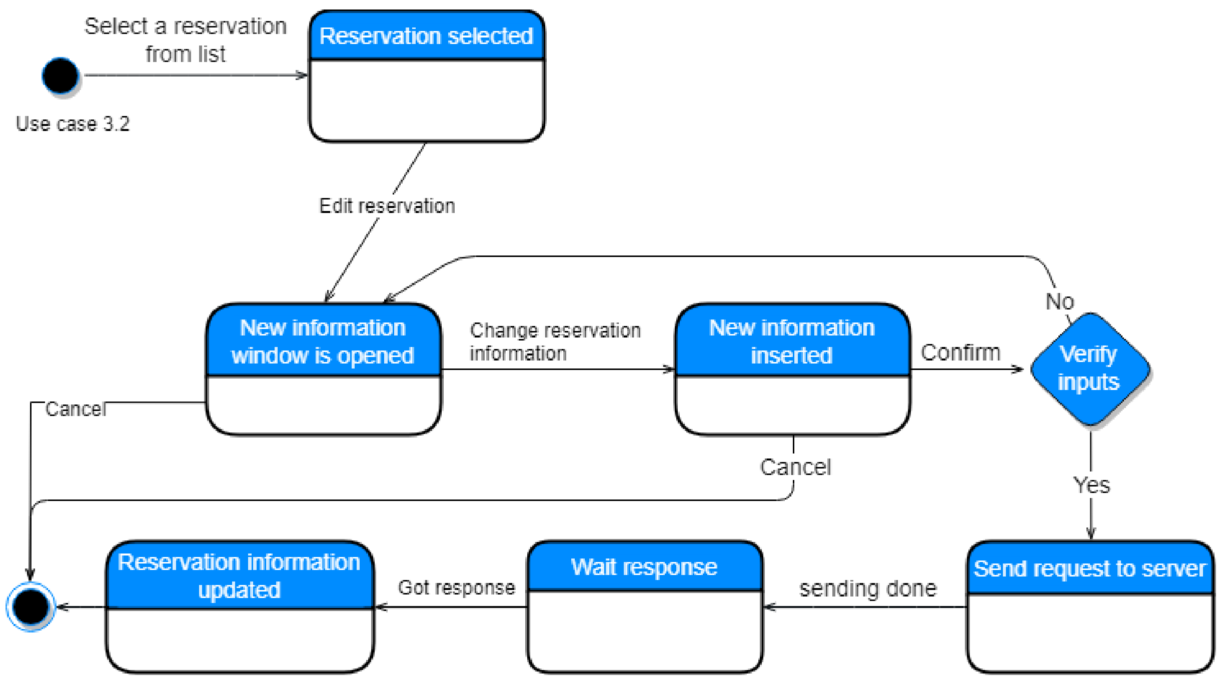

FIGURE 20 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO EDIT AN EXISTING PENDING-RESERVATION USE CASE

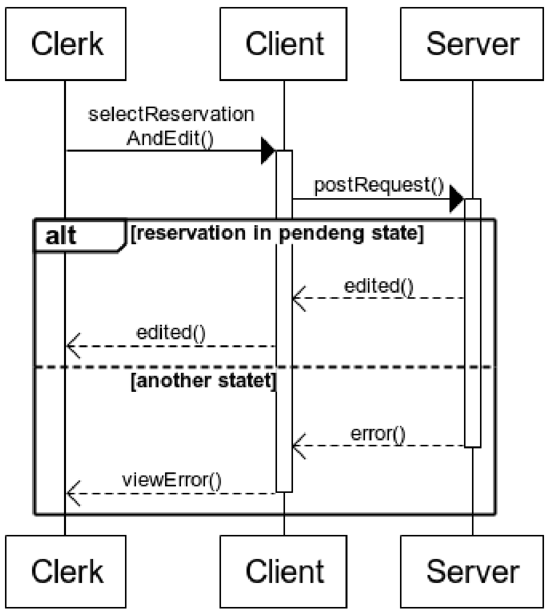

FIGURE 21 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM EDIT AN EXISTING PENDING-RESERVATION USE CASE

- **Use case #3.4**

Use case #3.4
Actors: Clerk
Description: Cancel a pending reservation (Changes the reservation's status and view the bill).
Pre-Conditions: Use case 3.2
Main flow: 1. The clerk selects a pending reservation and chooses to cancel it. 2. The program views a confirmation message. 3. The clerk confirms the cancellation. 4. The program cancel the reservation. 5. The program calculates the bill and views the result.
Post-Condition: The selected reservation is canceled
Alternative flow: **3.a The clerk chooses to cancel the cancellation.** 3.a.1 Continue to 1

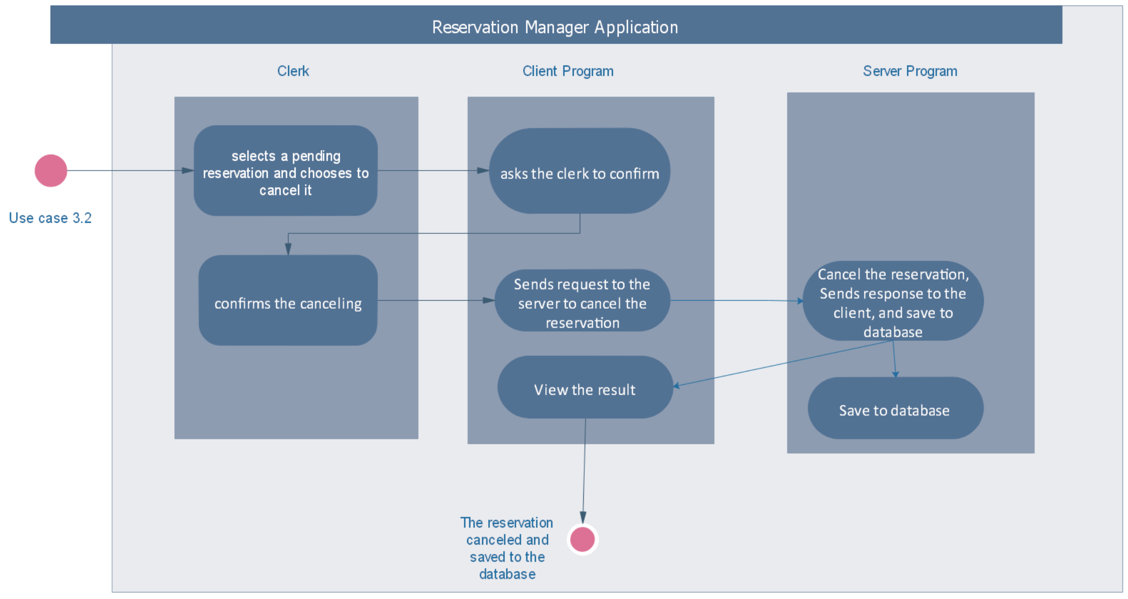
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a pending reservation and chooses to cancel it
asks the clerk to confirm
confirms the canceling
Sends request to the server to cancel the reservation
Cancel the reservation, Sends response to the client, and save to database
View the result
Save to database
The reservation canceled and saved to the database

