CS 319 Project :
Analysis Report

AutoHotel
Hotel Automation Software System

İrem Hergüner   21201077
Halit Onur Karabaş   20901806
Çağlar Alim Ağlar   21101255

Can Alkan

CS319
Object Oriented Software Engineering

Contents

List Of Figures

Analysis Report

AutoHotel : Hotel Automation Software System

1) Introduction

This project is an automation software system for the use of hotel employees. This system is designed to be a desktop application on hotel's computers. With this project, each employee can see and set the information of rooms and customers in order to maintain their works efficiently. So, the users of this software system is hotel employees, not customers. All adjustments are made by considering that every employee has computer access to enter information about their job into the system.

In this project, we aimed to create an automation system that all workers can communicate easily and efficiently. For this reason, we will try to maintain a system which is able to use easily by all workers without any training. Therefore, our system will be easy to use but also contains every need of an hotel.

2) Requirement Analysis

2.1 Overview

There are workers in hotels which are mainly receptionists, room keepers, managers, accountants and kitchen chef. Receptionist can make a reservation with the help of this software, managers can control operations, accountants can calculate economic issues, room keepers can see dirty and clean rooms and chef in kitchen can see order of customers that comes from room service. This system aims to ease the communication among hotel employees and improve efficiency of service.

2.2 Functional Requirements

- Users of this program are hotel employees, not customers.

- This program will be designed with considering the all employees have computer access to make changes on rooms and customers information.

- Each user types which are receptionist, manager, accountant, room keeper and kitchen chef have their own interface according to their worker ID and password on opening page.

- Receptionist will be able to access customer and room informations, make and cancel reservations depending on capacity of hotel.

- Manager will be able to access worker informations, set shifts, pay salaries and control the economic status.

- Accountant will be able to access calcualte workers salaries, pay hotel taxes, calculate economic status and report it to the manager and set currency.

- Room keeper will be able to access room informations that if they are empty, full, clean or dirty; get minibar status, laundry need and ordered room service.

- Kitchen chef will be able to access customers meal order detail to prepare them, update the their bill and manage the hotel meal plan.

2.3 Non-functional Requirements

- System's user interface is designed according to employees types that means each employee can make changes only on their job related informations.

- System is user friendly that every employee -even not related with higher technology- can easily manage.

2.3.1 User-friendly Interface

AutoHotel system is user friendly in many ways:

- The program is easy to install since it is desktop application and do not need any external database system. It can keep informations in text files.

- The system is intuitive that it can be work well with good GUI.

- Software is efficient in a way that users can complete their job, features are well located, not maste work time while manage the system.

- It is a menu-driven programs, employees are able to find their ways with toolbars unlikely than command-driven programs.

2.3.2 Extensibility

Extensibility means system ability about future growth according to requirements. Good extensibility feature of software system can be measured with effort while adding new functionalities to software.

Our project supports the white-box extensibility that modified with source code. It is flexible and less restrictive. This project is an object- oriented software that supports inheritance and dynamic binding. So with extensions, original system cannot be affected by changings.

As an example, during extensions developers can add new worker types without changing original source code because there are inheritance between worker class and worker types which are receptionist, manager, accountant, room keeper and kitchen chef. In order to add new worker type,the extension does not affect these classes under favour of object-oriented software structure and specifically glass-box extensibility.

2.4 Pseudo Requirements

- The program will be implemented in Java.

2.5 Scenarios

There will be different scenarios for different users. There will be six different type of user which are receptionist, cleaner, manager, accountant and kitchen workers. Our program will detect users by their login information. These users will be have usernames and passwords, and these usernames will contain the information about their job description. Therefore, these scenarios show us how can these different users use this program depend on their job description.

2.5.1 Scenarios for Receptionists

Participating Actor : Receptionist

Entry Condition : Actor logins with his ID and password

Exit Condition : Actor pushes Logout Button

Main Flow of Events:
1. Actor will see all the rooms and detailed information about them
2. Actor will push Check-In button for customers
3. Actor will enter informations about customer and room if customer does not have reservation
4. Actor will push Check-out button for leaving customer

Alternative Flow of Events:
1. Actor will push Make Reservation button for customers who wants to reserve room and enter customers information
2. Actor will choose Menu bar to see the information about meals for workers and customers
3. Actor will choose Room Service bar to enter customers' room services information in order to send Kitchen Chef.

2.5.2 Scenarios for Room Keeper

Participating Actor : Room Keeper

Entry Condition : Actor logins with his ID and password

Exit Condition : Actor pushes Logout Button

Main Flow of Events:
1. Actor will see all the rooms with detailed informations about rooms
2. Actor will enter Laundry price and update customer's bill if there is a laundry
3. Actor will fill minibar and update customer's bill if customer consume minibar

Alternative Flow of Events:
1. Actor will push Set Clean button after cleaning it and changes room status from dirty to clean

2.5.3 Scenarios for Accountant

Participating Actor : Accountant

Entry Condition : Actor logins with his ID and password

Exit Condition : Actor pushes Logout Button

Main Flow of Events:
1. Actor will choose type of a worker
2. Choose an employee
3. Actor will see employee's informations and calculate his salary depends on his job and shifts.

