IFSC3360: System Analysis and Design

Final Project Report

Hotel Reservation Information System

Hilma Svalander

7 december 2020

1. Introduction

The goal for this project was to design and plan a reservation information system for a hotel. The reason for designing this system is that I previously have experince with planning and designing a mobile application that records and calculate data that is inputed by users to help them in their everyday life of finding a healthy and balanced lifestyle. I have used this app development as a foundation for many of my school projects and figured it would be of benefit to myself if I broaden my view and experince of information systems expand my knowledge into other types of systems. Hence I chose to design a system more aimed to be used on a computer or as a website rather than a smartphone application. Hotel reservation systems are common and thus it would allow easy visualization of the entities, processes and data flows needed in this system; so that focus could be spent on thinking and designing the actual data flow and methods. Thus, the system covered in this report is not a brand new system. However it is not based on a specific existing system or hotel but thought of from ”scratch”.

Hotel reservation systems are needed since a hotel is a place that enables people a brief stay traveling outside of their own cities or even countries. Thus, a hotel needs to actively be able to store and update information about rooms, reservations and guests. The hotel need to keep track of what rooms have been reserved by which guest and for which days. As well as what rooms are available in order to successfully accommodate the guests. A successful hotel needs to accommodate the guest who has reserved rooms while keeping track of what rooms are still available to accommodate new guests. Therefore a hotel is in need of a reservation system that organizes and updates all of this information.

The requirements for this hotel reservation system is therefore that the system should allow guests to search for and reserve rooms at the hotel. The system will maintain information about which guest that have reserved which rooms and for what dates. Rooms will have prices applied to them that determine the price of the stay for the guests and these prices can only be updated by the system admin. Thus the system need data stores for rooms and for reservations. Further, the

system will apply and authorize payments of the reservations through a third-party payment authorization service. The system needs to seamlessly update the rooms as ”occupied” when they are reserved for specific dates and thus disabling them from being reserved by other guests. Likewise the system needs to change the room availability status to ”available” once the reserved stay of the guest is over or if the guest cancels their reservation.

2. System Requirements Model

2.1 Data Flow Diagrams

Data flow diagrams (DFD) identifies the data and processes needed in the system. Combining different levels of DFDs provide a logical model of *what* the system will do. The highest level of DFD is called a context diagram and show the scope and boundaries off the system. Figure 2.1.1 shows the context diagram of this system. The central process, numbered as 0 represents the entire Hotel Reservation System and how it interacts with the three entities Guest, Payment Authorization Service and System Admin. This diagram shows broad views of the data that will be passed between the system and entities. An example is that the guest will request to make a reservation and the hotel reservation system will generate a reservation for the guest.