FIGURE 22 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF CANCEL A PENDING RESERVATION USE CASE

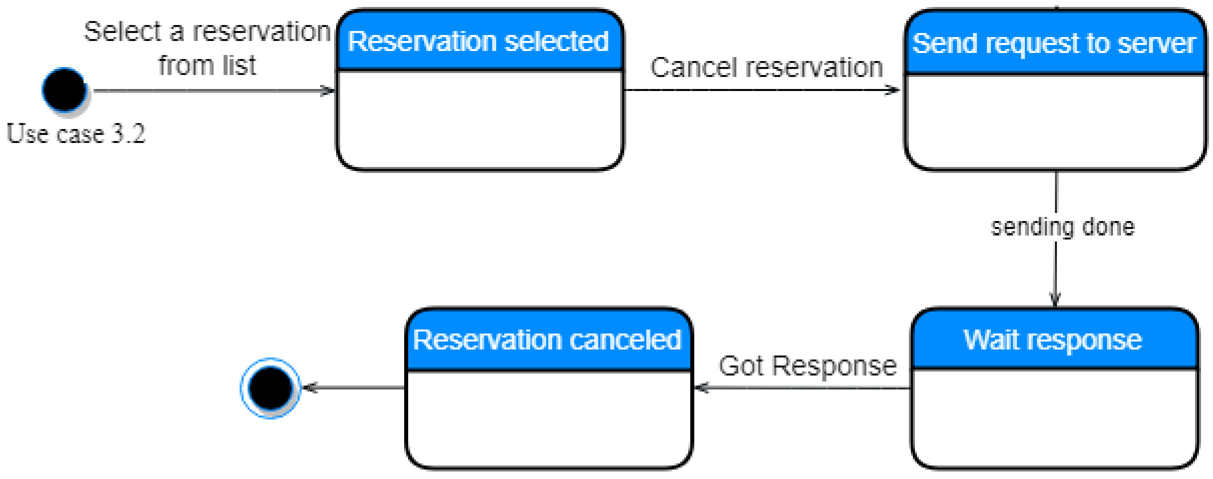

FIGURE 23 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO CANCEL A PENDING RESERVATION USE CASE

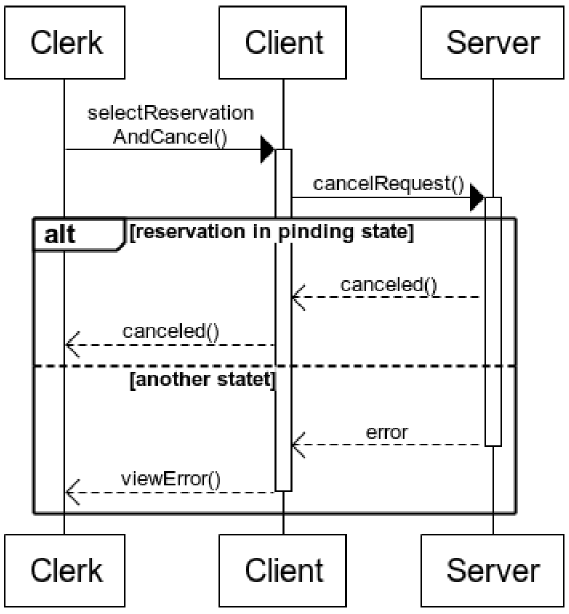

FIGURE 24 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM CANCEL A PENDING RESERVATION USE CASE

- **Use case #3.5**

Use case #3.5
Actors: Clerk
Description: Delete a pending or checked out reservation (Delete the reservation completely without bill).
Pre-Conditions: Use case 3.2
Main flow: 1. The clerk selects a reservation and chooses to delete it. 2. The program views a confirmation message. 3. The clerk confirms the deleting. 4. The program delete the reservation compeletally.
Post-Condition: The selected reservation is removed from the reservations-list.
Alternative flow: **3.a The clerk choose to cancel the deleting.** 3.a.1 Continue to 1

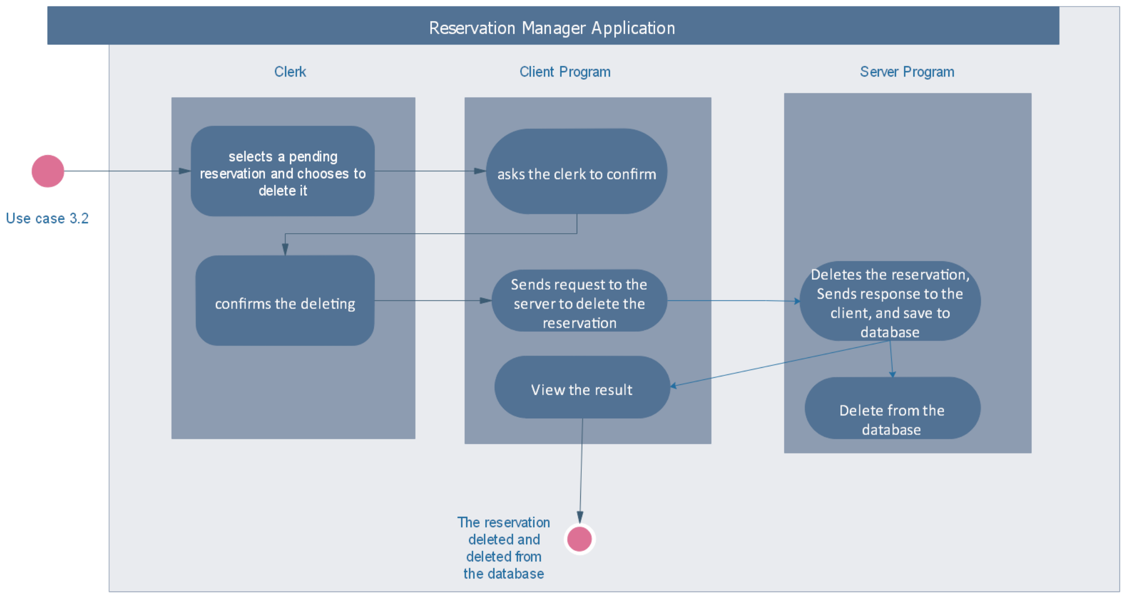
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a pending reservation and chooses to delete it
asks the clerk to confirm
confirms the deleting
Sends request to the server to delete the reservation
Deletes the reservation, Sends response to the client, and save to database
View the result
Delete from the database
The reservation deleted and deleted from the database

FIGURE 25 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF DELETE A PENDING OR CHECKED OUT RESERVATION USE CASE

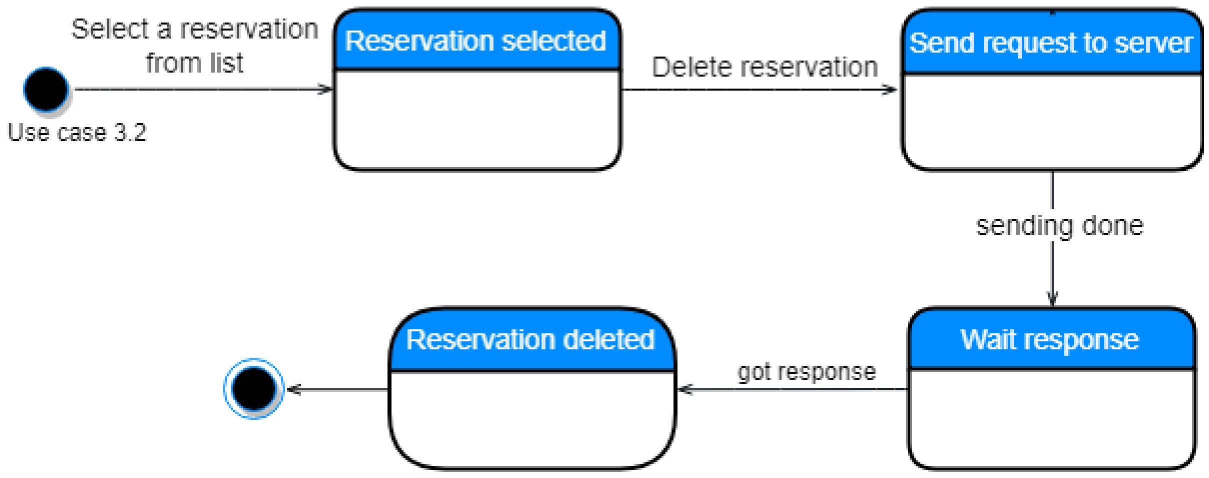

Figure 26 - UML State Machine Diagram Represents the Changes In the Status Of the System Over Time In Response To DELETE A PENDING OR CHECKED OUT RESERVATION Use Case

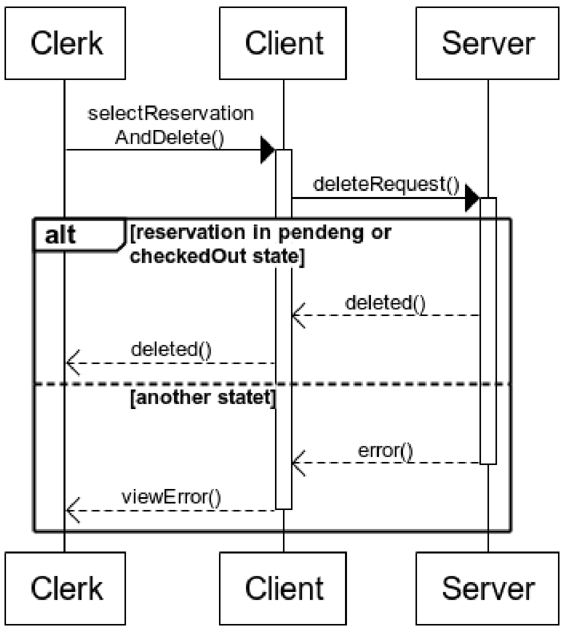

Figure 27 - UML Sequence Diagram Represents The Actors and the Interactions Exchanged Between Them to Perform DELETE A PENDING OR CHECKED OUT RESERVATION Use Case

- **Use case #3.6**

Use case #3.6
Actors: Clerk
Description: Add service to a reservation
Pre-Conditions: Use case 3.2
Main flow: 1. The clerk selects a reservation and chooses to add service. 2. The program asks the clerk to define a service. 3. The clerk defines the service and confirms. 4. The program add the service to the reservation.
Post-Condition: New service added to the reservation.
Alternative flow: **3.a The clerk choose to cancel the deleting.** 3.a.1 Continue to 1