Alternative Flow of Events:
1. Actor choose the Tax bar and calculate the tax to send Manager.
2. Actor choose the Economy bar and calculate the economical status of Hotel to send Manager.
3. Actor choose the Economy bar and set the currency

2.5.4 Scenarios for Manager

Participating Actor : Manager

Entry Condition : Actor logins with his ID and password

Exit Condition : Actor pushes Logout Button

Main Flow of Events:
1. Actor will choose Employees bar and Actor will see Employees in order depending on their job type
2. Actor can pay employees' salary and set their shifts.
3. Actor will choose the Economy bar and see the detailed informations about hotel's economic status

Alternative Flow of Events:
1. Actor will choose a room
2. Actor will see detailed information about room and customer
3. Actor will see occupancy rate of hotel

2.5.5 Scenarios for Kitchen Chef

Participating Actor : Kitchen Chef

Entry Condition : Actor logins with his ID and password

Exit Condition : Actor pushes Logout Button

Main Flow of Events:
1. Actor will get meal orders automatically from reception
2. Actor will see information about meal
3. Actor will send order and enter information

Alternative Flow of Events:
1. Actor will choose Menu bar to create meal menu for workers and guests

2.6 Use-Case Model

In AutoHotel Project, there will be five different user of the program and they have different screens and inputs about customers, hotel and employees. We use this use case model for requirements elicitation and analysis to represent the functionality of hotel automation system. The diagram represents the behaviour of the project from external view which is actor's view. This diagram also clarify boundary of the system. The actors which receptionist, accountant, manager, room keeper and kitchen chef are outside of the boundary in diagram. And use cases are inside the boundary. Each rectangle defines subsystems and dashed arrows defines the interactions between them.
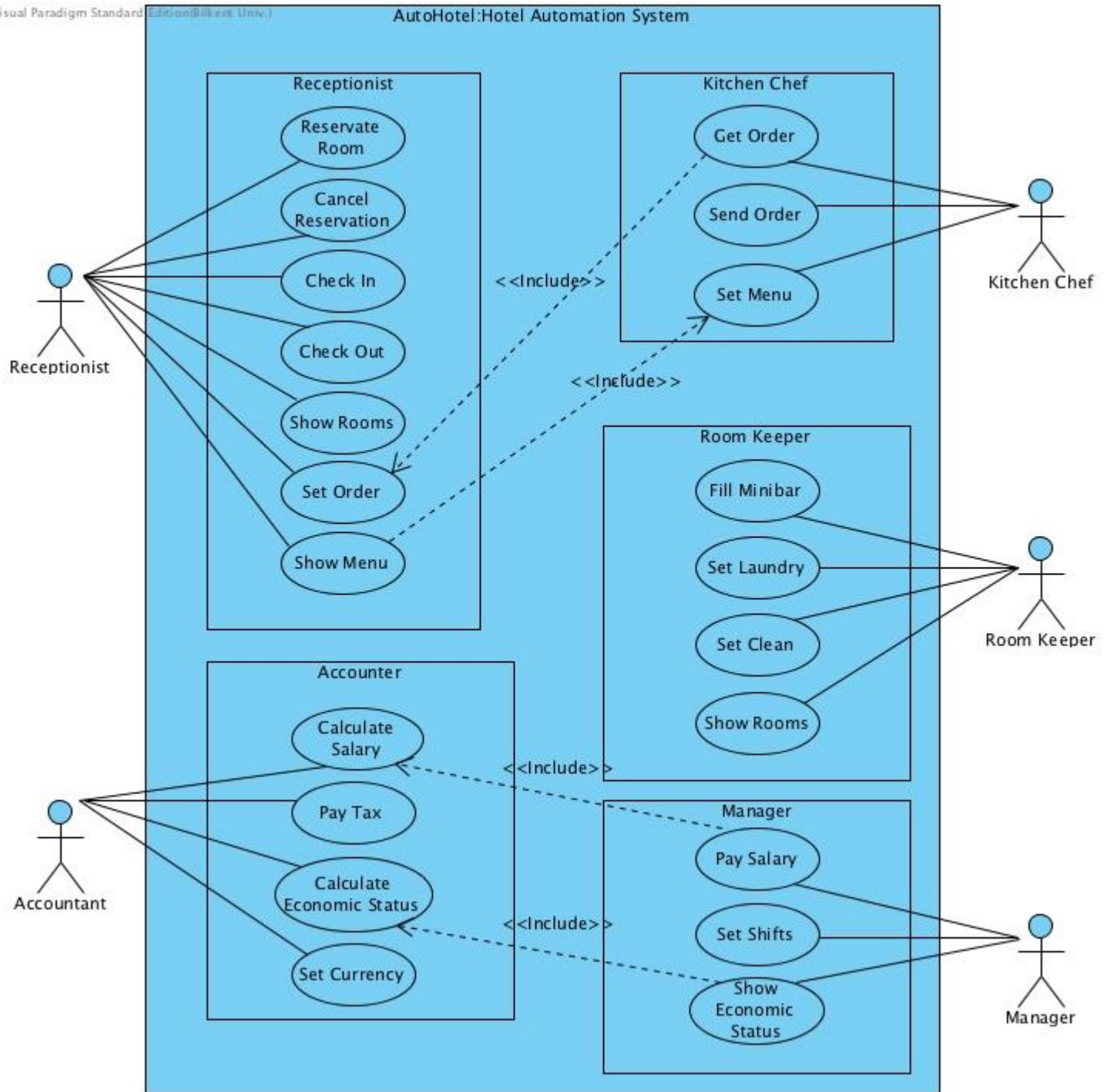
Figure 1 Use case model of whole system