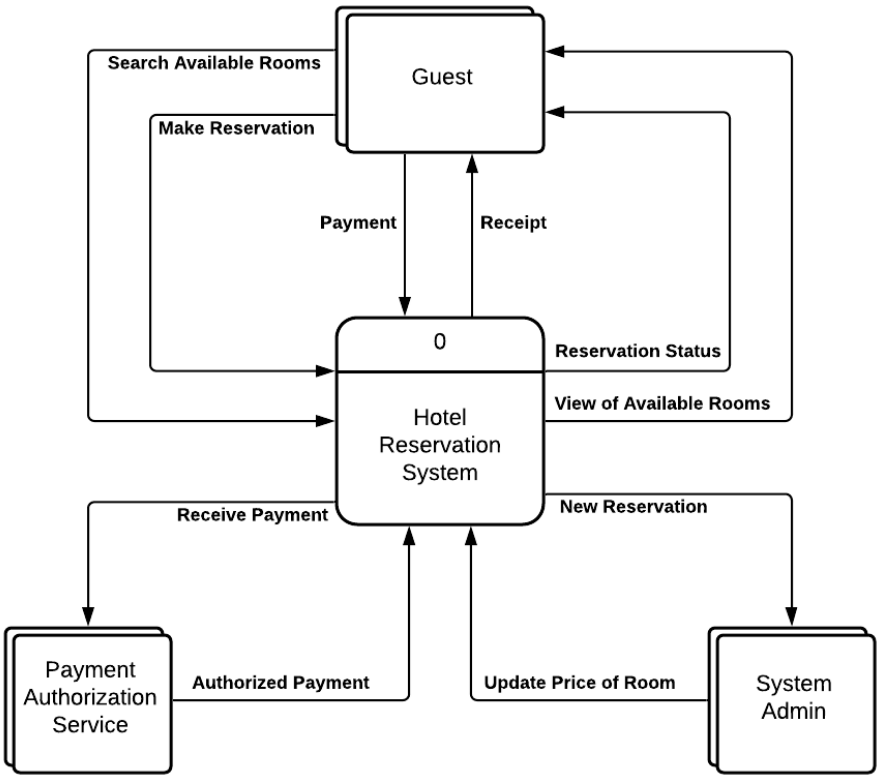

Figure 2.1.1 - Context Diagram

When zooming in on process 0, the Data Flow Diagram of Level 0 (figure 2.1.2) can be created. It shows the three external entities interacting with the subprocesses and data stores of the hotel reservation system. The main subprocess that the hotel reservation system is divided up in are: check room availability (1), create reservation (2), delete reservation (3), apply payment (4), and update room details (5). The DFD Level 0 Diagram show how these sub-processes interact with the two data stores room details (D1) and reservations (D2) and the general data flow that occurs between these processes, data stores and entities. To give an example of a specific process in this diagram we can look at the *check room availability* process (1). The guest will start the process by searching for rooms during preferred dates and the process will communicate with the data store D1 to return a list of available rooms to the guest.