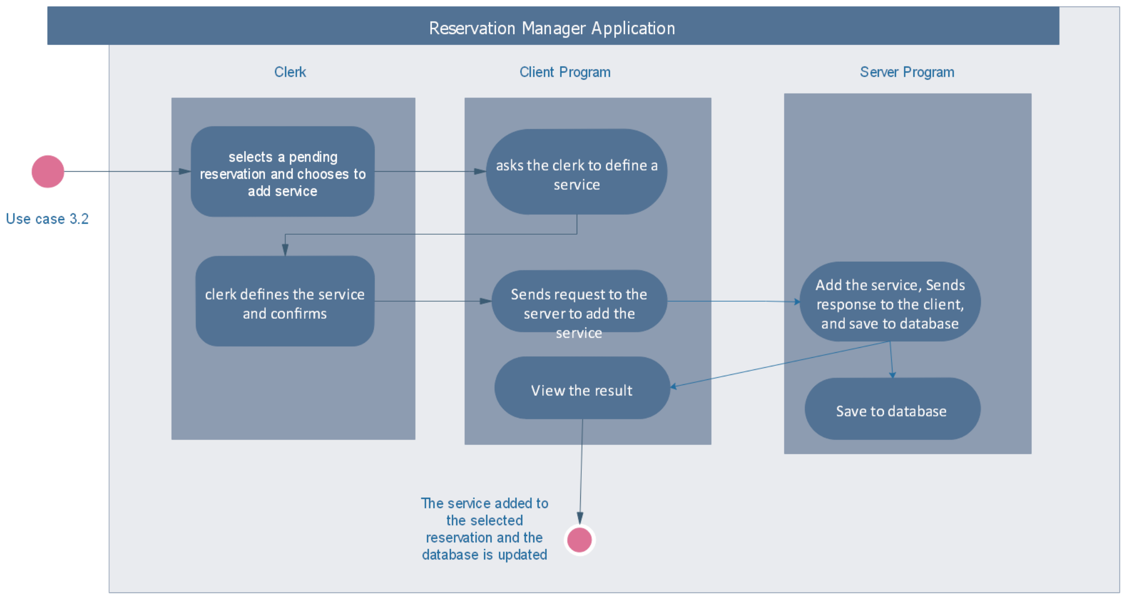
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a pending reservation and chooses to add service
asks the clerk to define a service
clerk defines the service and confirms
Sends request to the server to add the service
Add the service, Sends response to the client, and save to database
View the result
Save to database
The service added to the selected reservation and the database is updated

FIGURE 28 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF ADD A SERVICE TO A RESERVATION USE CASE

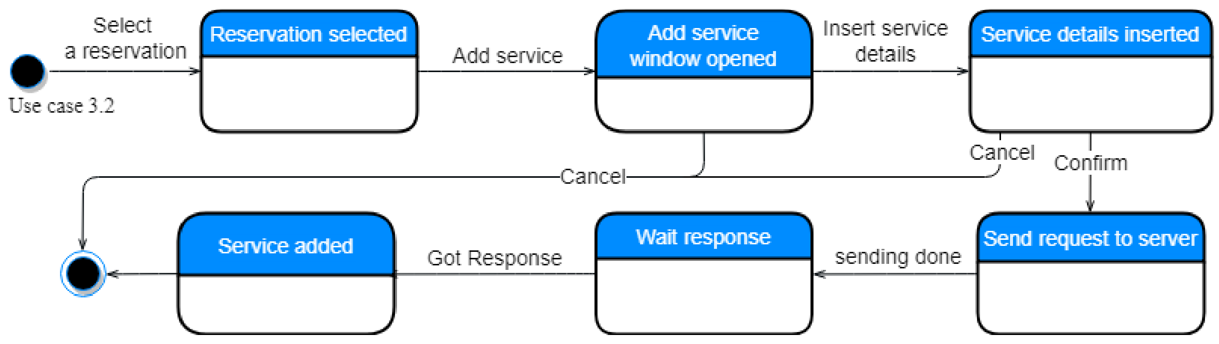

FIGURE 29 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO ADD A SERVICE TO A RESERVATION USE CASE

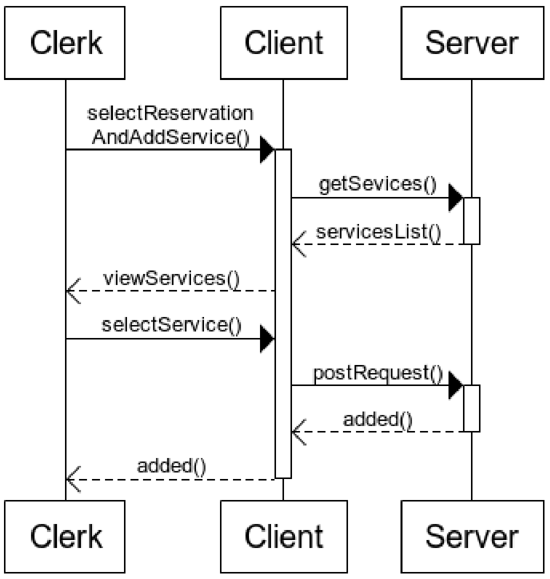

FIGURE 30 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM ADD A SERVICE TO A RESERVATION USE CASE

- ## Use case #4.1

Use case #4.1
Actors: Clerk
Description: Search for existing customers and view details
Pre-Conditions: - Use case 1. - Some reservations and customers are already registered.
Main flow: 1. The clerk chooses "Search customer " 2. The program asks the clerk to input the customer's details. 3. The clerk inputs the customer details and confirms. 4. The program shows the customers that match the search inputs (Table with their details).
Post-Condition: The program displays the customers that match the search inputs.
Alternative flow: **4.a The system did not find a customer matchs the details.** 4.a.1 The system show message that there is no customer matches the search inputs. 4.a.2 Continue to 2

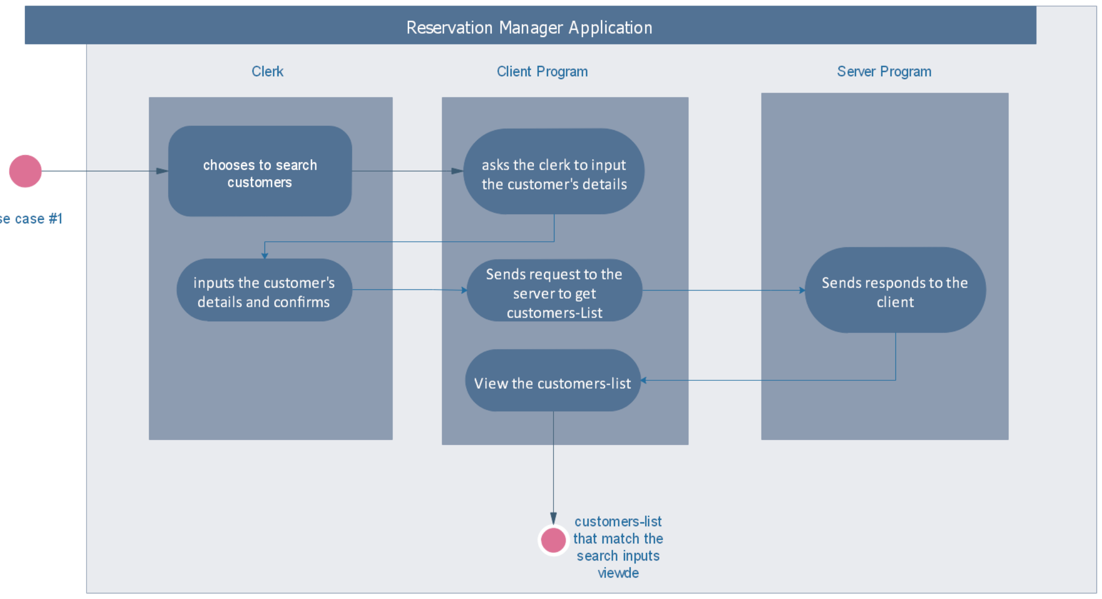
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #1
chooses to search customers
asks the clerk to input the customer's details
inputs the customer's details and confirms
Sends request to the server to get customers-List
Sends responds to the client
View the customers-list
customers-list that match the search inputs viewde