2.7 User Interface

In this section, there are user interface prototypes. They do not provide much information about functionality and source codes but represent interfaces for most users. When the application runs, all users will see the welcoming screen with two text fields for username and password, and a login button. After login, the system direct the users to different pages according to their authorisation.
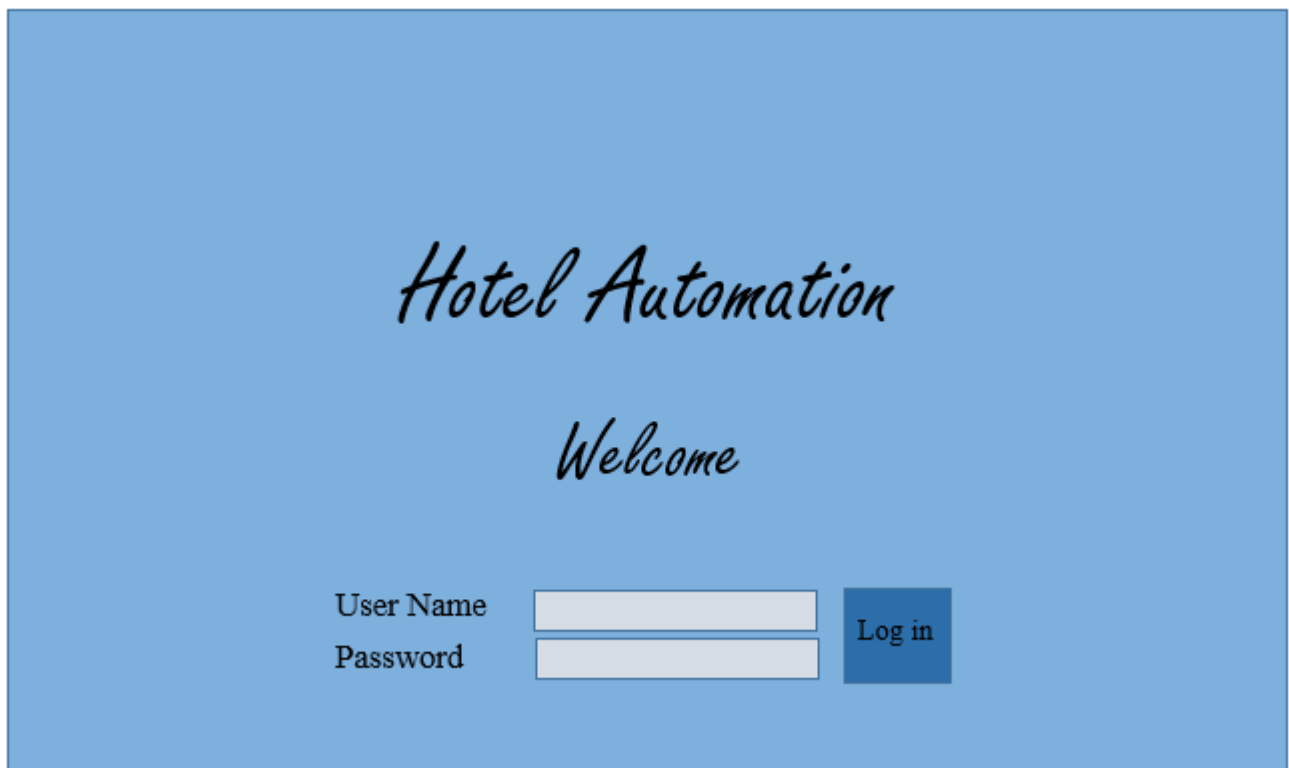

Figure 2 Opening Page

2.7.1 Receptionist Interface

Receptionists will be directed to the screen below. This is main control screen of receptionist. In the Rooms tab, a receptionist can see informations about all the rooms. To make an easier access to different groups of rooms, there are four buttons as follows, all rooms, reserved rooms, occupied rooms and empty rooms.
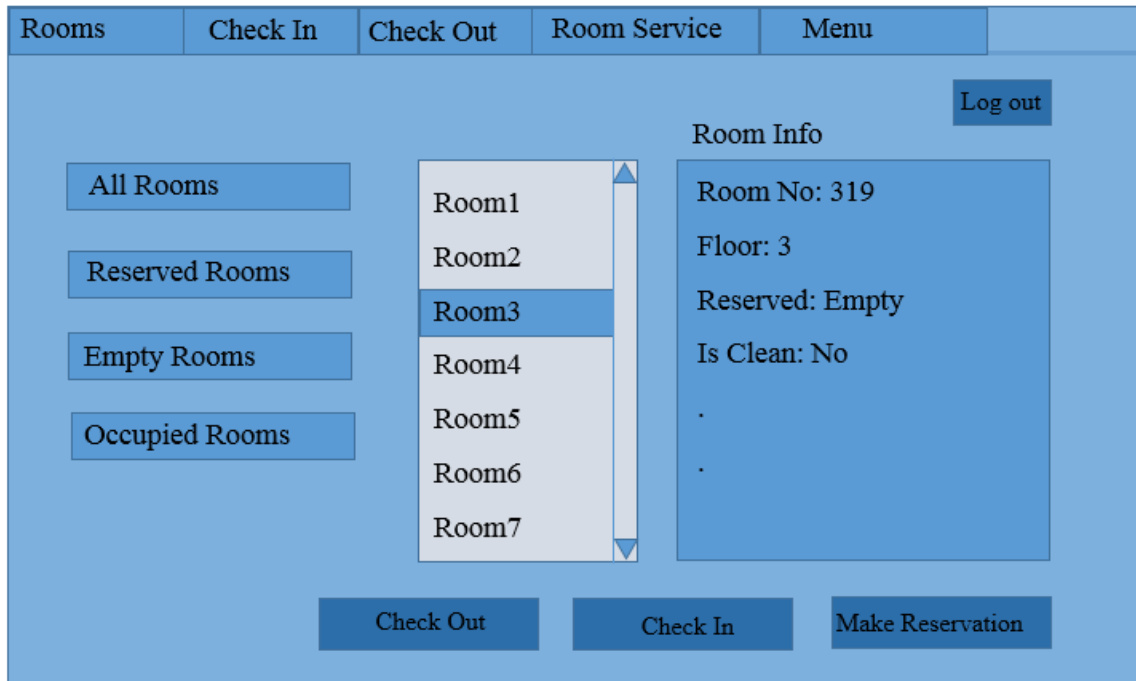
Figure 3 Receptionist Interface

Receptionists can see informations about each room, and with 'check in', 'check out' and 'make reservation' buttons, they can change the status of the selected room according to customer request. Only the receptionist has the rights to change reservations.