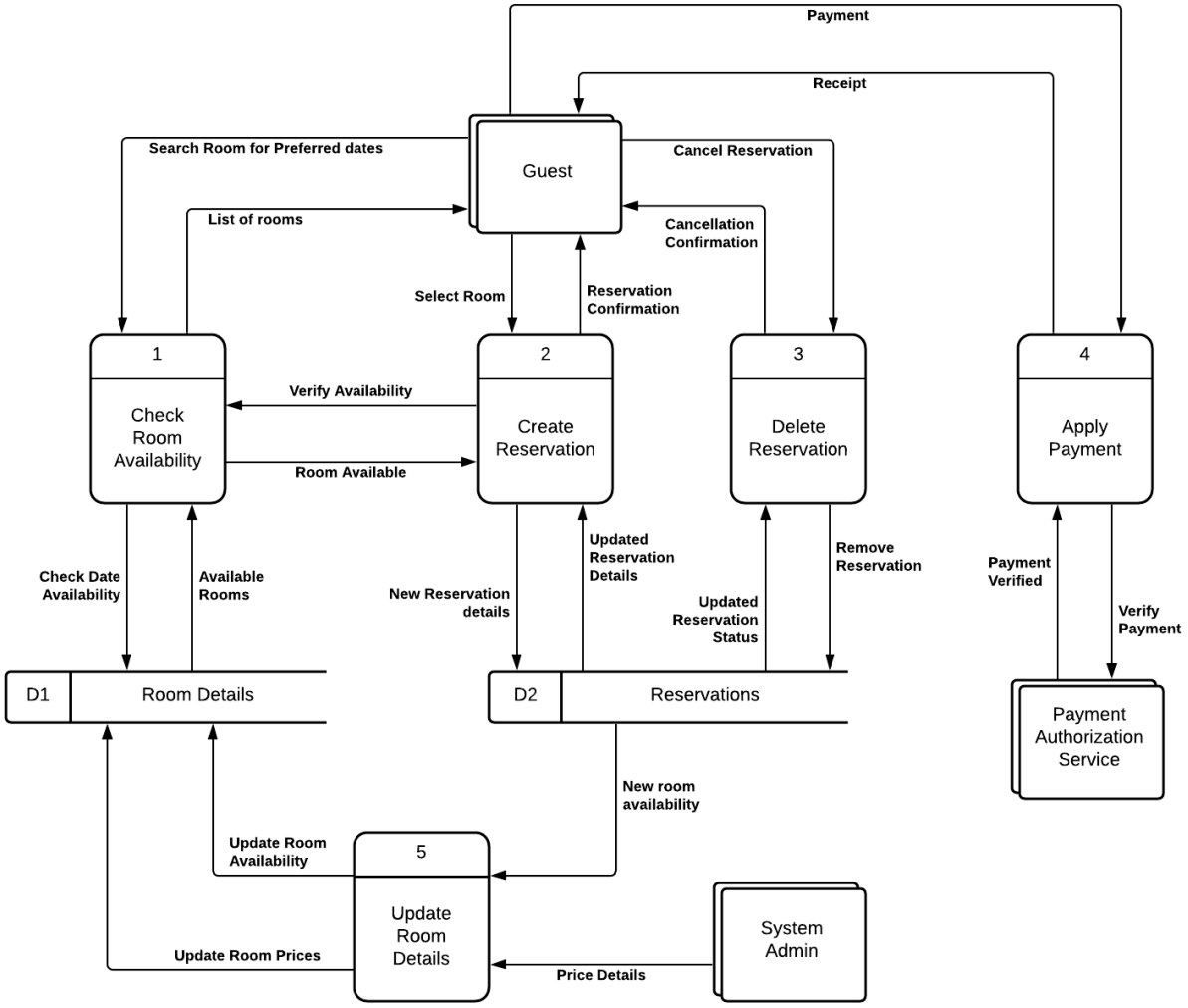

Figure 2.1.2 - DFD Level 0

2.2 Use Case Diagrams

Once the scope, boundaries and main processes have been defined through the data flow diagrams the use case diagram can be modeled. In a use case diagram the key actors that will interact with the system are identified. Further, it is identified what actions or tasks these actors want to perform or experience when using the system. A use case is therefore *what* a user wants to do when using the system. In the Hotel Reservation System there are three key actors: the *guest,* the *payment authorization service* and the *system admin*. Five use cases are identified and connected to at least actors. The use cases involving only the guest are: *search rooms, make reservation* and *cancel reservation*. The guest and payment authorization service are both actors involved with the use case *pay for reservation*. Lastly, the system admin is connected to the use case *update price of rooms*.

