ANALYSIS SPECIFICATION

HOtel Management System



11/06/2012

Johann Lutterodt

Yordan Lyubenov

Ermir Ismaili

# Contents

[1 Contents 2](#_Toc339982928)

[2 Introduction 3](#_Toc339982929)

[3 Analysis model 4](#_Toc339982930)

[Composite Class Diagram 4](#_Toc339982931)

[A. Use Case: Check-Out 5](#_Toc339982932)

[Class diagram 5](#_Toc339982933)

[Sequence Diagram 5](#_Toc339982934)

[Statechart Diagram –Payment 8](#_Toc339982935)

[B. Use Case: Login 9](#_Toc339982936)

[Class Diagram 9](#_Toc339982937)

[C. Use Case: Preparing Rooms (scheduling the tasks) 10](#_Toc339982938)

[Class Diagram 10](#_Toc339982939)

[D. Use Case: Checking for Tasks 11](#_Toc339982940)

[Class Diagram 11](#_Toc339982941)

[Sequence diagram 11](#_Toc339982942)

[Statechart diagram - Employees 12](#_Toc339982943)

[E. Use Case: Checkin guest 14](#_Toc339982944)

[Class Diagram 14](#_Toc339982945)

[Sequence diagram 14](#_Toc339982946)

[Statechart diagram – guest 16](#_Toc339982947)

[F. Use case: Booking 17](#_Toc339982948)

[Class diagram 17](#_Toc339982949)

[4 Miscellaneous 18](#_Toc339982950)

# Introduction

The purpose of this deliverable is to analyze each use case and construct a class diagram for each. Also, there are three functional requirements corresponding to a sequential diagram and three objects that correspond to a particular state chart diagram.

All the work is summed up into a composite class diagram that shows the interaction between classes.

There no changes or refinements made to the use cases presented in the last deliverable.

# Analysis model

## Composite Class Diagram

The composite class diagram shows how the several classes from the use case grouping interact as a whole. It mainly shows the entity objects with their respective methods and attributes.

NB. Please find attached, the figure corresponding to the composite class diagram. (fig. extra)

## Use Case: Check-Out

*By Johann Lutterodt*

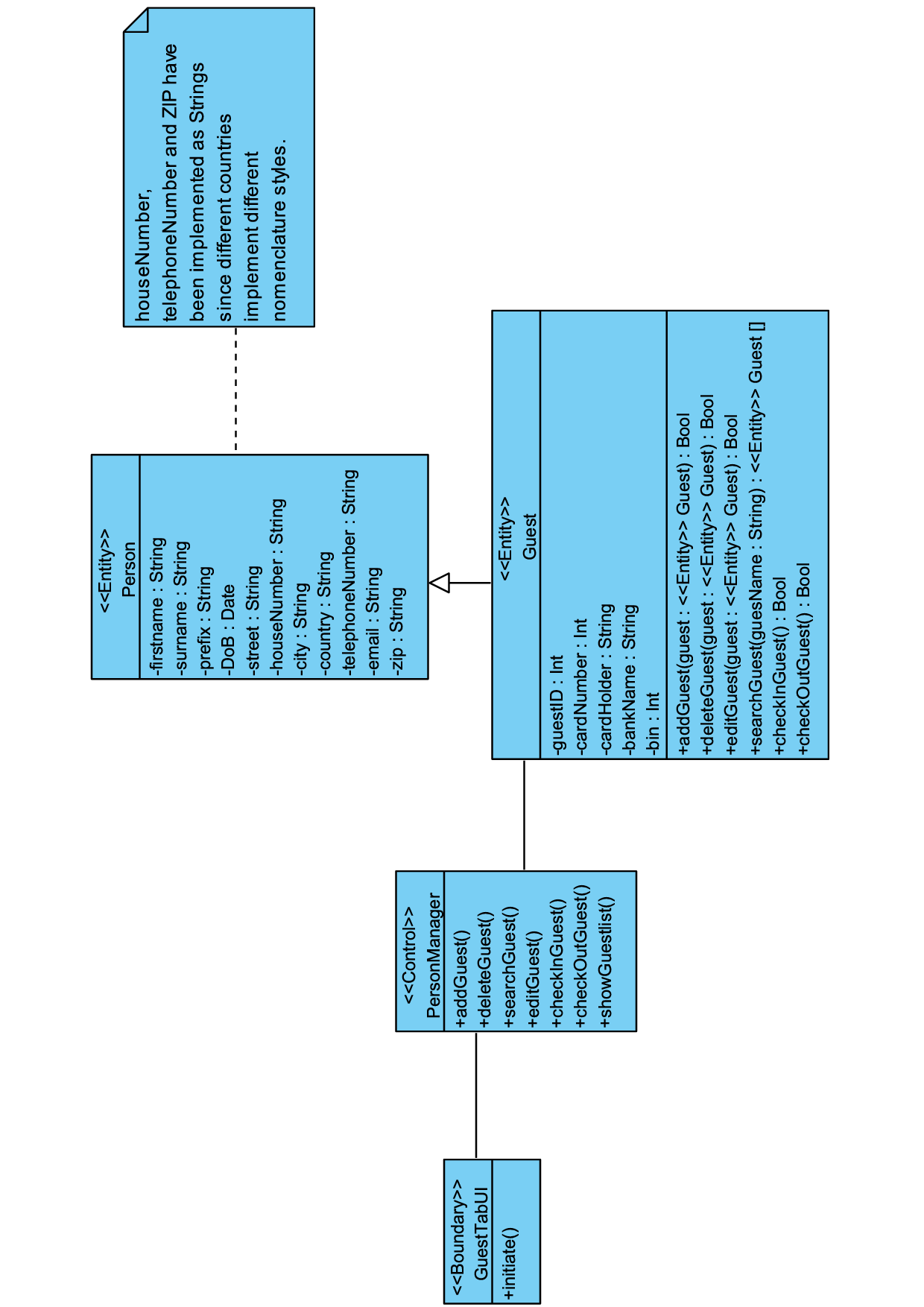
## Class diagram

The following objects were used in the creation of the class diagram.

