Linnaeus Hotel Management System Document

1. Domain Analysis 6

1.1. Introduction, Background and Problem. 6

1.1.1. General knowledge about the domain. 6

1.1.2. Clients and users 6

1.1.3. Environment 6

1.1.4. Tasks and procedures currently performed 6

1.1.5. Competing software 7

2. Analysis of Legacy System 8

2.1. Introduction 8

2.2. Analysis 8

2.3. Team Decision 9

2.3.1. Reasons of re-writing the system 9

2.3.2. Other Factors 9

3. Requirements Document 10

3.1. Introduction 10

3.1.1. Purpose 10

3.1.2. Scope 10

3.1.3. Stakeholders 10

3.1.4. Overview 10

3.2. Overall Description 10

3.2.1. Product Perspective 11

3.2.1.1. Hardware Interfaces 11

3.2.1.2. Software Interfaces 11

3.2.2. Product Function 11

3.2.3. User Characteristics 11

3.3. Specific Requirements 11

3.3.1. External Interfaces 11

3.3.1.1. User Interface 11

3.3.1.2. Hardware Interfaces 13

3.3.1.3. Software Interfaces 13
3.3.1.4. Communication Interface 13
3.3.2. Functional Requirements 13
3.3.3. Non- Functional Requirements 14
3.3.3.1. Performance Requirement 14
3.3.3.2. Database requirements (Data retention) 15
3.3.3.3. Availability 15
3.3.3.4. Security 15
3.3.3.5. Design Constraints 15
3.3.3.6. Standard Compliance 16
3.3.3.7. Usability 16
3.3.3.8. Maintainability 16
3.3.3.9. Portability 16
3.3.3.10. Learnability 16
3.3.4. Scenarios 16
3.3.4.1. Scenario 1 16
3.3.4.2. Scenario 2 16
3.3.4.3. Scenario 3 17
3.3.4.4. Scenario 4 17
3.3.4.5. Scenario 5 17
3.3.4.6. Scenario 6 18
3.3.4.7. Scenario 7 18
3.3.4.8. Scenario 8 18
3.3.5. Actors and their tasks 18
3.3.6. Use cases 19
3.3.6.1. Overview Use-cases 19
3.3.6.2. Use-case 1 20
3.3.6.3. Use-case 2 21
3.3.6.4. Use-case 3 22
3.3.6.5. Use-case 4 23

3.3.6.6. Use-case 5 24

3.3.6.7. Use-case 6 25

3.3.6.8. Use-case 7 25

3.3.6.9. Use-case 8 26

3.3.6.10. Use-case 9 27

3.3.6.11. Use-case 10 28

4. Design Document 40

4.1. Introduction 40

4.1.1. Purpose 40

4.1.2. Scope 41

4.1.3. Glossary 41

4.1.4. Intended Audience 41

4.1.5. Priorities 41

4.1.6. Overview 42

4.2. System Overview 42

4.2.1. Product functions 42

4.2.2. User Characteristics 42

4.2.3. Outline of the design **Ошибка! Закладка не определена.**

4.2.3.1. High-level description of the design 43

4.3. Major design issues 43

4.3.1. Re-writing 43

4.3.2. Architectural Pattern 44

4.3.3. GUI 44

4.3.4. Development language 44

4.4. Technical issues 45

4.4.1. Database management system 45

4.4.2. Applications that use MySQL: 45

4.4.3. Data format 46

4.5. Design details: 46

4.5.1. Software Architecture 46

4.5.1.1. Architectural Pattern 46

4.5.1.2. The set of components 46

4.5.2. Diagrams 48

4.5.2.1. The UML Component Diagram of the architecture: 48

4.5.2.2. Deployment Diagram: 49

4.5.2.3. Component Implementation 49

4.5.2.4. Model/Functional layer: 50

4.5.2.5. GUI/View: 51

4.5.2.6. Database 52

4.5.2.7. View Sequence diagrams 52

4.5.2.7.1. Log-in 52

4.5.2.7.2. Search room 53

1. Domain Analysis

1.1. Introduction, Background and Problem.

Linnaeus Hotel is a software system build to manage the front-desk activities. This app will replace the existing paper -based system to fulfil the customer desideratum as they believe an automated system would save their money and provide better services for their guests. The system main problems are to manage reservation, check guest in and out, and reserve rooms in both hotel location Växjö and Kalmar.

1.1.1. General knowledge about the domain.

- The hotel contains rooms for guests to stay in.
- The hotel has two branches in Växjö and Kalmar.
- Some rooms adjoin with other; which mean a room has a connection with other room through internal doors.
- Each room has a quality level of either size and location.
- Each room has a certain attribute; certain number and type of beds, a room number and a smoking / non-smoking status.
- Each quality level has a maximum daily rate, although the rate that a guest pays maybe less.

1.1.2. Clients and users

Stakeholder is the Linnaeus Hotel. The employees of Linnaeus Hotel are the users of this system. Our client has ordered to design and build this software system in order to improve their business and provide better services for their customers.

1.1.3. Environment

The actors have a running PC on their desks; MS-Windows 10 which will run the system. The system will be developed and written in Java. Hardware minimum intel core i3.

1.1.4. Tasks and procedures currently performed

Linnaeus Hotel has several tasks performed in order to host a guest:

- The hotel reception main task is to take make reservation to guests and provide rooms according to the guest wish.
- The hotel clerk can book both in Växjö and Kalmar.
- The hotel records basic information about each guest such as name, telephone number, credit card, passport number, etc.

- Check and verify availability of the rooms before reservation.

1.1.5. Competing software

Nowadays, every hotel has a similar hotel software system. Some popular systems are Cloudbeds and FCS Concierge which are well known in the market and they include additional features.

2. Analysis of Legacy System

This section analyses the legacy code of JHotel provided by the stakeholders. It elaborates on the points that led to the decisions made whether or not to use this code for the software, on the 'Design' section.

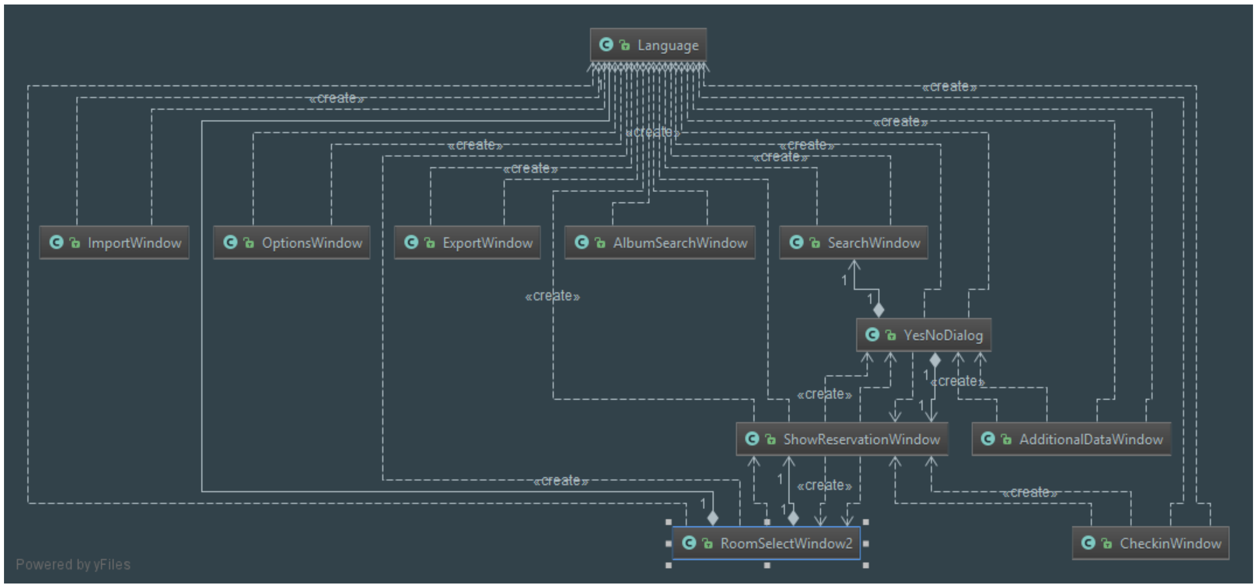
2.1. Introduction

This legacy system JHotel is coded in Java and gives a simplistic features to help a hotel serve its clients quicker. it uses a file storage to store data and has a basic user interface designed by the Java Spring Framework.

2.2. Analysis

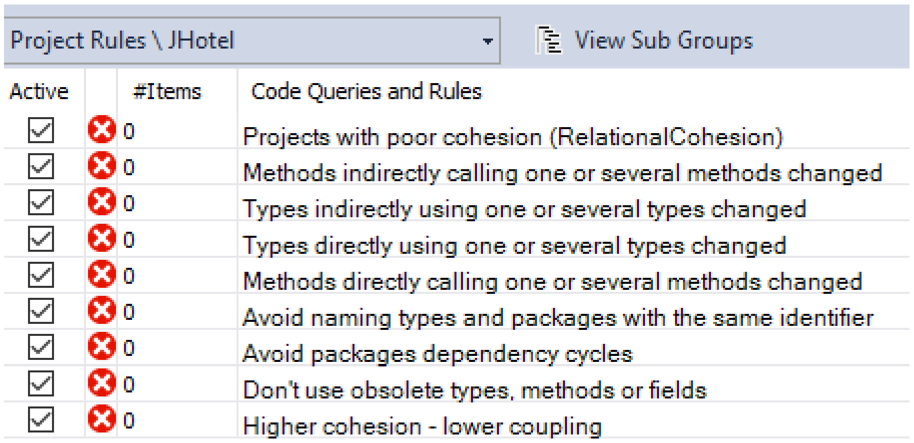
The software uses the Java Spring Framework which follows the Model View Controller pattern oriented software architecture. When run in a Windows OS, the application seems to run without a problem. It was responsive with a few delays when loading the calendar. There has not been any notable crashes but the user interface sometimes can be glitchy with full content of a page not visible in a window.

The classes in the system itself are not separated in different packages. By observing the class diagram it seems that there are many dependencies on some classes. For ex. (*See Image Below*) the language class has a lot of dependencies maily from the classes responsible for the graphical user interface.



The models in this software, such as Guest, Reservation and Room seems to access the database directly and are more of a data access layer objects rather than models.

After running the code in JArchitect there were found some rules and queries with errors (*See image below*)



Project Rules \ JHotel — View Sub Groups

Active	#Items	Code Queries and Rules
✓	✗ 0	Projects with poor cohesion (RelationalCohesion)
✓	✗ 0	Methods indirectly calling one or several methods changed
✓	✗ 0	Types indirectly using one or several types changed
✓	✗ 0	Types directly using one or several types changed
✓	✗ 0	Methods directly calling one or several methods changed
✓	✗ 0	Avoid naming types and packages with the same identifier
✓	✗ 0	Avoid packages dependency cycles
✓	✗ 0	Don't use obsolete types, methods or fields
✓	✗ 0	Higher cohesion - lower coupling

Looking at the results there seems to be naming issues, dependency cycles which is seen between Album.java and AlbumEindow.java, and poor cohesion and coupling between classes such as Reservation Managment and MainWindow.

2.3. Team Decision

After checking the analysis of the legacy system, the developing team decided to re-write the program

2.3.1. Reasons of re-writing the system

Other than the negatives mentioned in the analysis section,

- Poor exception handling.
- The system does not have Models.
- Code-wise it is not well commented which makes it harder to understand the functionality.
- User interface, was built on outdated visual toolkit (Swing widget) and uses the Spring framework which is replaced by JavaFX as of now.
- Class structure was not well organized, no packages to divide the classes properly.
- Some variables and methods are not in use.

2.3.2. Other Factors

The design of the system does not make it easy to reuse, to rebuild it components and follow its pattern. Therefore the team decided to re-write the system in a more object oriented approach using a pattern oriented architecture MVC to separate the model and view which would also make it much more efficient to parallely work for developers. this will also make it easier to maintain or improve the system in the future. It will also be built using JavaFX which will be replacing the old framework the JHotel is currently using.

3. Requirements

3.1. Introduction

The following section, Software Requirement Specification (SRS) provides the overview of the required specification of the software Hotel Management System (HMS)

1.1 3.1.1. Purpose

The purpose of this SRS document is to give a detailed description of the requirements of the HMS. It clearly describes the HMS and its functionality that is expected from the software, which will help develop the correct desired software to be provided to the end user. This document will be used by the developers of the software themselves and the end users.

3.1.2. Scope

The software product to be produced is Linnaeus Hotel, a HMS is a hotel management system build to manage the front-desk activities. This application will replace the existing paper -based system to fulfil the customer desideratum as they believe an automated system would save their money and provide better services for their guests. The system task to manage reservations, track guest check in and out, and track room availability in both hotel location Växjö and Kalmar. The HMS should be easy to use and appropriate to manage hotel activities for end users.

3.1.3. Stakeholders

Customer : Linnaeus Hotel Management which is represented by LNU's teaching team.
User: Authorized and authenticated employee in the Hotel to use the system.
Admin: Verified and registered Manager in the Hotel which has full control over the application and data.
Developers : programmers who build and developed the system and have access during the development of the system.

3.1.4. Overview

The rest of this document contains an overall description of the Linnaeus Hotel System and the constraints the Specific Functional Requirements for the system and the Use case and scenario modeling for functional requirements. All the diagrams that are needed, will be attached to this document.

3.2. Overall Description

This section describes general factors that affect the product and its requirements. The specific requirements will be handled in the section 3.

3.2.1. Product Perspective

The Hotel Management System is an independent system, which means its totally self contained.

3.2.1.1. Hardware Interfaces

The Hotel Management System will be set up on PC's in front Hotel desk and Manager office.

3.2.1.2. Software Interfaces

A free online Database will be configured using MySQL software management. The Database includes hotel rooms, guest information, employees information and reservations (name, address, id number, telephone number, and credit card) which also can be modified by the end users. Database includes reservation number, checking -in date and check-out date and amount owed by guest.

3.2.2. Product Function

The Three main functions of the HMS are to allow user to enter reservations as well as to check guests in and out of both hotel's building in Kalmar and Växjö. For administrative purposes the manager has ability to add, edit and delete rooms and accounts to verify the user of the hotel front-desk.

3.2.3. User Characteristics

User should be verified as defined in 1.4, but should have at least basic knowledge in using computers. The system is friendly user, so no need to have any special training in HMS.

3.3. Specific Requirements

The Software Requirement Specifications will provide a detailed description of the requirements for the Hotel Management System, that when combined with the system use cases and use case description is sufficient to enable developers to design a complete system that satisfy the software requirements.

3.3.1. External Interfaces

The Hotel Management System shall use standard input/output devices for a PC. which includes a Monitor, keyboard, Mouse and Printer.

3.3.1.1. User Interface

The Hotel System Application is a desktop application and its user interface screens can be described as follow:

Screen name	Description

Login	log into the system either as a Hotel clerk or Manager.
Hotel Clerk Interface:	
Menu	shows todays check in and check out activities.
Search	find a room on certain date, room type, bed type , room size, room location show-room list, add from room list to selected room view,. reset search input, back to menu, and refresh
enter Reservation	view chosen rooms list, add guest information, check in and check out dates, confirm/cancel reservation, choose guest number,
Reservations	edit reservation by input clients ID number on search field, edit/save/cancel the reservation
Check in	confirm check in when arrival of guests
Check out	confirm check out when guest leaves.
logout	log out from the system.
Manager Interface:	
Main	menu rooms, accounts, go to employee interface..
Room	Add new room information, room id, price, room size number of beds, location, room type, cancel/create room.
Manage room	edit existing room, delete room, change rooms rate
Create account	enter employee or manager information, make username and password, cancel/create account.
Manage account	edit existing employee or manager information/change username and password, delete account.
Menu	take manager to employee interface to make reservations for special room or do the employee activities.
Log out	log out from the system.

3.3.1.2. Hardware Interfaces

The HSM shall run on Microsoft Windows based system.

3.3.1.3. Software Interfaces

The HSM shall interface with online MySQL database.

3.3.1.4. Communication Interface

The communication between database and the User Interface will be over internet. MySQL database will be interested remotely over an internet connection, this will be explained more in the design document.

3.3.2. Functional Requirements

Linnaeus Hotel system has functional requirements which are very specific based on the client desideratum. The main activity of the system is to manage reservations for the hotel and organize the facility work to receive guests in the most comfortable and sufficient way.
Therefore, the system should be able to perform several activities in order to manage the reservations of the hotel's rooms.
Priorities:

- 1 = must important
- 2 = should be implemented

***ID* #**	**Description**	**Priority**
FR1	System shall accept room reservations.	1
FR2	System shall allow to modify reservations.(Edit or canceling the reservation).	1
FR3	System shall record the check-in date for a guest on the day of his arrival.	1
FR4	System shall record check-out a guest in the day of his departure.	1
FR5	System shall record guest information to make reservation, all the informations should be valid (Name, identification number(passport or Swedish ID), telephone number, and Credit Card). The validation is done by the User or Admin.	1
FR6	System shall allow guest information to be modified (Edit or delete).	1
FR7	System shall allow available rooms to be searched based on the client's preferences.	2

FR8	System shall allow user to check room availability, on the dates that the client wishes to book a room.	1
FR9	Systems shall allow new room to be created by Admin a with choosing room options (Room ID, Price, Size, Location, Room Type, Bed Type).	2
FR10	System shall allow room rate be adjustable by Admin.	1
FR11	System shall allow Admin to search and book a specific room for a guest.	1
FR12	System shall accept to book adjoint rooms.	1
FR13	System shall verify if rooms are available in both buildings on the chosen dates, when user is searching to make a reservation.	1
FR14	System shall allow access to different Hotel branch to offer a room to stay within the other campus, when there are no rooms available meeting the searching criteria in the campus that the client is searching for.	1
FR15	System shall print bill when the user is checking out a client.	1
FR16	System shall prints bill when a reservation is cancelled, declaring if the client should pay any fees depending if the cancellation is made to late.	1
FR17	System shall alter and store reservation´s information after the user edits a reservation.	2
FR18	System shall charge the guest if cancellation is made within 24 hours from the date of check-in. system deduct 50% of the reservation fee.	1
FR19	System shall allow Admin to add new user account	2
FR20	System shall allow Admin to modify user account	2
FR21	System shall allow room to be modified by Admin and change room information	2

3.3.3. Non- Functional Requirements

In this section, quality requirements is listed to describe the system attributes to meet the stakeholders expectations. In our case, the hotel reservation system should posses the functional requirements which describes the system needs in terms of performance, database requirements, availability, security, design constraints, standard compliance, reliability, maintainability, portability, learnability.

3.3.3.1. Performance Requirement

The performance requirement describes the responses times that are acceptable for system functionality

- System shall check for room availability in less than 2 seconds.
- System shall keep track of the guest’s account and print his or her bill in less than 60 seconds.
- System shall logn in user/admin in less than 5 seconds

3.3.3.2. Database requirements (Data retention)

logical database requirements which includes the following data elements.

- Guest name
- Guest Identification number.
- Guest address.
- Guest telephone number
- Guest Credit Card number.
- Check in date
- Check out date
- Bed type
- Room type
- Room ID
- Room size
- Room price
- Room location
- Room Bill
- Employee account
- Manager account

3.3.3.3. Availability

The system shall be available under working hours of the hotel, it depends on the internet connection availability inside the hotel to connect to the database management system.

3.3.3.4. Security

System shall be secured by limiting the access with logging authentication, only Hotel Clerk and Manager are able to view the guest information. Manager shall have access to the Management subsystem which allow him to manage rooms and reservations which is protected by username and password. User has limited access to book, edit , delete and check reservations.