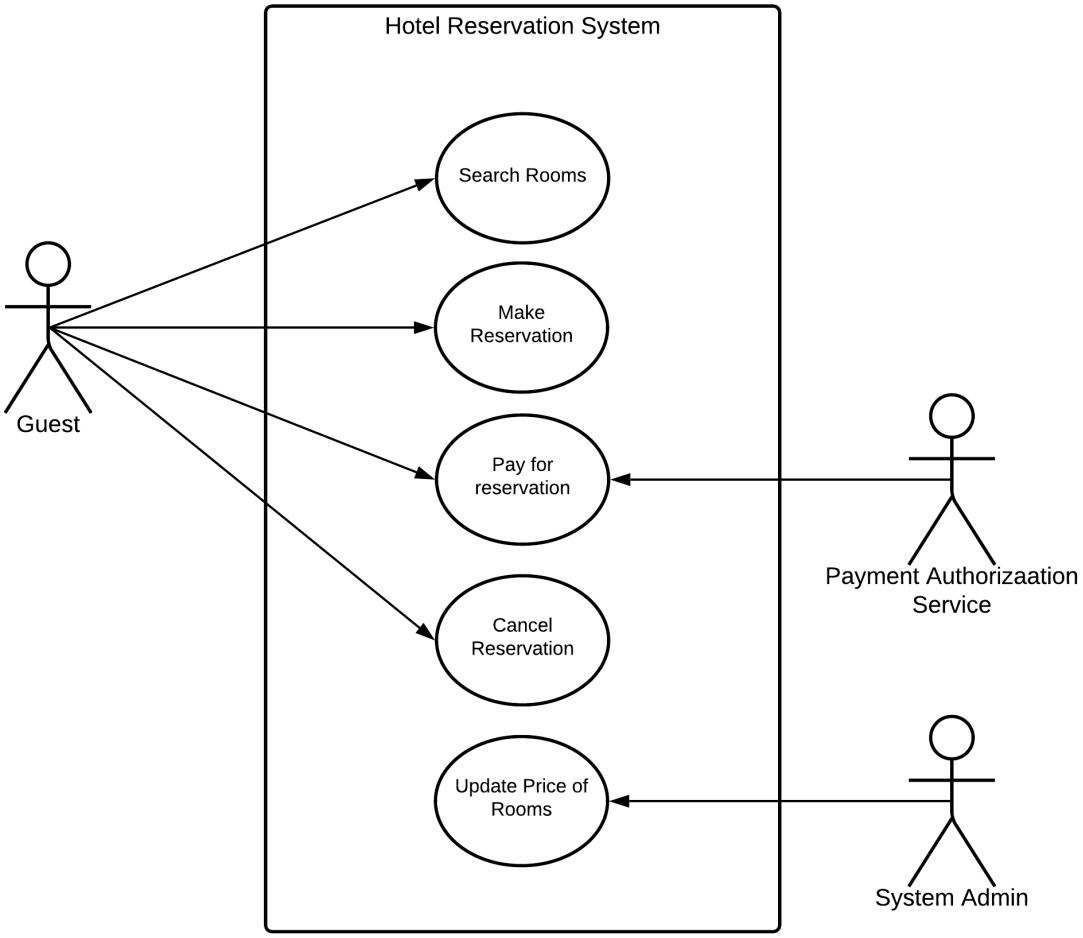

Figure 2.2.1 - Use Case Diagram

Each use case is further explained with details such as describing the use case, the actors involved, main success scenarios, alternative scenarios as well as any pre- and postconditions. This results in the five use cases being fully dressed. These dressed use cases are shown in the following five tables. To explain these fully dressed use cases the *make reservation* (table 2.2.1) use case will be explained. First the name of the use case and actors involved are defined in the table. In this case only the guest actor is involved. The description of the use case provides a brief description of what actions are taken and or what information that is used for this particular use case. Here, the make reservation use case describes the process used when a customer requests to reserve a room through the system. Successful completion of the use case involves the guest requesting to reserve a room that the hotel reservation system verifies is available. If the room is available the reservation is created and the room's availability is set to occupied. Then the guest is promoted to provide personal details and the reservation can be saved. The guest then receives a reservation confirmation. An alternative completion of this process would be if the room the guest selected not is available. Then, the guest will be prompted to select another room or try and search for other dates. The precondition of this process is that the room is available whereas the postcondition that the room as been reserved and set as occupied.

Table 2.2.1

MAKE RESERVATION use case	
Name:	Make Reservation
Actor:	Guest
Description:	Describes the process used for a guest to make a reservation
Successful completion:	1. Guest request to reserve selected room. 2. Hotel system checks if room is available. 3. If room is available a reservation request is created. 4. Room is set as occupied. 5. Guest is promoted to provide personal information. 6. Reservation is saved and guest receives confirmation.
Alternative completion:	3. If room is not available, guest is notified and prompted to select either another room or try other dates.
Precondition:	Room is available
Postcondition:	Room is successfully reserved and availability status is set to occupied.

Table 2.2.2

SEARCH ROOM use case	
Name:	Search Room
Actor:	Guest
Description:	Describes the process used for a guest to search and view rooms
Successful completion:	1. Guest search rooms for preferred dates. 2. Hotel System returns list of available rooms.
Alternative completion:	2. No rooms are available for selected dates i. Hotel Reservation System prompts guest try and search for other dates.
Precondition:	No rooms are shown
Postcondition:	List of rooms are showed successfully

Table 2.2.3

CANCEL RESERVATION use case	
Name:	Cancel Reservation
Actor:	Guest, Payment Authorization Service.
Description:	Describes the process used for a guest to cancel a reservation in their name
Successful completion:	1. Guest requests to cancel reservation. 2. Hotel system checks if request is sent earlier than 24 hours before reserved start date. 3. If request was sent more than 24 hours before reserved date the hotel system deletes the reservation. 4. The system checks the payment status of the reservation. 5. If the reservation is paid for, a request to apply a refund is sent to payment authorization service. 6. The availability status of the cancelled room is set as available. 7. Guest receives cancellation confirmation.
Alternative completion:	3. If request was sent within 24 hours of reserved start date the cancellation request is denied and payment is still required. 5. If the reservation is not paid for yet no request for refund is applied.
Precondition:	Room is reserved
Postcondition:	Room is successfully cancelled and availability status is set to a occupied

Table 2.2.4

PAY FOR RESERVATION use case	
Name:	Pay for reservation
Actor:	Guest, Payment authorization service
Description:	Describes the process used for a guest to pay for their reservation
Succesful completion:	1. Guest provides payment information and request to pay. 2. Hotel system redirects payment request to third party payment authorization service. 3. Payment is verified by payment authorization service. 4. The payment is applied. 5. A receipt is sent to guest.
Alternative completion:	3. Payment request t is denied i. Hotel Reservation System notifies that payment was denied and prompts user to try another payment method. ii. No payment is applied.
Precondition:	Room is reserved but not paid for
Postcondition:	Room is successfully paid for