Boundary Objects: - **GuestTabUI**

Control Objects: - **PersonManager**

Entity Objects: - **Guest, Person**



The class diagram below illustrates the interaction and associations between the objects stated earlier on. Notes added to the diagram are to clarify any unclearness.

Figure 1- Class diagram for Check out UC

## Sequence Diagram

In the following section, the “check out” scenario is going to be described; the later sequence diagram helps visualize the scenario in detail. The description explains the features and bridges gap between the diagram and the scenario. This diagram is part of the analysis and is thus not constrained to special classes or design decisions and has been consciously left relatively general, hence specific methods, concrete data types and return values have been omitted.

The check-in and check-out of guests is part of the management of the hotel and plays a major role in the daily use of the software. The check-out procedure can only be performed when guests are already checked in and is hence a prerequisite. The hotel management software has been designed to guide the guests quickly and smoothly through the check-out procedure.

The main role of receptionist is the check-in and -out guests. This task is carried out in the main lobby, where the staff is stationed. These employees have permission to use the software for both functions.

In order to identify the user and to get the correct permissions after the software has been started; the application invokes the login dialog box and prompts the user to enter his/her credentials. After the user clicks on the Login button the credentials are then validated by comparing the username and password to the entries in the database. The user then receives a corresponding notification message on whether the login was successful or not.

The user gets to see the main window after correct validation of the credentials. To check-out a guest the user has to select the “guest”-tab. The content of the tab is then called by the application automatically.

From the guest-tab, the guest to be checked-out is selected. After selection, the button to check out the guest is pressed. It is then verified whether the guest has been checked-in. When guests are not checked-in, the user receives a message that the guest cannot be checked out because the guest has not been checked-in. If the Guest is marked as checked-in, the check-out process can proceed. If the departure date is in the future, the user receives a warning. Once this warning has been taken by the user to acknowledgement the guest can be checked out. This warning should prevent the accidental checking-out of guests. If the departure date lies in the past, a message is displayed to notify the user that the guest has overstayed. The respective penalty charges are added to the bill. Is the date of departure on the same day, the user must confirm the check-out. Upon confirmation, the guest is removed from the room assignment and receives an invoice, if this has not been the case already. The user will receive a message to confirm that the check-out procedure was successful and that the guest has made the payment.

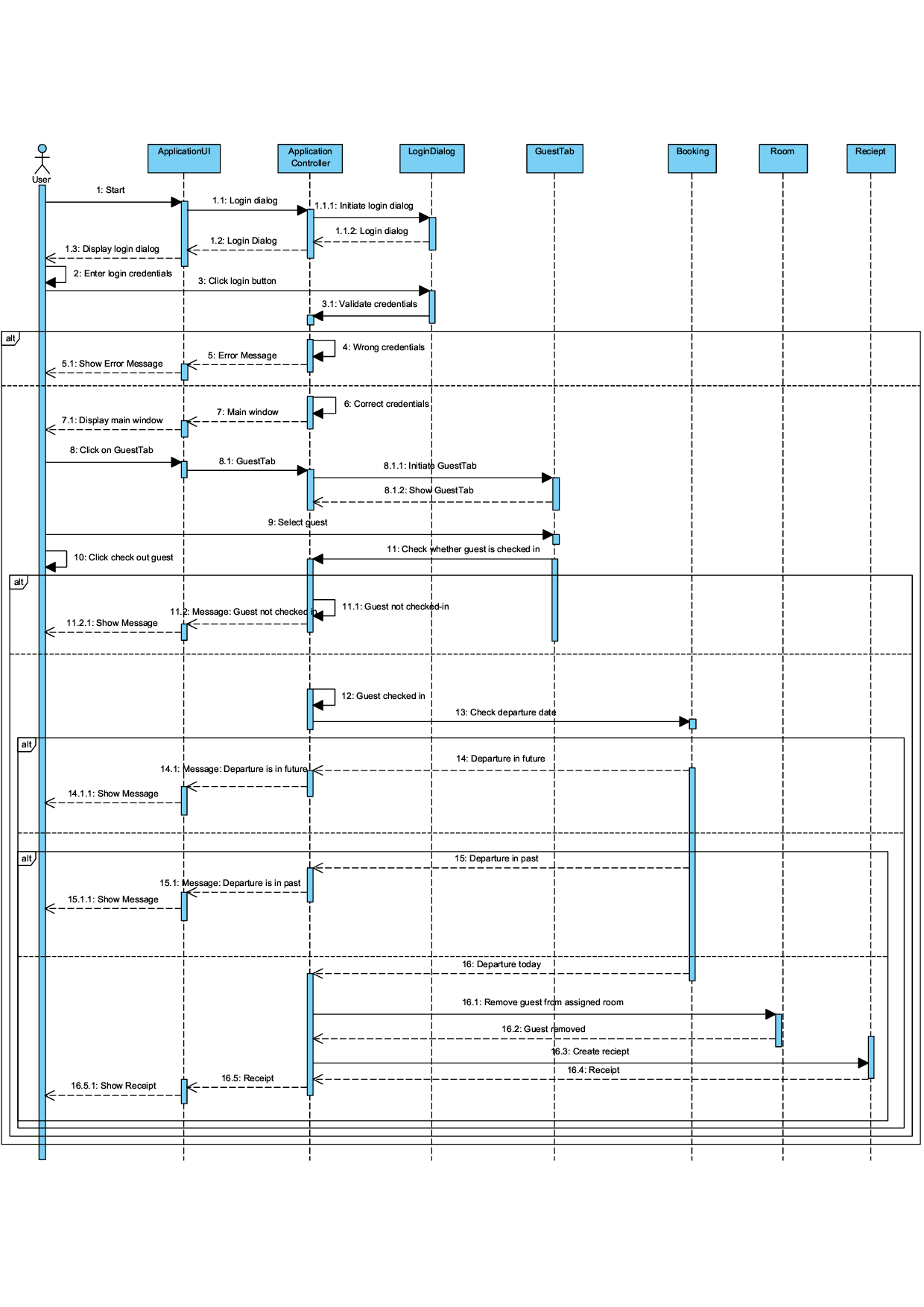


Figure 2- Sequential Diagram for Check out UC

## Statechart Diagram –Payment

The state diagram shows the several states of system operation and the events needed to trigger these different states. (fig. 3)