FIGURE 31 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF SEARCH FOR EXISTING CUSTOMERS AND VIEW DETAILS USE CASE

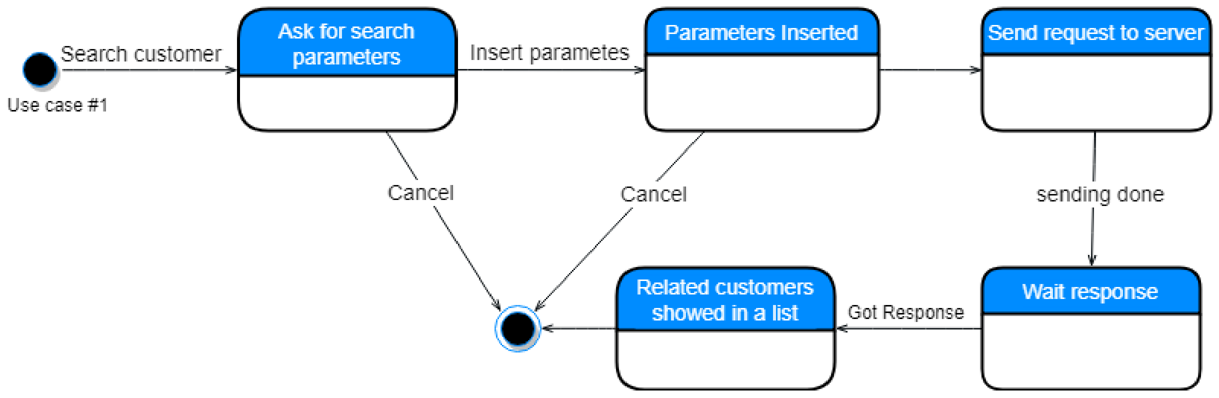

FIGURE 32 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO SEARCH FOR EXISTING CUSTOMERS AND VIEW DETAILS USE CASE

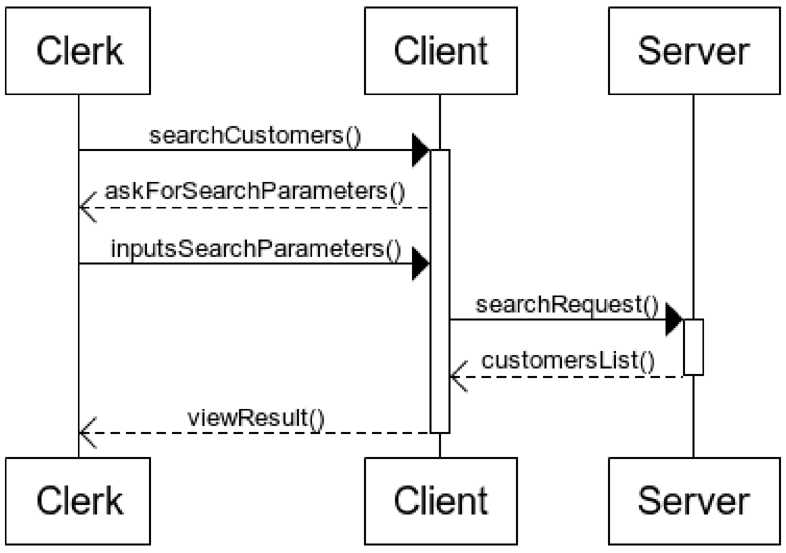

FIGURE 33 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM SEARCH FOR EXISTING CUSTOMERS AND VIEW DETAILS USE CASE

- **Use case #4.2**

Use case #4.2
Actors: Clerk
Description: Edit an existing customer‘s details
Pre-Conditions: Use case 4.1
Main flow: 1. The clerk selects a customer and chooses to edit it. 2. The program asks the clerk to new customer’s details. 3. The clerk changes the customer’s details and confirms. 4. The program update the customer’s details.
Post-Condition: The customer’s details are updated.
Alternative flow:

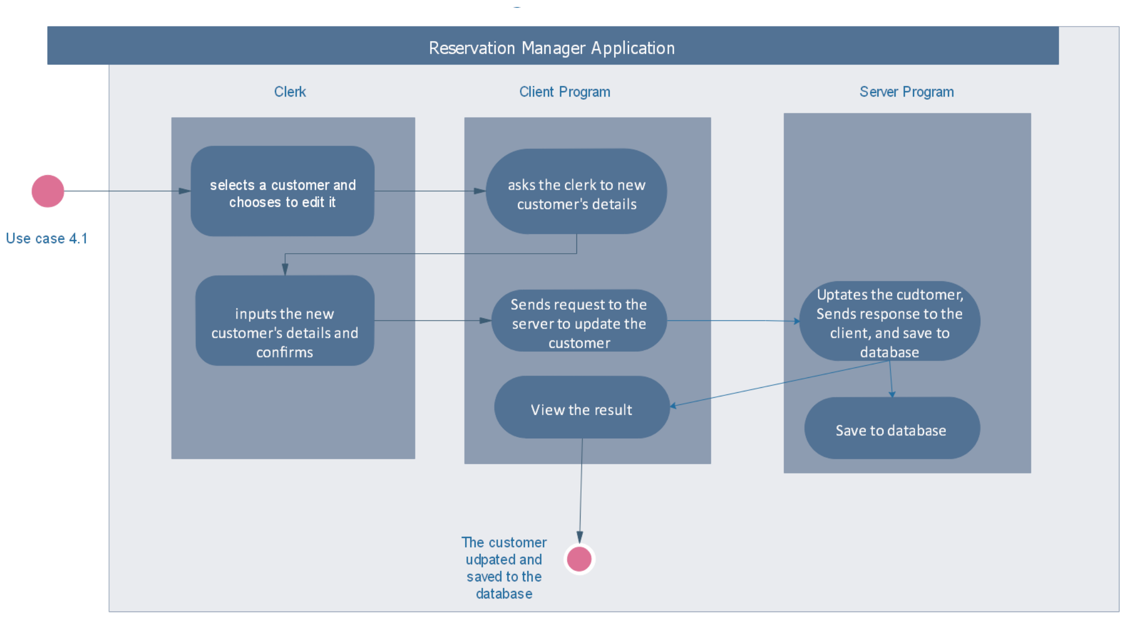
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 4.1
selects a customer and chooses to edit it
asks the clerk to new customer's details
inputs the new customer's details and confirms
Sends request to the server to update the customer
Uptates the cudtomer, Sends response to the client, and save to database
View the result
Save to database
The customer udpated and saved to the database

FIGURE 34 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF EDIT AN EXISTING CUSTOMER USE CASE

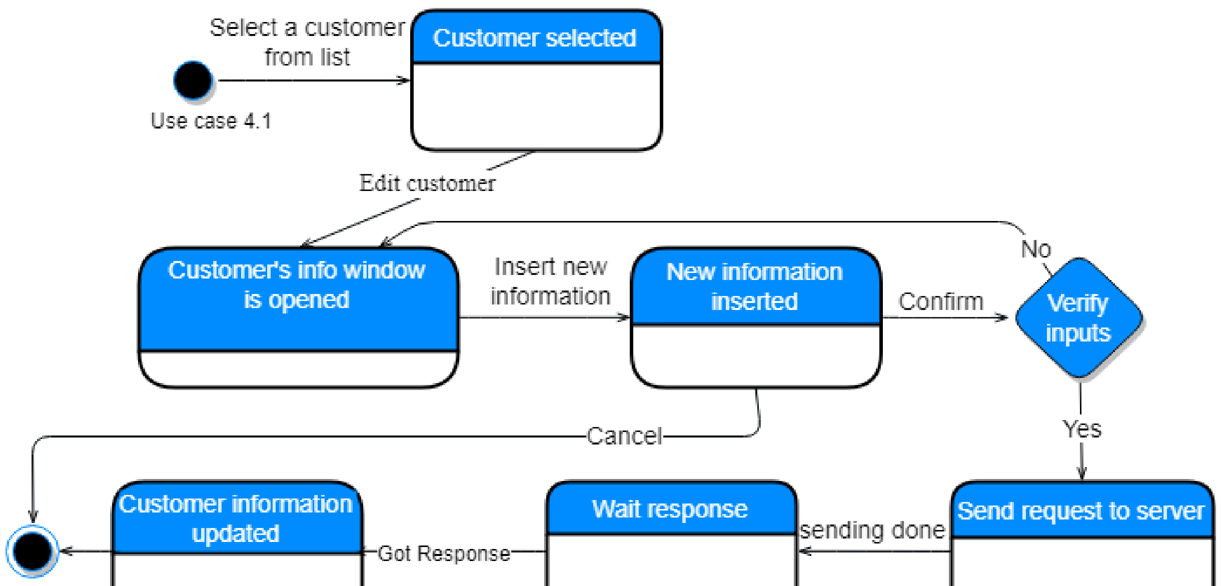

FIGURE 35 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO EDIT AN EXISTING CUSTOMER USE CASE

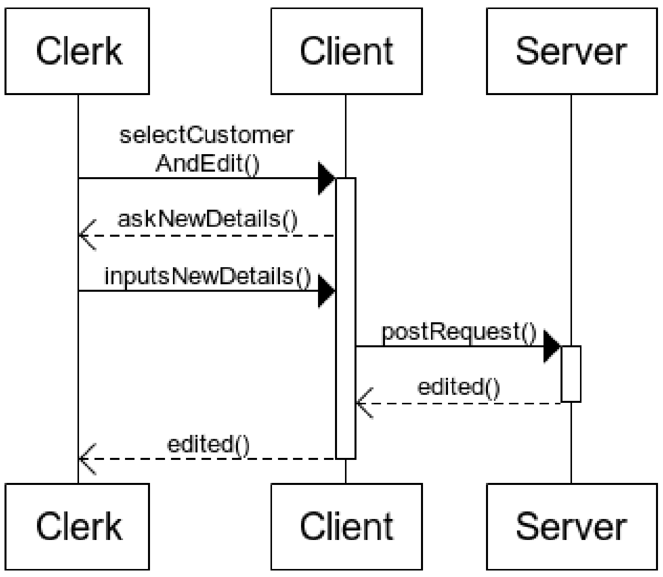

FIGURE 36 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM EDIT AN EXISTING CUSTOMER USE CASE

- **Use case #4.3**

Use case #4.3
Actors: Clerk
Description: Delete a customer (that does not has a pending or checked in reservation)
Pre-Conditions: Use case 4.1
Main flow: 1. The clerk selects a customer and chooses to delete it. 2. The program views a confirmation message. 3. The clerk confirms the deleting. 4. The program delete the customer compeletatlly.
Post-Condition: The selected customer is deleted.
Alternative flow: **3.a The clerk choose to cancel the deleting.** 3.a.1 Continue to 1

