Enera Hotel Management System Requirements Specification

Business Informatics

CEN302 – Software Engineering Course

Project Report

Prepared by:

Jona Cara

Halisa Quku

Fiori Fizi

Shejla Domnori

Valentina Roçi

Jorid Spaha

Supervised by:

Dr. Igli Hakrama

Epoka University

Department of Computer Engineering

Table of Contents

1. EXECUTIVE SUMMARY 4

1.1 PROJECT OVERVIEW 4
1.2 PURPOSE AND SCOPE OF THIS SPECIFICATION 5

2. PRODUCT/SERVICE DESCRIPTION 6

2.1 PRODUCT CONTEXT 6
2.2 USER CHARACTERISTICS 7
2.3 ASSUMPTIONS 8
2.4 CONSTRAINTS 8
2.5 DEPENDENCIES 8

3. REQUIREMENTS 9

3.1 FUNCTIONAL REQUIREMENTS 9
3.2 NON-FUNCTIONAL REQUIREMENTS 14
3.2.1 Product Requirements 14
3.2.1.1 User Interface Requirements 14
3.2.1.2 Usability Requirements 14
3.2.1.3 Learnability 14
3.2.1.4 Accessibility 15
3.2.1.5 Performance 15
3.2.1.6 Capacity 15
3.2.2 Organizational Requirements 16
3.2.2.1 Hotel Organizational Chart 16
3.2.2.2 Availability 17
3.2.2.3 Latency 17
3.2.2.4 Monitoring 17
3.2.2.5 Maintenance 17
3.2.2.6 Operations 18
3.2.2.7 Portability 18
3.2.3 External Requirements 18
3.2.3.1 System Interface/Integration 18
3.2.3.1.1 Network and Hardware Interfaces 18
3.2.3.2 Security 19
3.2.3.2.1 Protection 19
3.2.3.2.2 Authorization and Authentication 20
3.2.3.3 Data Management 20
3.2.3.4 Standards Compliance 20
3.2.4 Domain Requirements 20

4. SOFTWARE ANALYSIS AND DESIGN 21

4.1 USER SCENARIOS 21
4.1.1 User Scenarios Extended 24
4.2 USE CASES 35

5. BEHAVIORAL DIAGRAMS 45

5.1.1 Use Cases 45
5.1.1.1 Use Case – Basic Operations 45
5.1.1.2 Use Case – Admin 46
5.1.1.3 Use Case – Receptionist & Guest 47
5.1.1.4 Use Case – Cleaner 48
5.1.1.5 Use Case – General (All Users) 49
5.1.2 Activity Diagrams 50
5.1.3 State Diagrams 56

5.1.3.1 State Diagram – Admin 56
5.1.3.2 State Diagram – Receptionist 57
5.1.3.3 State Diagram – Customer 58
5.1.3.4 State Diagram – Cleaner 59
5.1.4 Interaction Diagram 60
5.1.4.1 Sequence Diagram 60
5.1.4.1.1 Sequence Diagram – Authentication 60
5.1.4.1.2 Sequence Diagram – Cleaning Process 61
5.1.4.1.3 Sequence Diagram – Booking Process 62
5.1.4.1.4 Sequence Diagram – Hotel Management 63
5.1.4.2 Collaboration Diagram 64
Bill-Customer-System 64
Cleaner Room Report 64
Dashboard – Admin – Room – Product – Staff 64
Guest – Message – Receptionist – Room 65

6. DFD & ERD 66

6.1.1 Data Flow Diagram (DFD) 66
6.1.1.1 DFD – Level 0 66
6.1.1.2 DFD – Level 1 67
6.1.1.3 DFD – Level 2 68
6.1.2 Entity Relation 69
6.1.2.1 ER - Diagram with Attributes 69
6.1.2.2 Database Schema Design 70

7. STRUCTURAL DIAGRAMS 72

7.1.1 Class Diagram 72
7.1.2 CRUD Class Diagram 73
7.1.3 Object Diagram 74
7.1.4 Component Diagram 74
7.1.5 Deployment Diagram 76

8. IMPLEMENTATION 77

8.1 For Programmers 77
8.2 For normal Users 80

9. PROJECT PLANNING 81

9.1 Task distribution 81
9.2 Network diagram 82
9.3 Grant chart 83

APPENDIX 84

Appendix A 84
Appendix B 85
Screenshots & User Guide 85
Appendix C 102
References 102
Appendix D – Code 103

1. Executive Summary

1.1 Project Overview

Hotels nowadays are focusing on maximizing their revenues as all other businesses do and the main reason for that is the ever-increasing competition. The online world has made it difficult for hotels and resorts to compete by providing guests a plenty of choices including Homestay, Hostel with bunk beds or even a modular option. Therefore there is an increasing demand for best hotel management system in order to facilitate the management of hotel operations and functions.

Operating a successful hotel business today is a challenge in itself. A hotelier has to manage various of proposals such as operations, staff and maintenance, meanwhile keeping costs under control and balances as it is one of the most important and critical issues for a hotel business to increase their revenues and to compete with other hotels. To improve the efficiency of this process, a good hotel management system which uses the modernizing techniques must be provided.

The key to reaping the benefits of an effective hotel management software system is to select the right one for your property. It's critical that you know exactly what this hotel management technology is, and why it is important for you to implement it at your hotel. These days every person can find different options of the hotel reservation software free on the internet, however one has to judge the solution with the quantum of features and quality that it is providing. There should be a complete functionality as a hotel management system can be both basic and advanced based on the pricing options that are available as well. Also we can say that the developers are making such software as per the pocket of the business and this is one of the main reasons why we have so many different options in the online world.

Hotel management is a key element for a very important branch of economy, which is tourism. Knowing this, two members of our group were familiar with different types of management software and they had analyzed the deficiencies that they have and decided to make this project based on the improvement of these deficiencies.

Our software aims to have all the features that a hotel needs in order to adapt to the management structure of the business, and by making practical and effective use of these features every hotelier's work life will be much easier than they have ever imagined by using this product.

1.2 Purpose and Scope of this Specification

This Software Project which will be a *Hotel Management System*, is going to be adapted to the management structure of *Hotel Enera* and will operate the major functionalities of the hotel.

Our Software Application will focus on automating functions such as guests booking, inquires, storing records of the guests, pre-booking and booking functions, managing inventories across the sales channel and keeping them always in sync so that a real-time update inventory is subjected to the online world, having a quick control of the rates across the channels and a easy to update control. The most important goal is to run the hotel properly while giving the current importance to each aspect of the business.