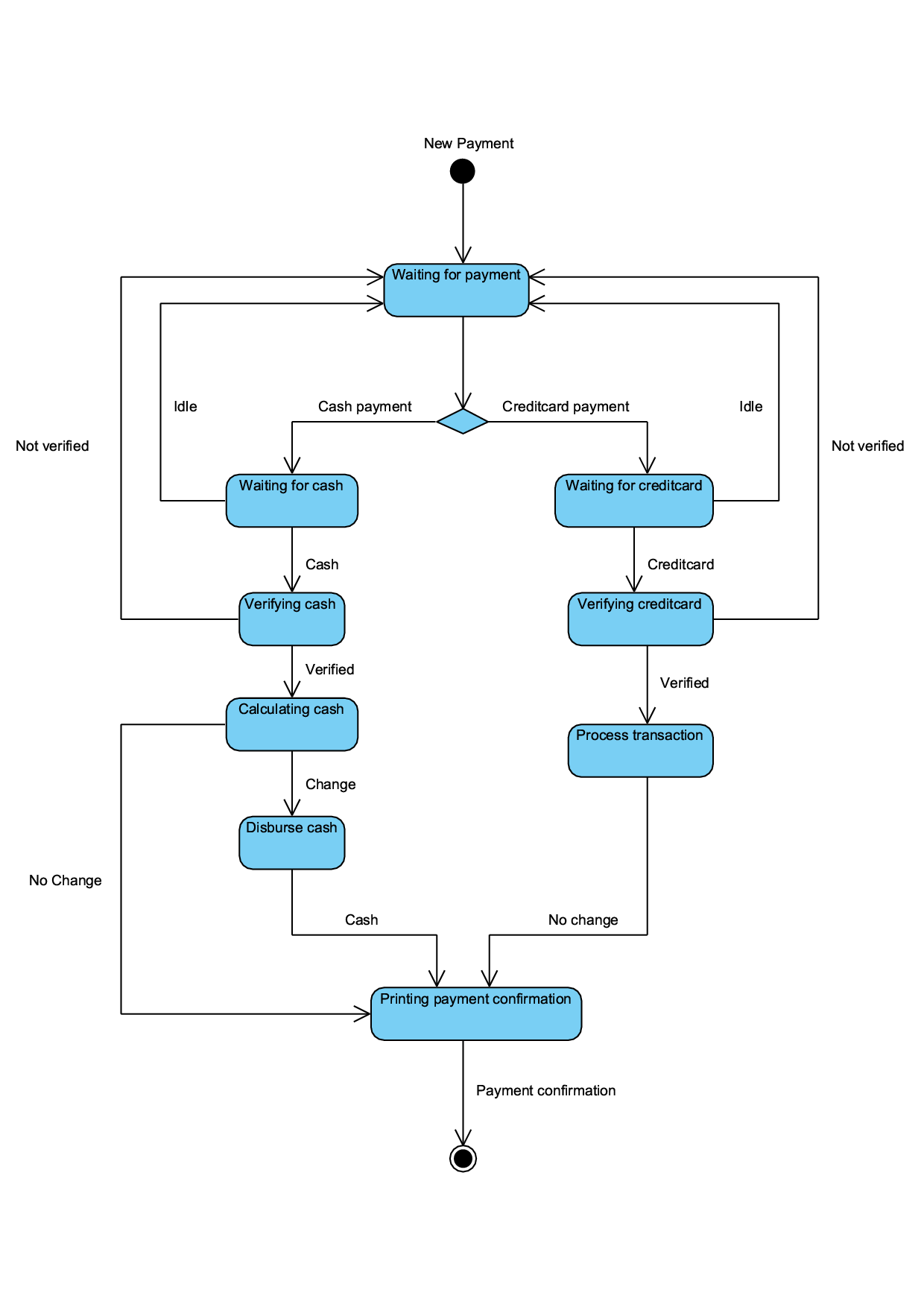


Figure 3- Make payment sequence diagram

## Use Case: Login

*By Johann Lutterodt*

## 

## Class Diagram

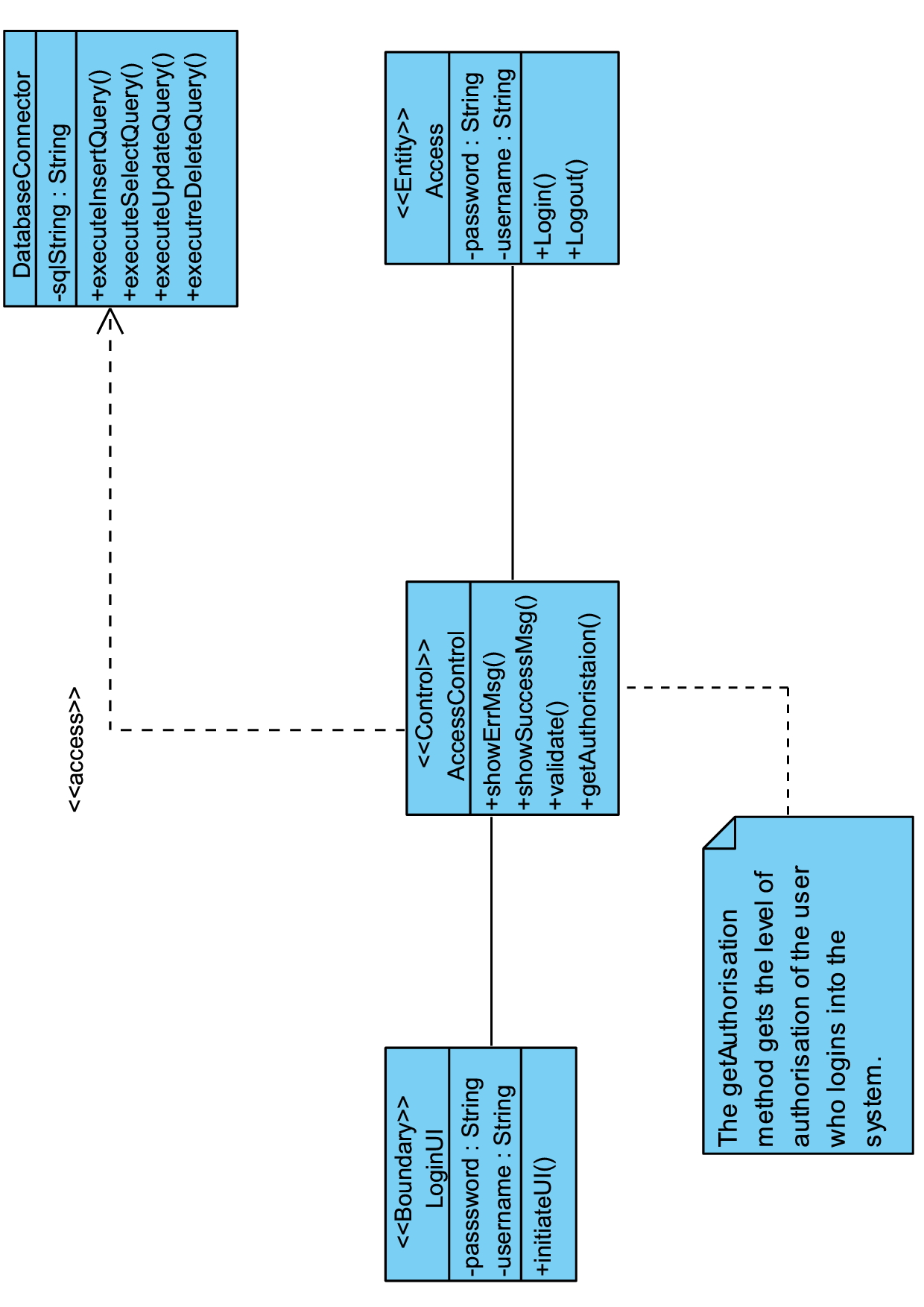
The following objects were used in the creation of the class diagram.

Boundary Objects: - **LoginUI**

Control Objects: - **AccessControl**

Entity Objects: - **Access**

Interface: - **DatabaseConnector**

****

The class diagram below illustrates the interaction and association between the objects stated earlier on.

Figure 4- Class diagram for Login UC

## Use Case: Preparing Rooms (scheduling the tasks)

*By Yordan Lyubenov*

## Class Diagram

The following objects were used in the creation of the class diagram.

Boundary Object – **managerUI**

Control Object – **receiptScheduler**

Entity Objects – **Staff, Receipt**