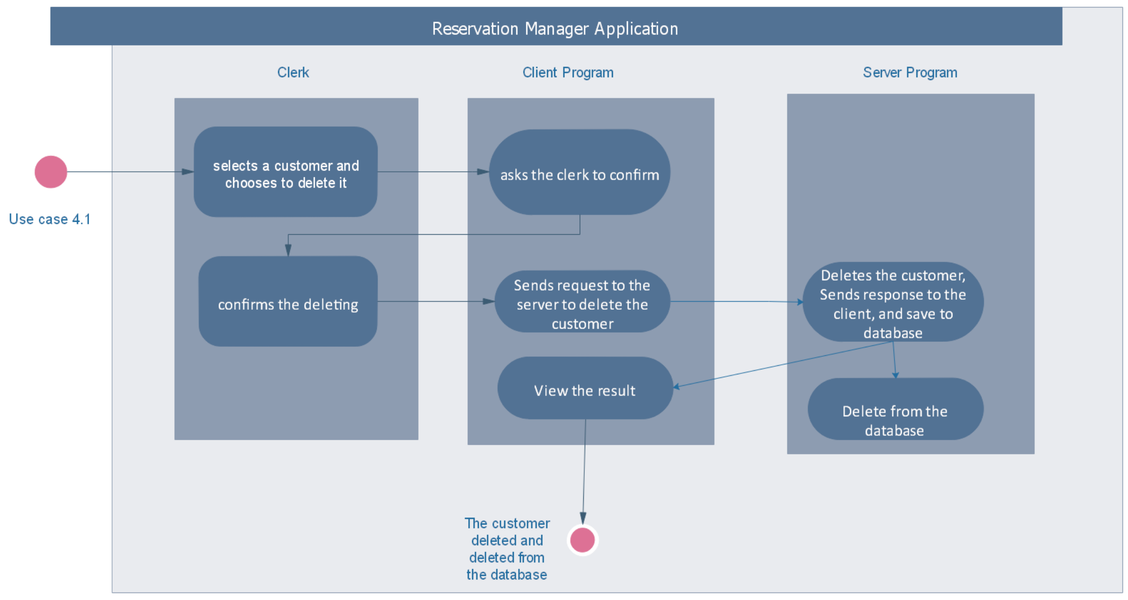
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 4.1
selects a customer and chooses to delete it
asks the clerk to confirm
confirms the deleting
Sends request to the server to delete the customer
Deletes the customer, Sends response to the client, and save to database
View the result
Delete from the database
The customer deleted and deleted from the database

FIGURE 37 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF DELETE CUSTOMER USE CASE

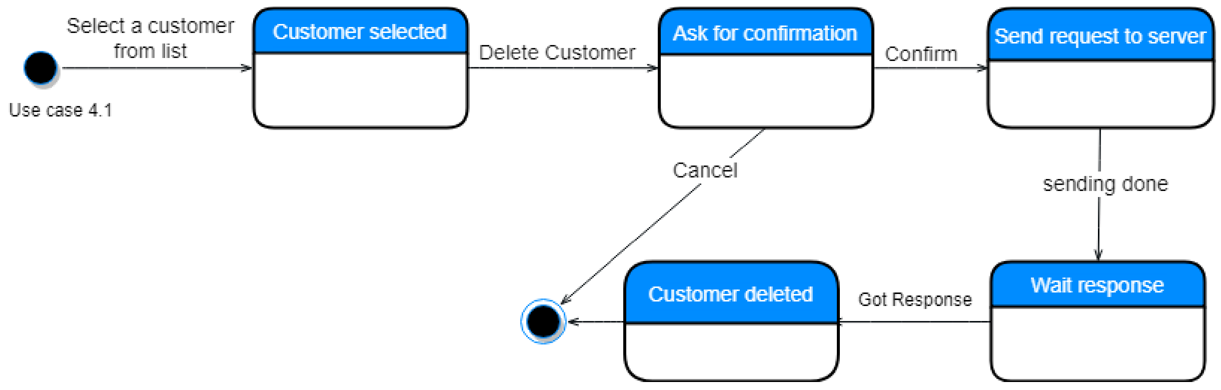

FIGURE 38 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO DELETE CUSTOMER USE CASE

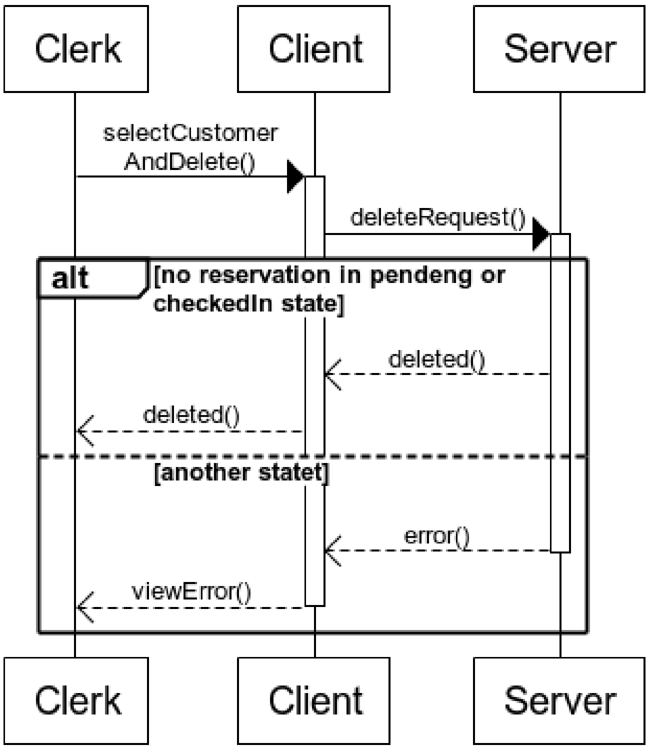

FIGURE 39 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM DELETE CUSTOMER USE CASE

- **Use case #5.1**

Use case #5.1
Actors: Clerk
Description: Search for bills
Pre-Conditions: - Use case 1. - Some reservations and customers are already registered.
Main flow: 1. The clerk chooses "Search bills " 2. The program asks the clerk to input the bill's details. 3. The clerk inputs the bill details and confirms. 4. The program shows the bills that match the search inputs (Table with brief details).
Post-Condition: The program displays the bills that match the search inputs.
Alternative flow: **4.a The system did not find any bill matchs the details.** 4.a.1 The system show message that there is no bill matches the search inputs. 4.a.2 Continue to 2

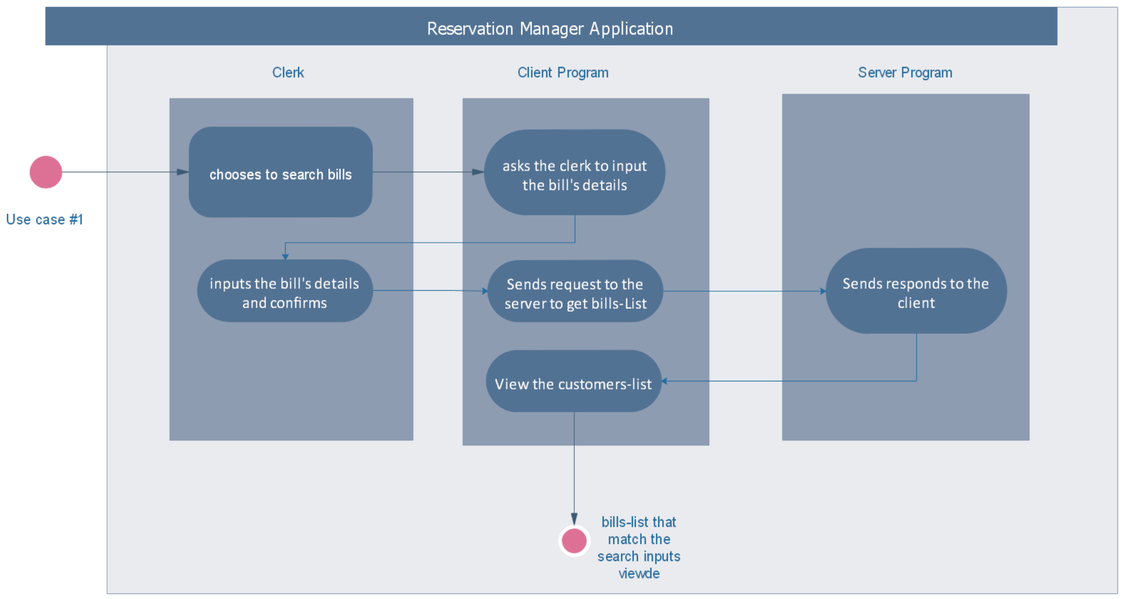
Reservation Manager Application
Clerk
Client Program
Server Program
Use case #1
chooses to search bills
asks the clerk to input the bill's details
inputs the bill's details and confirms
Sends request to the server to get bills-List
Sends responds to the client
View the customers-list
bills-list that match the search inputs viewde

FIGURE 40 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF SEARCH FOR BILLS USE CASE

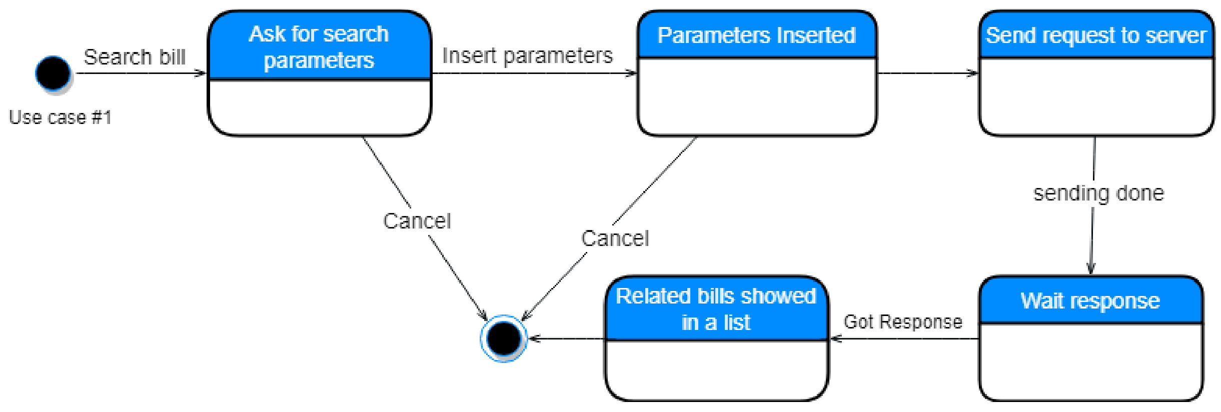

Figure 41 - UML State Machine Diagram Represents the Changes In the Status Of the System Over Time In Response To SEARCH FOR BILLS Use Case