The System should also ensure that accurate and precise information is consistently shared with the guest about the inclusions and amenities to him/her so there will be no room for misunderstanding by providing a complete transparency and this should be as much as possible human independent. Happy guests pay long term dividends.

Apart from all these functionalities our Software will also ensure that the front office manager will not fumble with his day to day operations, his operations will be automated to the maximum so that there will be no possibility for human errors. The software will also have a feature of room allocation and monitoring so that the room allocation will be done right from the time the booking is picked up and occupied into the system, but at the same time there will be a feature which will let these room allocations to be interchanged if required by giving an option of interchanging or updating the category and room.

Overall, in order to lower the costs and to grow the productivity of the business we see that businesses are oriented mostly on the usage of software. For this purpose we structured a software which aims to accomplish exactly this.

2. Product/Service Description

Hotels nowadays differ in size, culture and management structure. So, the perfect Software provided, needs to be adapted to specific business which will implement and use it.

Easy Hotel Management System is a web application which aims to facilitate the management system of a *Hotel Enera*. It will keep track of hotel reservations, inventory management, rooms to be cleaned and so much more.

Moreover, a mobile application version is added in order to provide better, easier and faster communication. Mobile version will let the user keep the Hotel Management system in its pocket. It is designed to be near to the guests anywhere and anytime.

2.1 Product Context

Easy Hotel Management System (EHMS) software is technology that allows hotel operators and owners to streamline their administrative tasks while also increasing their bookings in both the short- and long-term.

EHMS is not only important for day-to-day operations, but it's a vital part of the overall guest experience. From the beginning of guests' online booking journey until the completion of their stay and their feedback once they return home, it is necessary for the hotel management technology to enhance customer experience with the brand.

This product aims to bring together all possible stakeholders of Jona's Hotel, giving flexibility and facilitating the management process. The product main focus is to be robust in operation and user friendly in usage.

1. Platforms are available: web application and mobile application which is compatible with android operating system and further will be extended with the iOS.
2. Sign in option:
   - username & password
   - Gmail account

2.2 User Characteristics

Web Application:

The web version of the Software has multiple users with different functionalities such as:

1. Admin - the owner or manager of the business running this product
    - Is able to see booked rooms, can adjust room price, modify reservations
    - Is able to Add/Remove rooms or possible users of the system
    - Is able to observe Statistics
    - Inventory

2. Receptionist - the person who deals with guests checking-in & checking out of the hotel, answers calls
    - Is able to view Rooms to be cleaned and can print the receipt
    - Rooms available to customers
    - check-in , check-out dates
    - Make and view reservations
    - Have access to inventory & balance sheets

3. Guest - any user who is interested to the hotel and tend to make reservation
    - Can Make reservation, check rates
    - While is in its stay can view if the room is ready by the cleaner or not
    - Can modify his booking and edit his credentials

4. Cleaner - employer/s who take care of the cleaning part of the interior of the building.
    - Can see the rooms to be cleaned
    - Can make changes to the status of the room regarding cleaning services. (E.g. Room ready or not)

2.3 Assumptions

We assume:

1. All users have basic knowledge in English language;
2. All users have basic knowledge in computer and smartphone usage;
3. Stakeholders of the hotel have basic knowledge on how to use the system due to previous experiences with other systems;
4. Hotel is equipped with PC/Laptop/Tablet, printer, mobile phone;
5. Hotel must have internet connection all the time;
6. It is assumed that the Hotel provides Credit Card payments.

2.4 Constraints

1. All users have to be logged in in order to use the product and to access the information
2. Phone memory of the users is another problem, prices and quality of android phones varies and so the memory, most of the workers are likely to have cheaper phones and that can be a problem for the application usage

2.5 Dependencies

- As it is a web based application it is always dependent on internet connection. A connection is required to send commands and receive answers, usually in the form of a result set.

3. Requirements

The following definitions are intended as a guideline to prioritize requirements.

- Priority 1 – The requirement is a “must have” as outlined by policy/law
- Priority 2 – The requirement is needed for improved processing, and the fulfillment of the requirement will create immediate benefits
- Priority 3 – The requirement is a “nice to have” which may include new functionality

It may be helpful to phrase the requirement in terms of its priority, e.g., "The value of the employee status sent to DIS must be either A or I" or "It would be nice if the application warned the user that the expiration date was 3 business days away". Another approach would be to group requirements by priority category.

3.1 Functional Requirements

Req#	Requirement	Comments	Priority	Date Rvwd	SME Reviewed / Approved
BR_01	Different views for account types	There will be 4 account types: Admin, receptionist, client and cleaner each having its own view	1	30/04/2019	Shejla Domnori
BR_02	Login constraint	All users have to be logged in in order to access the data	1	02/05/2019	Jona Cara
BR_03	The system should provide the possibility to view booked rooms	Administrator, and receptionist can check if rooms are booked or not	1	04/05/2019	Halisa Quku
BR_04	Add, remove rooms	Only admin can add or remove rooms	1	05/05/2019	Fiori Fizi
BR_05	The system should provide to admin possibility to adjust room prices	It will allow admin to change rates according to the season and its peaks	1	05/05/2019	Jona Cara

BR_06	The system should provide to admin the possibility to add/remove employees	In case of staff shortage or an increase in the members' of the system	2	06/05/2019	Jona Cara Jorid Spaha
BR_07	The system should provide email notifications	The client will be notified with an email for each successful booking	3	08/05/2019	Jona Cara
BR_08	Provide statistics	Based on statistics administrator can declare the top employee	3	09/05/2019	Valentina Roci
BR_09	Block dates for specific rooms	Some rooms may need to be renovated thus guests should not be able to book those rooms	1	09/05/2019	Jona Cara
BR_10	Receptionist should be able to book rooms	The reservation may come from a phone call or the client can come directly to the hotel so the receptionist will be the one to book the requested room.	2	10/05/2019	Jona Cara
BR_11	Print invoices	Administrator can print daily, monthly, yearly invoices	2	15/05/2019	Jona Cara
BR_12	Manage inventory	Only administrator can manage the inventory	1	16/05/2019	Fiori Fizi
BR_13	View rooms availability	Receptionist can see if rooms are available or not	1	18/05/2019	Halisa Quku
BR_14	View rooms available to customers	Receptionist and client can check if the room Is	1	18/05/2019	Jona Cara