Enumeration was needed for the different level of the employees

The idea behind it is that when someone logs in, the enumeration will consider the possible tasks depending on the level of the employee authorization. After that, if one can change anything like adding a cleaning task to a current customer, this will be added to the total receipt.

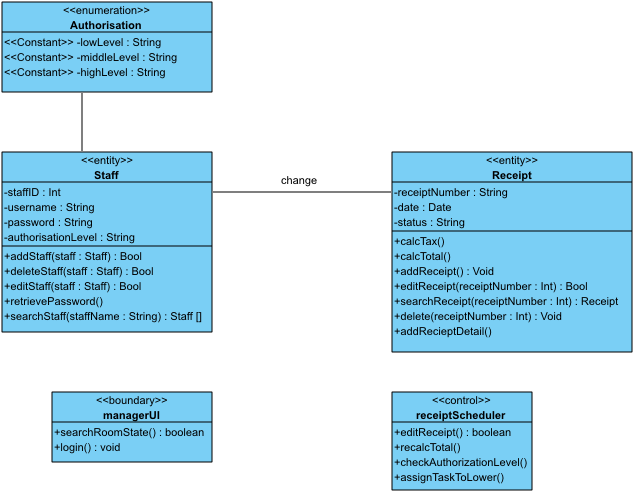


Figure 5 – Class diagram for checking for preparing rooms UC

## Use Case: Checking for Tasks

*By Yordan Lyubenov*

## Class Diagram

The following objects were used in the creation of the class diagram.

Boundary Object – **managerUI**

Control Object – **taskScheduler**

Entity Objects – **Staff, Room**

Enumeration for room was necessary so we can add the cleaning and mini bar jobs. When something is done, the controller will change that.