Table 2.2.5

UPDATE PRICE OF ROOM use case	
Name:	Update Price Details
Actor:	System Admin
Description:	Describes the process used for the System Administrator to update the price details for rooms
Succesful completion:	1. System Admin selects room to update price for 2. System Admin provides new price details 3. Room price is updated 4. System Admin receives confirmation of price update 5. New price details are shown to guests
Alternative completion:	1. Price entered is same as previous price i. Hotel Reservation System prompts user to provide new price information that is not same as current price.
Precondition:	Already set price is shown to customers
Postcondition:	New price is successfully updated for selected room and showed to customers.

2.3 Interaction Diagrams - System Sequence Diagram

The fully dressed use cases are then used to create the System Sequence Diagram. This is an interaction diagram that shows the logical order of how the different actors interact with the entire system. It also shows in what order messages are sent within the system as well as depicts the messages sent and returned for the five use cases. In this diagram, shown as figure 2.3.1, the two actors guest and system admin that interacts with the hotel reservation system are shown as stick figures. The third actor, payment authorization service, is shown as a separate system that the hotel reservation system interacts with. To give an example the first message in the digram shows the use case *search room* where the guest actor passes the message ”search rooms” to the hotel reservation system which returns a list of available rooms to the guest.

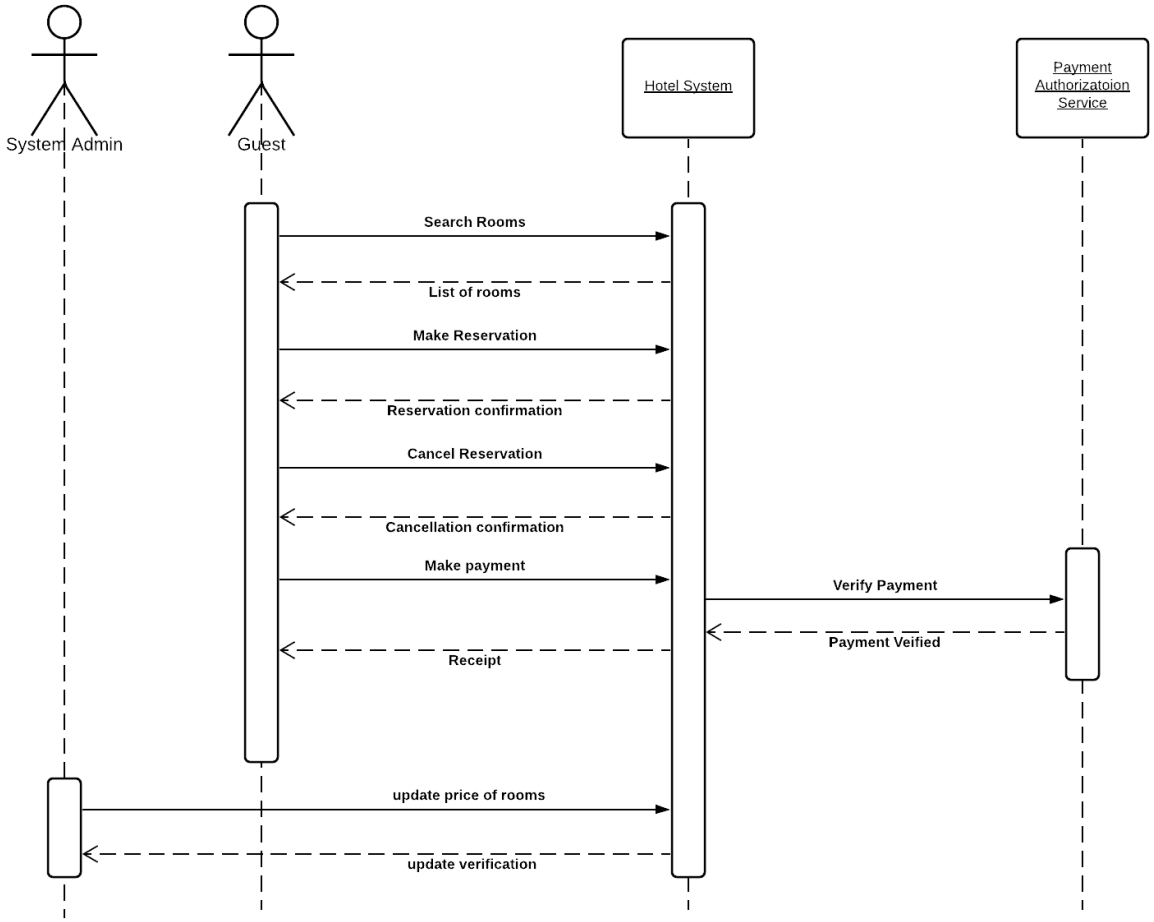

Figure 2.3.1 - System Sequence Diagram

2.4 Interaction Diagrams - Sequence Diagrams

Once the System Sequence Diagram (SSD) is made Sequence Diagrams (SD) for each use case can be created. Sequence diagrams help determining what methods and objects will be needed when creating the system. The following five sequence diagrams show the five different use cases.