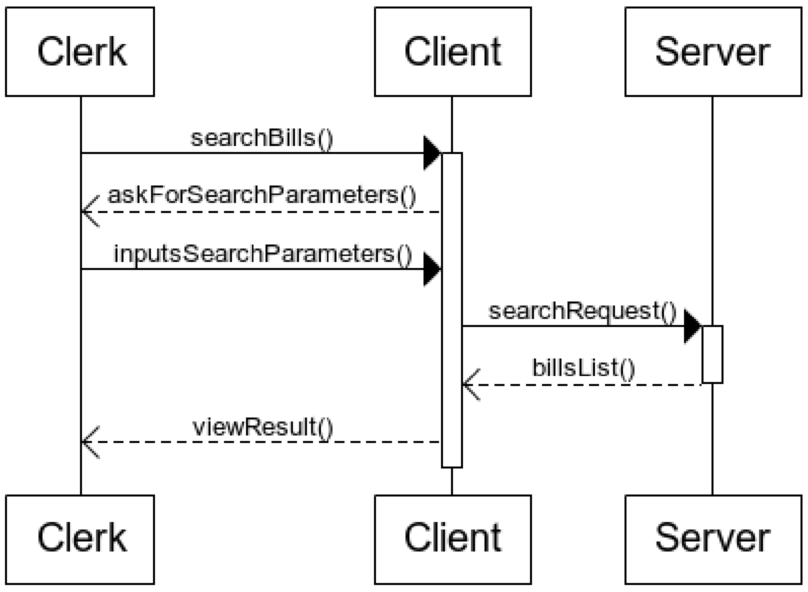

Figure 42 - UML Sequence Diagram Represents The Actors and the Interactions Exchanged Between Them to Perform SEARCH FOR BILLS Use Case

- **Use case #5.2**

Use case #5.2
Actors: Clerk
Description: View bill as PDF
Pre-Conditions: - Use case 5.1 - Adobe PDF reader is installed on the computer
Main flow: 1. The clerk selects a bill and chooses to view it. 2. The program generate a PDF file and try to open it by the PDF reader in the computer. 3. The bill is viewd
Post-Condition: The bill is viewd as PDF
Alternative flow:

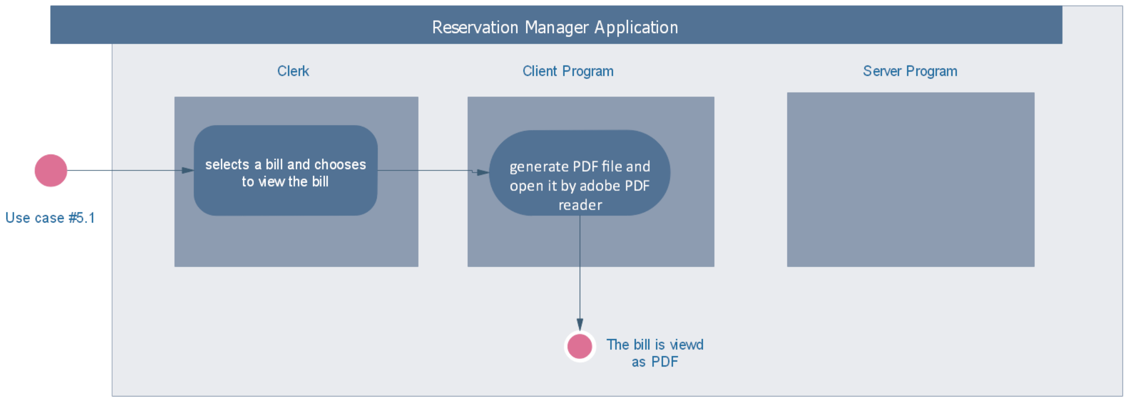
Reservation Manager Application
Clerk
Client Program
Server Program
selects a bill and chooses to view the bill
generate PDF file and open it by adobe PDF reader
Use case #5.1
The bill is viewd as PDF

FIGURE 43 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF VIEW A BILL AS PDF USE CASE

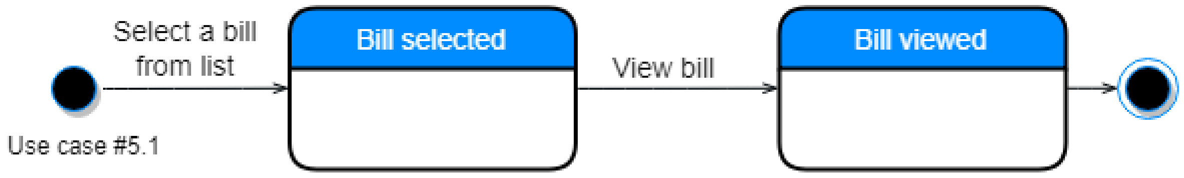

FIGURE 44 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO VIEW A BILL AS PDF USE CASE

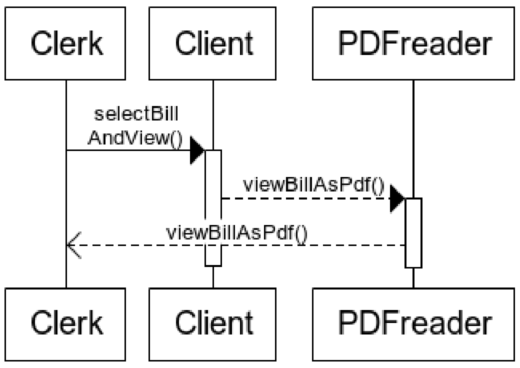

FIGURE 45 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM VIEW A BILL AS PDF USE CASE

- **Use case #5.3**

Use case #5.3
Actors: Clerk
Description: Print bill
Pre-Conditions: - Use case 5.2
Main flow: 1. The clerk chooses to print the bill in the Adobe PDF reader.   2. The bill is printed.
Post-Condition: The bill is printed
Alternative flow:

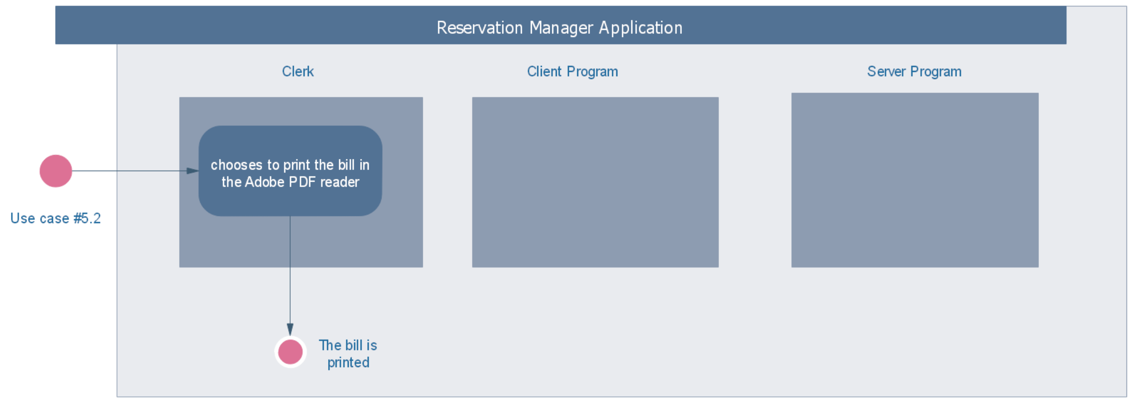
Reservation Manager Application
Clerk
Client Program
Server Program
chooses to print the bill in the Adobe PDF reader
Use case #5.2
The bill is printed

FIGURE 46 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF PRINT BILL USE CASE

Since the printing will be done by a third party program (PDF reader), the program (client and server) will not be a part of printing process. Therfore, they do not have any act in the activity, sequence diagrams.