In 'Check in' tab, customer check in can be made by the receptionists. Receptionists will enter the customer info, and choose a room type depends on customer's request. According to chosen room type, list of available rooms will be displayed on the screen. Receptionist will be able to choose a room from the list, and by clicking the 'make reservation' button, receptionist can take the customer's information and store them into the system.

Figure 4 Room Reservation Interface

Here is a reservation page that customer's information is stored. Customer informations are needed to provide a better hotel service. Beside this, if customer reenter the system in further dates, system can remember customer information to ease reservation.

In 'Check Out' tab, customer check out will be made by the receptionists. Receptionists will be able to search according to customer name or room number. After search, customer info and room info will be displayed on the screen. By clicking the 'checkout

button receptionist is able to end the checkout process.

Figure 5 Room Information Page

In the menu tab, receptionists will only see the daily menu. Receptionist cannot change meal plan, it is kitchen chef's duty. They will just inform the customers about the menu if it's needed.
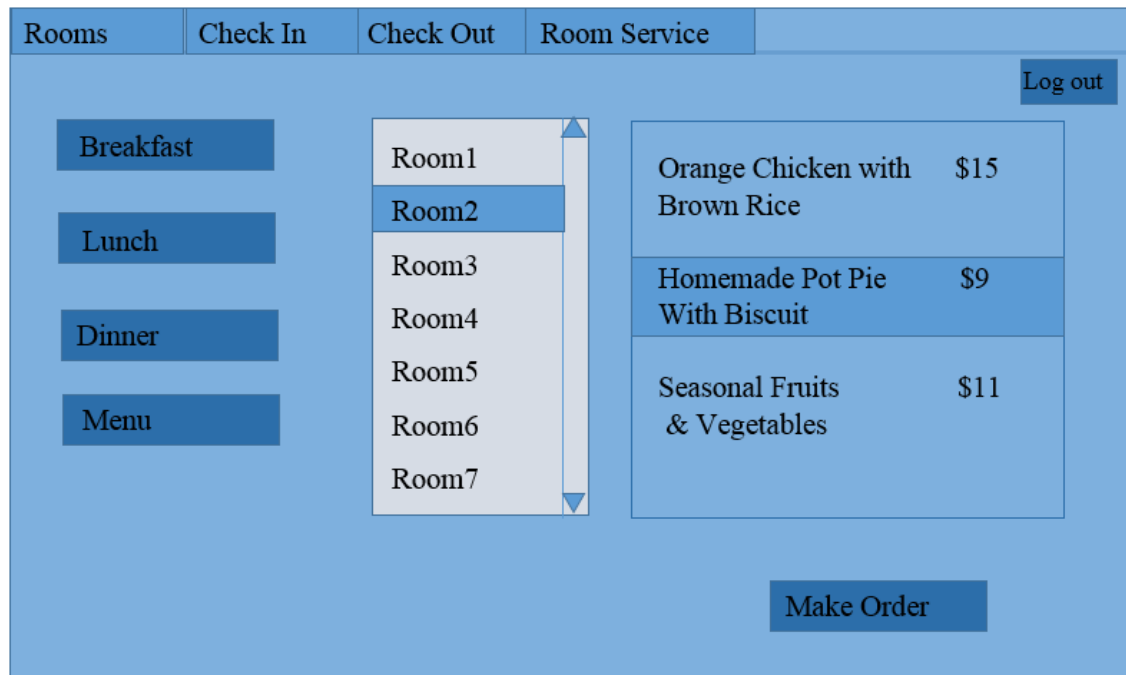
Figure 6 View of Meal Plan

2.7.2 Manager Interface

Manager will be directed to the screen below where he can see the general informations of the hotel and the rooms. The Hotel tab is similar with 'Rooms' tab where in the receptionist interface but the difference is that the manager can not change anything.

Figure 7 View of Room Status

In 'Employees' tab, manager is able to see employees from different departments. The manager check the informations about the employees, set their shits, and pay salaries.

Figure 8 View of Employee Information

In the 'Economy' tab, the manager can see the economic status of the hotel, which is organized by the accountant.

| Hotel | Employees | Economy | | Log out |

……… +20
……… -3
………+12
………-35
………+65
………+100
.
.
.

Profit: +30000
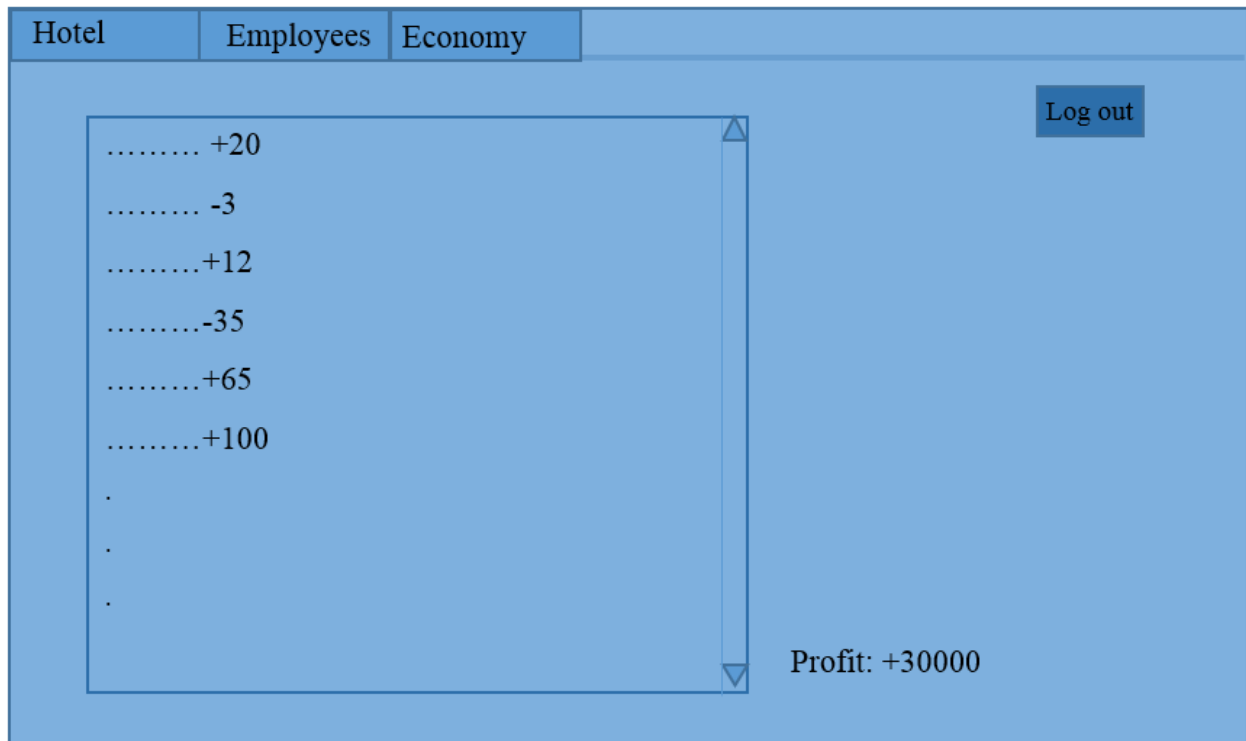
Figure 9 View of Economic Status