The first message in the SSD shows the search room process. Figure 2.4.1 below shows the sequence diagram of it. The guest will search for available rooms and the room object will return a list of available rooms.

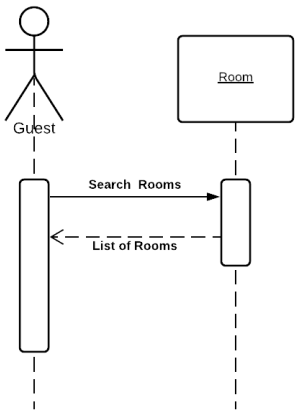

Figure 2.4.1 - Search Room SD

The make reservation process (figure 2.4.2) will start with the guest requesting to reserve a room. The room object will, through a message to itself, verify that the room is available and if it is, the room object will send a message to create the reservation. This will create the reservation object where the room, start and end dates as well as guest data will be stored. The reservation object will interact with the room object in order to change the availability status of the room into occupied. The guest will then be notified that the room has been selected and prompted to provide personal information that will be stored in the guest and reservation objects. Once this is done the reservation is saved and a reservation confirmation is sent to the guest.

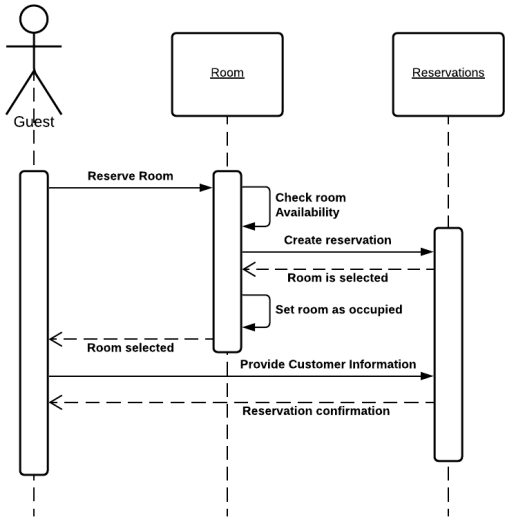

Figure 2.4.2 - Make Reservation SD

The third sequence diagram shows the cancel reservation process (figure 2.4.3). The guest will request to cancel their reservation and in the main success scenario the reservation object will then verify the 24-hour cancellation constraint. If the request was sent earlier than 24 hours from start date of reservation the process proceeds. The reservation object checks the payment status of the reservation and if it is not paid for the reservation is deleted. The reservation object sends a message to the room object to update the availability status into occupied. The guest receives a cancellation confirmation.

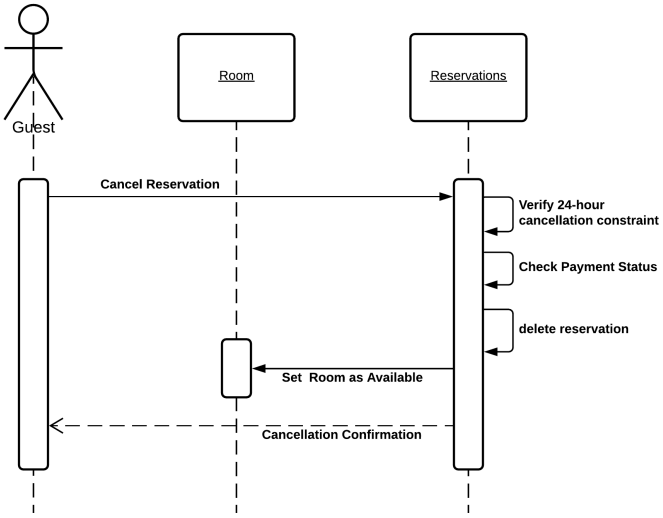

Figure 2.4.3 - Cancel Reservation SD

Next is the make payment sequence diagram (figure 2.4.4). The guest will provide payment details to the system that will redirect the payment to the payment authorization system. In the main success scenario the payment is verified and the payment status is updated in the reservations object and a receipt is sent to the guest.