3.3.3.5. Design Constraints

The System shall be running in a Microsoft Windows 10 or Mac OS X environment. the system shall be developed using Java and MySql online database.

3.3.3.6. Standard Compliance

GUI shall have the same design to give a consistent look and feel.

3.3.3.7. Usability

The HMS is user friendly, easy to use and understand its functionality.

3.3.3.8. Maintainability

The HMS is developed in Java an object oriented programming language and shall be easy to maintain.

3.3.3.9. Portability

The HMS shall run in any Microsoft Windows 10 or Mac OS X environment that contains Java Runtime and internet connection to connect MySql database.

3.3.3.10. Learnability

The Hotel Manager System shall be easy to learn and to understand where a new user should require about 30 minutes of training.

3.3.4. Scenarios

These scenarios are based on the client requirement.

3.3.4.1. Scenario 1

Tony visit Linnaeus university in Växjö and decided to stay in Linnaeus Hotel for two nights from 11 April till 13 April. Tony walks to reception desk in the hotel and ask the hotel clerk Andesh to make him a reservation. Andesh nicely ask Tony for from which date to which date he would like to stay in the hotel, and what kind of room he would like to have and how many guest to expect. Tony chose to have a single room with smoking allowed. Andesh enters location växjö and look for available rooms with same specification in the system, after inputting the required choices made by the guest. Room is found, so Andesh inform Tony the price. Tony accepts and asks to book the room. Then Andesh asks Tony for his passport or identification card to register his information, and asks for his address, phone number and credit card. Andesh enters the information of the guest into the system, finally a reservation is made and the guest will be given the room keys, and Andesh will confirm Tony check in in the system. on check out date Andesh confirm check-out activity in the system and the system will print a bill for Tony to pay the room price.

3.3.4.2. Scenario 2

Yoel and Beysim made a room reservation in Hotel Linnaeus through the phone, but they have change in plans and will visit other city instead of Växjö. Yoel calls the reception desk to cancel the reservation. Andesh finds the reservation by searching Yoels unique id number that is registered in the system and cancel the reservation. the system checks if the cancellation is made before 24 hours or within to deduct 30% of the room price according to the days of stay in their reservation. The reservation is cancelled before 24 hours from the check in date so its is cancelled from the system and no bill will be needed to print out. The room status will change to available for another guest.

3.3.4.3. Scenario 3

Alex visit Hotel Linnaeus to make a group of 4 guest reservation, so he walk to the front desk and find Andesh. Alex ask for a good room that fits for total 5 guest which includes him as well. Andesh ask nicely the reason of the visit to suggest good rooms, Alex informs Andesh that they are a group of Japanese students coming to visit for a friend graduation so any proper room with no smoking and good view of the lake would be nice. Andesh search in the system and find an adjoint room for 3 guest and on double bed room with good view and a bit expensive. Andesh suggest those rooms to Alex and Alex accepts, then Andesh makes the reservation. Few days later, Alex called the Hotel to edit his reservation from 5 guests to only 4 as he decided to stay at a friend place in Växjö. Therefore, he asks to change the double room to single room and change the reservation to a different guest name. Andesh check the reservation and edit the guest information with the new information given by the Alex's Japanese friend , then cancel the double room in the system. Afterwards, he makes a new reservation after he found a single room with similar description as Alex asked for and make the reservation.

3.3.4.4. Scenario 4

Waeel has visited Hotel Linnaeus several times before. He likes a certain room where he enjoy the view and smoking his cigar. So Waeel call the reception desk to ask Andesh to make him a specific room reservation, but Andesh has no authorization in the system to do such a room search. Andesh asks Waeel to hold a minute so he would transfer him to the manager which has such access to the system. The Manager speaks to Waeel and asks him what is the room number he is looking for. The Manager search for the room and check if it's available or not, then he tells Waeel when the specific room is available, Waeel agrees on the date and the Manager makes the reservation himself.

3.3.4.5. Scenario 5

The manager has redecorate the rooms and update their types, so firstly he wanted to change room V107 from single and non-smoking room to smoking room as there are alot of demand on such rooms. The Manager goes to the system to his account and choose to manage rooms and update the room type. in addition, he has a storage room in the second floor that is empty and can make a guest room to rent for the hotel, so he goes to the system using his account to add new room and enter the information needed to use this room (room id, room price, room type, bed type, and room size) then create room, the system now includes a new room.

3.3.4.6. Scenario 6

Narges is a new employee that will work night shifts in the hotel as Andesh works on day shifts. The Manager will ask Narges to come his office and make her an account in the system so she could use the Hotel Management Software for booking. Narges goes to the Manager office, The Manager ask her to give him her ID to register her information in the system, The Manager will go to his account in the system and enter her information (name,id number, address, telephone) then create her a unique username and password to access the system.

3.3.4.7. Scenario 7

Narges in her first day of work forgot her username and password, so she goes to the Manager and ask him for it. The Manager finds it and suggest he would make her an easier one that she can remember easily. So the Manager go to his account and change the username and password to new one and the system save it. Now Nerges could access the system.

3.3.4.8. Scenario 8

Andesh wants to move to his girlfriend in Stockholm, and he found a job there. He resigned from his job at the reception desk. The Manager goes to his account and delete Andesh account from the system. Now Andesh can not access the system anymore.

3.3.5. Actors and their tasks

In this section, the actors that are interacting with the system should be defined and breakwond their tasks using Hotel Management System.

1- Hotel Clerk (User):

- Enter reservations in the system, from guest information, date of arrival and departure, and the type of room requested by the guest.
- Search for rooms on the date of arrival and departure requested by the guest, room type and guest number.
- Edit or cancel reservations.
- Check in the hotel guest.
- Checks out the hotel guest and print the room bill.
- Checks all the guests that are going to checks in and out during the day.
- Check room availability and room location either in Växjö or Kalamar.

2- Hotel Manager (Admin):

- Create accounts for the employee who are going to use the Hotel Management System.
- Edit or delete accounts for employees, incase someone would change his information or forgets his username and password. Delete incase the employee has resigned for the job.
- Create new rooms in the system, either if the room redecorates and has new bed types or if the hotel has changed the building and has new formation of rooms.

- Change the rate of the room in the system.
- update or delete rooms, in case of the management has new way of distributing the bed types or would change from smoking to non smoking. Delete room incase the room is no longer to be used for guests and would be made as storage or something else.
- Make a reservation for special guests who ask for specific room, which the the clerk has no access to such a reservation in the system.
- Make the normal activities as the clerk does in the system by accessing his interface from manager account.

3.3.6. Use cases

The following use cases are derived from the scenarios.

3.3.6.1. Overview Use-cases

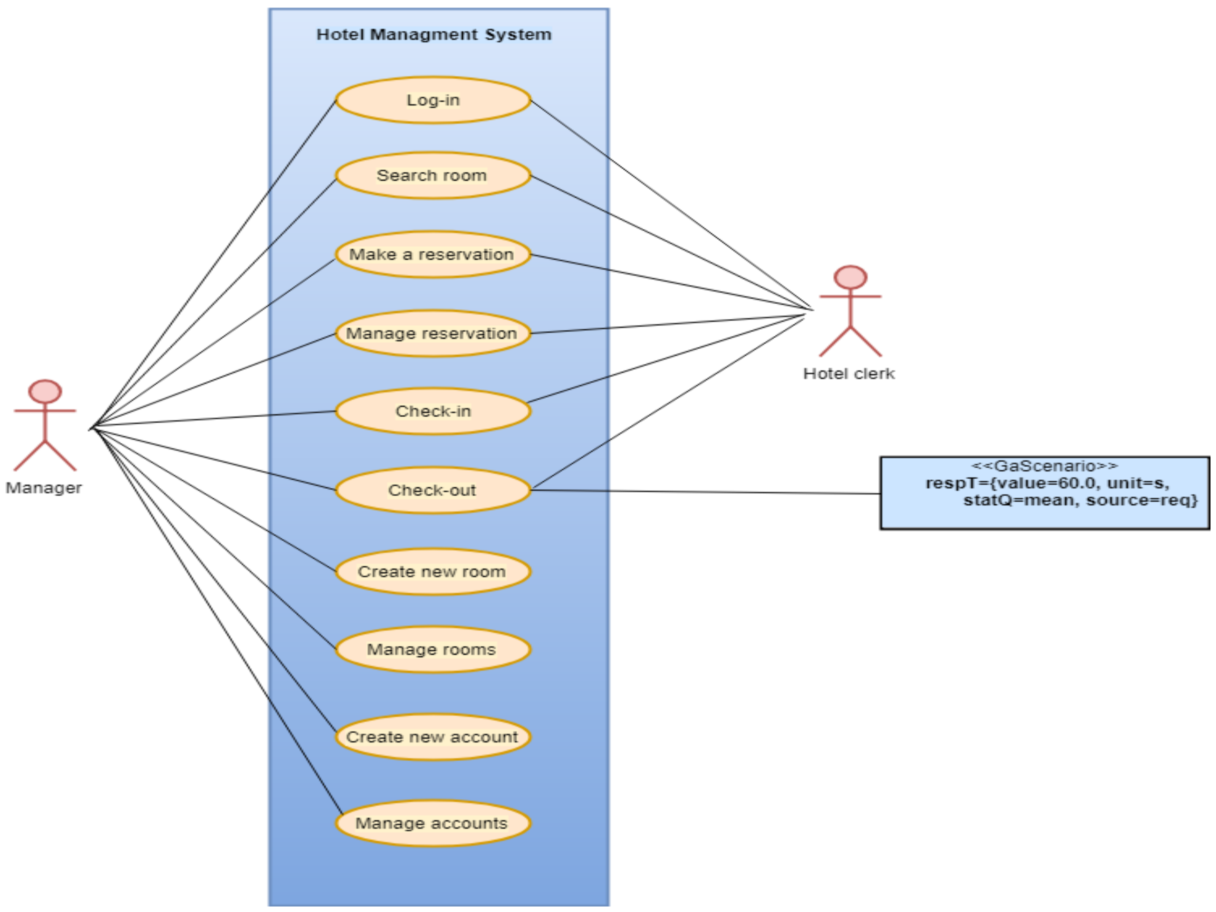


Figure 1: Use-cases diagram

Overview of the Use-cases that is made for the Hotel Management System and the actors who interact with the system.

3.3.6.2. Use-case 1

Use-Case Name	Login
ID requirement	3.3.4, 3.3.1
Actors	User, Admin
Pre-Condition	User has an account with username and password save in the system.
Flow of Events	1- User enter his username and password. 2- System shows User interface(menu). 3- User can use the system to book rooms .
Post condition	User access the software application.
Main Scenario	1- The User wants login to start using the software. 2- User opens the system. 3- System asks for authentication to access. 4- User will enter his uniques username and password.

	5- System validate his information. 6- System allows the user to access the system.
Alternative Scenario	1- The wants login to start using the software. 2- Admin opens the system.. 3- System asks for authentication to access. 4- Admin will enter his uniques username and password. 5- System validate his information. 6- System allows the Admin to access the system.
Error Scenario	User or Admin enters a mistake username or password, they system will display an error message.

3.3.6.3. Use-case 2

Use-Case Name	Search room
ID requirement	FR3, FR4, FR7, FR8, FR12, FR18
Actors	User
Pre-Condition	User has access to the software.
Flow of Events	1- User check rooms list 2- System shows all rooms available/non-available in the list. 3- User choose hotel location to search room. 4- User make a reservation 5- System save the reservation.
Post condition	Room is chosen with with check in and check out dates in chosen location Växjö/Kalmar..
Main Scenario	1- The user receives the guest order of type of room and dates of arrival and departure. 2- The user check room list to find an available room in the system. 3- System display available rooms that match the search in less than 2 seconds. 4- System will request to add guest information. 5- The user adds the guest information to the system (Name, address, telephone number, credit card and passport number). 6- Then the user chose the date of check in and out 7- User book It. 8- System will save the new reservation with a unique reservation number.
Alternative Scenario	1- User search for rooms in Växjö but it's all booked. 2- User choose different location (Kalmar) to find available rooms.

	3- System show the available room in Kalamar. 4- User book the room and system save operation.
Alternative Scenario 2	1. User wants to book adjoined rooms for the client 2. User searches for adjoined rooms in the given dates. 3. System checks if there are available adjoined rooms during those dates 4. System shows available adjoined rooms 5. System performs the adobe step in less than two seconds. 6. User books the adjoined rooms
Error Scenario	1- The user enters guest info incorrectly in the system, so error messages from the system will popup, so the user cancel the reservation. 2- The systems shows no rooms available on required dates, so the user cancel the reservation. 3- all rooms are booked in chosen location, the system will send a pop up window to notify the user.

3.3.6.4. Use-case 3

Use-Case Name	Make a reservation
ID requirement	FR1, FR3, FR4, FR5, FR7, FR8, FR11, FR12, FR14, FR17, 3.3.1, 3.3.3
Actors	Hotel clerk(User). Hotel Manager(Admin)
Pre-Condition	User has access to the software.
Flow of Events	1- User check rooms list 2- System shows all rooms available in the list. 3- System shows available rooms in less than two seconds. 4- User choose an available room. 5- User enter guest information and reserve it. 6- System save the reservation.
Post condition	Room is reserved under guest name with unique reservation id.
Main Scenario	1- The user receives the guest order of type of room and dates of arrival and departure. 2- The user check room list then chose the available room that match the guest order in 2 seconds. 3- System display rooms then user add guest 4- System will request to add guest information. 5- The user adds the guest information to the system (Name, address, telephone number, credit card and passport number). 6- Then the user chose the date of check in and out 7- User book It.

	8- System will save the new reservation with a unique reservation id.
Alternative Scenario	1- Guest want a certain room with specific status. 2- User asks for Admin to make the reservation 3- Admin log in with his account to the system 4- System shows the admin a unique search for rooms. 5- Admin finds the room, system shows if room is available or not. 6- Admin make reservation if available and system save the reservation.
Alternative Scenario 2	1. Guest wants to book adjoined rooms 2. The user check room list then chose the available adjoined rooms that match the guest order in 2 seconds. 3. System display rooms then user add guest 4. System will request to add guest information. 5. The user adds the guest information to the system (Name, address, telephone number, credit card and passport number). 6. Then the user chose the date of check in and out 7. User books It. 8. System will save the new reservation for the adjoined rooms with a unique reservation id.
Error Scenario	1- The user enters guest info incorrectly in the system, so error messages from the system will popup, so the user cancel the reservation. 2- The systems shows no rooms available on required dates, so the user cancel the reservation.

3.3.6.5. Use-case 4

Use-Case Name	Manage reservation
ID requirement	FR2, FR6, FR16, 3.3.2
Actors	Hotel clerk (User)
Pre-Condition	1- There is a room reservation in the system 2- User find reservation with a unique reservation number.
Flow of Events	1- User check reservato room list. 2- User find reservation and change the info that wanted to change then save or delete to cancel reservation. 3- System will send a confirmation message.
Post condition	Room list is displayed with updated info.
Main Scenario	1- Guest want to edit his reservation (Cancel, change guest info or change check in/out date).

	2- User go to reservation list and find the guest's reserved room through searching with reservation number. 3- System show information of the room and guest. 4- User edit reservation and fill new info. 5- User save or cancel the whole reservation in the system. 6- The system will send a message for confirmation. 7- System will update the room list with the new data. 8- If guest cancelled less than 24 hours before check in date 15% percent of room price will be charged.
Alternative Scenario	1- Guest want to switch room or upgrade his room. 2- User delete first reservation from the system 3- User choose new room from the list. 4- Room is available in the system 5- User make reserve the new room with guest info.
Error Scenario	1- User enter wrong reservation number, error message will show. 2- User update information but forget some fields, error message will show

3.3.6.6. Use-case 5

Use-Case Name	Check in
ID requirement	FR3
Actors	Hotel clerk, Hotel Manager
Pre-Condition	Guest has reservation for a certain room Guest has an Id or passport for identification
Flow of Events	1- User check guest reservation in Today check in list. 2- Guest info and room type display on system. 3- User confirm check- in room in the system. 4- System save the operation.
Post condition	System save the operation in guest enters the room.
Main Scenario	1- The user checks the reservation in Today check in list. 2- System shows all the check-ins for the day in view list. 3- The user confirms guest arrival in system. 4- System save the operation.
Alternative Scenario	1- User could not access the system because he forgets his username. 2- Manager goes the user interface in the system and do the check-in 3- System saves the operation.
Error Scenario	User try to check in with no reservation is made, system shows error messages.

3.3.6.7. Use-case 6

Use-Case Name	Check out
ID requirement	FR4, FR15, 3.3.1
Actors	Hotel clerk, Hotel Manager
Pre-Condition	Guest has checked in a room
Flow of Events	1- Guest want to check out of room. 2- System shows the user today check out list. 3- User confirm check out in the system. 4- System track the guest account and print the room bill. 5- System performs the adobe step in less than sixty seconds.
Post condition	System will update the room status to available.
Main Scenario	1- System will show the daily activity of check out of the day in the view list. 2- After the guest check out the user confirm check out in the system on the room in the room list. 3- System will save the operation. 4- System track the guest account and print the room bill. 5- User print room's bill and charge the guest. 6- System performs the adobe step in less than sixty seconds.
Alternative Scenario	1- Guest wants to stay more days in the hotel 2- User will make a check out in the system. 3- User try to search for new reservation in the system.
Error Scenario	1- User make a check out with no reservation, system will send error message. 2- System can not know if credit cards are valid or not.

3.3.6.8. Use-case 7

Use-Case Name	Manage rooms
ID requirement	FR21, 3.3.2
Actors	Manager (Admin)
Pre-Condition	1- Manager has access to system 2- Rooms exist in room list.
Flow of Events	1- Admin goes to room list 2- Admin choose room that wanted to change its information. 3- System display room info 4- Admin change the room status and save changes in system 5- System sends a message for confirmation

Post condition	1- System will update the room status to new status. 2- User can view new rooms in room list.
Main Scenario	1- Admin wish to change rooms status from smoking to non-smoking etc. 2- Admin has a private access to room list in System. 3- Admin goes to room list choose the room he wants to update 4- Admin update the room and save the operation 5- System will save the operation to room list 6- New updated room list is shown on the system.
Alternative Scenario	1- Rooms has a damage by guest or has constructional problems or will change to storage. 2- Admin access to the room list. 3- System display rooms in room list. 4- Admin delete room from the system. 5- System save update to room list. 6- System will not view this room till its updated again by Admin.
Error Scenario	User delete some fields and leave some empty, system will send an error message.

3.3.6.9. Use-case 8

Use-Case Name	Create new room
ID requirement	FR9, 3.3.2
Actors	Manager (Admin)
Pre-Condition	1- Manager has access to system 2- Room doesn't exist in room list.
Flow of Events	1- Admin go to his account in the system 2- Admin ask system to add room. 3- System shows empty fields to add room. 4- Admin create the room and save it.
Post condition	1- System will save the new room with its new attribute. 2- Room is available in room list in the system.
Main Scenario	1- Admin want to add a new room in the system. 2- Admin access through his account to add a new room. 3- Admin fill the information for the new room (Room number, room price, room size, room type, number of bed, location) 4- Admin save the operation in the system 5- System save the new room in room list and show it to User.
Alternative Scenario	1- Admin want to create an adjoint room. 2- Admin access through his account to add a new room.

	3- Admin fill the information for the new room (Room number, room price, room size, room type, number of bed, location) 4- Admin choose adjoint room in the system. 5- System asks to add which rooms to be adjoint. 6- Admin choose suitable rooms and save. 7- System save the new adjoint room in room list and show it to User.
Error Scenario	1- Admin did not fill all the information, system send error message. 2- Admin enters room number that already exist, system send error message.