2.7.3 Accountant Interface

Accountant will be directed to the screen below where he can see the employees from different departments and calculate their salaries according to their shifts.

18

Figure 10 View of Employee Information

2.7.4 Kitchen Interface

Employees from the kitchen department will be directed to the screen below where they can enter the price of ordered meals after they prepare them. The meals are ordered by the customers as a room service.

Figure 11 View of Food Order Information

2.7.5 Room Keeper Interface

Employees from the cleaning department which are room keepers will be directed to the screen below where they can enter customers' laundry price, change the cleaning status of the room and change the packness status of the minibar.

Figure 12 View of Room Keeper

3 Analysis

3.1 Object Model

3.1.1 Class Diagrams

Hotel

+hotelName : String
+roomCapacity : int
+address : String
+customerCapacity : int
+currentCustomerNumber : int
+currentWorkerNumber : int
-income : double
-expense : double

Room 1..*

+roomNumber : int
-customerID : int
-free : boolean
-clean : boolean
-roomType : RoomType
-roomPrice : double
-miniBarPrice : double
-loundryPrice : double
-roomServicePrice : double
-mealOrder : String

+showAllRoom() : ArrayList<Room>
+calculateBill(roomNo : int) : double
+setClean(roomNo : int, c : boolean) : void
+setFree(roomNo : int, c : boolean) : void
+setMiniBarPrice(roomNo : int, price : double) : void
+setLaundryPrice(roomNo : int, price : double) : void
+setRoomServicePrice(roomNo : int, price : double) : void
+setMealOrder(meal : String) : void

Customer 0..*

-customerID : int
-name : String
-surname : String
-gender : String
-address : String
-phoneNumber : int
-birthDate : Date
-roomNo : int
-checkIn : Date
-checkOut : Date
-TCNo : int
-allCustomers : ArrayList<Customer>

+addCustomer(c : Customer) : boolean
+deleteCustomer(w : Worker) : boolean
+showAllCustomer() : ArrayList<Customer>
+getCustomer(name : String) : Customer
+getRoomNo() : int
+setRoomNo() : boolean
+setCheckInDate() : boolean
+setCheckOutDate() : boolean
+getCustomerID(customerName : String) : int