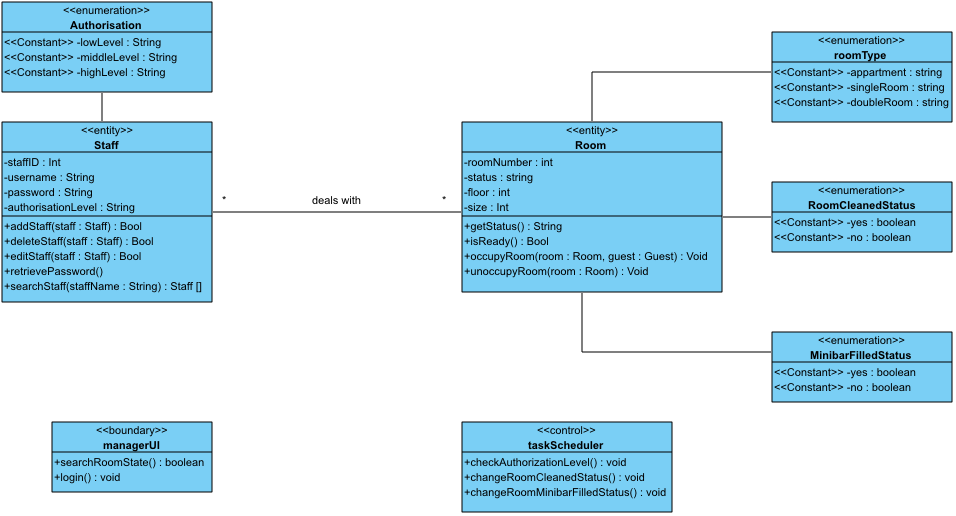


Figure 6 – Class diagram for Checking for tasks UC

## Sequence diagram

What this sequence diagram shows is how the low level employee will proceed when entering the system through the application UI. Login dialog will pop up and one has to input credentials there. The application controller invokes it, and it is also concerned with the verification of the credentials. After that, the employee will check via manager UI that will pop up afterwards. It will have different view in different levels of authorization. For the low level employee, he will only be able to deal with tasks that were announced to be done. He will check if there are room statuses that are unfinished. In other words, he/she will check if there are tasks to be done. If not, he can exit the system. If yes, he/she will mark them and then proceed with doing them. After that he/she will exit the system.

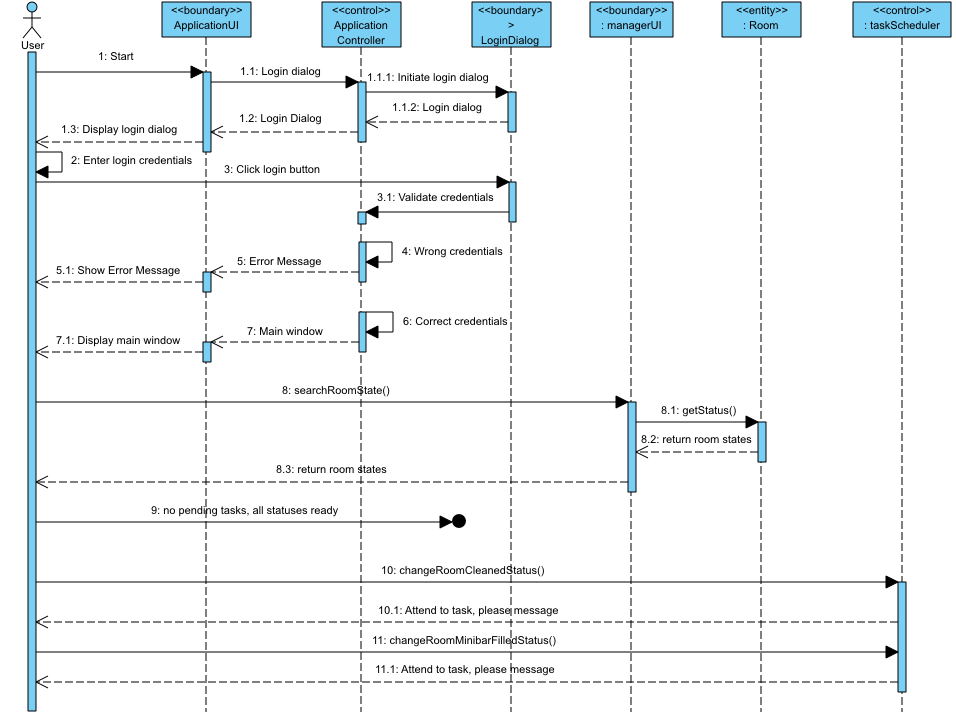


Figure 7- Sequence diagram for Checking for tasks UC

## Statechart diagram - Employees

Every employee level has own responsibilities, although higher ones can do lower ones’ jobs as well. The figure 8, representing the state chart diagram is rotated for a better display.

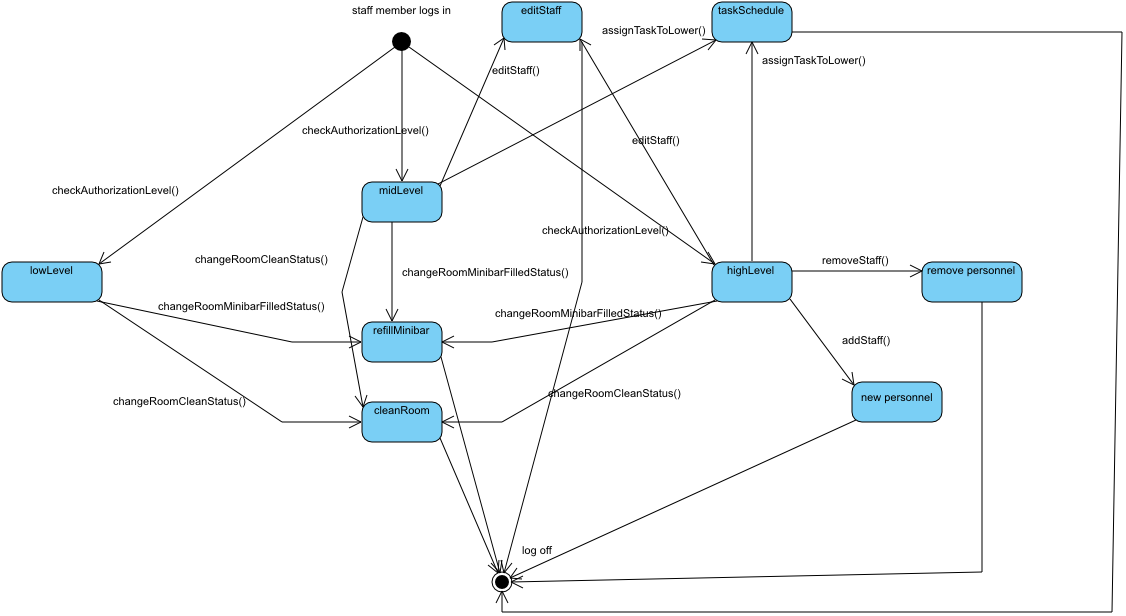


Figure 8- State chart diagram for Employees object

## Use Case: Checkin guest

*By Ermir Ismaili*

## Class Diagram

The following objects were used in the creation of the class diagram.