3.3.6.10. Use-case 9

Use-Case Name	Create account
ID requirement	FR19, 3.3.2
Actors	Manager (Admin)
Pre-Condition	1- Manager has access to system 2- Account doesn't exist in System..
Flow of Events	5- Admin go to his account in the system 6- Admin ask system to create account. 7- System shows empty fields to create account. 8- Admin create the account and save it.
Post condition	1- System will save the account for the new user.. 2- User can log in to system using the new account.
Main Scenario	1- Admin has a new user to add in the system. 2- Admin access through his account to create new account. 3- Admin fill the information for the new room (Name, id number, address, phone number, user name , and password). 4- Admin save the operation in the system 5- System save the new account in account list. 6- User log in the system using his username and password
Alternative Scenario	1- Admin wants to add a new admin to the system. 2- Admin access through his account to create new account. 3- Admin fill the information for the new room (Name, id number, address, phone number, user name , and password, manager). 4- Admin save the operation in the system 5- System save the new account in account list. 6- New admin log in the system using his username and password
Error Scenario	1- Admin did not fill all the information, system send error message. 2- Admin enters username or password that already exist, system send error

	message.

3.3.6.11. Use-case 10

Use-Case Name	Manage accounts
ID requirement	FR20, 3.3.2
Actors	Manager (Admin)
Pre-Condition	1- Manager has access to system 2- Account exist in System.
Flow of Events	1- Admin go to his account in the system 2- Admin ask system to update account to change.. 3- System shows account list . 4- Admin find the account to be changed . 5- Admin change it and save it.
Post condition	1- System will save the updated account with new changes. 2- User can log in to system using the updated account.
Main Scenario	1- Admin has a user who want to change his username or password or edit his phone number. 2- Admin access through his account to update account. 3- Admin change the user username or password or phone number. 4- Admin save the operation in the system 5- System save the updated account in account list.
Alternative Scenario	1- Admin wants to delete a user account who no more works in the hotel. 2- Admin access through his account to update account. 3- Admin delete the account in the system. 4- System save the operation in account list. 5- User is deleted from account list. 6- User can not log in the system anymore.
Error Scenario	1- Admin did not fill all the information, system send error message. 2- Admin delete without choosing a certain account, system send error message.

3.3.7. Diagrams for use cases

In this section, use cases are going to be explained in form of diagrams to show the functionality of every use cases. State machine diagrams and Activity diagrams are presented to model the behaviour of the system.

3.3.7.1. Log-in

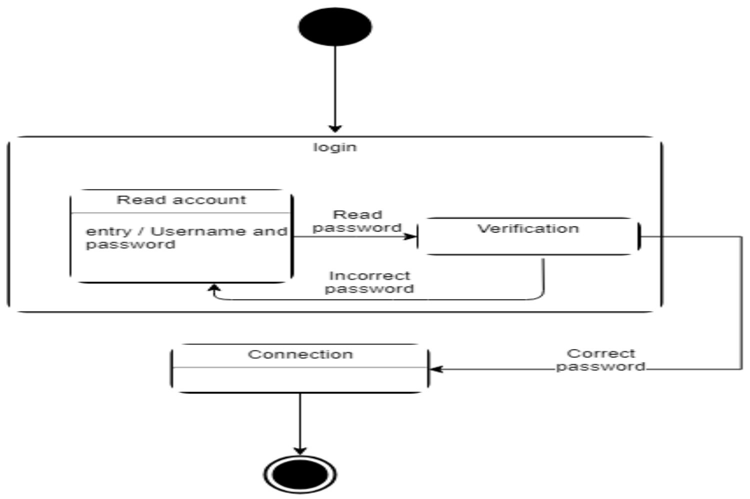


Figure 2: Log-in State Machine diagram

The diagram represent the changes in the system when the system log in a User to use the system.

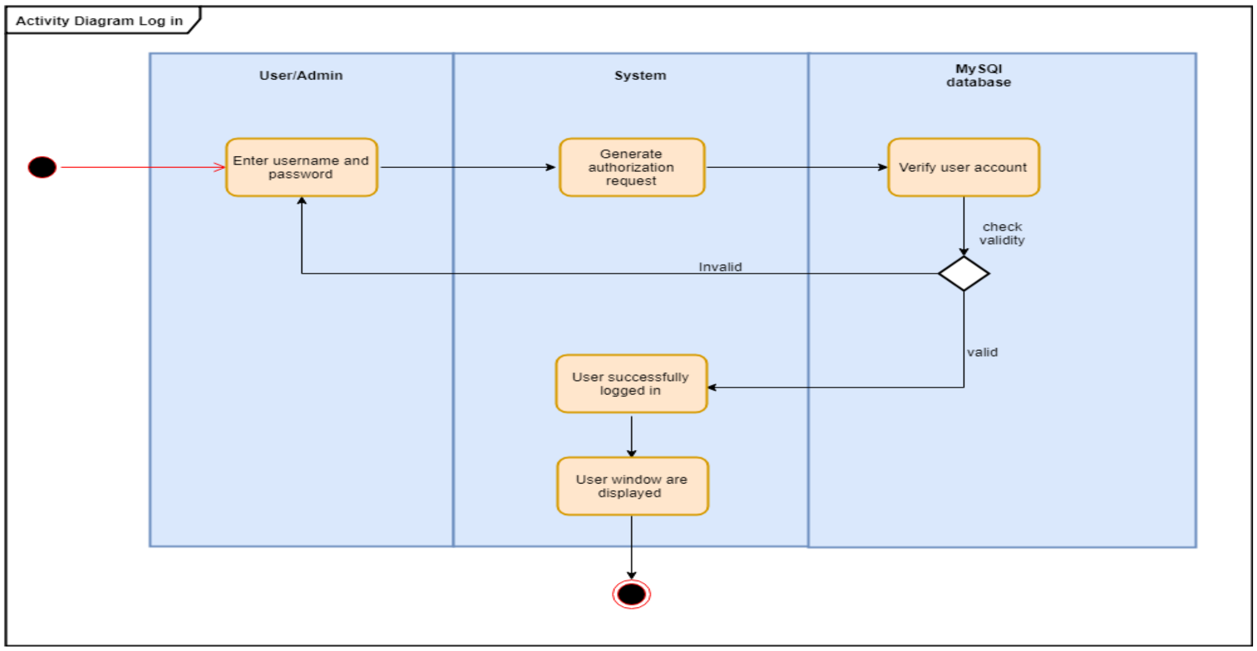

Activity Diagram Log in
User/Admin
System
MySQl database
Enter username and password
Generate authorization request
Verify user account
check validity
Invalid
valid
User successfully logged in
User window are displayed

Figure 3: Log-in Activity diagram

The activity starts when User enters his username and password. System receive the information and request from remote database to verify if the account valid or not. Then the system receive the validation from the database and allow user to access the system and activity ends.

3.3.7.2. Search room

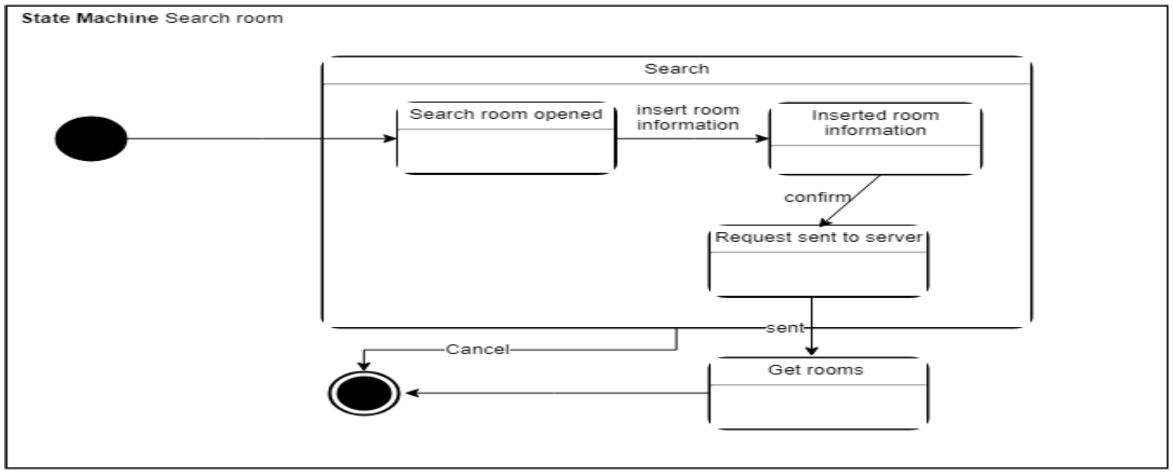


Figure 4: Search room State Machine diagram

The diagram represent the changes in the system when searching for a room and view the room list with the new results.

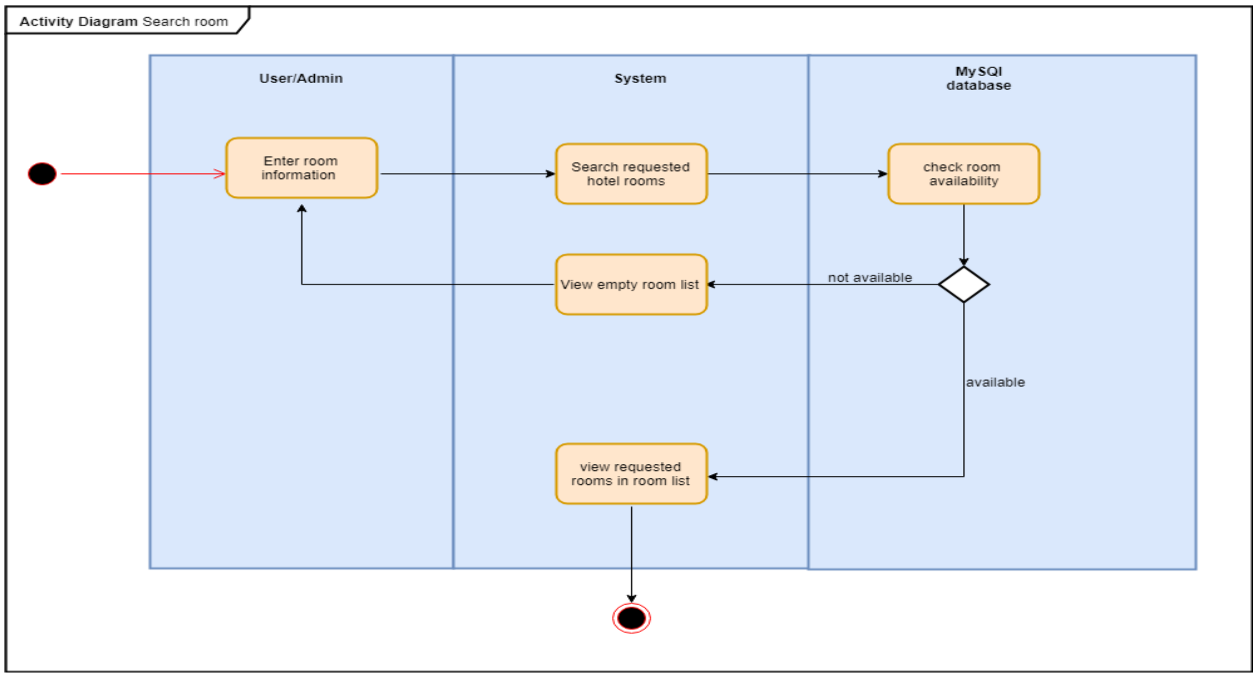


Figure 5: Search room Activity diagram

To search a room for a guest, the activity starts when the User enters room information according to guest demand (room type, check-in date, check out date, hotel location, and bed type). System will read the input and send it to remote database to check rooms availability, if room is available database will send empty room list otherwise it will display the available room on given date and activity ends.

3.3.7.3. Make a reservation

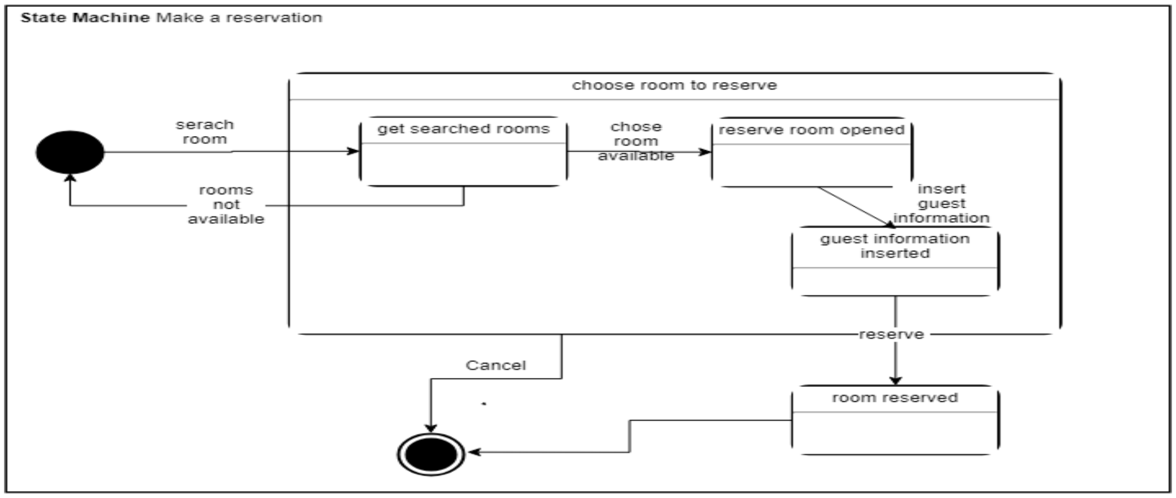


Figure 7: Make a reservation State Machine diagram

The diagram represent the changes in the system when reserving a room in the system.

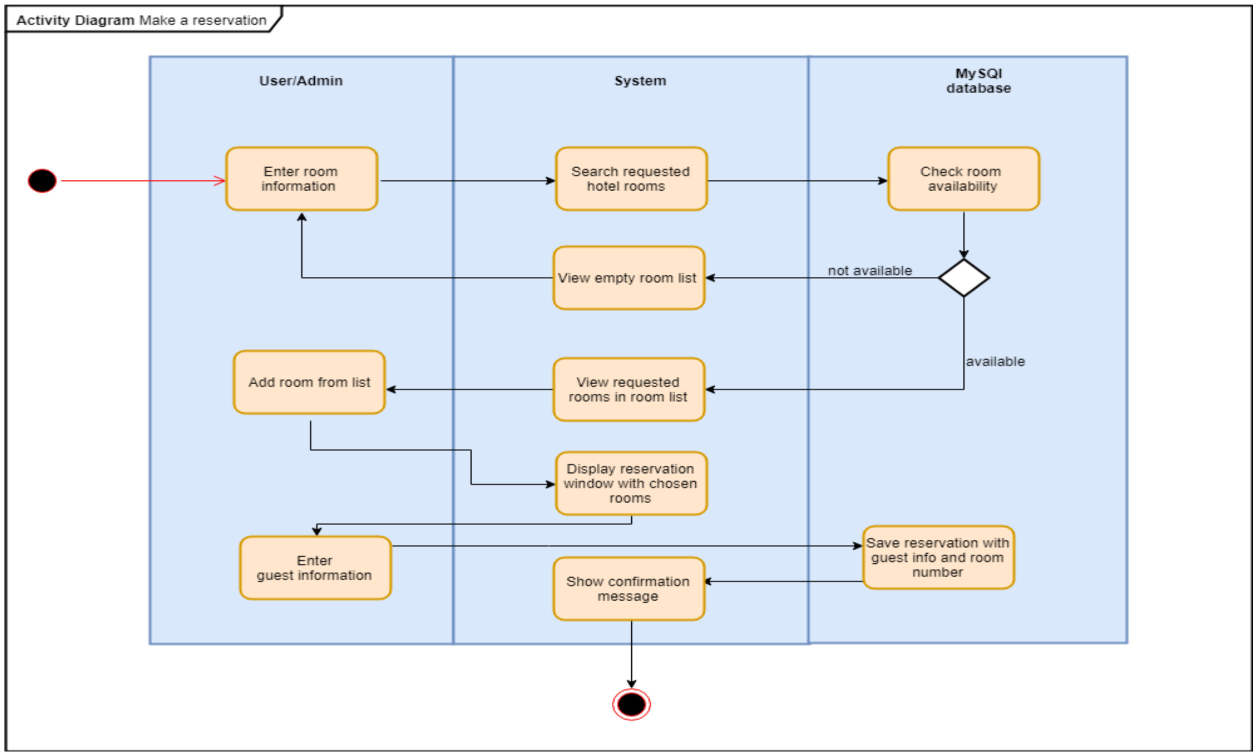

Activity Diagram Make a reservation
User/Admin
System
MySQl database
Enter room information
Search requested hotel rooms
Check room availability
View empty room list
not available
available
Add room from list
View requested rooms in room list
Display reservation window with chosen rooms
Enter guest information
Save reservation with guest info and room number
Show confirmation message

Figure 8: Make a reservation Activity diagram

The activity starts when User make a reservation for a guest, so User search for room and if room is available the system will display the reservation window and User shall enter the guest information. The reservation will be sent to the database to be saved with unique reservation id, then the remote database will send a confirmation message that the reservation is made and activity ends.

3.3.7.4. Manage reservation

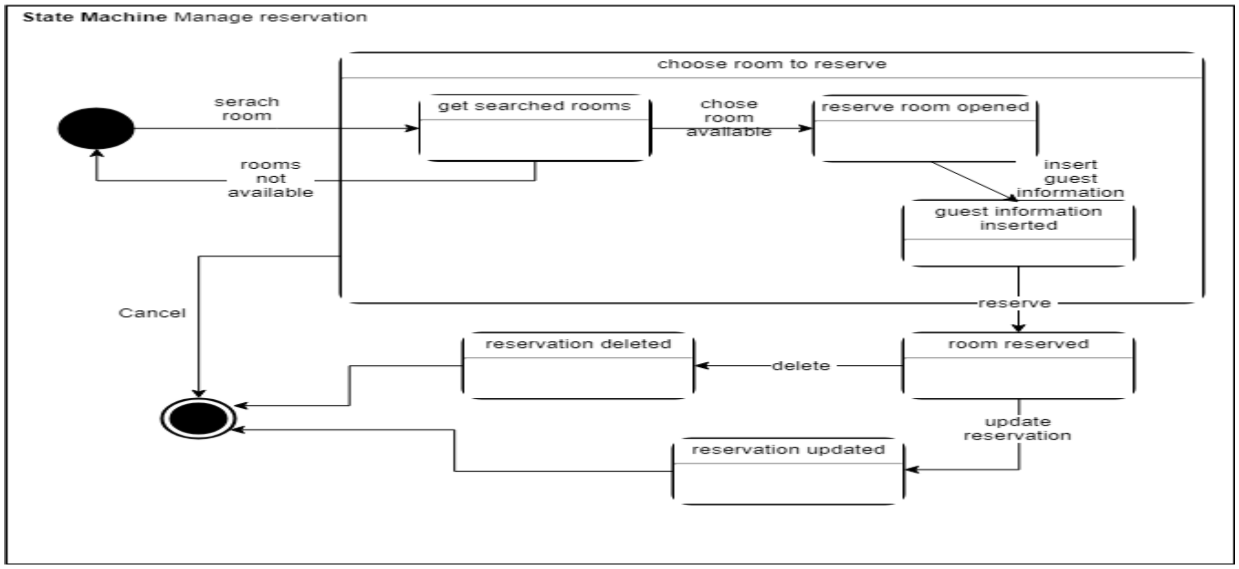


Figure 9: Manage reservation State Machine diagram

The diagram represent the changes in the system when reservation is updated or delete it from the system.