Worker 0..*

-name : String
-surname : String
-ID : int
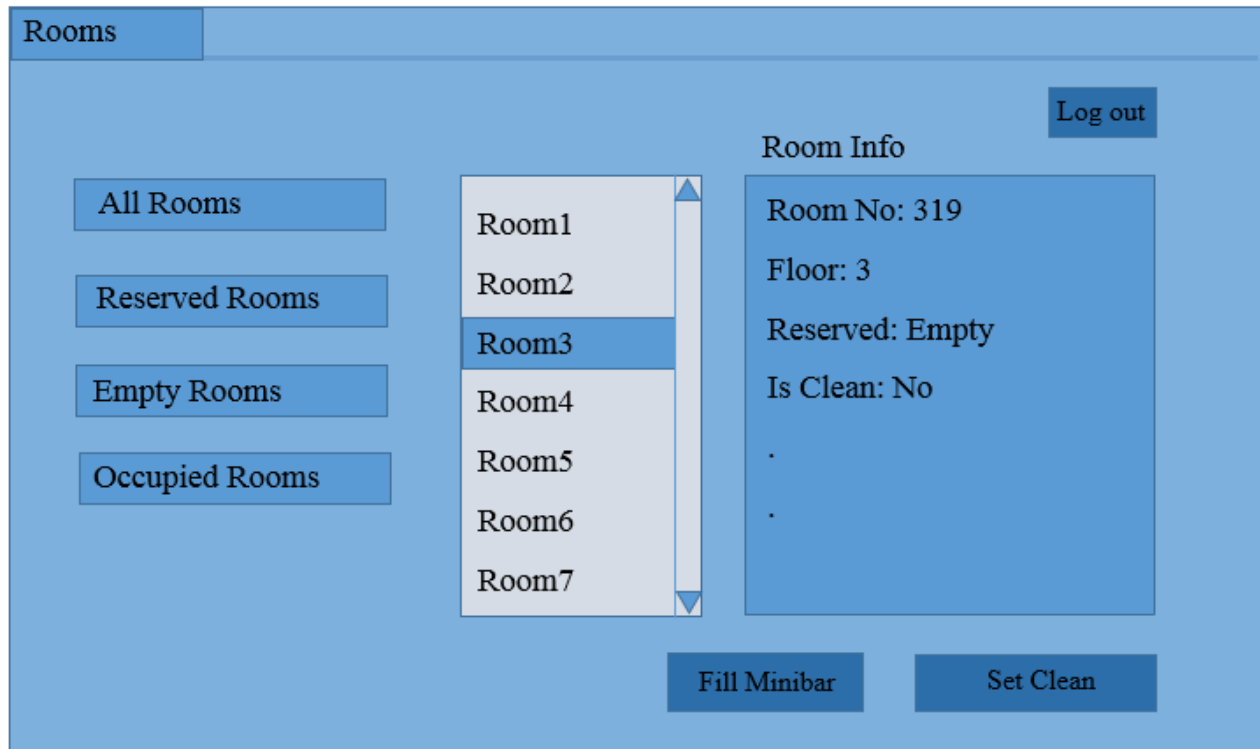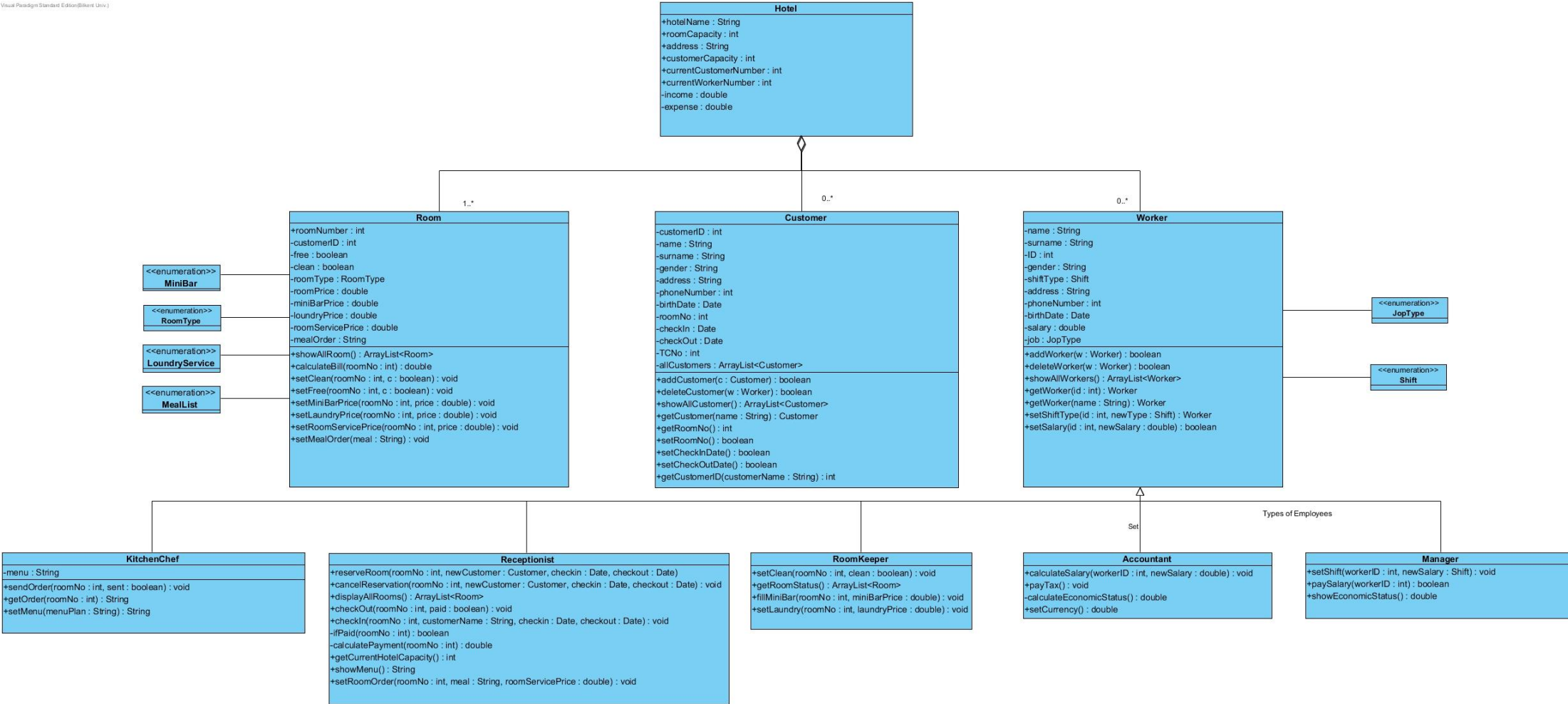-gender : String
-shiftType : Shift
-address : String
-phoneNumber : int
-birthDate : Date
-salary : double
-job : JopType

+addWorker(w : Worker) : boolean
+deleteWorker(w : Worker) : boolean
+showAllWorkers() : ArrayList<Worker>
+getWorker(id : int) : Worker
+getWorker(name : String) : Worker
+setShiftType(id : int, newType : Shift) : Worker
+setSalary(id : int, newSalary : double) : boolean

<<enumeration>> MiniBar

<<enumeration>> RoomType

<<enumeration>> LoundryService

<<enumeration>> MealList

<<enumeration>> JopType

<<enumeration>> Shift

Types of Employees

Set

KitchenChef

-menu : String

+sendOrder(roomNo : int, sent : boolean) : void
+getOrder(roomNo : int) : String
+setMenu(menuPlan : String) : String