Boundary Objects – **GuestTabUI**

Control Objects – **Reservation**

Entity Objects – **Guest, Person, Booking, Room**

Enumaration objects **– roomType, miniBarFilledStatus, RoomCleaned Status**

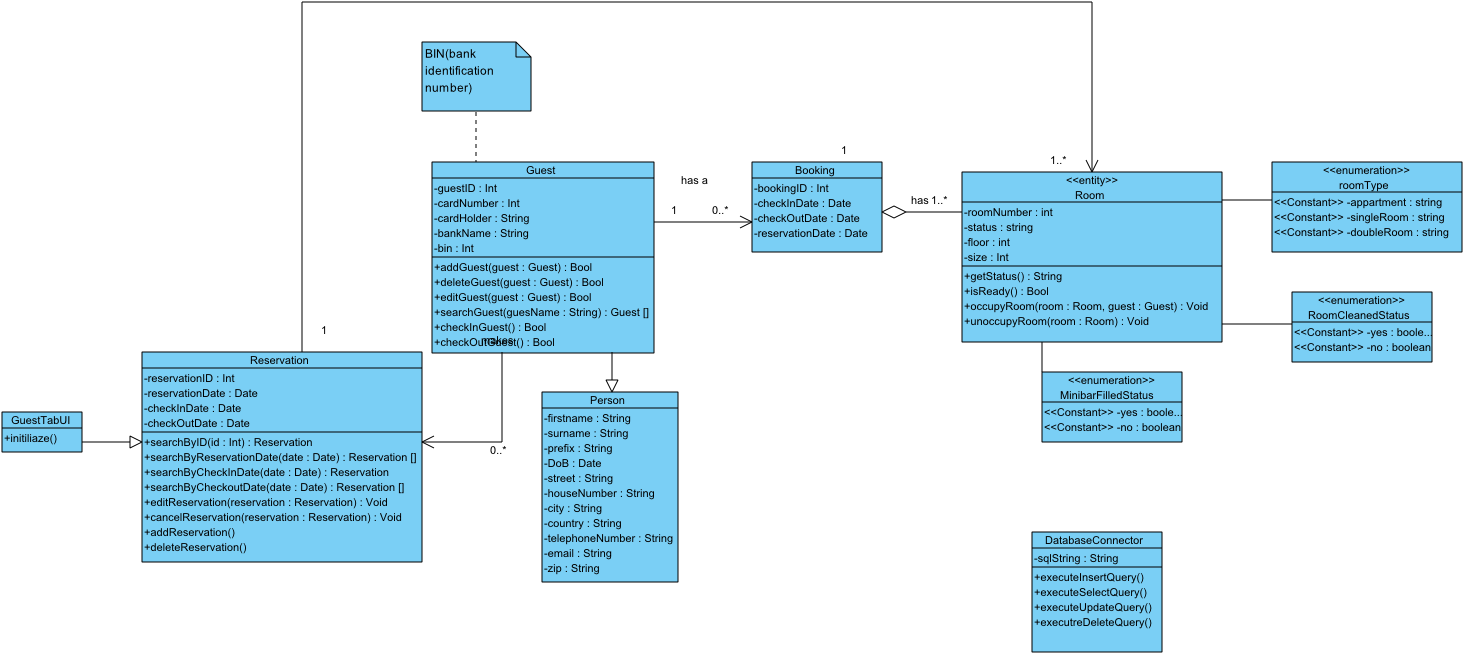


Figure 9- Class diagram for Checkin guest use case

## Sequence diagram

The following diagram illustrates the sequence of events performed by the Receptionist to check in a guest. This diagram is divided into two cases: First, if the guest at the reception has a reservation or second, it is a walk-in guest. This is depicted in the alt. combined fragments in figure 11. For the first case: when the receptionist types the reservation code, the input data is compared to the database the database. If it is not found, the guest is acknowledged. If it is found, the application retrieves guest information and displays it. If the information is correct, the receptionist confirms it otherwise updates it. Then the room status is asked. If it is not ready, the receptionist acknowledges the guest by telling them to wait. Otherwise, if it is ready, the receptionist starts guest’s bill and give them keys.

In the second case: If there is a walk-in guest, the receptionist starts a booking session, searches for a room as specified by the guest. If none is found, the guest is acknowledged. Otherwise, the receptionist occupies the room and room status is changed. Than the guest profile with all the information is created. The guest ID passed to his/her reservation and success message is prompted. Then a bill is created.