		available for a new customer			
BR_15	Make reservations	After being logged in the client can make a reservation for rooms	1	19/05/2019	Shejla Domnori
BR_16	Check rates	The client can check the rates and reviews of the hotel	2	19/05/2019	Jorid Spaha
BR_17	Check rooms to be cleaned	Cleaner can check room to be cleaned	2	20/05/2019	Jona Cara
BR_18	Check room's status	Cleaner can check if the room is cleaned or not	2	20/05/2019	Jona Cara
BR_19	The system should assign to each cleaner the rooms to be cleaned for each day	Name of worker ="Transcript with Room numbers"	3	21/05/2019	Jona Cara
BR_20	Book Multiple Rooms	The system should provide the possibility to book more than one room from a single client account	2	21/05/2019	Jona Cara
BR_21	Guest note	The client may have extra requirements that are not listed on our software thus they can leave a note to specify their extra needs	3	21/05/2019	Fiori Fizi
BR_22	The system will provide a mobile application for the administrator	The administrator of the hotel will have a mobile application from where he/she will have full access	3	22/05/2019	Jona Cara

		of all the features that we will offer			
BR_23	The system will provide a web application for all types of users	Every user can use the web application but according to the type of user that is logged in, *there will be different interfaces provided.*	1	22/05/2019	Jona Cara
BR_24	The system should calculate VAT according to room price	According to Albanian legislation about hotels, 6% of room price goes for VAT. This will be calculated in the printed receipt.	2	23/05/2019	Shejla Domnori
BR_25	The system should provide the Admin to add product to the inventory	Admin can add specific product to the inventory so it will be easier to keep track of the hotel's product.	2	26/05/2019	Jorid Spaha
BR_26	The system should provide the Admin to assign and control tasks	Admin assigns different tasks related to the hotel issues to the staff.	3	28/05/2019	Halisa Quku
BR_27	The system should provide the admin with possibility to modify booked rooms	Admin can delete or modify the booked room according to guest needs. It can also extend the guest stay	1	31/05/2019	Jona Cara
BR_28	The system should provide users the ability to edit information	Users can edit their information and change the profile picture	2	02/06/2019	Valentina Roci

BR_29	The system should provide the admin to leave notes to employees	Admin can send a message to other staff members of the system for a specific reason	2	03/06/2019	Jona Cara
BR_30	The system should provide to the cleaner the information of the rooms to be cleaned	Cleaner can see in its page which room it is assigned to clean.	3	04/06/2019	Jona Cara

3.2 Non-Functional Requirements

3.2.1 Product Requirements

3.2.1.1 User Interface Requirements

- A simple and responsive system in a short time
- Web app, consistent in all interfacing screens or devices
- Details of any user (Client, Admin, Receptionist) will be activated in the displayed mode and in the database real quick
- Flexible navigation to and from displayed panels or pages

3.2.1.2 Usability Requirements

This Web app will have an easy interaction with every user from each device connected to an internet service also from every browser or platform. We think we can conduct also a mobile application which can be downloaded only from Androids, but we aren't sure if we will achieve a complete version because of the time available. Every user will have its own type of interface, with the attributes and actions they can perform there. Admin interface records all the information for the overall management of the hotel. The most important commands for each type of interface will be visible at first view so the software can be easily accessed. It will be designed in a way that allows modifications to be made, as it has to be updated time after time to fulfill hotel's management requirements and to manage possible error occurrences.

3.2.1.3 Learnability

- Our product is user friendly – everyone can easily learn the commands following the guidelines provided by us;
- Even though the software is in English, it can be understood by someone with basic knowledge of English since every functionality will be graphically shown;
- A .pdf file with all specifics of the product will be provided to the user of the software, where every functionality will be graphically explain.

3.2.1.4 **Accessibility**

- The software will be able to perform 24/7.
- Probability that the users will accomplish a desired task (reservation) in the first time without errors is 99, 5%.
- Different levels of users will achieve different tasks in the same time as the system guarantees to perform multiple actions per minute.
- System data are automatically saved in the database and backed up daily.

3.2.1.5 **Performance**

The web and mobile application will be an online platform with multiple users. It uses multi-tier architecture which consists of; user client, middle tier, and application server.

Web application performance is dependent on:

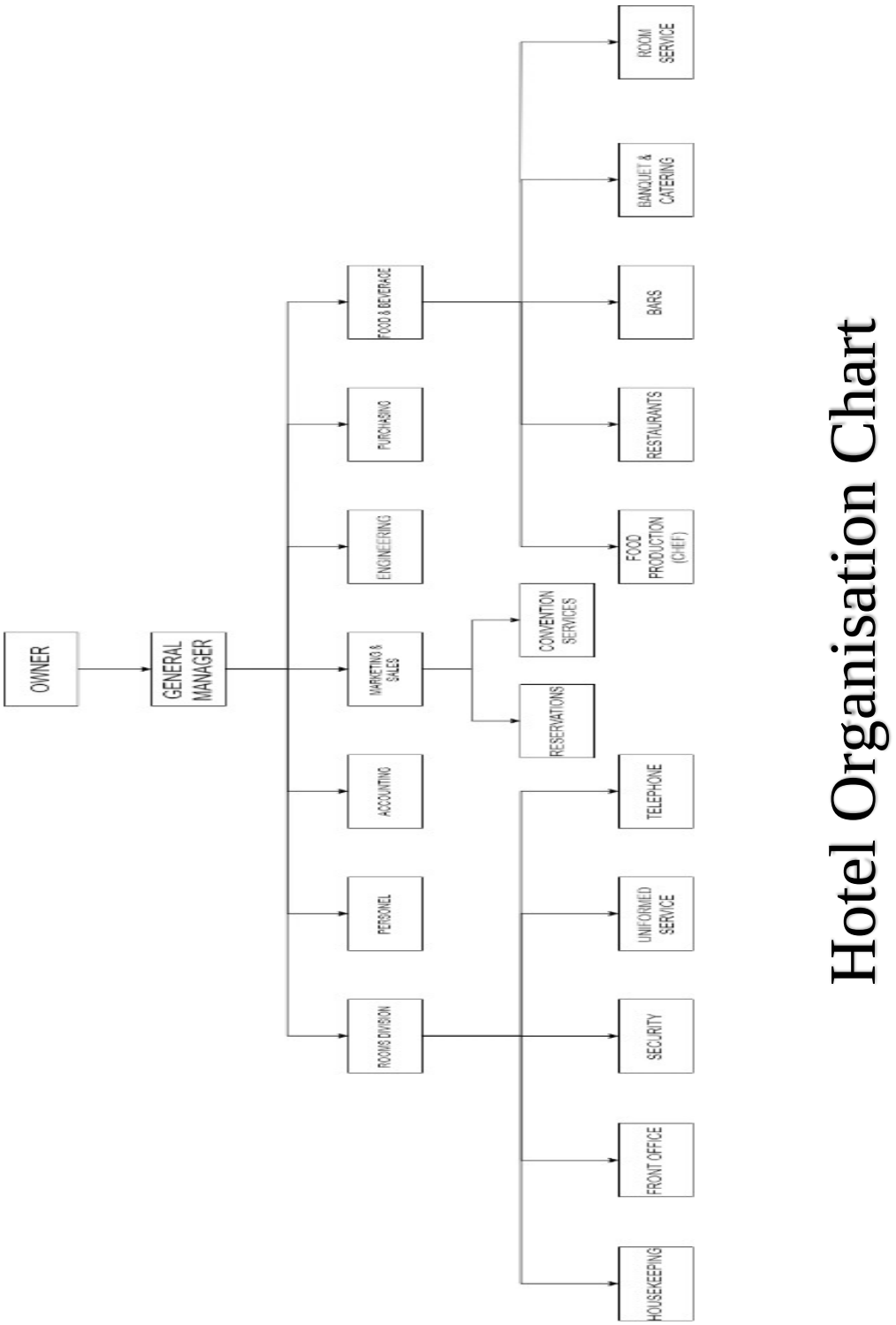
1. Written Code - in order to make the performance of the software better we have insured that developers are using the optimal code practices, as well as automated tools such as profilers to check time complexity etc.
2. Responsive time - ***Example:*** Search results for room availability should be populated within an acceptable time limits.
3. Speed of internet connection
4. Specific HTML title tags - tags sum up the entire content of the website to major search engines such as Google. However, a lack of specificity in your domain name can lower its visibility.

3.2.1.6 **Capacity**

The applications runs better and faster on stronger internet connection but it loads perfectly even with slow internet connection. Moreover, it is fast even with multiple users accessing the page at the same time. Web and mobile application are stored in a server which needs to have a minimum of 150MB of size.

3.2.2 Organizational Requirements

3.2.2.1 Hotel Organizational Chart



3.2.2.2 **Availability**

- The website will ensure to be available all the time, everyday 24/7. It will have a high availability to achieve the higher as possible the percentage of time the system is functioning.
- Even though it is in English language the system can be used wide world as it is a web application, the same for the android application
- Our product will have a downtime as minimal as possible as long as the software will be used with reliable web browsers
- While talking about the mobile application it will be available only in android phones

3.2.2.3 **Latency**

The software will be used in a strong internet connection and the website will be loaded within less than 2 seconds.

3.2.2.4 **Monitoring**

The software will be evaluated often. In case of errors the administrator will be able to follow specific validations because everything will be well documented in files.

3.2.2.5 **Maintenance**

System is maintainable and useable, it is made in the form that later on if required it can be improved by adding more functionalities. This software uses Doctrine which is focused on database storage and object mapping, Symfony for the back-end and React.js for the front-end. The system will be updated continuously with different features and extra features based on guests reviews and business requirements. Moreover, the software will be observed and maintained by the administrator of the system. In case there is any error in the system in the site will appear a message informing users to be patient while the system is being maintained.

Example: "We are performing site maintenance and will be back shortly! – Please be patient"

3.2.2.6 Operations

- The admin can log in and have access on all the software's features.
- The admin can view statistics for booked rooms, revenues, guest insight.
- CRUD functionalities for staff, rooms and reservations.
- Guest can book rooms online and modify it afterwards.
- The Guest can leave reviews after they finish their staying in the hotel.
- The receptionist can log in and access only information for the rooms and guests.
- The cleaner can log in and see which room he/she is assigned to clean.

3.2.2.7 Portability

Web application can be accessed in any platform and browser as far as they have internet connection, while the mobile application is accessible only to android users. In the near future the iOS version will be developed.

3.2.3 External Requirements

3.2.3.1 System Interface/Integration

Unauthorized parties aren't allowed to access the system.
Access in different subsystems with important recorded data will be protected by user log in with a username and an encrypted password. This will ensure different views and functions of user levels through the software. Customer Representatives will have access on Reservation and Personal Information subsystems. Receptionist will have access on the Reservation data of all customers. Only the admin user has the maximum privilege to access all the subsystems.

3.2.3.1.1 Network and Hardware Interfaces

This web application version of the Hotel Management System can run on any browser such as Internet Explorer, Firefox, Opera, Safari and Google Chrome. Clients should use latest version of these browsers to take advance all of newest features. In our working process, we use some software applications to create and design websites by Photoshop. This software uses Doctrine which is focused on database storage and object mapping, Symfony for the back-end and React.js for the front-end.
Mobile application is only supported by android user as for the moment.

3.2.3.2 **Security**

Our web application:

- Ensures that users and client applications are identified and that their identities are properly verified.
- Ensures that users and client applications can only access data and services for which they have been properly authorized.
- Detects attempted intrusions by unauthorized persons and client applications.
- Ensures that unauthorized malicious programs do not infect the application or component.
- Ensure that communications and data are not intentionally corrupted.
- Ensure that parties to interactions with the application or component cannot later repudiate those interactions.
- Ensure that confidential communications and data are kept private.
- Enable security personnel to audit the status and usage of the security mechanisms.
- Ensure that applications and centers survive attack, possibly in degraded mode.
- Ensure that centers and their components and personnel are protected against destruction, damage, theft, or surreptitious replacement.
- Ensure that system maintenance does not unintentionally disrupt the security mechanisms of the application, component, or center.

3.2.3.2.1 Protection

Symfony provides a complete security system for web development. It offers the facilities for authenticating using HTTP basic authentication, interactive form login and also allows the developers to implement their own authentication strategies. Moreover it provides ways to authorize authenticated users based on their roles.

The security component of symphony contains subcomponents which can be used separately to guarantee a secure application.