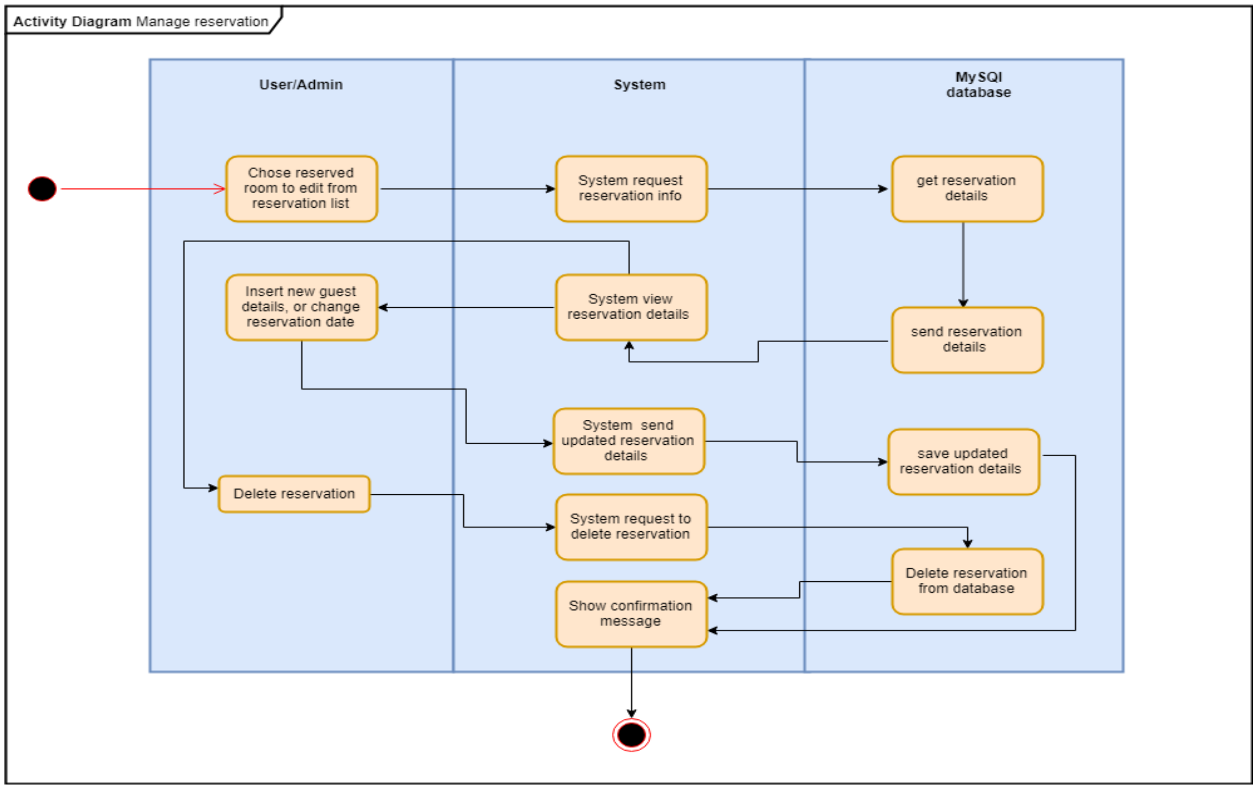

Activity Diagram Manage reservation
User/Admin
System
MySQl database
Chose reserved room to edit from reservation list
System request reservation info
get reservation details
Insert new guest details, or change reservation date
System view reservation details
send reservation details
System send updated reservation details
save updated reservation details
Delete reservation
System request to delete reservation
Delete reservation from database
Show confirmation message

Figure 10: Manage reservation Activity diagram

To manage a reservation, the activity starts when the User chooses a reservation from reservation list to edit it or delete it. System requests reservation information from database to display it. System show reservation window where User can either edit the guest information or delete reservation.

User decides the action is sent through the system to database and is saved there. Database sends confirmation message and activity ends.

3.3.7.5. Check-in

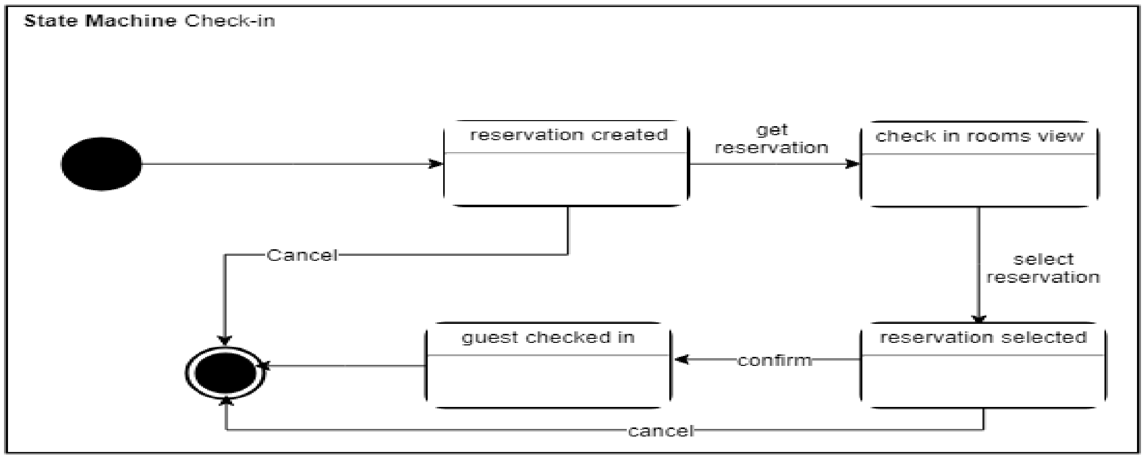
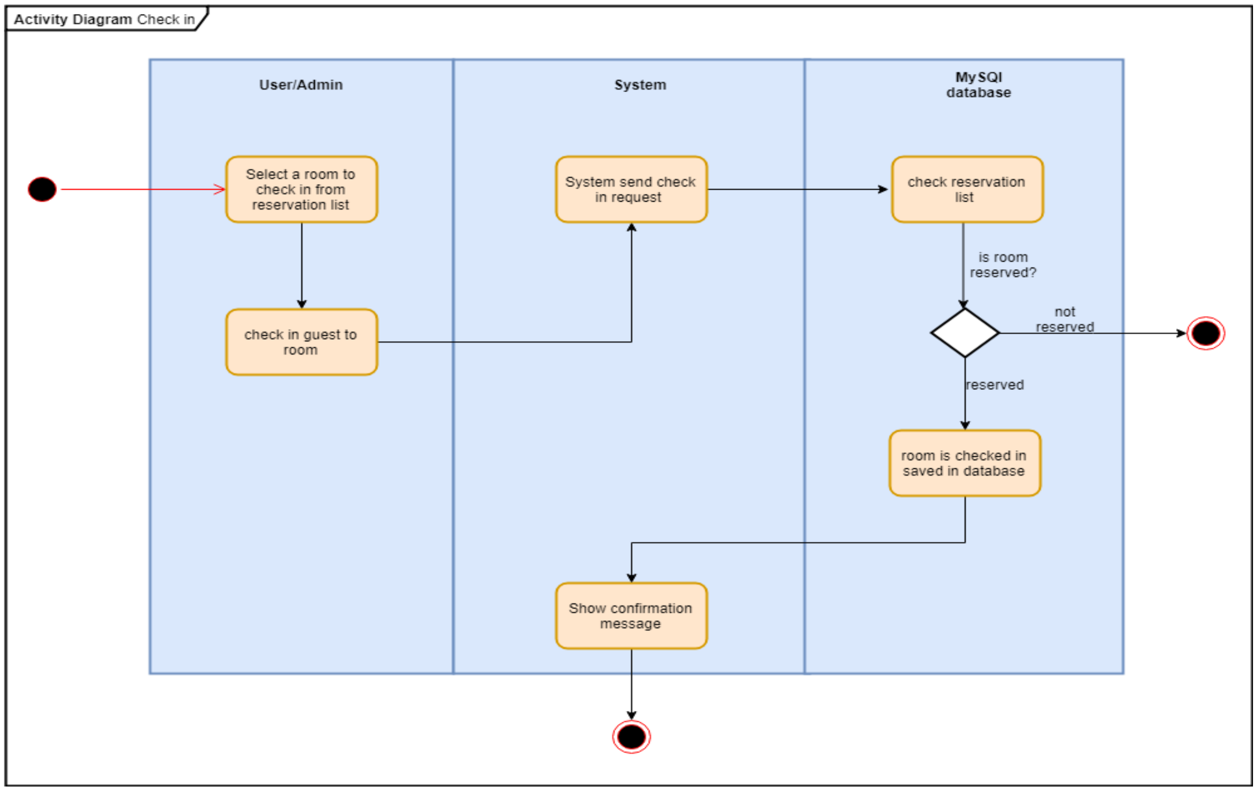


Figure 11: Check-in State Machine diagram

The diagram represent the changes in the system when the system check-in a guest



Activity Diagram Check in
User/Admin
System
MySQl database
Select a room to check in from reservation list
System send check in request
check reservation list
is room reserved?
check in guest to room
not reserved
reserved
room is checked in saved in database
Show confirmation message

Figure 12: Check-in Activity diagram

The activity starts when User check in a guest by going to reservation list and chooses the rooms that is going to be checked-in for the guest. System sends check in request to database to check reservation list and change the room status to check in, then sends confirmation message to the system to display to the user and activity ends.

3.3.7.6. Check-out

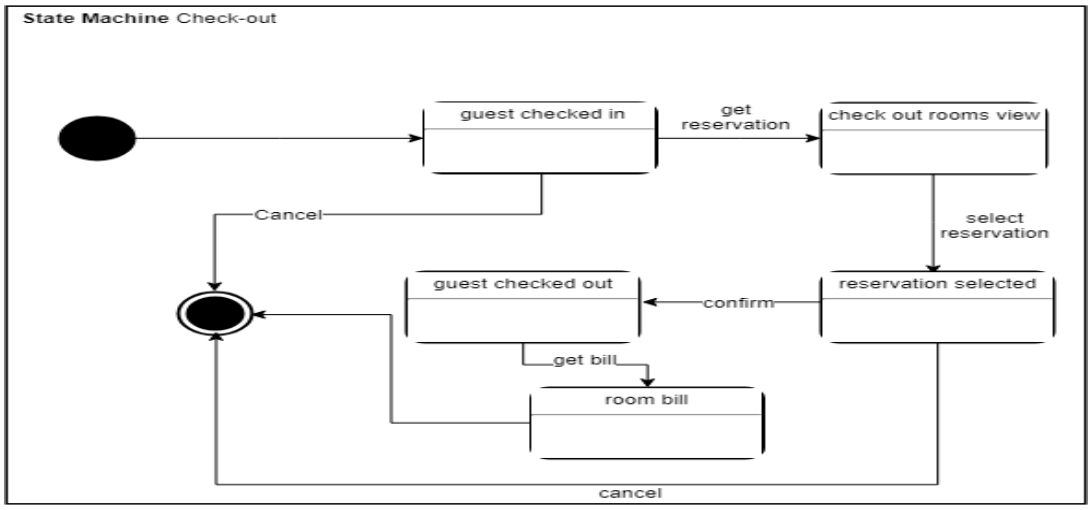


Figure 13: Check-out State Machine diagram

The diagram represent the changes in the system when the system check-out a guest and print out the guest bill and change the room status.

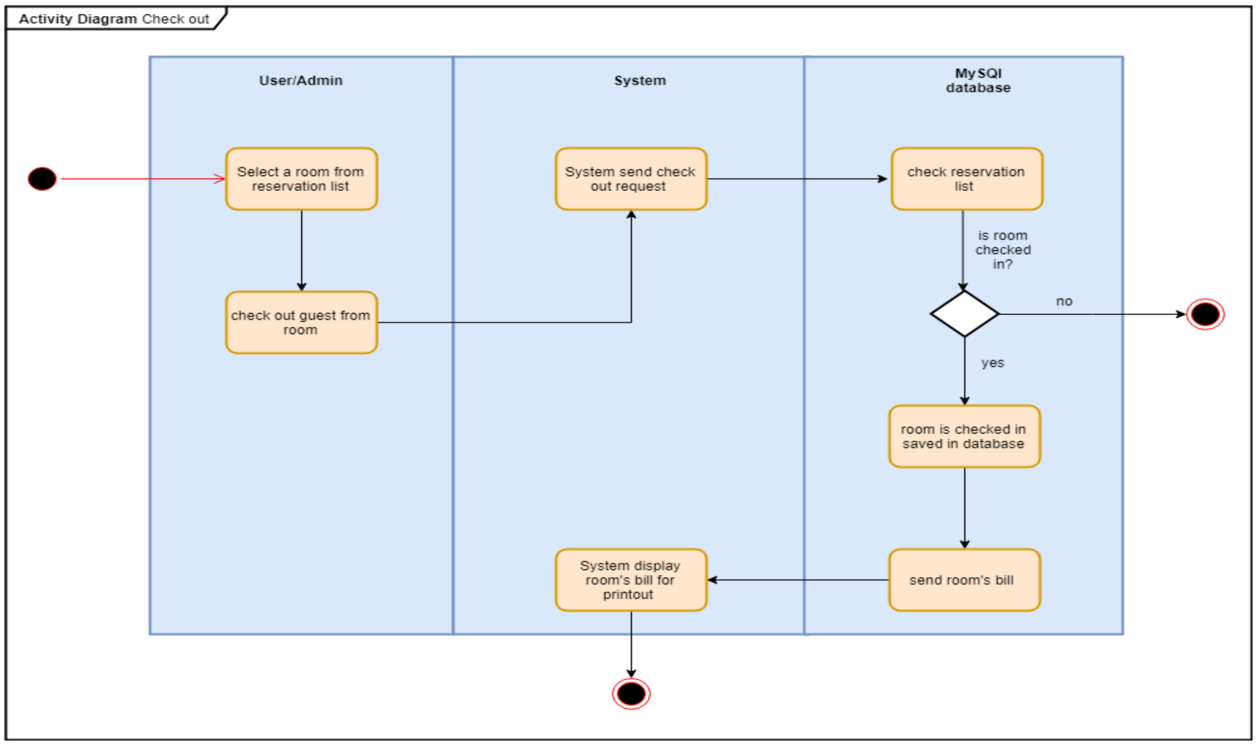

Activity Diagram Check out
User/Admin
System
My SQl database
Select a room from reservation list
check out guest from room
System send check out request
check reservation list
is room checked in?
no
yes
room is checked in saved in database
send room's bill
System display room's bill for printout

Figure 14: Check-out Activity diagram

Activity starts when User selects a room from reservation list to check out guest. System sends a check request to database where it check the reservation list and make sure guest is checked in the room. Database saves the check out activity and return room to available status. Database sends a bill to the system to display and activity ends.

3.3.7.7. Create new room

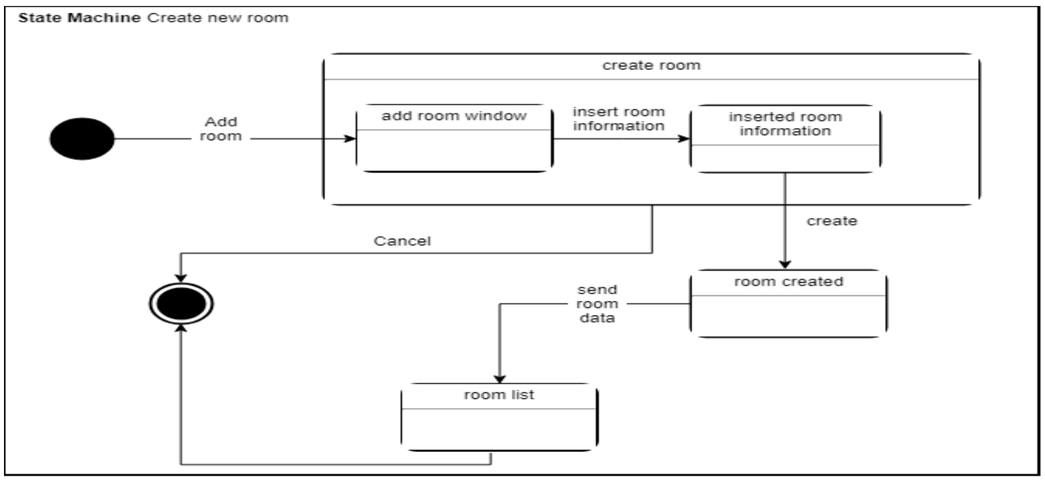


Figure 15: Create new room State Machine diagram

The diagram represent the changes in the system when the system creates a new room and save it in the database.

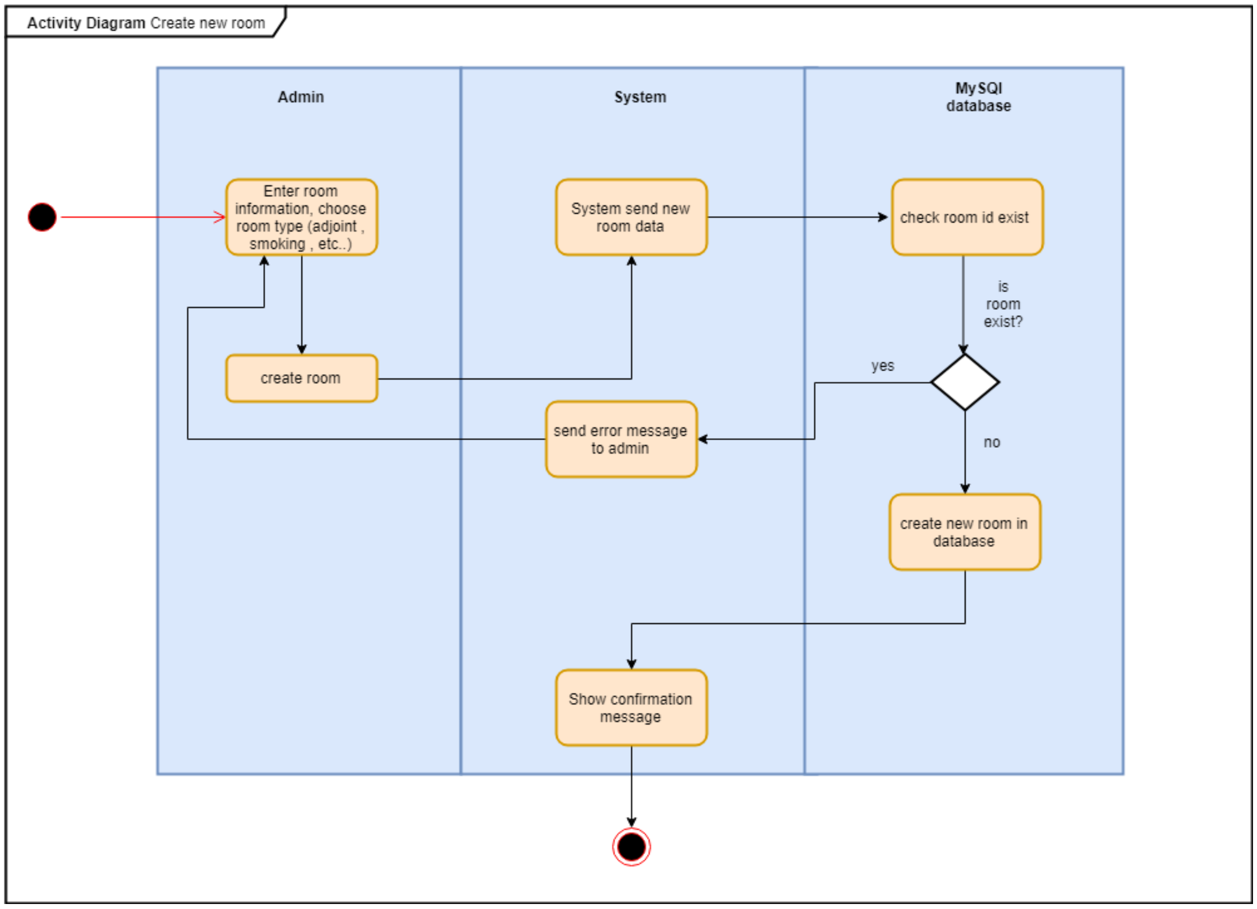

Activity Diagram Create new room
Admin
System
My SQl database
Enter room information, choose room type (adjoint , smoking , etc..)
create room
System send new room data
check room id exist
is room exist?
yes
no
send error message to admin
create new room in database
Show confirmation message

Figure 16: Create new room Activity diagram

To create a new room, the activity starts when Admin enter room information to the system. System reads the input and sends it to the database to check if given room id exist or not. Database saves the new room information then send a confirmation message to the system and activity ends.