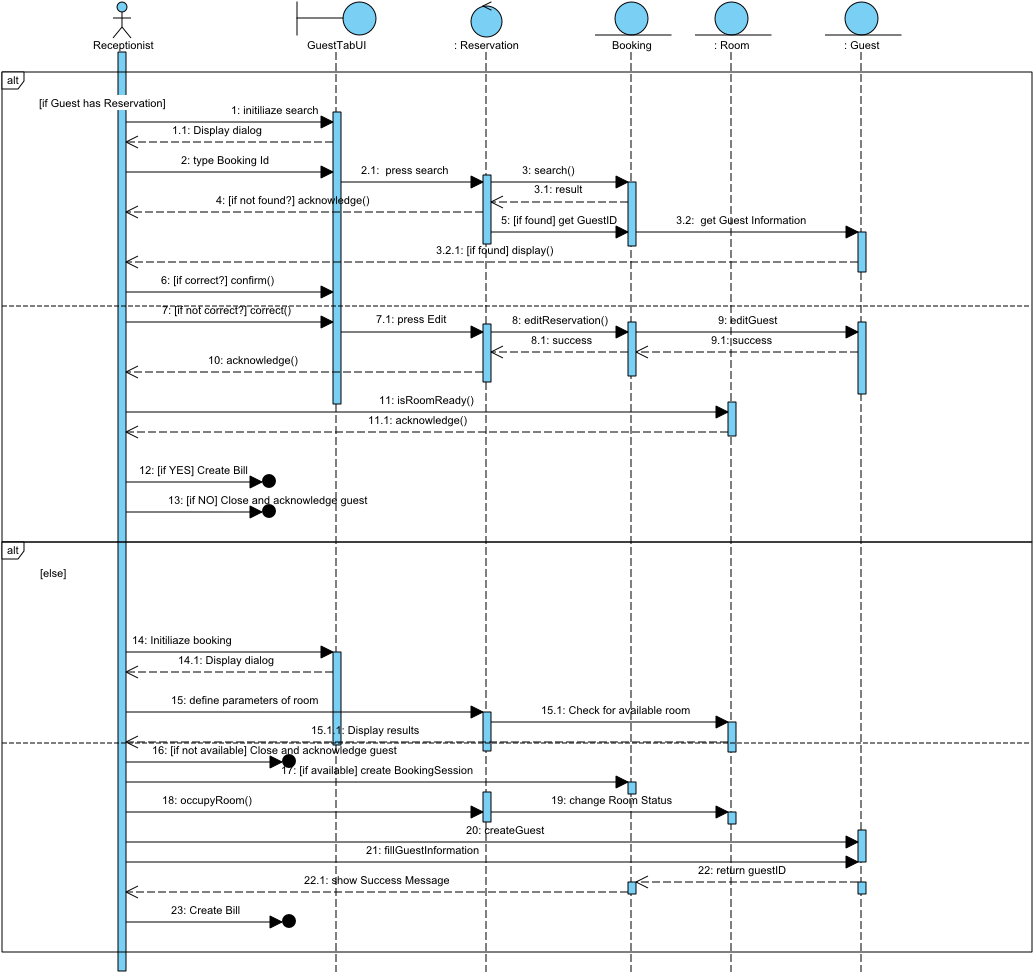


Figure 10 – Sequence diagram for checkin guest UC

## Statechart diagram – guest

The following diagram describes the states of the Guest as an object during an event for instance check-in.

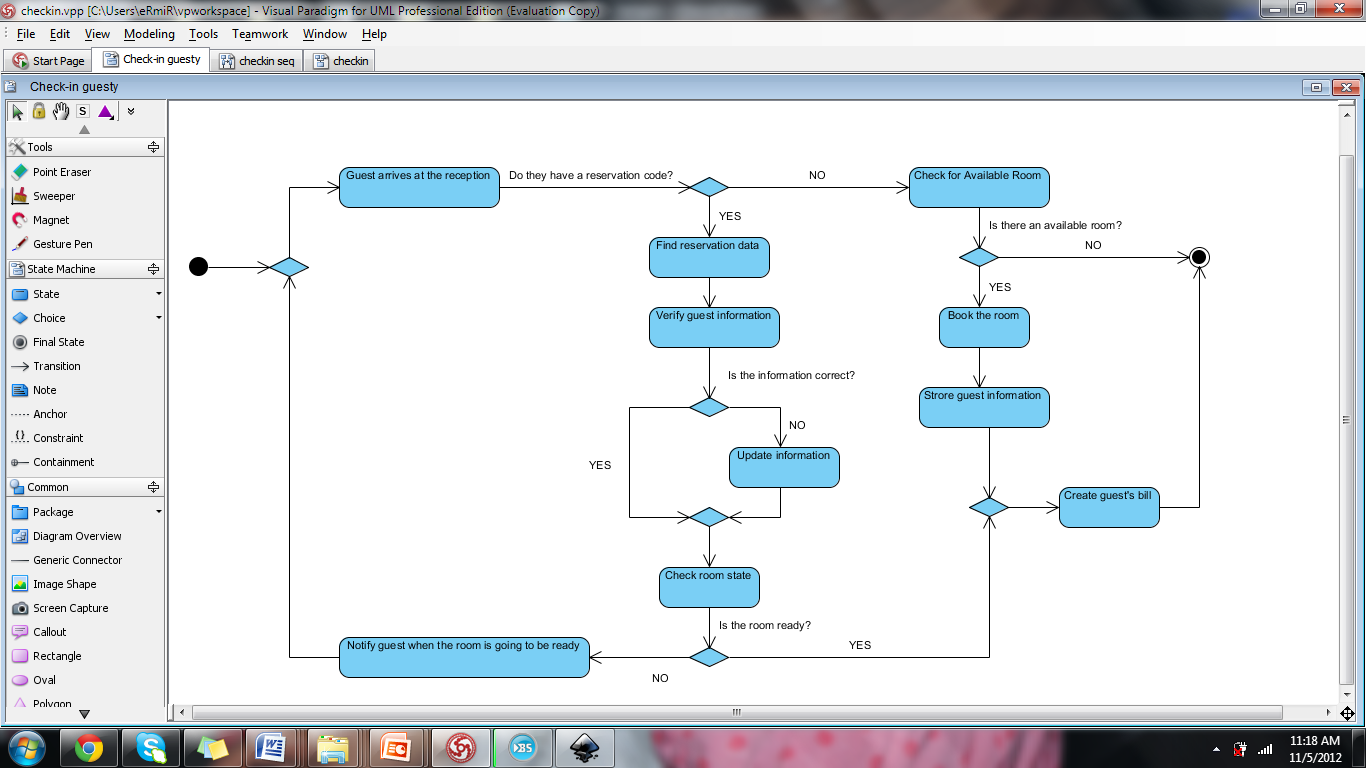


Figure 11- State chart diagram for Guest object

## Use case: Booking

*By Ermir Ismaili*

## Class diagram

The following objects were used in the creation of the class diagram.