Receptionist

+reserveRoom(roomNo : int, newCustomer : Customer, checkin : Date, checkout : Date)
+cancelReservation(roomNo : int, newCustomer : Customer, checkin : Date, checkout : Date) : void
+displayAllRooms() : ArrayList<Room>
+checkOut(roomNo : int, paid : boolean) : void
+checkIn(roomNo : int, customerName : String, checkin : Date, checkout : Date) : void
-ifPaid(roomNo : int) : boolean
-calculatePayment(roomNo : int) : double
+getCurrentHotelCapacity() : int
+showMenu() : String
+setRoomOrder(roomNo : int, meal : String, roomServicePrice : double) : void

RoomKeeper

+setClean(roomNo : int, clean : boolean) : void
+getRoomStatus() : ArrayList<Room>
+fillMiniBar(roomNo : int, miniBarPrice : double) : void
+setLaundry(roomNo : int, laundryPrice : double) : void

Accountant

+calculateSalary(workerID : int, newSalary : double) : void
+payTax() : void
-calculateEconomicStatus() : double
+setCurrency() : double

Manager

+setShift(workerID : int, newSalary : Shift) : void
+paySalary(workerID : int) : boolean
+showEconomicStatus() : double

21

3.2.1 State Chart Diagram

3.2.2 Sequence Diagram

In this section, there are sequence diagrams that are used to formalize the dynamic behaviour of the system and to visualize the communication among objects. In this diagrams, left most column represents the actor who initiates the use case. Labeled arrows represents the functions that actor or object send s to other object which functions are shown more clearly in class diagram. In receptionist sequence diagram there are 2 combined fragments that are for if- else consditions that activity change among them.
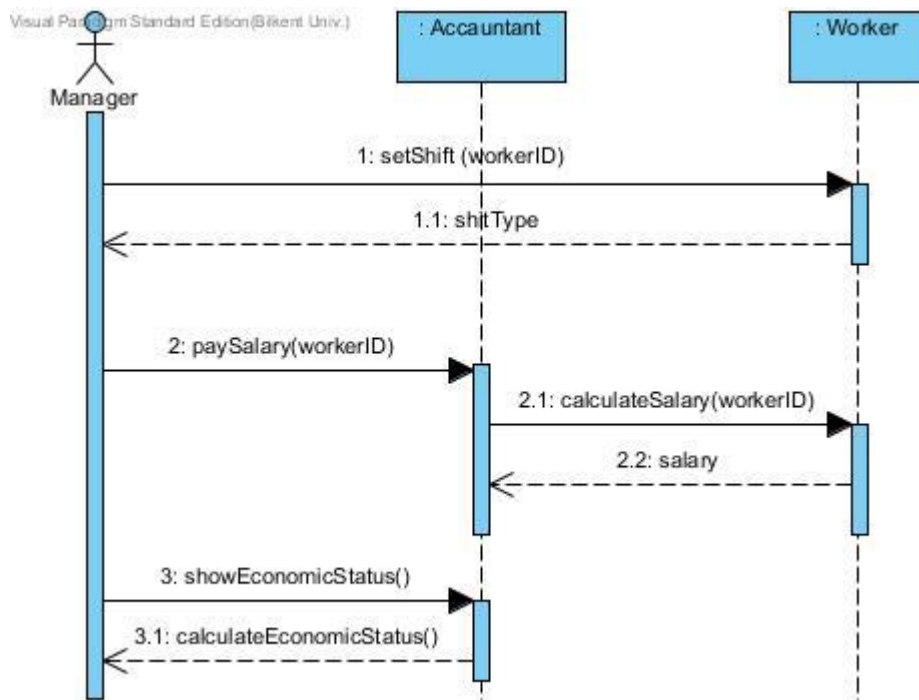
: Hotel   : Room   KitchenChef

Receptionist

1: getCurrentHotelCapacity()

1.1: capacity

alt

[capacity > 0]

2: showRooms()

2.1: emptyRooms

2.2: reservedRooms

3: reserveRoom(roomNo,customerName, checkIn, checkOut)

4: checkIn(roomNo, customerName, checkIn,checkOut

[capacity <= 0]

5: cancelReservation(roomNo, customerName, checkIn, checkOut)

alt

[customerNo > 0]

6: checkOut (roomNo)

[customerNo = 0]

7: [system sleep until customer enter]

8: [phone call meal order]

9: setRoomOrder(roomNo, meal, roomSerivceprice)

10: getOrder(roomNo)

10.1: meal

11: sendOrder(roomNo)

12: showMenu()

13: menu

menu is a lost message since the
receiving event does not occur in
this sequence diagram

24

Figure 14 Sequence diagram for the Manager use case
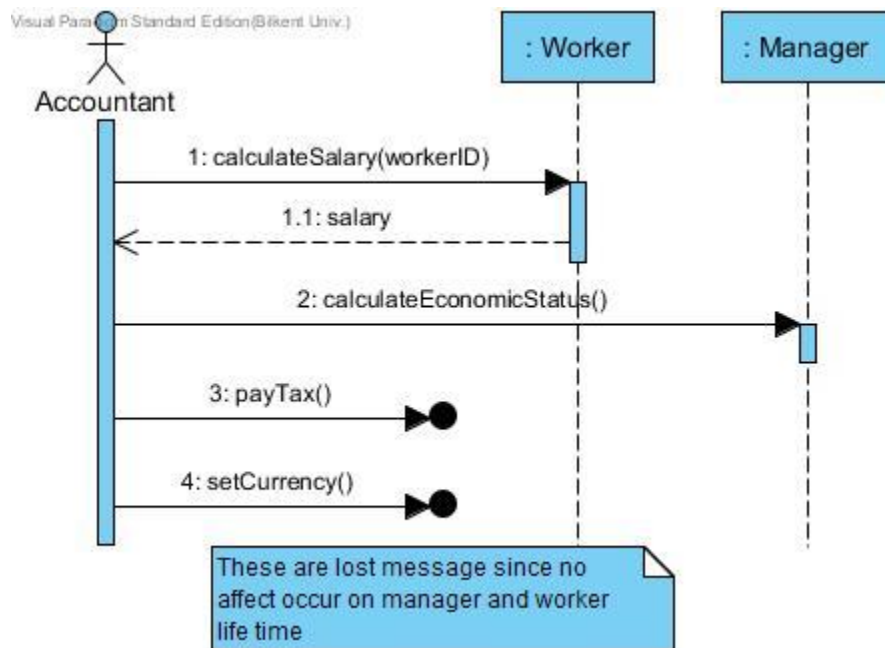
Figure 15 Sequence diagram for the Accountant use case