3.3.7.8. Manage rooms

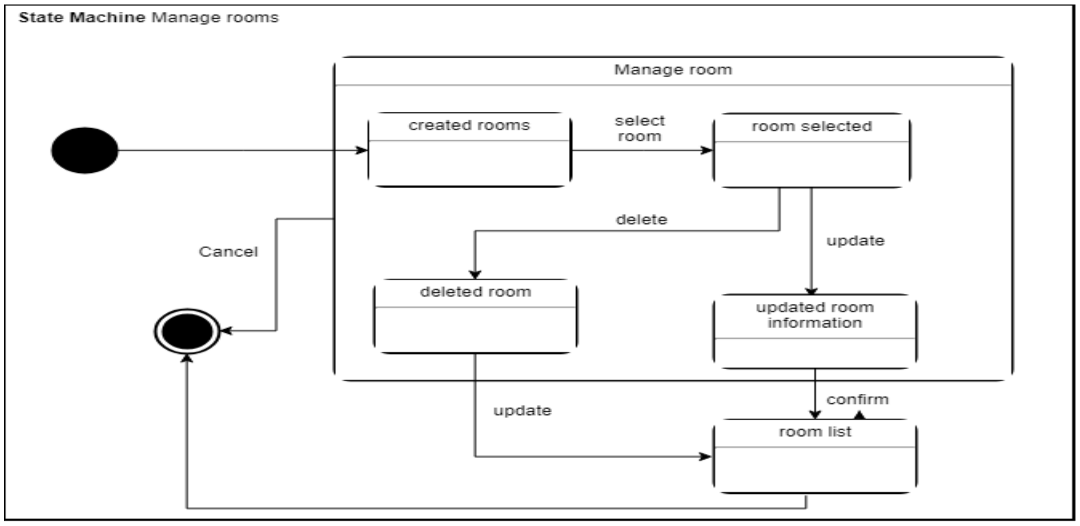


Figure 17: Manage rooms State Machine diagram

The diagram represent the changes in the system when the system edit or delete a room from room list in database.

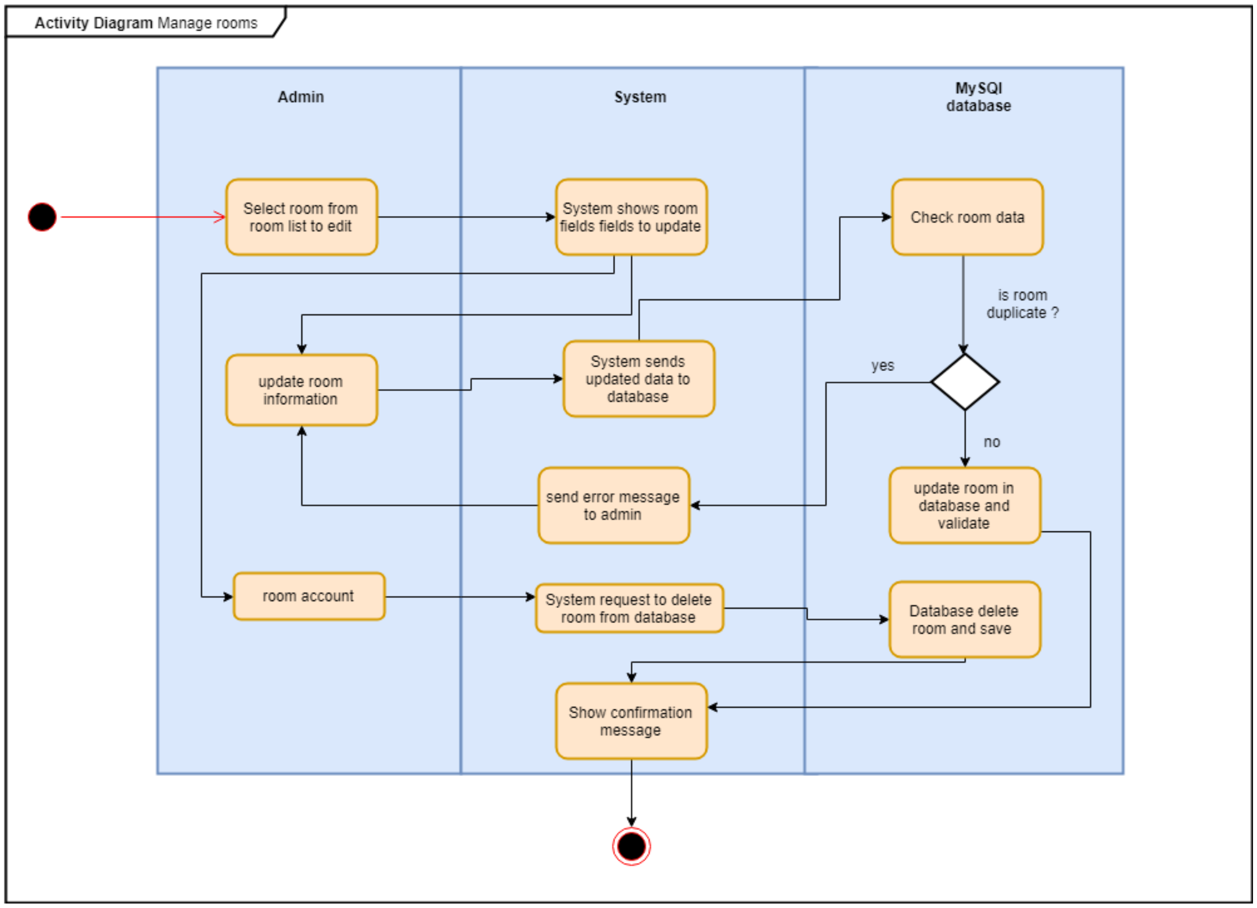

Activity Diagram Manage rooms
Admin
System
MySQl database
Select room from room list to edit
System shows room fields fields to update
Check room data
is room duplicate ?
yes
no
update room information
System sends updated data to database
send error message to admin
update room in database and validate
room account
System request to delete room from database
Database delete room and save
Show confirmation message

Figure 18: Manage rooms Activity diagram

To manage rooms, the activity starts when Admin select a room from room list either to update it or delete it. The system sends the chosen action to the database, then database checks if the room doesn't has same room id. Database validate the updated information and send a confirmation message to the system and the activity ends.

3.3.7.9. Create new account

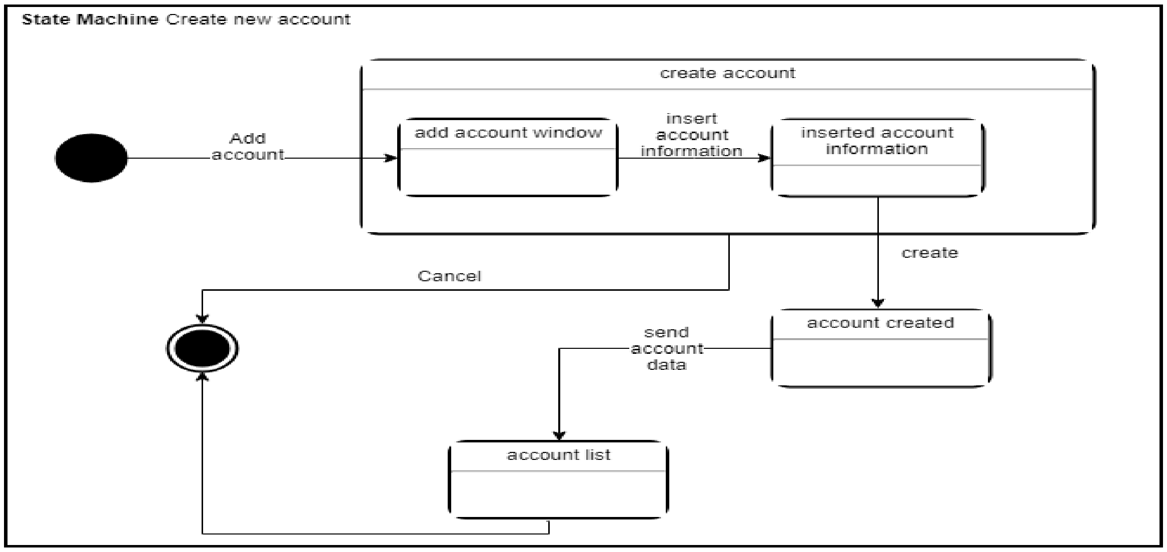


Figure 19: Create new account State Machine diagram

The diagram represent the changes in the system when the system creates a new account , authorize it and save it in the database.

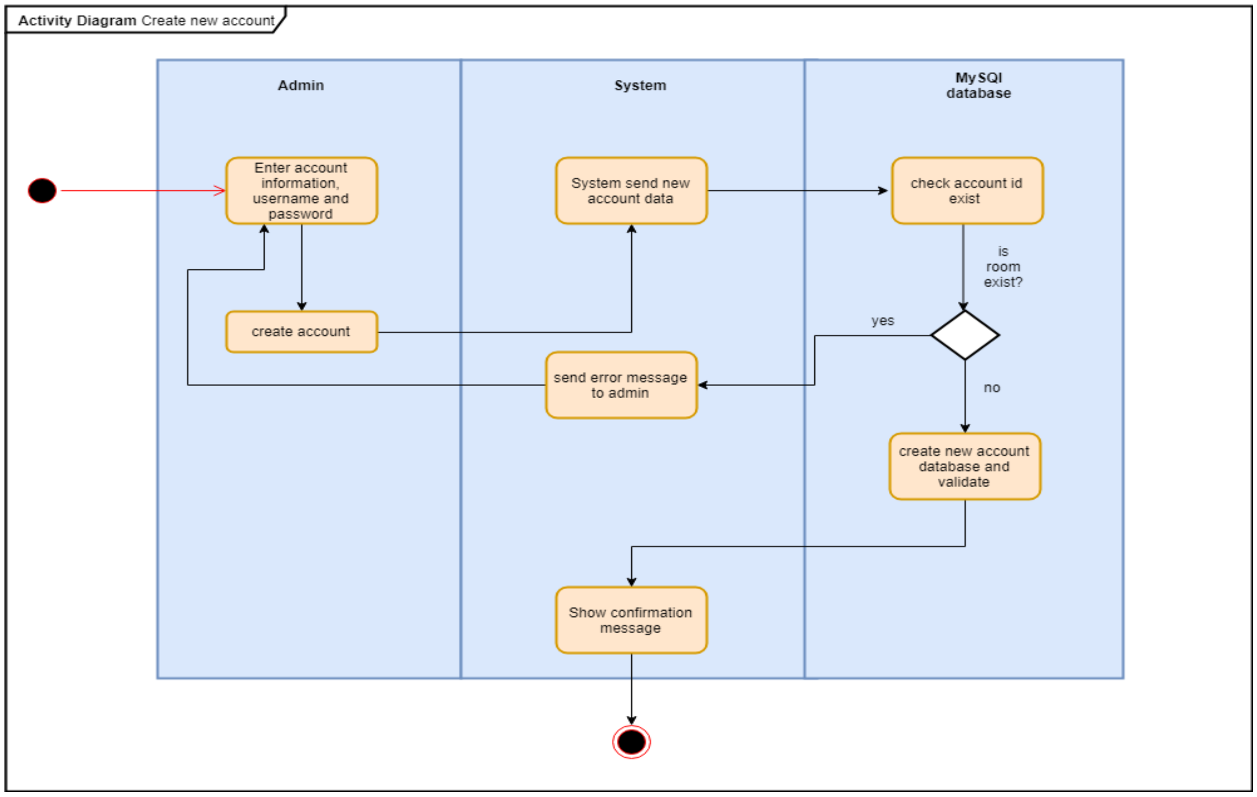

Activity Diagram Create new account
Admin
System
MySQl database
Enter account information, username and password
System send new account data
check account id exist
is room exist?
create account
yes
send error message to admin
no
create new account database and validate
Show confirmation message

Figure 20: Create new account Activity diagram

To create a new account for users, the activity starts when Admin enters account information to the system. System reads the input and sends it to the database to check if given user account or password exist or not. Database saves the new account information then send a confirmation message to the system and activity ends.

3.3.7.10. Manage accounts

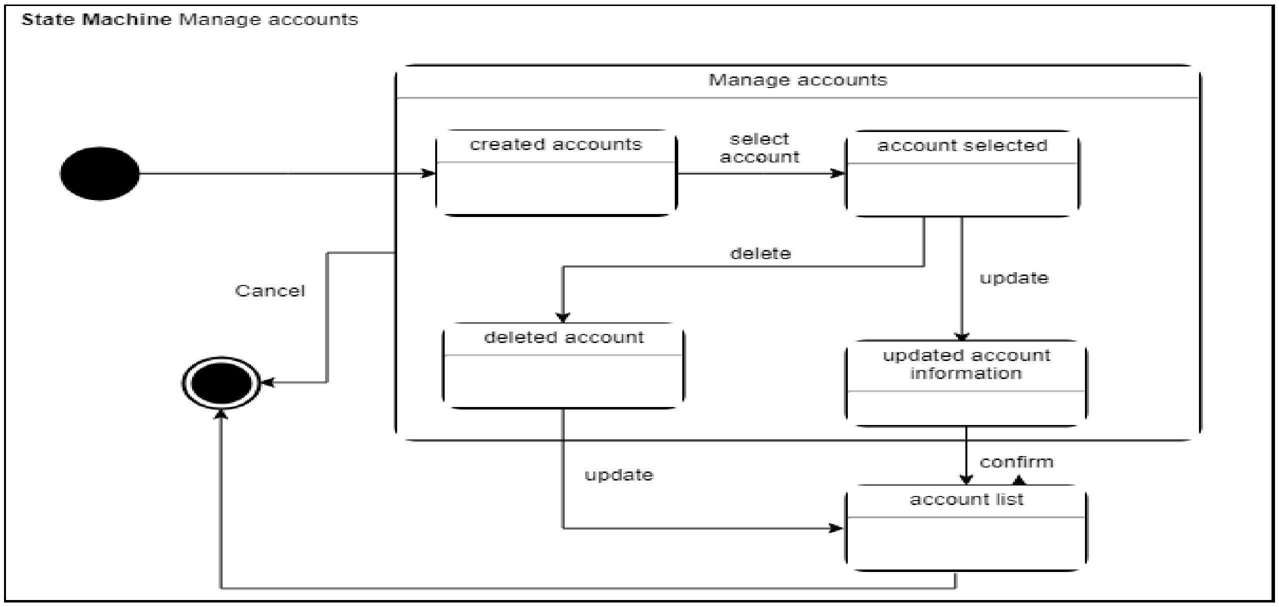


Figure 21: Manage accounts State Machine diagram

The diagram represent the changes in the system when the system edit or delete an account from account list in database.

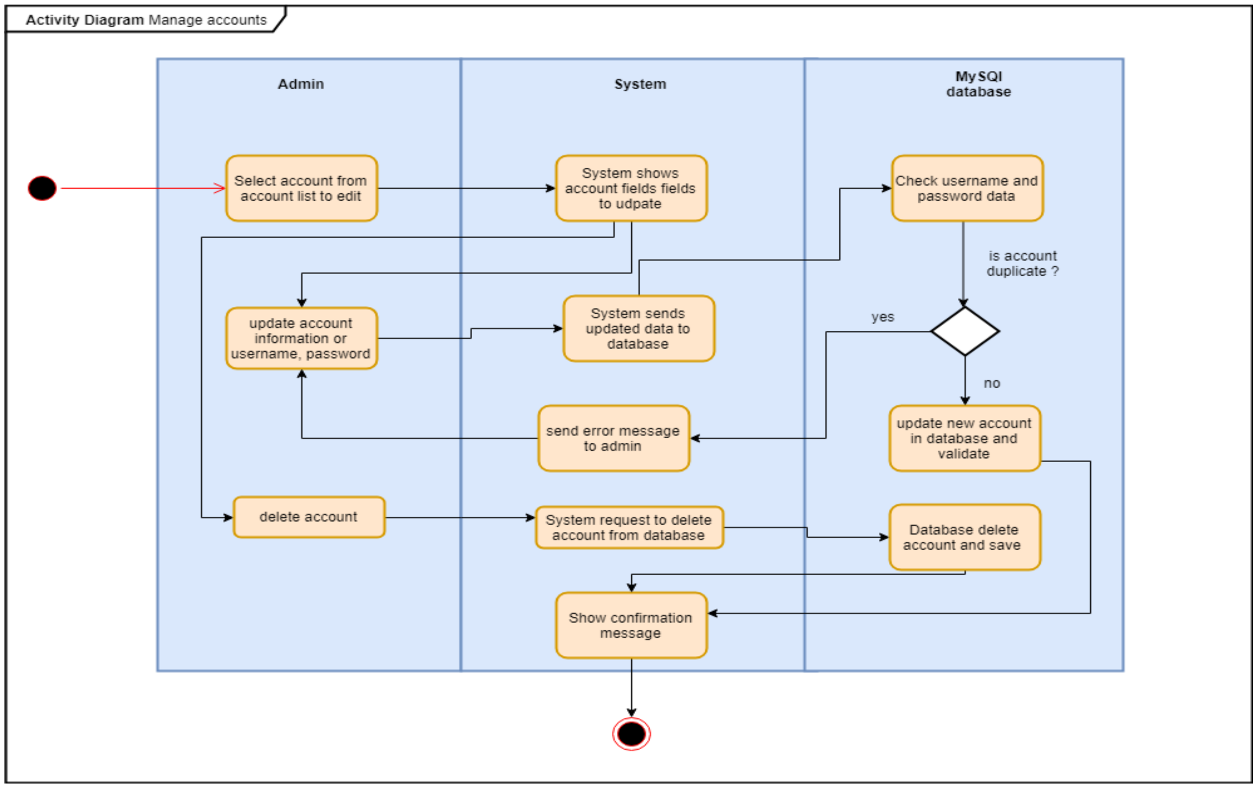

Activity Diagram Manage accounts
Admin
System
MySQl database
Select account from account list to edit
System shows account fields fields to udpate
Check username and password data
is account duplicate ?
yes
no
update account information or username, password
System sends updated data to database
send error message to admin
update new account in database and validate
delete account
System request to delete account from database
Database delete account and save
Show confirmation message

Figure 22: Manage accounts Activity diagram

To manage accounts, the activity starts when Admin select an account from account list either to update it or delete it. The system sends the chosen action to the database, then database checks if the account doesn't duplicate with another account. Database validate the updated information and send a confirmation message to the system and activity ends.