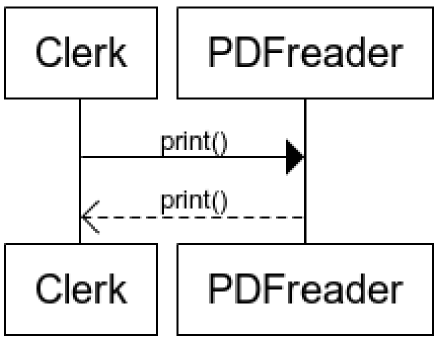

FIGURE 47 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM PRINT BILL USE CASE

- **Use case #5.4**

Use case #5.4
Actors: Clerk
Description: Mark bill as paid
Pre-Conditions: - Use case 5.1
Main flow: 3. The clerk selects a bill and chooses to mark it as paid.   4. The program marks the bill as paid.
Post-Condition: The bill is marked as paid
Alternative flow:

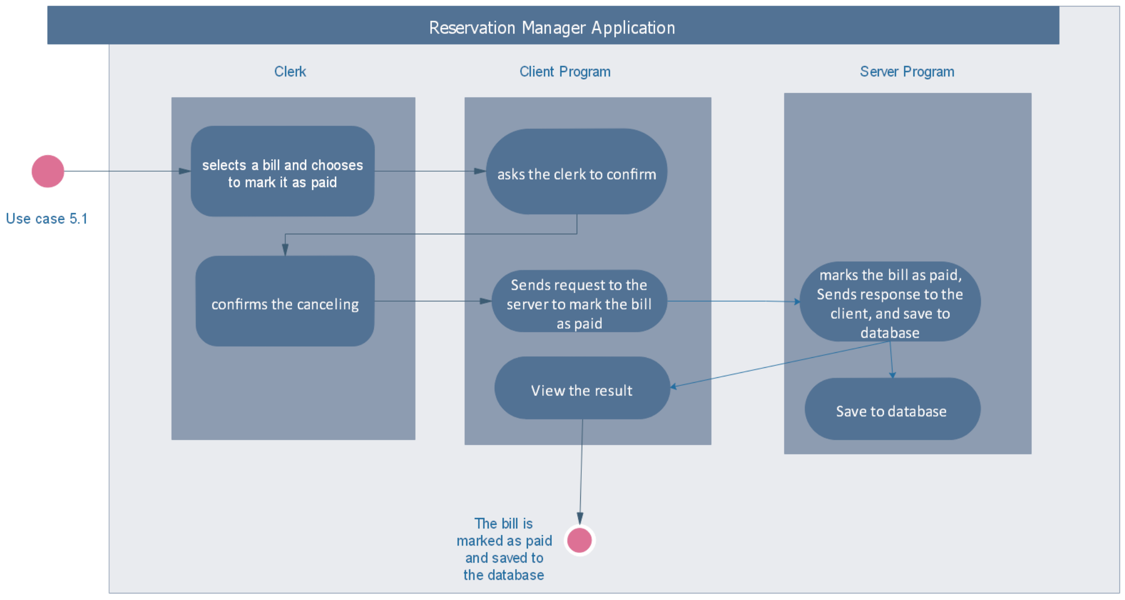
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 5.1
selects a bill and chooses to mark it as paid
asks the clerk to confirm
confirms the canceling
Sends request to the server to mark the bill as paid
marks the bill as paid, Sends response to the client, and save to database
View the result
Save to database
The bill is marked as paid and saved to the database

FIGURE 48 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF MARK BILL AS PAID USE CASE

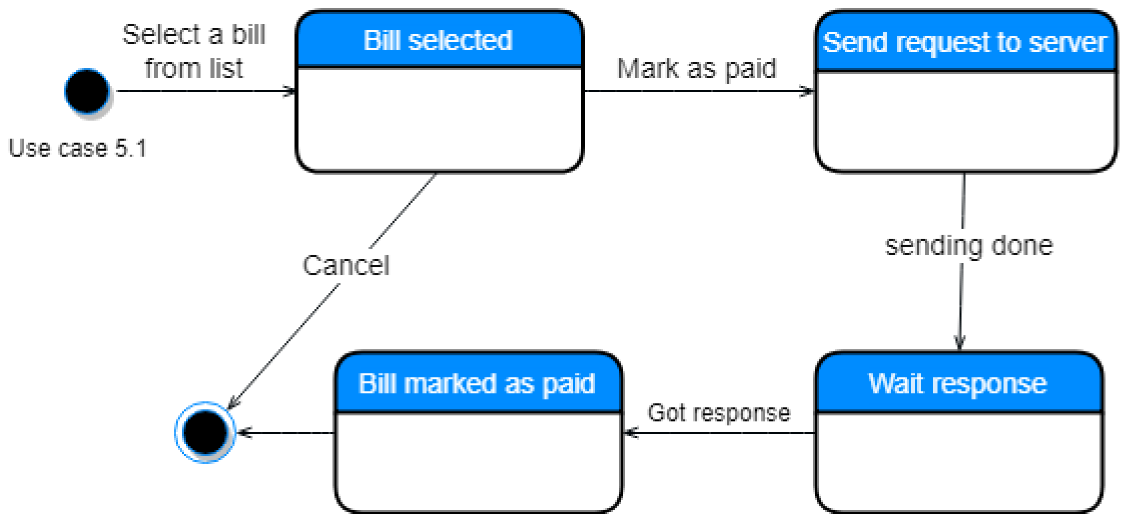

FIGURE 49 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO MARK BILL AS PAID USE CASE

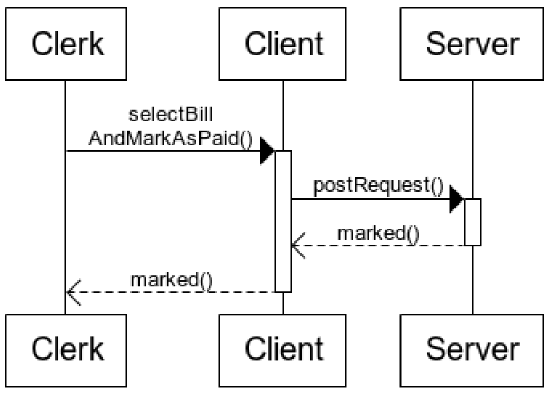

FIGURE 50 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM MARK BILL AS PAID USE CASE

- **Use case #6**

Use case #6
Actors: Clerk
Description: Check in a pending reservation
Pre-Conditions: Use case 3.2
Main flow: 1. The clerk selects a pending reservation and chooses to change the reservation's status to checked-in 2. The system changes the reservation's status to checked-in.
Post-Condition: The reservation's status is changed to checked-in
Alternative flow: **3.a The clerk choose to cancel the changing.** 3.a.1 Continue to 1

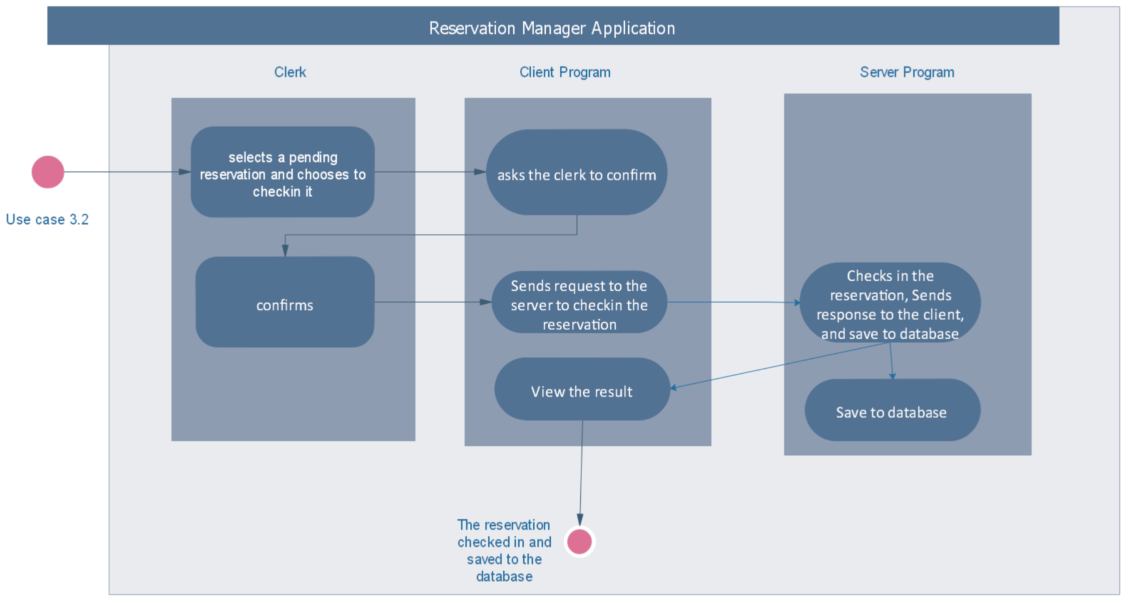
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a pending reservation and chooses to checkin it
asks the clerk to confirm
confirms
Sends request to the server to checkin the reservation
Checks in the reservation, Sends response to the client, and save to database
View the result
Save to database
The reservation checked in and saved to the database