Boundary Objects – **webGuestUI**

Control Objects – **BookingManager**

Entity Objects – **Guest, Person, Booking, Room**

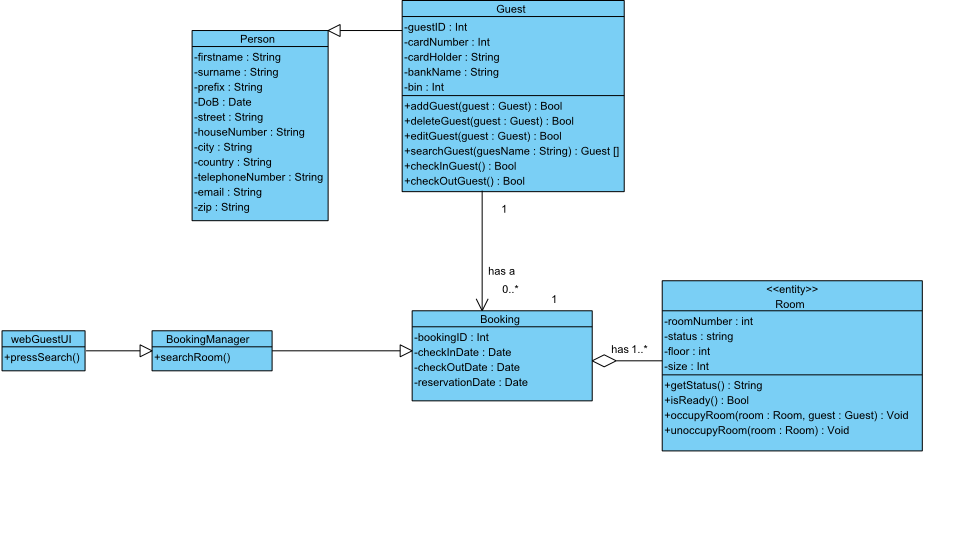
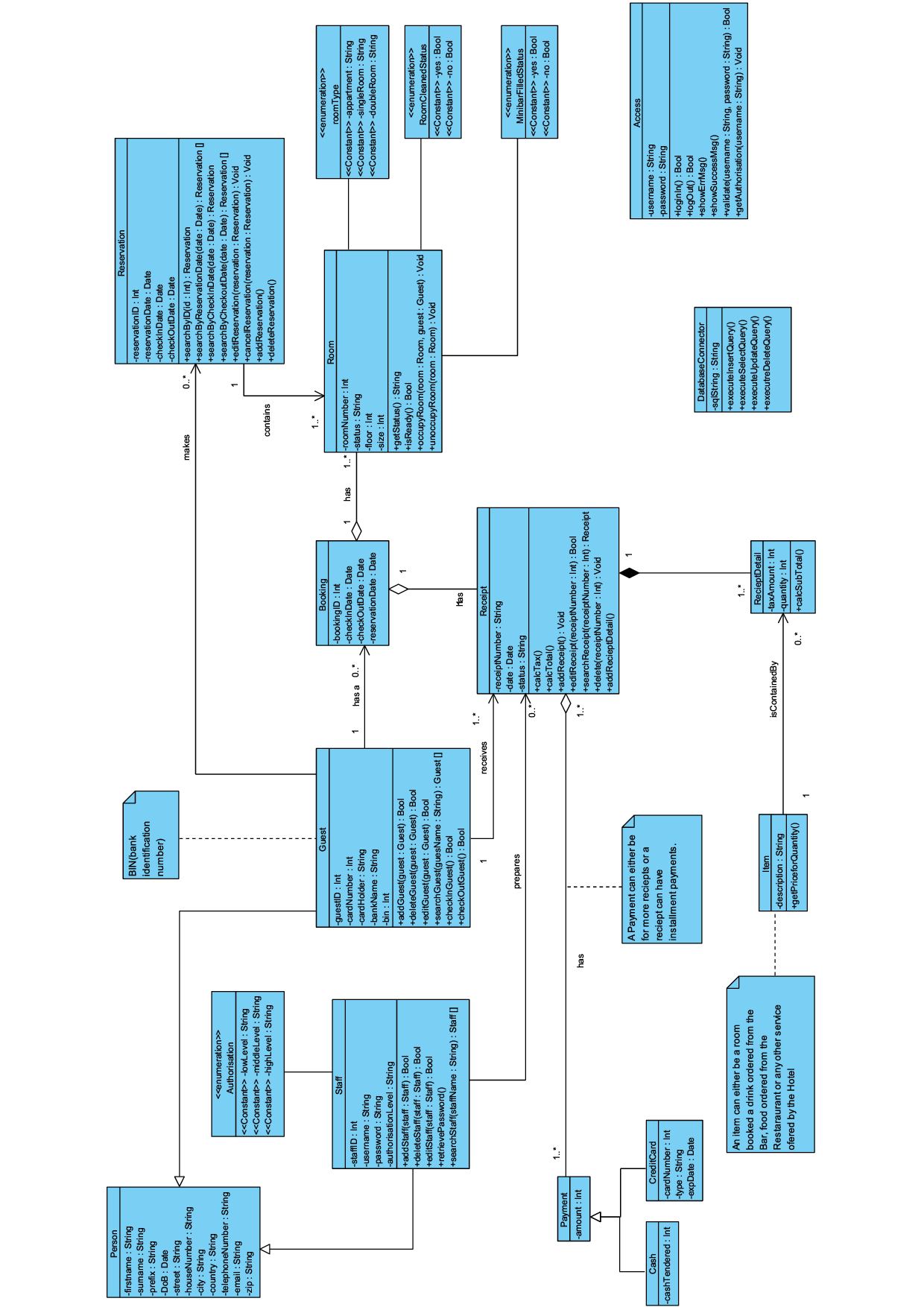


Figure 12 – Class diagram for Online Booking UC

# Miscellaneous

For a Database Access the interface Database Connector is provided, which is integrated in the class diagram. This serves as a "placeholder" for a class which implements this interface. This interface provides all the functions required to accomplish a database access:



It must not be ensured that data of the database cannot be accessed simultaneously, i.e. there is no locking mechanism required.