4. Design Document

4.1. Introduction

This document describes the design process of Hotel Management System. Hotel Management System is a software coded through an Object oriented approach and was created to provide easy management tools for Linnaeus Hotel's daily work.

4.1.1. Purpose

The intention of this Design Document is to describe the implementation of the new Linnaeus Hotel Management Software described in the Linnaeus Hotel requirement document. it includes the architecture and plan that are necessary to complete this project which is being created to assist Linnaeus Hotel with booking and managing of rooms.

4.1.2. Scope

The major functions of this Software is to handle the Reservation, Checking-in, Checking-out Calculating the bills of the customers, plus other minor functions to reduce waiting time and help Linnaeus Hotel work more efficiently. This software will not handle the payment process itself. A security risk this software faces includes the use of a third party, web based sql database. This could be dangerous as the private information of clients, including credit card information is stored within this database.

4.1.3. Glossary

User: Users of the system that can reserve, book, check-in, check-out and cancel a reservation, users are the hotel clerk.
Admin: Admin is the manager of the hotel, an admin would have access to everything that a user has access to, plus adding rooms, reserving specific rooms and editing a room.
FR: Functional Requirements are mentioned as FR*. The FR stands for Functional Requirement and the * will be replaced with a different number corresponding to each requirement.
NFR: NON Functional Requirements are mentioned as NFR*. The NFR stands for Non Functional Requirement and the * will be replaced with a different number corresponding to each requirement.
SRS: Software Requirement Specification Document.
MySQL + SQL: Structured Query Language.
Legacy Application: an enterprise application that is based on languages, platforms, and/or techniques that predate current technology.
DA: Domain Analysis Document.

4.1.4. Intended Audience

Primary audience of this document are the developers of this system. Furthermore, the users of this system in Linnaeus Hotel as well.

4.1.5. Priorities

Speed: The **speed** of the system is vital because there are set amount of seconds required for several processes and any delay could lead to the inconvenience of the customer and the clients. Some processes need to access the database and it should be assured that this takes less than the required time by testing, maintaining and developing the system mainly focusing on the speed.

User friendly: The system should be user friendly for the reason that, the hotel employees would use it daily to manage reservations and sensitive information. Thus, it should be easy to understand easy to manage and clear on its functionality.

Security: For the reason that our system will store and will have access to clients private and sensitive information, our system should be **secure**. Also the database between the two campuses will be shared, so the software should keep the information visible as few as possible and give limited access to users. In order to achieve this, the system will require authentication and restriction to who is able to access the different database information.

Reliability: Due to the hectic environment of a Hotel the system needs to be **reliable** at all times, no data can be lost and any crash or difficulties could lead to frustration of customers. The system needs to be up and running at all times. This will be assured choosing a remote database host that is up and running throughout the day

4.1.6. Overview

This design document is for Linnaeus Hotel software. The main purpose of this software is to ease the work of the hotel employees of Linnaeus Hotel by assisting them with the Reservation, booking room, checking in and out of customers. The system will include a secure database to store the information of the customers which can only be accessed by registered hotel workers for safety reasons. Some functionalities like editing rooms and reserving rooms beforehand will only be available for managers of the hotel. The system should also be able to show registered rooms with their details, prices and should also be able to calculate the expenses a customer will be billed.

4.2. System Overview

This section introduces the system context and design and will discuss the background to the project.

4.2.1. Product functions

- Check available rooms
- Reserve
- Check-In/Out
- Manage reservation
- Manage employees (Manager)
- Manage rooms (Manager)
- Reserve specific room (Manager)
- Calculate bill

4.2.2. User Characteristics

- Employee
- Manager

Employee/User: Employee have access to most of the functions. Employee can search for a room, reserve the room by inputting clients information, check-in/out and edit existing reservations.
Manager/Admin: Manager have all the access the employee has, plus a few extra. A manager can manage room which includes changing prices, bed numbers, smoking/non-smoking, add view to a room and even adding a new room. Other than that, the manager can manage employees' details such as name, address, phone number, username, password and add a new user as a manager or as an employee.

4.2.3. Outline of the design

Re-writing

After discussing our options our team came to the conclusion that we will re-write the software. As its denoted in the legacy software document the legacy software is lacking some attributes such as:

- Poor exception handling.

- The system does not have Models.
- Code-wise it is not well commented which makes it harder to understand the functionality.
- User interface, was built on outdated visual toolkit (Swing widget) and uses the Spring framework which is replaced by JavaFX as of now.

Furthermore, the design of the system is not reusable, to rebuild it components and follow its pattern. The structure of the classes is really an issue, as they have many dependencies that prevent from develop the codes in a good period of time. simple example could be shown in Reservation class which is dependent of Reservation Management class which should be the opposite. The codes were unrunnable in the beginning, Java 8 platform was used to run the system but it took couple days to find how to run which used old libraries and old version of Java 5 libraries. Moreover, the Exception were not handled well as sometimes exceptions were caught with no action being taken. Therefore, re-building the legacy system was a better decision to avoid wasting time on refactoring a legacy system which has some unfigured issues that faces a modern developer to understand and deal with.

4.2.3.1. High-level description of the design

This design document is made for a hotel management software. The hotel management software is build to help the hotel management to save money and help them to serve guests better. The software will be used by the hotel employees and will allow them to add and edit reservations, print bills, etc. The software should be able to store manage and retrieve data, so a database is necessary(SRS 3.3.2). Also the software should prioritize on speed, security, User friendly and reliable as it is denoted in (1.5). Thus, we would focus on providing those futures for our software. We would describe every major decision on how would we provide these prioritized futures, and why did we choose to provide them the way that we did.

4.3. Major design issues

This section will discuss the major design issues. It will also include the possible alternatives, the final decision and the rationale for the decision.

4.3.1. Re-writing

As it is denoted in (4.2.3), After analysing and running the legacy application it was decided to re-write the program. Some reasons behind this were that:

- The current program has several classes that include too many methods, which made it hard to work with.
- Other than that, the software was not following a specific design architecture or pattern.

Hence, it would be time consuming and inefficient to redesign the existing system. After discussing the different options we agreed on dividing the work between us is the most efficient and fast way of developing for this project. We would try to make our own components and make them as reusable and flexible as we can.

4.3.2. Architectural Pattern

For our architectural design low coupling, abstraction, divide and conquer and flexibility would be needed. The reason is, for our development we choose to divide the software into smaller parts and each one of us would be responsible for developing one or more of these parts. So our system´s components should be as independent as possible so we can divide easier the development between us safer and more productive. We have chosen to divide the work between us for the reason that for this project is faster and more productive, than developing the software as a hole.
 Our choices, that allow us to divide the development as much as we can and safe:

- Multi-Layer pattern.
- MVC.

For this project we have chosen as an architectural pattern the MVC pattern. The reasoning behind this choice is that, we want **divide and conquer, low coupling, abstraction, and flexibility.** MVC provides all of them, plus design for **testability** which would allow us to test different components without the need of other components, this would also help in our development choice. In advance, divide and conquer low coupling and abstraction was found to be the most efficient as developers can work on tasks parallelly. MVC pattern gives us more independency on development, which is what we want.

4.3.3. GUI

In the (**SRS** 3.3.7, 3.3.8, 3.3.10) and 1.6. is denoted that the User Interface should be user friendly-easy to use, easy to learn and easy to maintainable. In advance, in the **SRS**3.3.4, and 1.6 the system should be secure. So we should make our software easy to use and do not give easy access to sensitive data.

- One way is, to develop the GUI so it won't have access at any data so it would be just provided with data and display it. This will ensure the safety of sensitive data, and the independence of the GUI would make it easier for its design and changeability.
- Our other option is to decide in the GUI which data will be displayed and managed. This practice will also be safe if it is carefully designed and it could make the design of the GUI easier since we can retrieve and manage all data at all time.

Our decision was to make the GUI as independent as possible from the rest of the software. For the reasons that it would be safer when it comes to data management, maintainable as the GUI can be changed in the future. Also, the design of it would be easier as the developer of the GUI will not have to separate the GUI functions from the rest of the software. Lastly, we can separate it and assign it to one of our group members to work with it, this would make the GUI development faster and more productive.

4.3.4. Development language

As the (**SRS**3.3.6,3.3.7) state the GUI should be user friendly, and as (**SRS**3.3.2) states a database connection is crucial. By using a graphical interface construction tool and build in libraries the development of the software and the design of the GUI would be faster and more productive. Last and most important the chosen development language should be a language that our team has experience with, in order to work more productive and complete the development faster. Possible development languages:

- C++
- Java
- C

We have chosen Java as a development language for our project. Java is one of the most widely used languages when it comes to desktop applications.
Reasons will be listed bellow:

- Java has the ability to support a number of frameworks, external tools and a number of build in libraries that can be used.
- It has a big community support.
- Java supports a number of databases.
- All members of our team have experience with java.

4.4. Technical issues

4.4.1. Database management system

As in the (**SRS** FR13) is denoted the software should have access to the data of other locations that the software is been used. Thus, the database should be able to store and share data for more than one locations. In order to archive that the database should be on a network. In advance, like the (**SRS** 3.3.1) denotes the database should be able to store and retrieve data fast. Also as we are using Java as a programing language it should be compatible with it. Possible Database management systems:

- SQL
- MYSQL
- SQLite

We have chosen MYSQL as our database management system, MYSQL is one of the most used and reliable Database management systems. Reasons are listed below:

- MySQL is a stable, fast and free open-source database management system.
- It has a big community support.
- It is compatible with Java
- Easy to use.
- It uses external connections, it can be accessed through the network.

4.4.2. Applications that use MySQL:

The Application that would be used should be able to run on a server platform in order to allow access from different locations simultaneously and use MYSQL as the 2.2.4.2. section states. The application should be fast as that as the (**SRS**3.3.1) states, easy to use and easy access for faster development. Our options are listed below:

- MySQL Workbench
- MySQL GUI Tools

Our final decision was to choose MySQL GUI Tools because it is easier to manage and maintain, it can be accessed from different clients so all of us could work in the same page, also:

- Free use for small scale applications (development).
- Easy access

4.4.3. Data format

As the (**SRS** 2.1.2) states MYSQL will be used as a database management system and JAVA as a developments language. Thus, the data format will have to change in order to be able to be shared between MYSQL and JAVA. The data will be formatted into Variable Character Field, integers, TINYINT and Dates when they are stored in the database. When they are used in the system(not database) they would be used as strings for Variable Character Field , integers and dates will stay as it is and booleans for TINYINT values.

4.5. Design details:

4.5.1. Software Architecture

4.5.1.1. Architectural Pattern

As denoted in 4.3.2, MVC will be used as our architectural pattern.

The view layer will handle the GUI, taking inputs and presenting the outputs. The view layer will be composed of a number of fxml files that each one will present a different window to the user. The user will be able to navigate through the fxml files and depending on the file different functionalities will be provided to the user.

The controller layer will take the information from the view layer and ask for the model to analyze and retrieve data that would be displayed later in the view. After getting the analyzed data from the model it would send it to the view and it would be presented there.

The model will analyse the data and perform certain functionalities that the controller will use to present through the view the results of them. The model would be a set of classes that interact with each other in order to analyse data and perform the functionalities needed.

4.5.1.2. The set of components

Authentication: The authentication component is providing security and divides the system´s access depending on if the connected client is a user or a admin as (**SRS** FR19 FR18 FR10) state the admin should have access to functionality that the user does not. The providing component is the GUI, which will require for a username and a password in order to have further access. After providing username and password it would validate. In order to be able to validate the required interface is the database component and model. Database would receive a query that states what to search for. The database would search if the username belongs to an account and if it is, it would check if the password matches to that account password. If it is the username and password match the database would send back to the provider the corresponding Employee that would be filtered in the model.

That means that the user exists and the Employee contains the information if they are a user or admin, so the appropriate would be provided to them.

Model/Functional layer: The model layer is the component where the system's functionality nests. The required interface is the database which the model component will use to store retrieve and manage data. The model layer is able of calculating the bill (**SRS** FR14 FR17 FR15), searching for the available rooms (**SRS** FR7, FR8), reservations that would check in or out (R.E FR3, FR4). The provider interface is the controller, which “feeds” the model layer with the information that the controller wants to be processed and receive.

GUI/View: The GUI in our software is the component that allows a user or admin to interact with the software, it only allows inputs or displays outputs (**SRS** 3.1.1) no calculations or algorithms are made or placed in the GUI. The provider interface is the controller which provides GUI with what to display by taking the GUI inputs and filtering them into the Model. Model is the required interface, which would filter/calculate the data that the controller sends to it.

Database: The database component is the component that will store retrieve or manage data. Moreover, Database component´s provider is the Model which will provide the database with queries stating what to do. Database component is crucial as (**SRS**2.1.2) denotes, it would provide access to every data that the software needs to be functional. It uses MYSQL to manage the data.

4.5.2. Diagrams

4.5.2.1. The UML Component Diagram of the architecture:

The authentication component depending on the login information that the user or admin would provide it would provide the corresponding access window. The access window is the view component which would provide different abilities for the user or admin that is logged in, by exchanging the information through the model layer. Moreover, the authentication component leads to the view component which depending on the user or admin actions would process the data through the model component and then display it to the user or admin. The model component is communicating with the database, retrieving, managing and storing data to the database component.

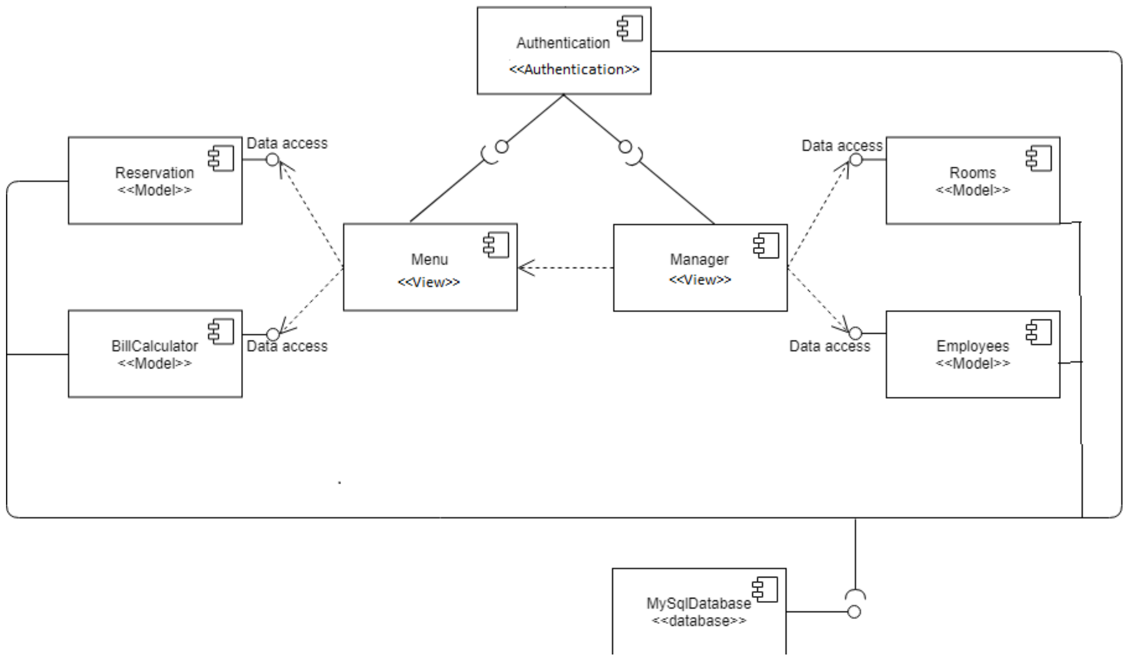


Figure 23: UML component diagram

4.5.2.2. Deployment Diagram:

There are two nodes used for our software the PC which refers to the user's computers and the server which will contain the database. Our software will run on the user's computer and connect with the database via JDBC. The database will contain the necessary data for the software to run and the software will continue to add, manage and retrieve data to the database via JDBC.

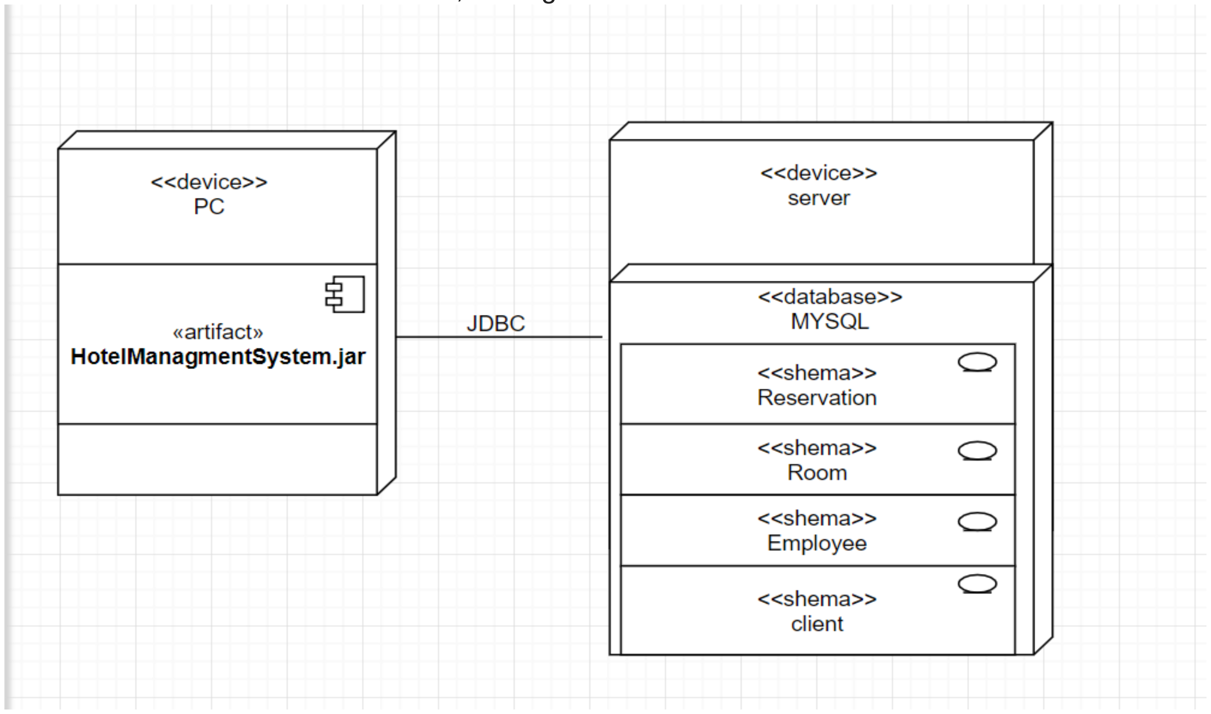


Figure 24: Deployment Diagram

4.5.2.3. Component Implementation

Authentication:

Authentication will be divided into two layers, the view layer and the model layer. The view layer will take care of the inputs and outputs of the authentication and the model will make sure that the inputs that are given are correct. For the reason that the authentication component will handle sensitive data, it will be implemented with the testability principle. We will implement the view and model separately and test them separately to make sure every part works correctly.