FIGURE 51 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF CHECK-IN A PENDING RESERVATION USE CASE

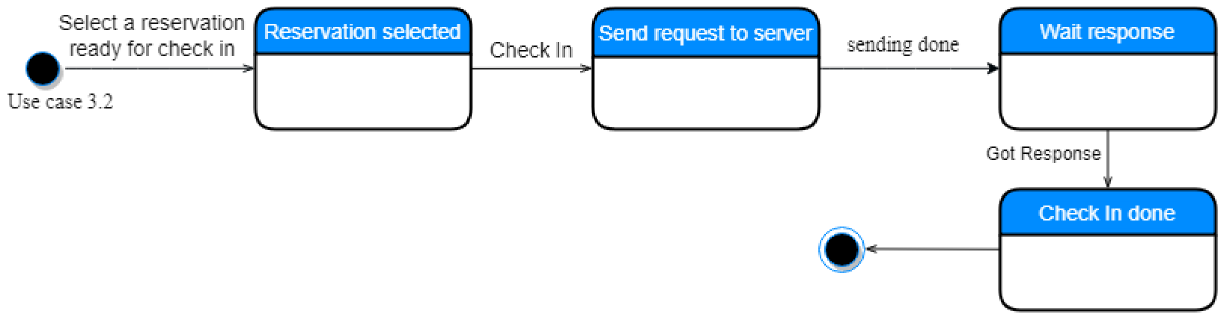

FIGURE 52 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO CHECK-IN A PENDING RESERVATION USE CASE

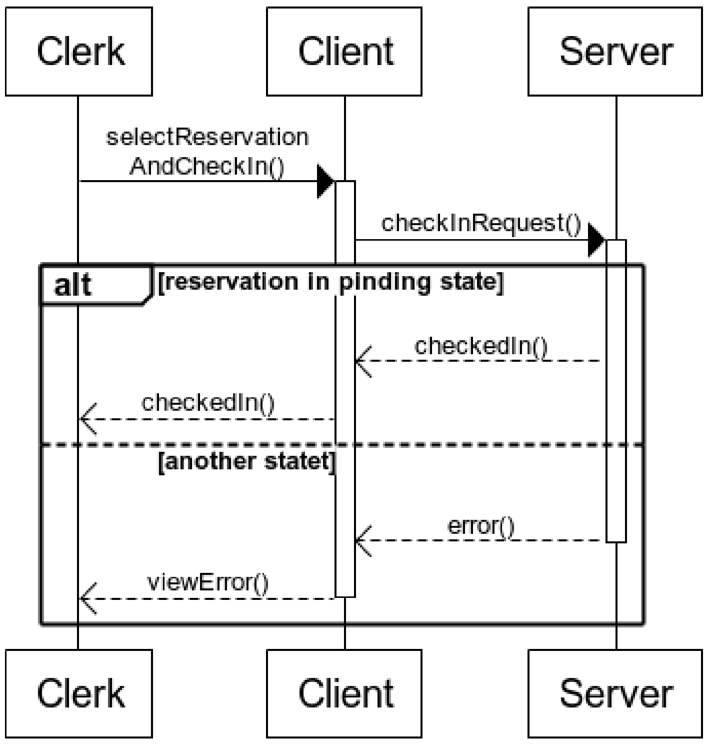

FIGURE 53 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM CHECK-IN A PENDING RESERVATION USE CASE

- **Use case #7**

Use case #7

Actors: Clerk

Description: Check out a checked in reservation

Pre-Conditions: Use case 3.2

Main flow:

1. The clerk selects a checked in reservation and chooses to check out.
2. The system check out the reservation.
3. The system changes the reservation's status to checked-out and calculate the bill.
4. The system asks the clerk to view the bill.
5. The clerk choose to view the bill.
6. The program views the bill.
7. The program ask the clerk to mark the bill as paid.
8. The clerk choose to mark the bill as paid.
9. The program marks the bill as paid.
10. The reservation is checked-out.

Post-Condition: The reservation is checked-out.

Alternative flows:

5.a The clerk does not want to view the bill.

5.a.1 Continue to 10.

8.a The clerk does not want to mark the bill as paid.

8.a.1 Continue to 10.

Special Requirements: The whole checkout procedure must take in average less than 60 seconds to complete.

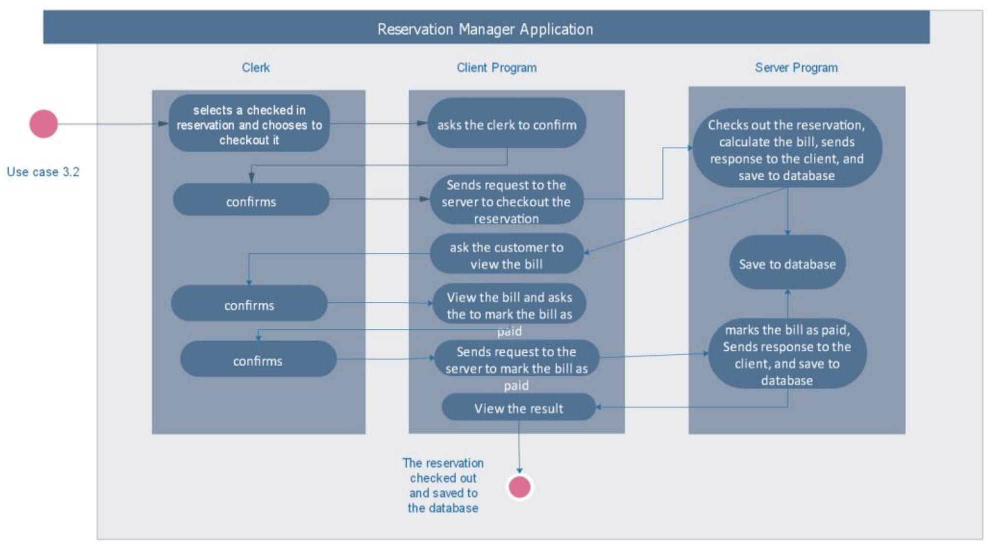
Reservation Manager Application
Clerk
Client Program
Server Program
Use case 3.2
selects a checked in reservation and chooses to checkout it
asks the clerk to confirm
Checks out the reservation, calculate the bill, sends response to the client, and save to database
confirms
Sends request to the server to checkout the reservation
ask the customer to view the bill
Save to database
confirms
View the bill and asks the to mark the bill as paid
marks the bill as paid, Sends response to the client, and save to database
confirms
Sends request to the server to mark the bill as paid
View the result
The reservation checked out and saved to the database

FIGURE 54 - UML ACTIVITY DIAGRAM REPRESENTS THE ACTIONS STEPS OF CHECK-OUT A CHECKED-IN RESERVATION USE CASE

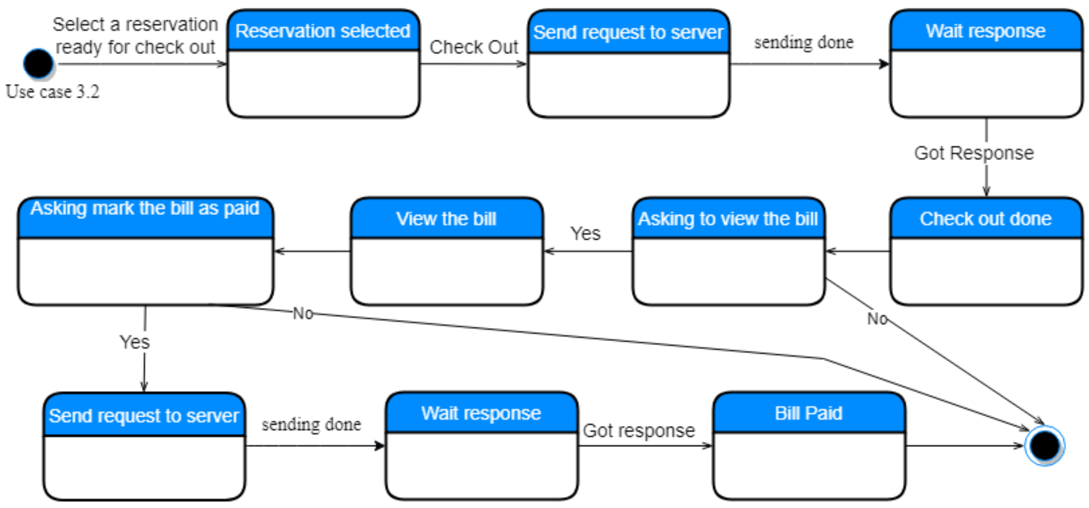

FIGURE 55 - UML STATE MACHINE DIAGRAM REPRESENTS THE CHANGES IN THE STATUS OF THE SYSTEM OVER TIME IN RESPONSE TO CHECK-OUT A CHECKED-IN RESERVATION USE CASE

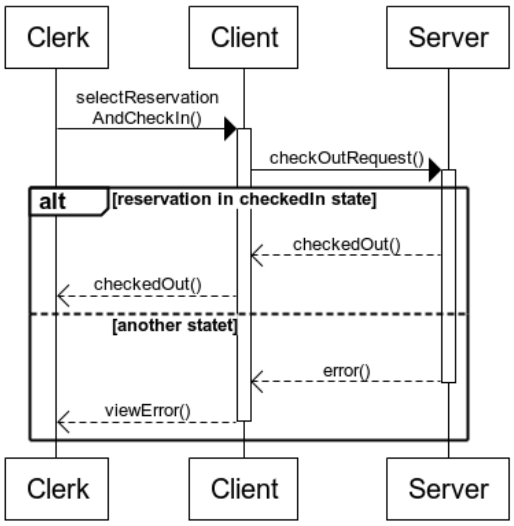

FIGURE 56 - UML SEQUENCE DIAGRAM REPRESENTS THE ACTORS AND THE INTERACTIONS EXCHANGED BETWEEN THEM TO PERFORM CHECK-OUT A CHECKED-IN RESERVATION USE CASE

5. The analysis of the legacy system

- Old: The legacy software is about 14 years old.
    - The method (setNextFocusableComponent) in JHotel_Translator class is deprecated.
    - The method getType() in the class RoomSelectWindow.java is not allowed in JDK 8 anymore since it is not compatible with the new operating systems.
- The software is inherited from previous developers.
- The the legacy system is not documented
- The legacy system is using a local data base.
- The legacy system is untested.
- It is hard to reuse or maintain the legacy system. All classes are gathered in one package, and there is no design pattern used (such as Model-View or another pattern)
- Inflexible code:
    - Some classes are responsible for more than one feature.
    - Almost all the existing features are implemented without following any pattern. For example, the searching feature does not follow any search pattern.
- The legacy system does not follow any architecture pattern.