Room Keeper

: Room

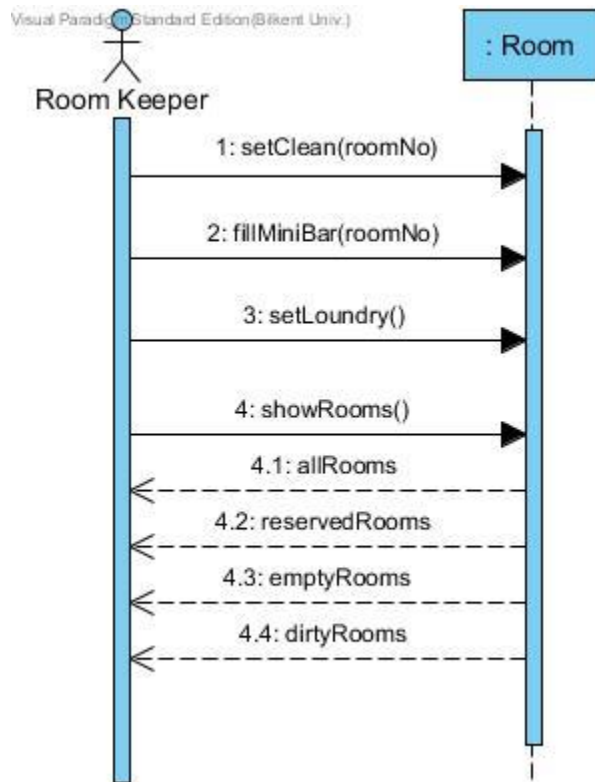1: setClean(roomNo)

2: fillMiniBar(roomNo)

3: setLoundry()

4: showRooms()

4.1: allRooms

4.2: reservedRooms

4.3: emptyRooms

4.4: dirtyRooms

Figure 16 Sequence diagram for the Room Keeper use case

Kitchen Chef

: Room

Receptionist

1: setRoomOrder(roomNo, meal, roomSerivceprice)

2: getOrder(roomNo)

2.1: meal

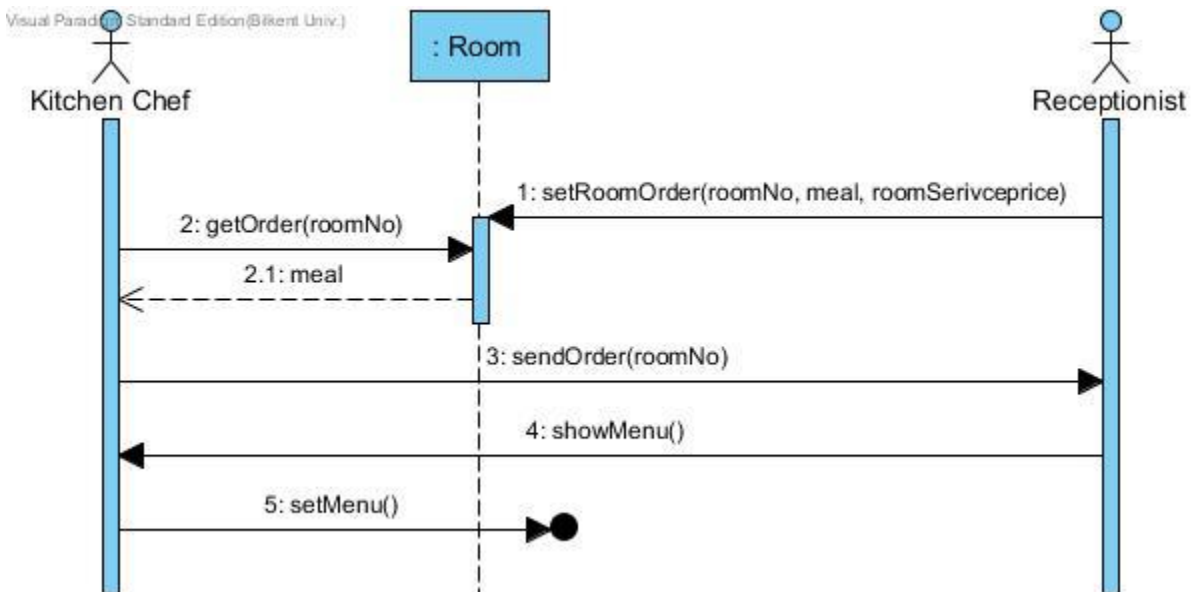3: sendOrder(roomNo)

4: showMenu()

5: setMenu()

Figure 17 Sequence diagram for the Kitchen Chef use case

4 Conclusion

When we were making our analysis, we used Visual Paradigm to prepare models and it was very useful tool to create our diagrams.

During this analysis process, we learned how to start to create this project efficiently instead of jumping directly to writing code. We deeply discuss the structure of project, user communities and requirements of them, class diagrams, which storage technique will be useful and how user interface will be efficient. Also we improved our skills while decide the structure of classes and objects, visual representation of them and handle possible logic mistakes without any implementation. As we were creating this report, we focused on requirements of our users and tried to create a program which realistic, user-friendly and implementable. When we create our diagrams, we always care about implementation in order to create a realistic and meaningful analysis report.

As a result, the users of this software system is hotel employees not customers. All adjustments are made by considering that every employee has computer access to insert information about their job into the system.