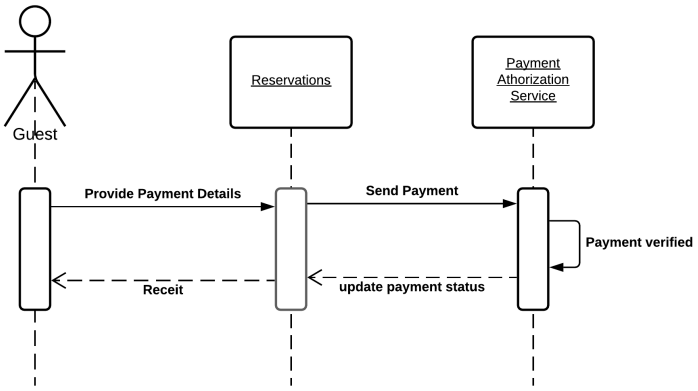

Figure 2.4.4 - Make Payment SD

The last sequence diagram shows the update price of room process (figure 2.4.5). Here the actor is the System Admin that send a message to the room object to update the price details of selected rooms. The room object will make the updates and confirm the system admin that the update has been applied. This return message might not seem as important from a system perspective. However, when it comes to user experience it is important for a system to continuously verify and confirm its users that requested actions have been applied. Therefore this return message is shown here.

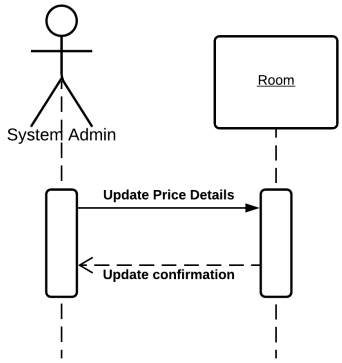

Figure 2.4.5 - Update Price of Rooms SD

2.5 Class Diagrams

The last step of the requirement modeling is to create the class diagrams. Class diagrams show the classes, attributes, operations as well as the relationships between a systems' classes. It is a convenient step of system design as class diagrams easy can be transitioned into object-oriented programming languages. The classes needed can be defined by looking at what objects are used in the SDs and the messages sent between these objects are operations used by the classes. Hence, once the SDs are created the class diagram can be designed. In this project and for this system five classes were identified: *user, system admin, guest, room* and *reservation*. Each class is equipped with attributes and operations that are either private, public or protected. Further, how these classes interact with each other are identified through their relationships. This diagram shows the three types of relationships between the classes: association, dependency and inheritance.