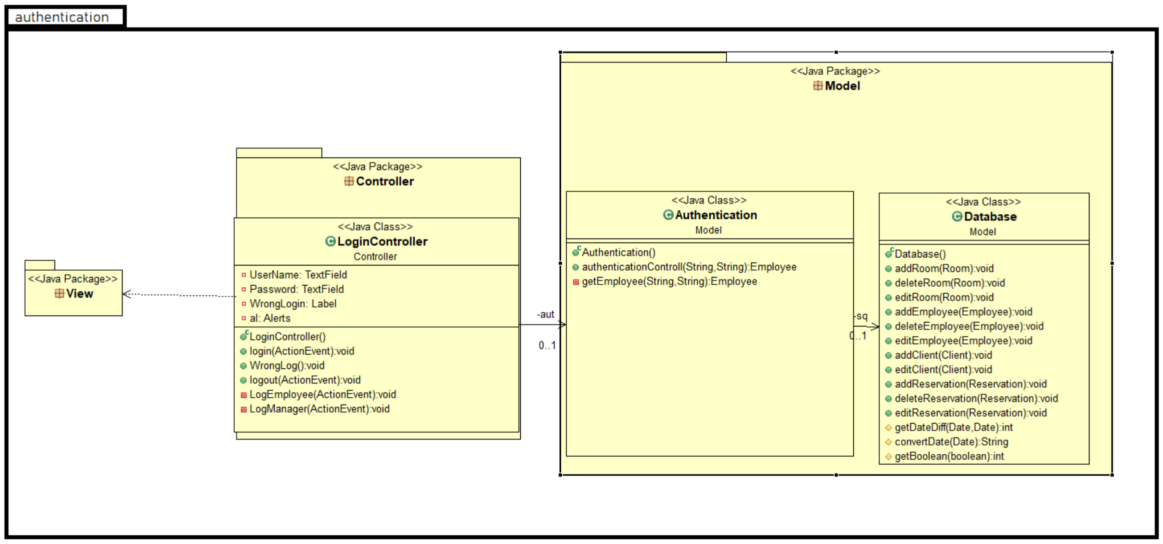


Figure 25 : Authentication layer

4.5.2.4. Model/Functional layer:

The model layer is a number of classes that our system´s functionality will be based on. Some of the classes will communicate with each other depending on the functionality needed and providing the corresponding result. This component will be implemented with defensive, testability and abstractive principle. Furthermore, a number of functions will be nesting here so they must only interact when needed, taking correct inputs and be as much abstracted as possible.

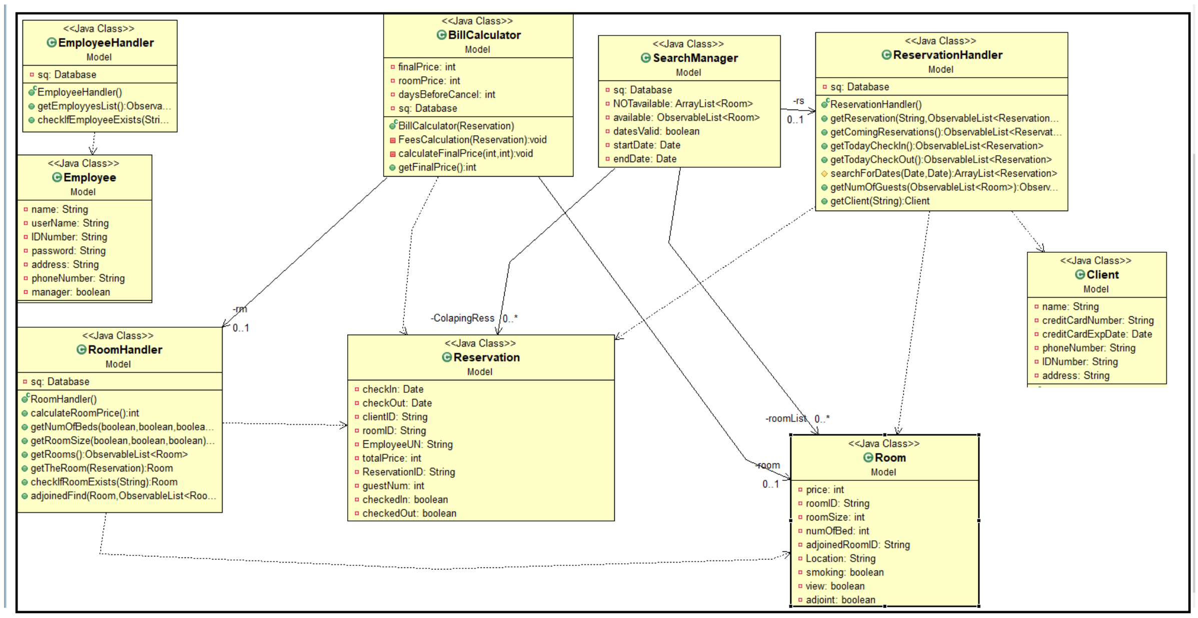


Figure 26: Model layer

4.5.2.5. GUI/View:

The view component includes a number of classes that are used for displaying the GUI of our system, each class will display something else depending on where the user wants to navigate. The view layer will be responsible for taking and presenting information. It would be implemented with flexibility principle so it can be changed and modified at any time.

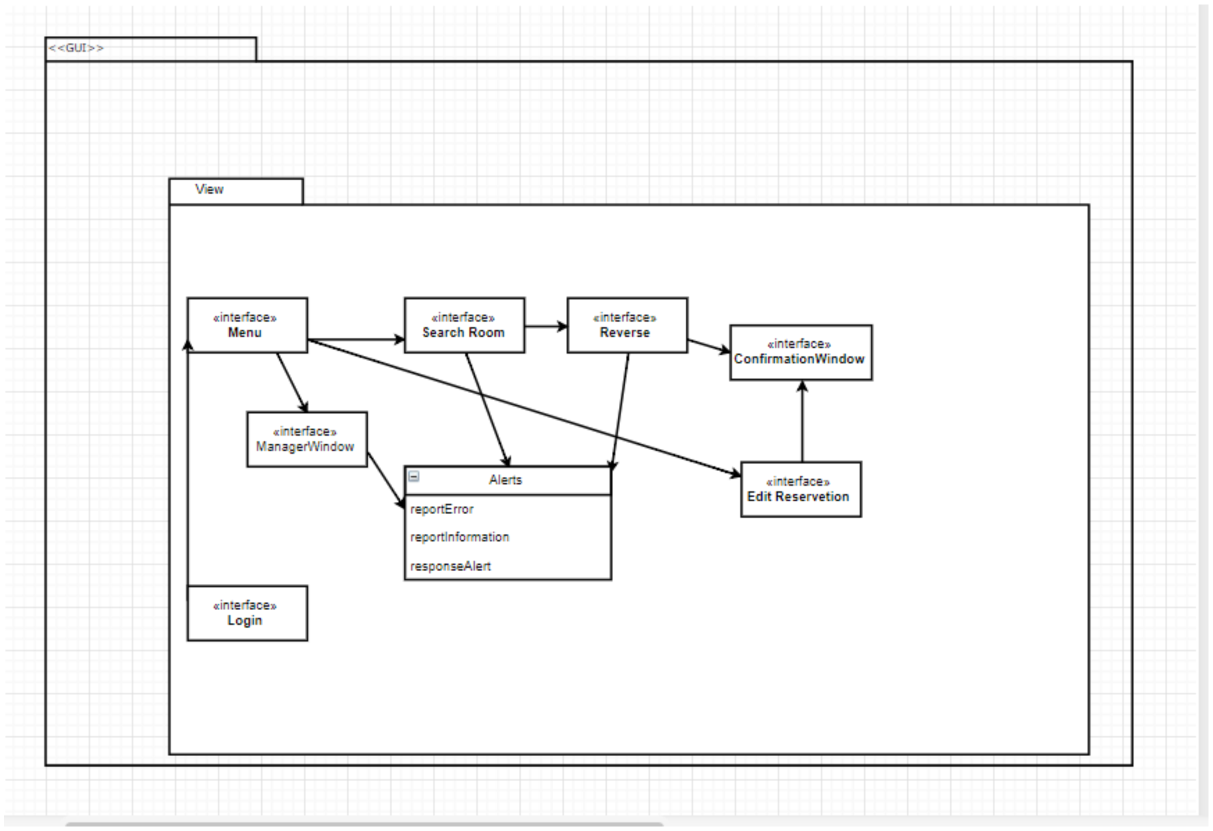


Figure 27: View layer

4.5.2.6. Database

The database will be just one class responsible for managing, adding and retrieving data.

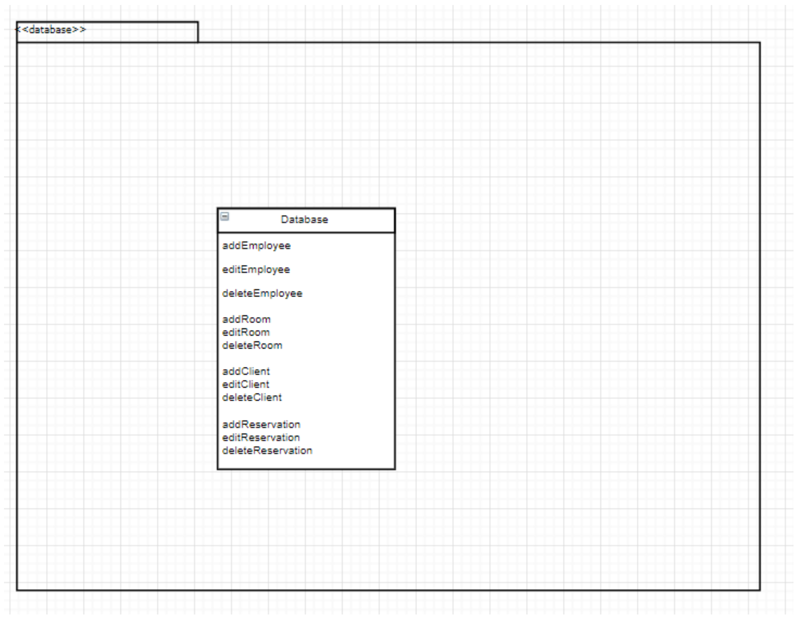


Figure 28: Database layer

4.5.2.7. View Sequence diagrams

4.5.2.7.1. Log-in

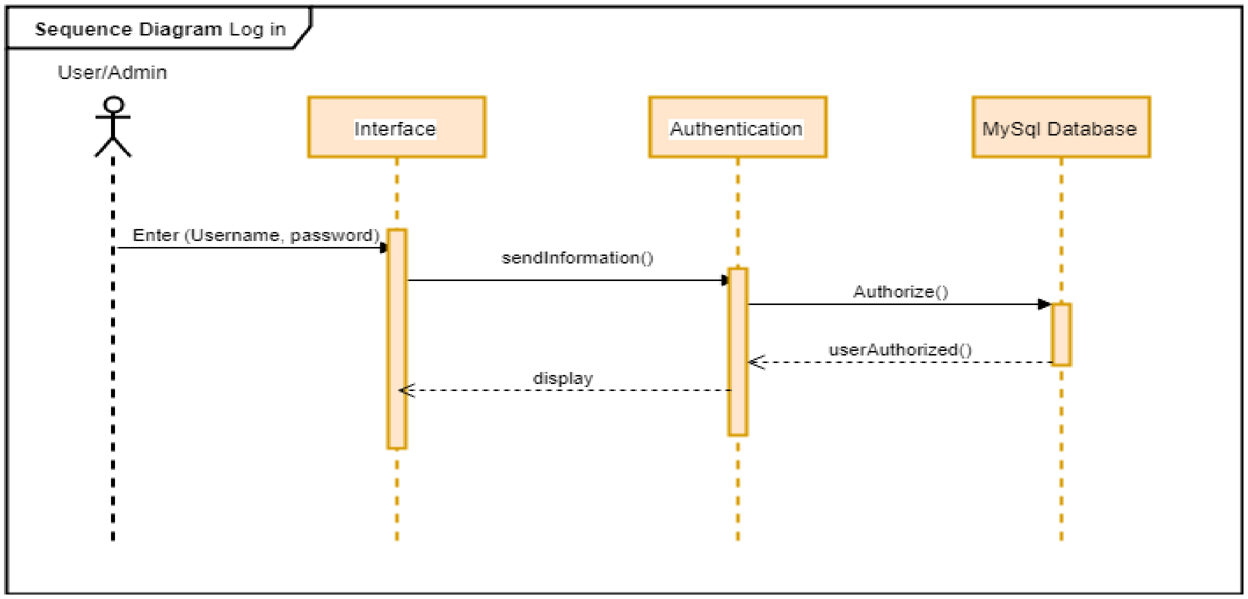


Figure 29: log-in sequence diagram

The diagram shows how the user or admin log-in successfully to the system. The user/admin enter their username and password in to the system, the information is sent to be checked and authorized. If its correct account the process continue to database and return to display the right account

4.5.2.7.2. Search room

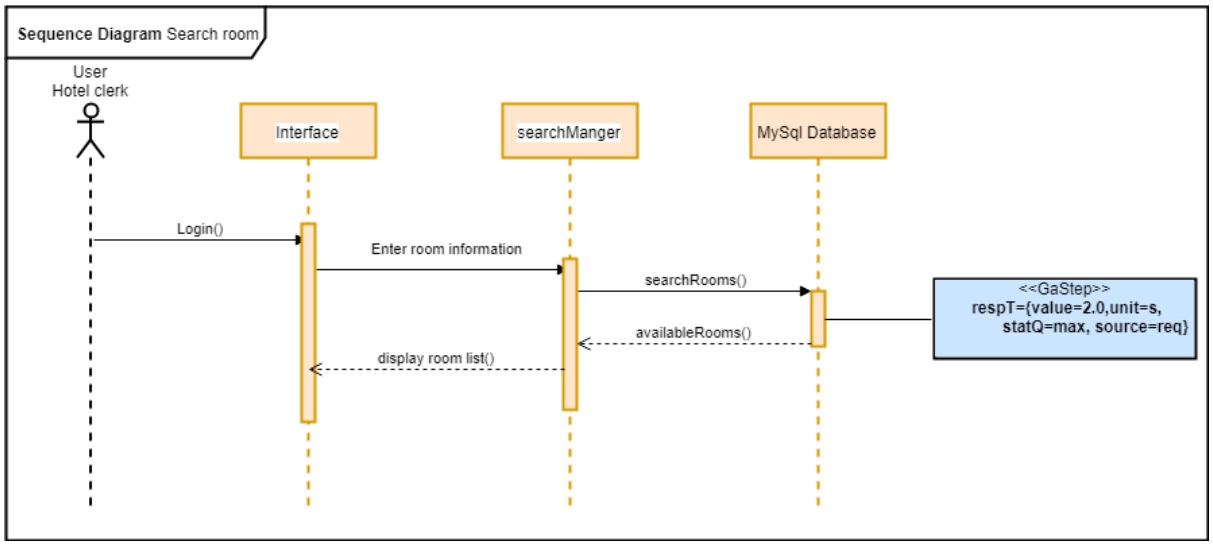


Figure 30: Search room sequence diagram

The diagram shows how the hotel clerk successfully searches a room in the system. Hotel clerk gets quest order of room, then search it the system gets information and get it from the database. Finally, display all the rooms that match the search to the hotel clerk.

4.5.2.7.3. Make a reservation

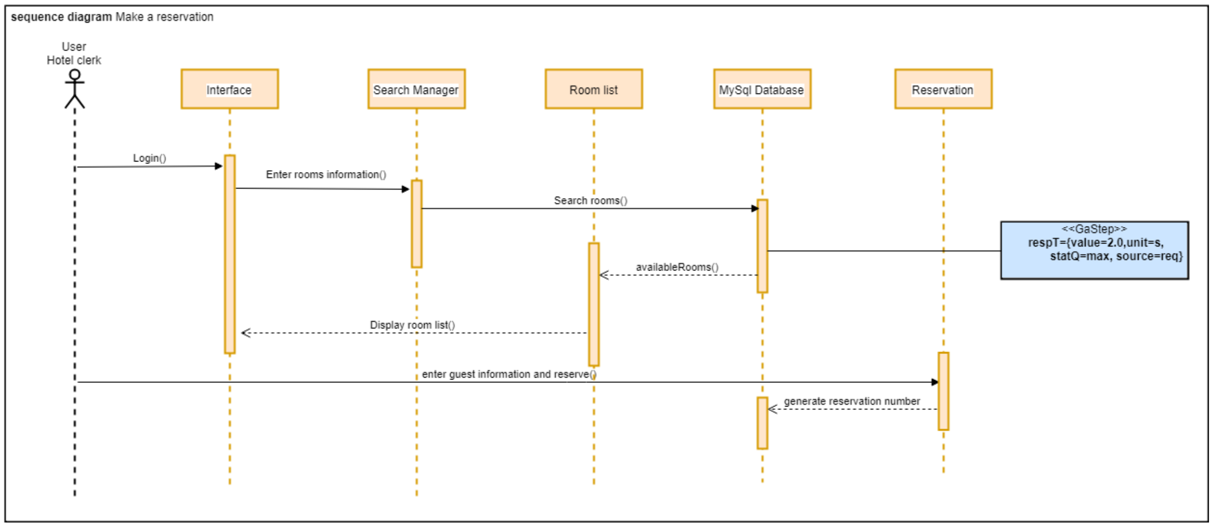


Figure 31: Make a reservation sequence diagram

Guest ask for a room in the Hotel, Hotel clerk will log in to the system. The system will ask for rooms preference to find it in room list, the user will search according to the guest wish. Then the database will send the rooms that match the order, and the available rooms only, the rooms from room list will view to the Hotel clerk. The Hotel clerk then enters guest information and make the reservation. Reservation will save information and send reservation number to the database, now reservation is saved.

4.5.2.7.4. Manage reservation

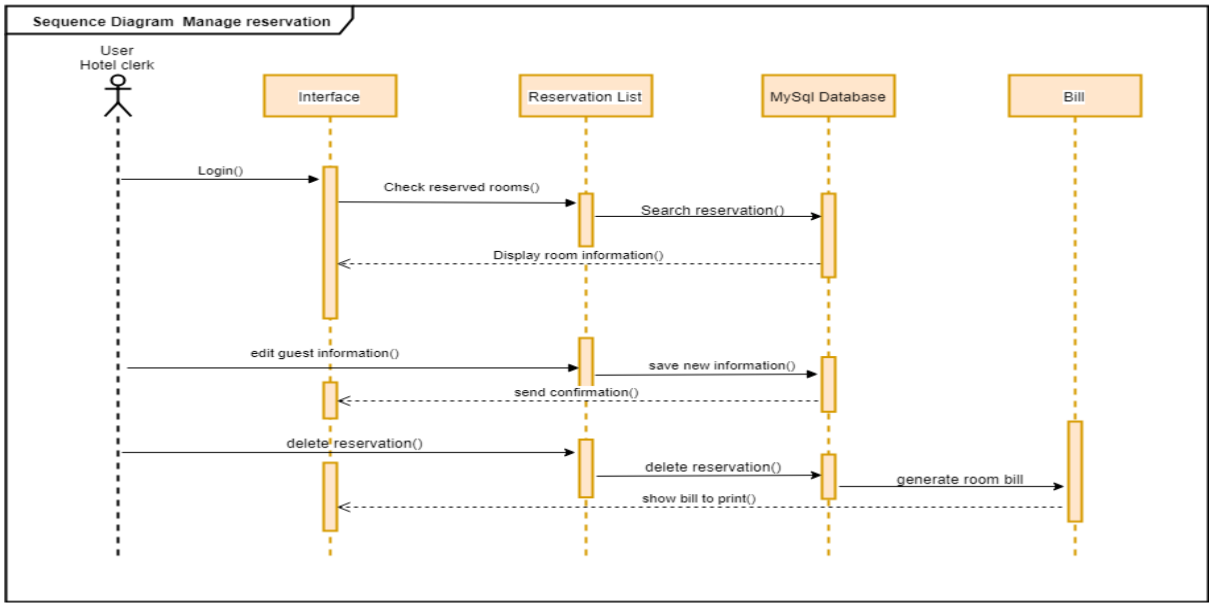


Figure 32: Manage reservation sequence diagram

The diagram shows how the Manager successfully edits a room list and the database. The manager checks room list where it displays all existing room, Manager chooses a room in the list then it edits it, the database updated room list and send confirmation message. The manager wants to delete a room from the database, manager checks room list where it display all existing rooms then manager chooses a room in the list then it deletes the room, the database updates room list and sends a confirmation message.