Symphony / security-core

- It provides all the common security features, from authentication to authorization and from encoding passwords to loading users.

Symphony / security-http

- It integrates the core sub-component with the HTTP protocol to handle HTTP requests and responses.

Symphony / security-csrf

- It provides protection against Cross-site Request Forgery attack.

3.2.3.2.2 Authorization and Authentication

Symfony comes in help for the Authorization and Authentication part. It uses Authentication Providers by implemecnting AuthenticationProviderInterface and it has a method supports() by which the AuthenticationProviderManager determines if it supports the given token or not. If the token is supported by the AuthenticationProviderManager, it will call the provider's method authenticate(). This method should return an authenticated token or throw an AuthenticationException (or any other exception extending it). The providers offers authentication as below.

- Authenticating Users by their Username and Password
- The Password Encoder Factory
- Creating a custom Password Encoder
- Using Password Encoders

For a successful Authorization Symfony uses the access listener and the Authorization Checker.

3.2.3.3 **Data Management**

Data management is a way of managing knowledge and information and make it accessible for specific purposes. In our case, data management involves using effectively the data and implementing the current policies of the business. It is a major task for the hotel, as the data entered on the system needs to integrate according to the demand and requirement of the guest with their preferences.

3.2.3.4 **Standards Compliance**

The software is designed to be able to endow the customer with the certain receipt for his purchase. This receipt will have added also the value of standards which are ensured by the release of our product as a liability for this project. In the receipt the customers can see that the VAT only adds 6% of the amount of the receipt, as it is the defined percentage for all the hotels.

3.2.4 Domain Requirements

Only admin can create, update and delete employees. If sign up option is clicked in the web application the user will be of type “Guest” of the hotel. Rooms to be cleaned are automatically assigned to the worker by the system, if the worker is absent in a specific day, its work will be distributed to its coworkers. The user interface will be standard for all types of users. System should take into account the exact time of check-out of the leaving guest and check-in of the new guest in order to avoid collisions between bookings. The system should also take into account that there will be different currencies for online payments.

4. Software Analysis and Design

4.1 User Scenarios

Number	User Scenario	Description
1	Admin logins into the system	Admin user insert his/her own credentials (username/email and password) to login into the system
2	Admin fails to login into the system	Admin provides wrong username/email or password thus the login will fail
3	Admin changes his/her credentials	Admin changes his/her stored information on the system (email, username, password etc.)
4	Admin logs out of the system	Admin clicks the logout button to initiate logout process. Then the home page will be displayed
5	Admin views room details of the system	Admin can check all room details that exist on the system including their number, price, type etc.
6	Admin adds new room to the system	Admin can add new rooms on the system by providing room number, price, type and other specification
7	Admin modifies room properties	Admin can modify room specification if there is such a need
8	Admin deletes room from the system	Admin can delete room from the system. He will be demanded to confirm the removal process if he/she clicks the delete room button.
9	Admin views staff/employee details	Admin can view all the details defined on the system for the registered staff/employee
10	Admin adds new staff/employee to the system	Admin can add new employee on the system. She/he will enter her/his information and give the login credentials to the new registered staff for her/his account
11	Admin modifies staff/employee credentials	Admin can modify insights for the staff. Most of the time it will be the salary that the admin chooses to change.
12	Admin deletes staff/employee from the system	Admin removes an employee from the system after she/he leaves the job
13	Admin views product details	Admin can check details of the products that exists on the system.
14	Admin adds product to the inventory	Admin can add specific product to the inventory so it will be easier to keep track of the hotel's product.
15	Admin modifies product	Admin can modify the available products.
16	Admin deletes products	Admin can delete a specific product from the system's inventory.

17	Admin views financial report	Admin views the financial report of his/her hotel for a specific time that he/she can choose
18	Admin views guest details	Admin views the details provided by the guests that has made a reservation at his/her hotel
19	Admin views rooms availability	Admin views rooms availability at a specific time for each room
20	Admin views booking details	Admin will be notified for every new booking, booking modification and booking cancellation that will be done.
21	Admin modifies booked rooms	Admin will be notified will an email or notification changes that have to be done for a reservation. This taking into the consideration that the guest and receptionist are not available to modify it.
22	Admin views guest reviews	Admin checks reviews done from the guest after their staying in the hotel and also can leave a reply to their comments if he wants to
23	Admin prints room cleanliness status	Admin can see which of the cleanser has to clean each specific room and have a printed receipt with these information
24	Admin leaves notes for the employees	Admin can send a message to other staff members of the system for a specific reason
25	Admin assigns and controls tasks	Admin can assign different tasks related to the hotel issues which have to be done and later approved by the staff member.
26	Receptionist logs in into the system	Receptionist user insert his/her own credentials (username/email and password) to login into the system
27	Receptionist fails to login into the system	Receptionist provides wrong username/email or password thus the login will fail
28	Receptionist changes his/her credentials	Receptionist can changes his/her stored information on the system (email, username, password etc.)
29	Receptionist logs out of the system	Receptionist clicks the logout button to initiate logout process.
30	Receptionist makes a reservation	Receptionist can make a reservation to the system by entering guests credentials and clicking ‘Book’ button
31	Receptionist replies to guest messages	Receptionist gets messages of the guests in the system and replies in the real time.
32	Receptionist can check who is Checking-in/ Checking-out of the hotel.	Receptionist can view check-in/check-out, in the dashboard of the system. So he can provide specific attention to the client

33	Receptionist is notified by the system for new bookings	Receptionist assures that the booking is done correctly, by being in touch with the client
34	Receptionist views booked rooms and availability	Receptionist looks at availability in case of reservations from phone call.
35	Receptionist clicks settings	Receptionist can edit its credentials and can also upload a profile picture
36	Guest logs in the system	Guest insert his/her own credentials (username/email and password) to login into the system
37	Guest fails to login into the system	Guest provides wrong username/email or password thus the login will fail
38	Guest logs out of the system	Guest clicks the logout button to initiate logout process. Then the web page of Hotel Enera will be displayed.
39	Guest opens Profile	Guest receives a welcoming message.
40	Guest enters check-in, check-out dates and clicks 'Check availability' button	Guest can see room types available in the dates entered in the system
41	Guest clicks Book Now	Guest will have to fill out its credentials to make the request valid
42	Guest clicks Send Message	Guest can Contact in real time with the receptionist or manager of the hotel
43	Guest clicks setting	Guest can modify its credentials and also it can upload a picture.
44	Cleaner logs in the system	Cleaner is asked to insert his/her own credentials (username/email and password) to login into the system
45	Cleaner logs out of the system	Cleaner clicks the logout button to initiate logout process. Then the home page will be displayed
46	Cleaner views Dashboard	The information with the rooms he is assigned to clean is shown
47	Cleaner clicks setting	Cleaner can modify its credentials and also it can upload a picture.
48	Cleaner clicks tick or cross button in the rooms to be cleaned section	Cleaner changes the state of the room as clean or unclean due to certain reasons.
49	Cleaner clicks print	If there is a printer connected Cleaner can get the receipt with the information.