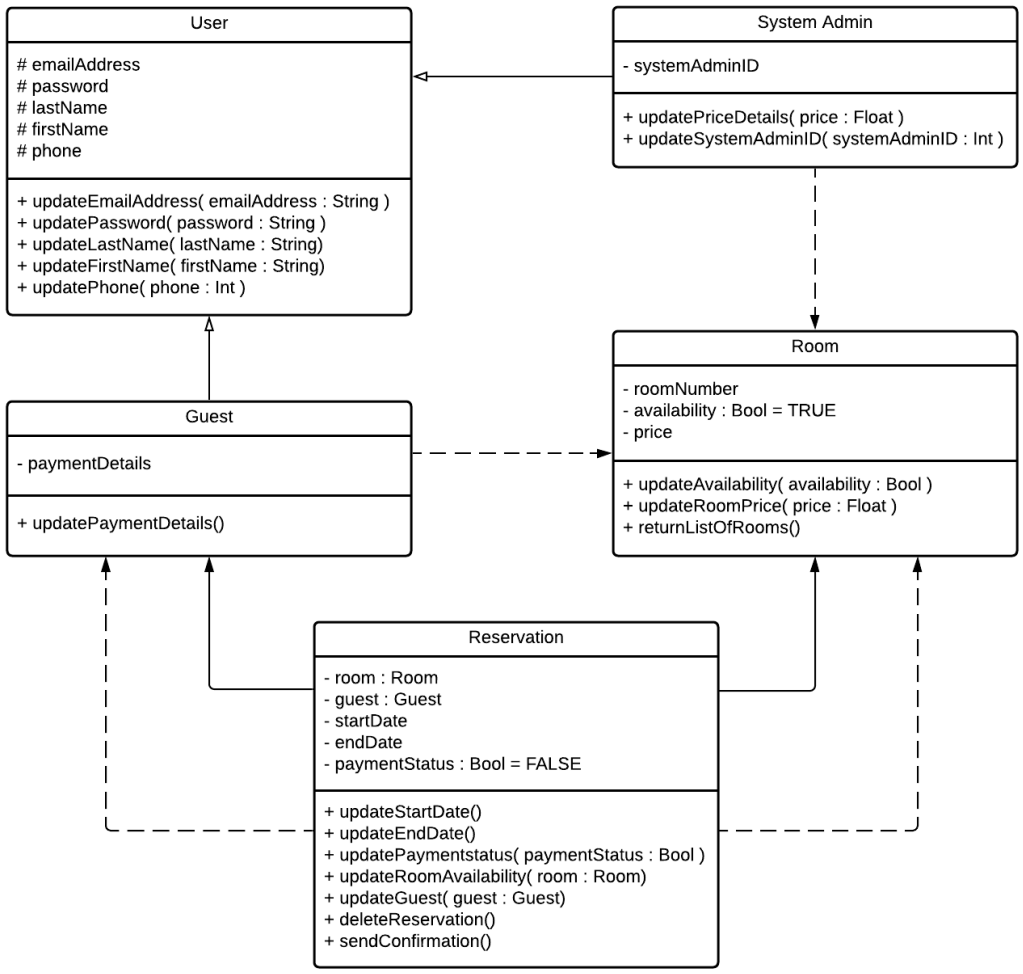

Figure 2.5.1 - Class Diagram

The system admin and guest classes both have an inheritance relationship with the user class. Inheritance is when a subclass inherits a superclass and thus the subclass gets equipped with all attributes and operations of their superclass. This is showed with a dashed line with a flat triangular arrow head. In order for the subclasses, guest and system admin, to access the attributes of the user class, they were defined as protected. This was done since making them public would have negative effects as it would allow personal and sensitive information to be accessed from anywhere in the system.

The second relationship type is shown where the reservation class has association relationships with the guest and room classes since it has attributes of type guest and type room. This is represented by a solid line and arrow head.

The last relationship type shown in this diagram is the dependency relationship. This relationship occurs when a class sends a message to or receives parameters from another class. It is represented by a dashed line with an open arrow head. In this system there are multiple dependency relationships. First, the system admin class and the guest class sends messages to the room class and therefore have dependency relationships with the room class. Further, the reservation class receives parameters of type room and type guest which gives the reservation class dependency relationships with these two classes.

3. Lessons Learned

This project was very educational and I learned a lot from it. First and foremost, I learned through this project that system design is something I really want to explore as a career path. The critical thinking, problem solving and analytics skills are all skills I have and enjoy using. Second, as the project progressed I realized how easy it is to design and revise a system while going through this types of requirement modeling as was done for this project. For example, I had not intended to have a user class in my class diagram but after creating the system admin and guest class I realized that both of them very users of the system and shared many attributes. The user class was then created to enable the system to have different types of users and to let the other classes inherit from it.

If I were to revise this project and expand on it, I would like to add a *create account* and a *log in* feature to the system. This would enable guests to save have their data saved which would make the user experience for them more smooth as well as enabling the system to have their personal details stored for easy access. Another feature I would like to add is a fourth actor/external entity like a *receptionist*. A receptionist would be at the front desk of the hotel for which the system was designed and would add essential features to a reservation system. One process would be for the receptionist to make reservations for guest that call the hotel to reserve a room. Another would be to check the guest in and out from their room. Lastly, the receptionist would be able to handle payments if the guest wished to pay in person after their stay rather than do it through the system when reserving the room. One critical part to think about then would be if the hotel system should have a payment system of their own, or if it still should be managed by a third party service.

In conclusion I'm happy with how my project turned out as well as that I learned many different types of modeling with UML, that will be very helpful when designing systems in the future and as I go further down my career path.