4.5.2.7.5. Check-in

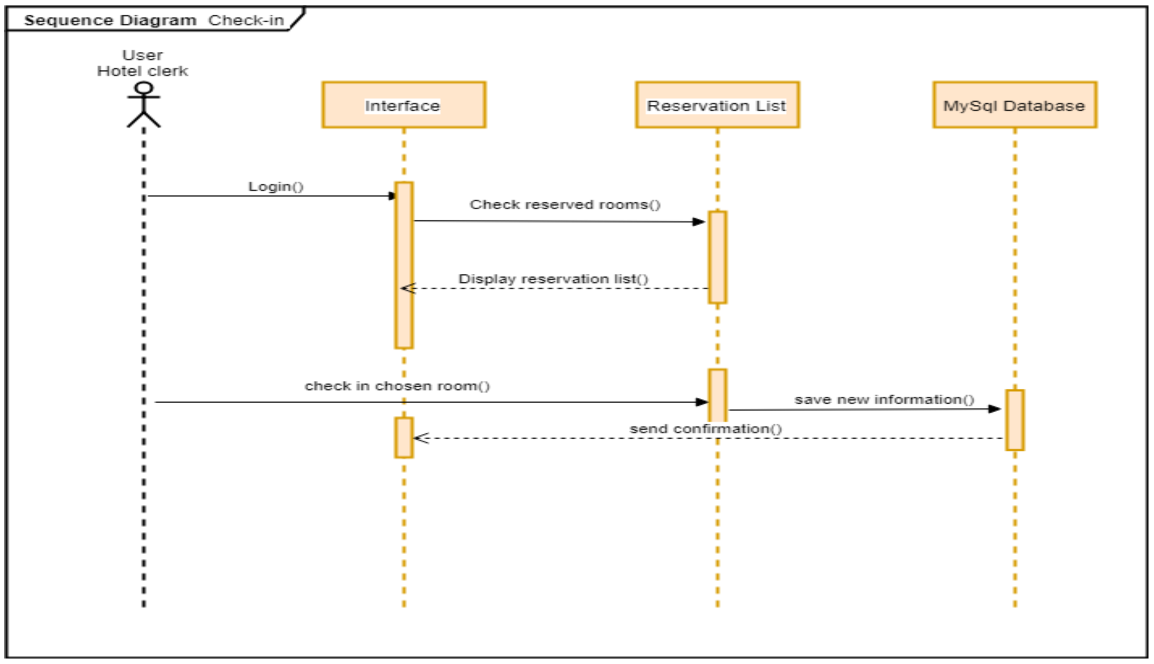


Figure 33: Check-in sequence diagram

Guest want to check-in to his/her room, the User logs in to the system. User checks reservation list of that day, reservation list will display. The user chooses the room to check in, then it processes to the database to save the operation. The database will send a confirmation message that the operation is saved.

4.5.2.7.6. Check-out

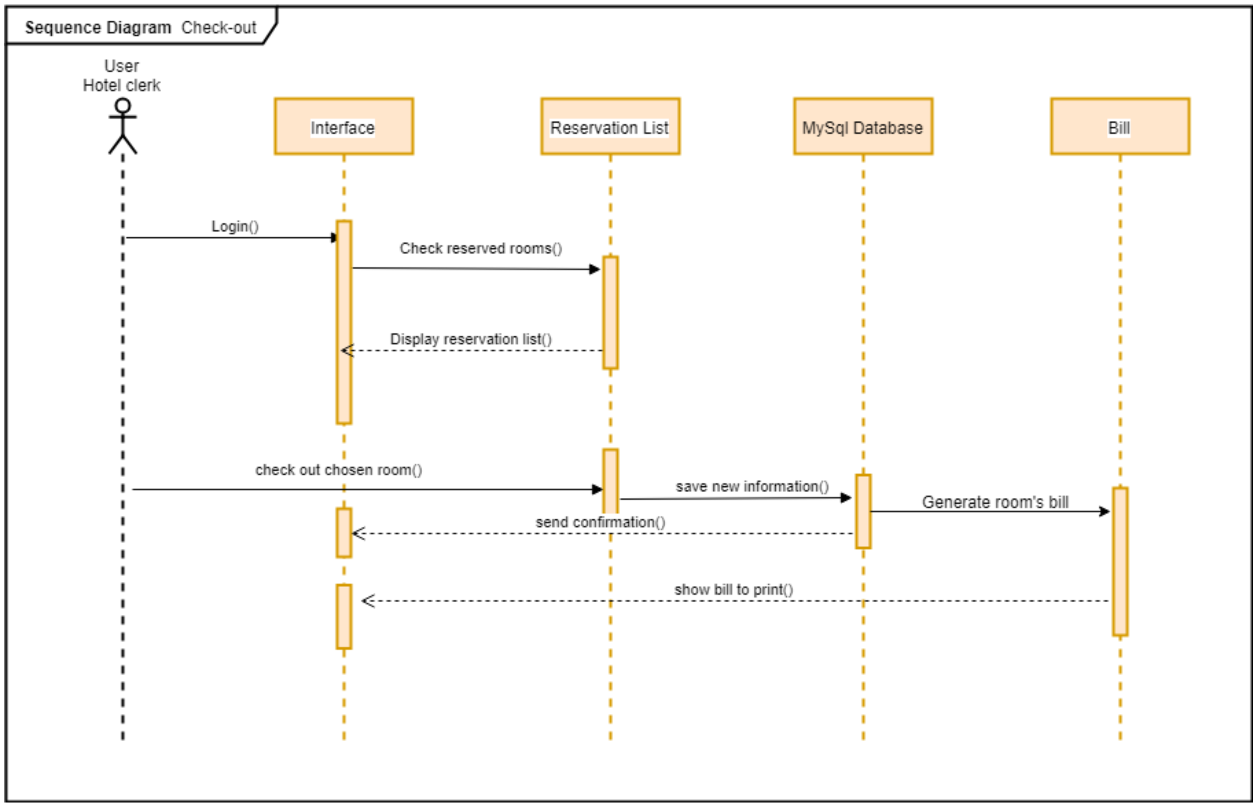


Figure 34: Check-out sequence diagram

Guest want to check-out from his/her room, the User logs in to the system. The user checks reservation list of that day, reservation list will display rooms that are checked in. The user chooses the room to check out, then it processes to the database to save the operation. The database will send a confirmation message that the operation is saved. The database will also generate a bill for the room expenses and send it to the user interface.

4.5.2.7.7. Create new room

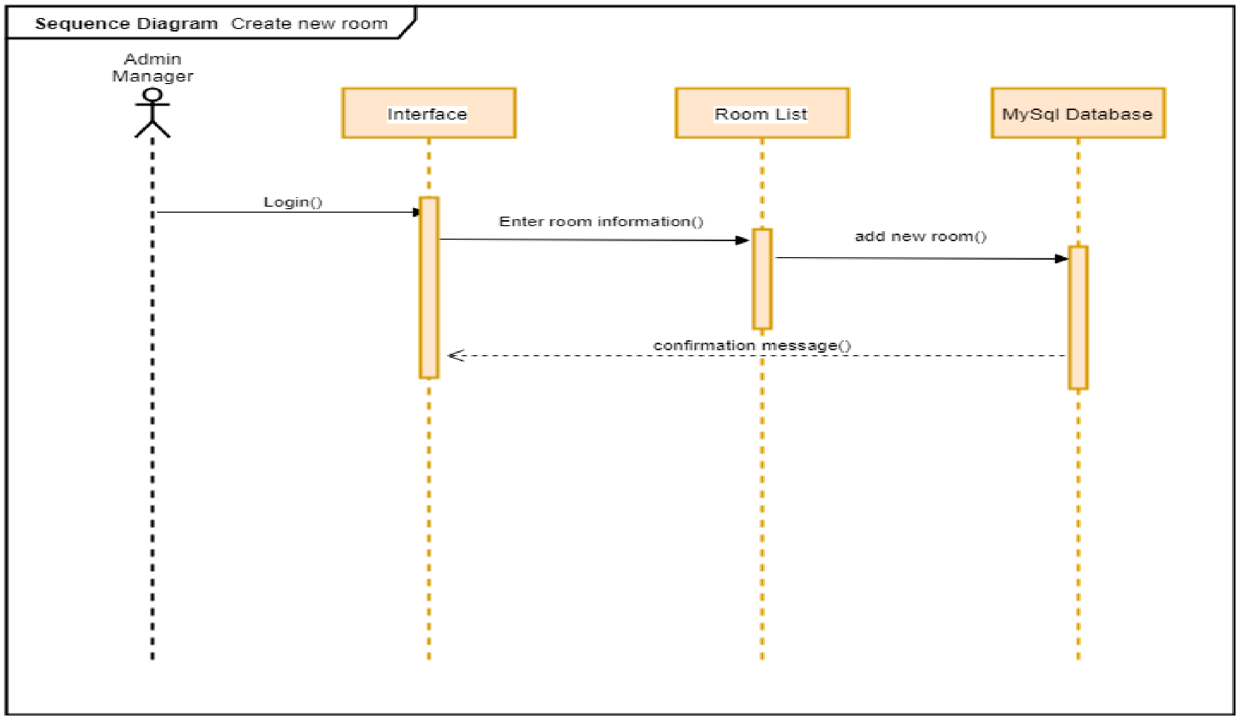


Figure 35: Create new room sequence diagram

The diagram shows how the Manager successfully adds a room to the database. The manager enters the room information to room list then it process to database and be saved. Afterwards the database send a confirmation message.

4.5.2.7.8. Manage rooms

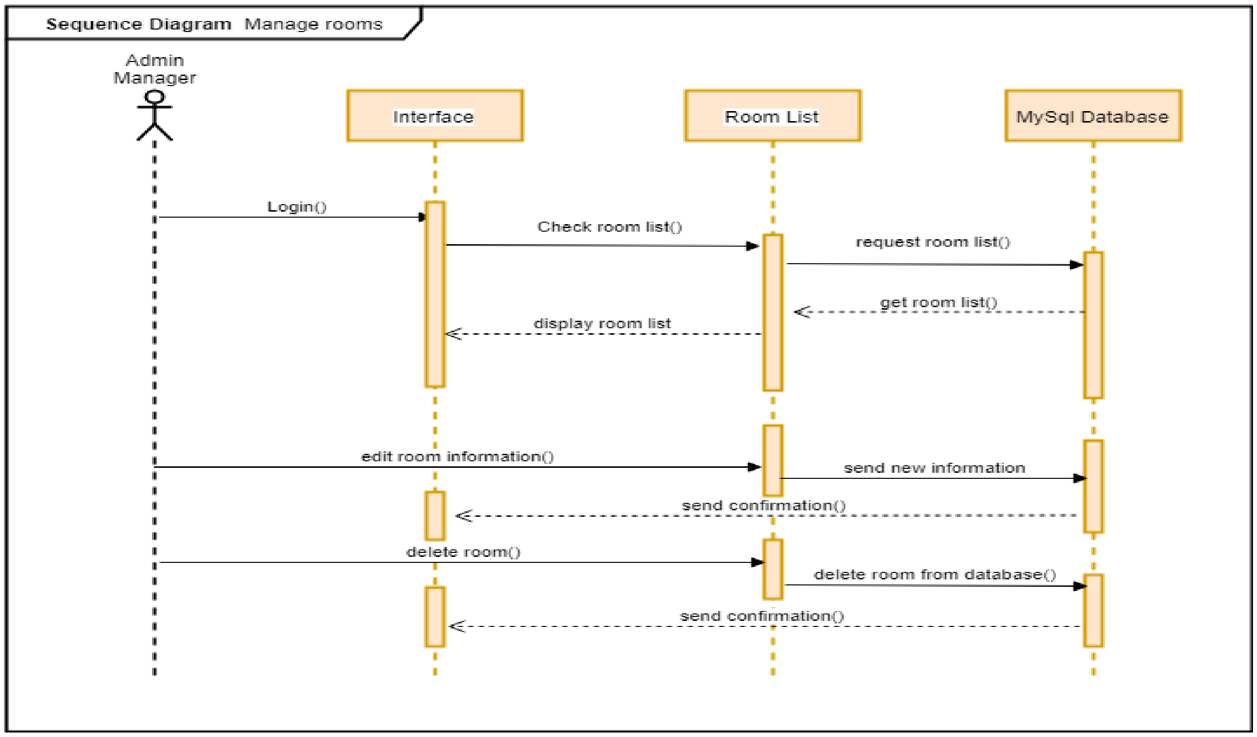


Figure 36: Manage rooms sequence diagram

The diagram shows how the Manager successfully edits a room in room list and the database. The manager checks room list where it displays all existing rooms. Manager chooses a room in the list then it edits it, the database updated room list and send a confirmation message. The Manager wants to delete a room from the database. The manager checks room list where it displays all existing rooms. Manager chooses a room in the list then it delete it, the database updated room list and sends a confirmation message.

4.5.2.7.9. Create new account

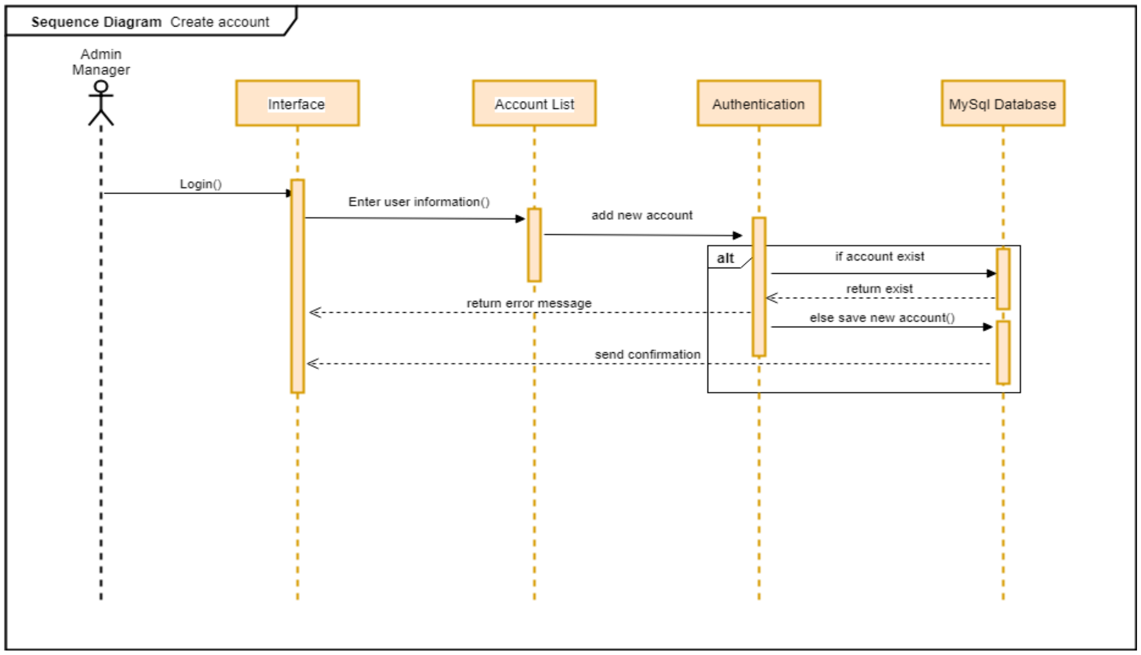


Figure 37: Create new account sequence diagram

The diagram shows how the Manager successfully adds a room to the database. The manager enters the room information to room list then it processes to check if the account is exist the database will send an error message, otherwise the account will be added to the database and be saved. Afterwards the database send a confirmation message

4.5.2.7.10. Manage accounts

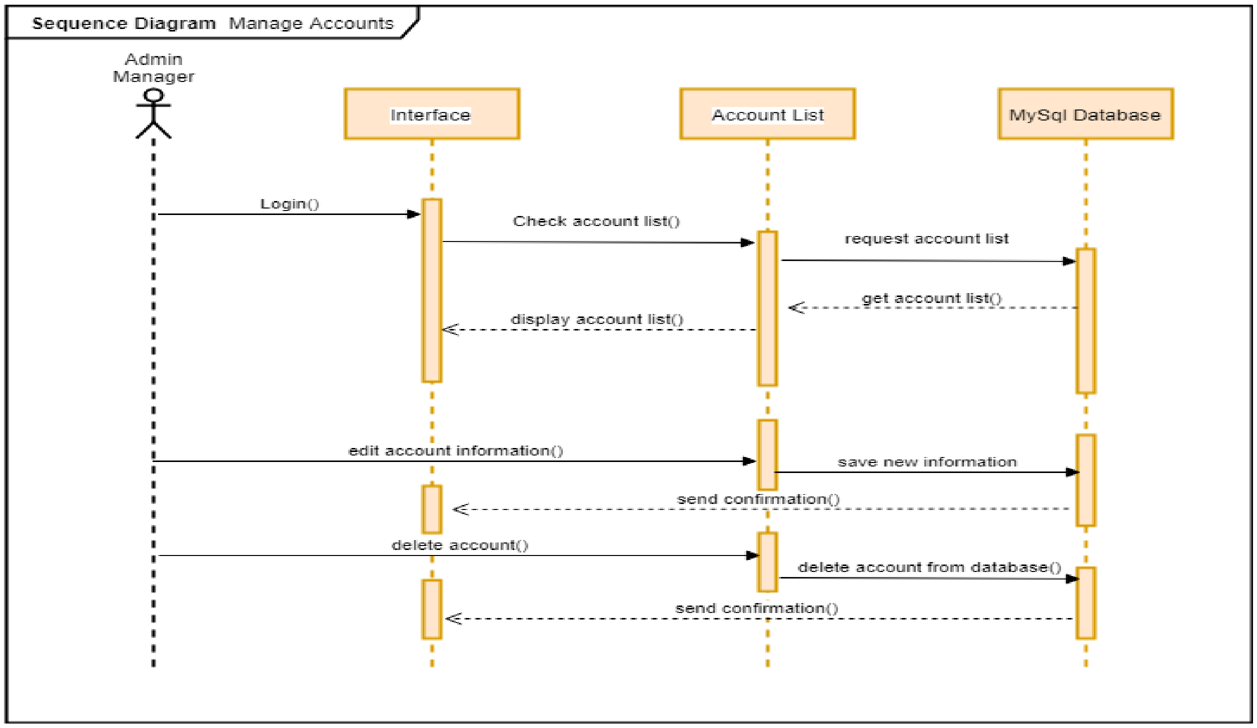


Figure 37: Manage accounts sequence diagram

The diagram shows how the Manager successfully edits an account in account list in the database. The manager checks account list where it displays all existing accounts. Manager chooses an account in the list then it edits it, the database updated account list and sends a confirmation message. The Manager wants to delete an account from the database. The manager checks room list where it displays all existing accounts. Manager chooses an account in the list then deletes it, the database updated account list and sends a confirmation message.