4.1.1 User Scenarios Extended

1. Admin logins into the system

a. Admin opens the login page of the system
b. Admin is asked to enter his/her credentials (username/password)
c. Admin proves that he/she is not a robot by checking the Captcha
d. Admin clicks Login button
e. If his/her credentials matches with any of the data in the current database, the admin is successfully logged in
f. Admin gets redirected to the main view (dashboard) of the web page

2. Admin fails to login into the system

a. Admin opens the login page of the system
b. Admin is asked to enter his/her credentials (username/password)
c. Admin proves that he/she is not a robot by checking the Captcha
d. Admin clicks Login button
e. Admin types one of his/her credentials wrong therefore these data are not found on the database.
f. Admin will get e message error telling him/her that he has typed wrong credentials thus he will have to try to login again
g. Admin can click forgot password if the password is wrong or type his credentials again

3. Admin changes his/her credentials

a. Admin is first logged in
b. Admin clicks on the profile tab under settings which will open the current profile of the logged in user in this case Admin
c. Admin can view all his/her information on this page
d. Admin edits or changes all the information that he/she wants to modify
e. Admin clicks Save button
f. The changed information is stored now on database and the page is refreshed with the new entered information

4. Admin logs out of the system

a. Admin is logged in
b. Admin clicks the logout button under Settings tab
c. Admin's session is over, and he will be successfully logged out
d. The system is directed to the login page
e. If Admin wants to login again, he has to write his credentials again otherwise a successful login will not be executed.

5. **Admin views room details of the system**
    a. Admin is logged in the system
    b. Admin clicks on the settings tab in the navigation bar
    c. Admin clicks the view Room details button in the settings tab
    d. Admin will be redirected to the Room's page where he can check all details of a specific room

6. **Admin adds new room to the system**
    a. Admin is logged in the system
    b. Admin clicks Add Room button in the settings tab
    c. Admin fills the form for the new Room with the proper information about the new room
    d. Admin clicks Submit button in order to add the new room or Cancel button if he/she wants to abort the action
    e. If Submit is clicked the new room with the given information will be added on the database
    f. The page will be refreshed and the room will be shown on the room details.

7. **Admin modifies room properties**
    a. Admin is logged in
    b. Admin clicks view Rooms button on the settings tab in the navigation bar
    c. Admin clicks on the room that he/she wants to modify
    d. Admin updates the new information by filling the form which already contains the current information about the room.
    e. Admin clicks save button to save the new entered information
    f. The new information will now be stored in the database by replacing the old one and the page will be refreshed showing the new information about the room

8. **Admin deletes room from the system**
    a. Admin is logged in
    b. Admin clicks Remove Room button in the settings tab
    c. Admin selects the Room that he/she wants to remove and click on the remove room button
    d. A confirmation dialog will be showed asking if he/she is sure to remove the specific room
    e. If yes is clicked the room will be removed from database together with all its information
    f. The page will be refreshed and the deleted room will not be shown there anymore.

9. **Admin views staff/employee details**
    a. Admin is logged in the system
    b. Admin clicks on the settings tab in the navigation bar
    c. Admin clicks the view Staffdetails button in the settings tab
    d. Admin will be redirected to the Staff's page where he can check all details for a specific employee

10. Admin adds new staff/employee to the system

a. Admin is logged in the system
b. Admin clicks Add Staff/Employee button in the settings tab
c. Admin fills the form for the new employee with the proper information about him/her
d. Admin clicks Submit button in order to add the new employee
e. The new employee will be added on the database with the given information
f. The page will be refreshed and the employee will be shown on the employees' view.

11. Admin modifies staff/employee credentials

a. Admin is logged in
b. Admin clicks on the view employees button on the settings tab in the navigation bar
c. Admin clicks on the employee that he/she wants to modify
d. Admin updates the new information by filling the form which already contains the current information about the employee.
e. Admin clicks save button to save the new entered information
f. The new information will now be stored in the database by replacing the old one and the page will be refreshed showing the new information about the employee

12. Admin deletes staff/employee from the system

a. Admin is logged in
b. Admin clicks Remove Employee button in the settings tab
c. Admin selects the Employee that he/she wants to remove and click on the remove Employee button
d. A confirmation dialog will be showed asking if he/she is sure to remove the specific employee
e. If yes is clicked the employee will be removed from database together with all his/her information
f. The page will be refreshed and the deleted employee will not be shown there anymore.

13. Admin views product details

a. Admin is logged in the system
b. Admin clicks on the settings tab in the navigation bar
c. Admin clicks the view Staffdetails button in the settings tab
d. Admin will be redirected to the Staff's page where he can check all details for a specific employee

14. Admin adds product to the inventory

a. Admin is logged in
b. Admin clicks Inventory button on the navigation bar
c. Admin clicks Add Product button and will be redirected to a fillable empty form where he/she can add information about the new entered product (name, amount, type etc.)
d. Admin clicks Submit button in order to add the new product in the inventory
e. The new product will be added on the inventory database with the given information

f. The page will be refreshed, and the product will now be shown on the inventory's page.

15. Admin modifies product's information

a. Admin is logged in
b. Admin clicks Inventory button on the navigation bar
c. Admin clicks Edit Product button and is redirected to a form which contains the information for the current product from where he can update that information
d. Admin clicks Save button to save the new entered information for the product
e. The new information is stored in the database and the old information now is being replaced
f. The inventory page is refreshed, and the product will now be shown on the inventory's page with the new information

16. Admin deletes product

a. Admin is logged in
b. Admin clicks Inventory button on the navigation bar
c. Admin points to the product that he wants to remove and clicks Remove button from the products view page
d. A confirmation dialog will be showed asking if he/she is sure to remove the specific product
e. If yes is clicked the product will be removed from database together with all its information
f. The inventory page will be refreshed, and the deleted product will not be shown there anymore.

17. Admin views financial report

a. Admin is logged in
b. Admin clicks on the Statistics Tab in the navigation bar
c. Admin Clicks on Reports tab
d. Admin can choose two date (first date indicated from which date the report will be calculated and the second date indicates until what day the report will be calculated.
e. The report will be shown with different models (Line graph, Pie charts, Bar graph etc.) and with normal numbers as well.
f. Print financial report

18. Admin views guest details

a. Admin is logged in
b. Admin clicks on Booked Rooms tab in the navigation bar
c. Admin can view the name/surname of the guest who has booked a specific room in the schedule viewer
d. Admin clicks on the active guest view or all guests history to view details for each registered guest.

19. Admin views rooms availability

a. Admin is logged in
b. Admin clicks on Booked Rooms tab in the navigation bar
c. Admin can view the availability of each room on the schedule located in the Booked Rooms page
d. Admin can choose Booked Room list view from where he can see all rooms and their availability on specific days
e. Admin searches for a specific room number or type to filter the rooms availability view

20. Admin view booking details

a. Admin is logged in
b. Admin clicks on the Booking details button in the dashboard page
c. Admin will have a list view for each booking that is made
d. Admin can click on any of the transactions to have a more details view about that specific booking (guest, number of days, price etc.)

21. Admin modifies booked rooms

a. Admin is logged in
b. Admin clicks on Booked Rooms tab in the navigation bar
c. Admin can click in one of the booked rooms on the schedule view or he/she can search for the room they want to apply changes on the search bar
d. Admin makes the proper changes of the booking and an automatic email claiming about this change is sent to the guest who has booked that room

22. Admin views guest reviews

a. Admin is logged in
b. Admin clicks on Guest Reviews in the dashboard page
c. Admin can view all the reviews sorted by date as default or he/she can sort them based on other criteria
d. Admin can reply to any of the reviews and the guest will be modified that his review is being replied by the admin

23. Admin prints room cleanliness status

a. Admin is logged in
b. Admin clicks on Booked rooms tab in the navigation bar
c. Admin can see the cleanliness status in the schedule view in the right corner of the room or in the list view.
d. Admin click print room cleanliness status to get an overview of the rooms that have to be cleaned and the cleaner to who it is assigned

24. Admin leaves notes for the employees

a. Admin is logged in
b. Admin clicks on assign tasks in the dashboard page
c. Admin can assign task to each of the employees if he/she is not available to reach them in other ways
d. The tasks are shown on the dashboard of each user

25. Admin assigns and controls tasks

a. Admin is logged in
b. Admin clicks on assign tasks in the dashboard page
c. Admin can assign task to each of the employees if he/she is not available to reach them in other ways
d. The tasks are shown on the dashboard of each user

26. Receptionist logs in into the system

a. Receptionist opens the login page of the system
b. Receptionist is asked to enter his/her credentials (username/password)
c. Receptionist proves that he/she is not a robot by checking the Captcha
d. Receptionist clicks Login button
e. If his/her credentials matches with any of the data in the current database, the Receptionist is successfully logged in
f. Receptionist gets redirected to the main view (dashboard) of the web page adapted for the receptionist only

27. Receptionist fails to login into the system

a. Receptionist opens the login page of the system
b. Receptionist is asked to enter his/her credentials (username/password)
c. Receptionist proves that he/she is not a robot by checking the Captcha
d. Receptionist clicks Login button
e. Receptionist types one of his/her credentials wrong therefore these data are not found on the database.
f. Receptionist will get e message error telling him/her that he has typed wrong credentials thus he will have to try to login again
g. Receptionist can click forgot password if the password is wrong or type his credentials again

28. Receptionist changes his/her credentials

a. Receptionist is first logged in
b. Receptionist clicks on the profile tab under settings which will open the current profile of the logged in user in this case Receptionist
c. Receptionist can view all his/her information on this page
d. Receptionist edits or changes all the information that he/she wants to modify
e. Receptionist clicks Save button
f. The changed information is stored now on database and the page is refreshed with the new entered information

29. Receptionist logs out of the system

a. Receptionist is logged in
b. Receptionist clicks the logout button under Settings tab
c. Receptionist's session is over, and he/she will be successfully logged out
d. The system is directed to the login page
e. If the receptionist wants to login again, he/she has to write his credentials again otherwise a successful login will not be executed.

30. Receptionist makes a reservation

a. Receptionist is logged in
b. Receptionist has taken a notification or message from a specific guest to complete his/her booking
c. Receptionist clicks on Booked Rooms tab in the navigation bar
d. Receptionist clicks on add new Booking button or he/she can directly click on the schedule view in one of the rooms for a specific date
e. Receptionist fills the booking with the guest information and requirements
f. Receptionist clicks on Book Now button and the room is booked with the information provided by the receptionist
g. An email is sent to the guest who requested this booking that the room is booked successfully

31. Receptionist replies to guest messages

a. Receptionist is logged in
b. Receptionist sees that he/she has received a new message and clicks on the messages panel in the dashboard page
c. Receptionist sees all the messages and clicks on the guest that he/she wants to reply
d. Receptionist writes a reply message for the guest and clicks Send Message button

32. Receptionist can check who is Checking-in/Checking-out of the hotel

a. Receptionist is logged in
b. Receptionist clicks on Today Check-in/Check-out panel to view all the guests that are coming on that day at the hotel or are leaving that day from the hotel
c. Receptionists can also view who has checked-in/checked-out of the hotel for other days that he/she wants to check.

33. Receptionist is notified by the system for new bookings

a. Receptionist is logged in
b. The notifications icon badge will show the receptionist the number of unread notifications
c. Receptionist clicks on the notification icon and checks the notifications for the new bookings that are processed
d. The receptionist can also check the new bookings on the booking details panel

34. Receptionist views booked rooms and availability

a. Receptionist is logged in
b. Receptionist clicks on Booked Rooms tab in the navigation bar
c. Receptionist can view the availability of each room on the schedule located in the Booked Rooms page
d. Receptionist can choose Booked Room list view from where he can see all rooms and their availability on specific days
e. Receptionist searches for a specific room number or type to filter the rooms availability view

35. Receptionist clicks settings (change credentials)

a. Receptionist is logged in
b. Receptionist clicks on the profile tab under settings which will open the current profile of the logged in user in this case the guest view.
c. Receptionist can view all his/her information on this page
d. Receptionist edits or changes all the information that he/she wants to modify
e. Receptionist clicks Save button
f. The changed information is stored now on database and the page is refreshed with the new entered information

36. Guest logs in the system

a. Guest opens the login page of the system
b. Guest is asked to enter his/her credentials (username/password)
c. Guest proves that he/she is not a robot by checking the Captcha
d. Guest clicks Login button
e. If his/her credentials matches with any of the data in the current database, the guest is successfully logged in
f. Guest gets redirected to the main view of the web page adapted for the guest only

37. Guest fails to login into the system

a. Guest opens the login page of the system
b. Guest is asked to enter his/her credentials (username/password)
c. Guest proves that he/she is not a robot by checking the Captcha
d. Guest clicks Login button
e. Guest types one of his/her credentials wrong therefore these data are not found on the database.
f. Guest will get e message error telling him/her that he has typed wrong credentials thus he will have to try to login again
g. Guest can click forgot password if the password is wrong or type his credentials again

38. Guest logs out of the system

a. Guest is logged in
b. Guest clicks the logout button under Settings tab
c. Guest session is over, and he/she will be successfully logged out
d. The system is directed to the login page
e. If the guest wants to login again, he/she has to write his credentials again otherwise a successful login will not be executed.

39. Guest opens profile

a. Guest is not registered yet on the system thus he opens the sign up page of the system
b. Guest fills the registration form with all his/her information
c. Guest will take a confirmation email to confirm his/her account
d. Guest will be directed on the login page to enter his credentials to login on the system
e. The 35 Scenario will take part from now

40. Guest checks room availability

a. Guest is logged in
b. Guest chooses two dates (check-in and check-out) which defines the period that he wants to stay in the hotel
c. Guest will have a list of rooms that are available during these days with some information about the room
d. Guest can click in one of the rooms available on these dates to view full details of that room

41. Guest books one or several rooms

a. Guest is logged in
b. Guest performs scenario 40 to view rooms availability for the period that he/she wants to stay on the hotel
c. Guest clicks book room within the dates that he/she has chosen
d. Guest will be notified that the room is successfully booked

42. Guest sends message

a. Guest is logged in
b. Guest clicks on send message button to open an text area where he/she can write his/her message
c. Guest types the message with any issue or need that he/she has and clicks on the send button to send the message to the receptionist

43. Guest clicks settings (change his/her credentials)

a. Guest is logged in
b. Guest clicks on the profile tab under settings which will open the current profile of the logged in user in this case the guest view.
c. Guest can view all his/her information on this page
d. Guest edits or changes all the information that he/she wants to modify
e. Guest clicks Save button
f. The changed information is stored now on database and the page is refreshed with the new entered information

44. Cleaner logs in the system

a. Cleaner opens the login page of the system
b. Cleaner is asked to enter his/her credentials (username/password)
c. Cleaner proves that he/she is not a robot by checking the Captcha
d. Cleaner clicks Login button
e. If his/her credentials match with any of the data in the current database, the cleaner is successfully logged in
f. Cleaner gets redirected to the main view of the web page adapted for the cleaner only

45. Cleaner logs out of the system

a. Cleaner is logged in
b. Cleaner clicks the logout button under Settings tab
c. Cleaner session is over, and he/she will be successfully logged out
d. The system is directed to the login page
e. If the cleaner wants to login again, he/she has to write his credentials again otherwise a successful login will not be executed.

46. Cleaner views Dashboard(checks room status)

a. Cleaner is logged in
b. Cleaner clicks on the Rooms to be Cleaned panel
c. Cleaner views which rooms are assigned to him/her for cleaning

47. Cleaner clicks settings (change his/her credentials)

a. Cleaneris logged in
b. Cleanerclicks on the profile tab under settings which will open the current profile of the logged in user in this case the cleaner view.
c. Cleanercan view all his/her information on this page
d. Cleaneredits or changes all the information that he/she wants to modify
e. Cleanerclicks Save button
f. The changed information is stored now on database and the page is refreshed with the new entered information

48. Cleaner sets room status to cleaned

a. Cleaner is logged in
b. Cleaner performs scenario 46 to check for the rooms that he/she has to clean
c. After cleaning one of the assigned rooms he/she sets the status of the room to cleaned

49. Cleaner clicks print room status

a. Cleaner is logged in
b. Cleaner clicks on The Rooms to be cleaner panel
c. Cleaner views the rooms that are assigned to him/her for cleaning
d. Cleaner clicks print report button to have a more transparent view of the room status

4.2 Use Cases

U1

Name	**User login**
Summary	User is successfully logged in after he/she provides correct credentials to the system
Actor	Admin, Receptionist, Guest, Cleaner
Description	The user opens login page and in order to gain access to the main page of the system and other features offered, he/she must provide his/her valid credentials to the login page or can directly sign in using Google Sign-in. The user will later be directed to the main page which is different for different types of user. a. User opens the login page of the system b. User is asked by the system to enter his/her credentials (username/password) c. User proves that he/she is not a robot by checking the Captcha d. User clicks Login button e. If his/her credentials matches with any of the data in the current database, the user is successfully logged in f. User gets redirected to the main view (dashboard) of the web page
Precondition	The user must first have been registered in the software using his/her personal credentials
Alternatives	There are no alternative options
Post Condition	The user gains access to their profile which differs from the user's role

U2

Name	**User manages his/her profile**
Summary	User goes to Profile and edit data
Actor	Admin, receptionist, guest, cleaner
Description	• User click settings, under which will find the button Profile. There user can edit his/her details: name, surname, email, password, birthdate, address, edit picture, upload a new picture. • User edits or changes all the information that he/she wants to modify • User clicks Save button • The changed information is stored now on database and the page is refreshed with the new entered information
Precondition	User must be logged in
Alternatives	User types his/her credentials wrong so user will get e message error telling him/her that he has typed wrong credentials thus he will have to try to login again
Post Condition	The changed information is stored now on database and the page is refreshed with the new entered information

U3

Name	**Admin manages staff**
Summary	Admin can view, edit, add, delete guests and employees information
Actor	Admin
Description	Manage employees • Admin clicks on the settings tab in the navigation bar • Admin clicks the view Staff details button in the settings tab • Admin will be redirected to the Staff's page where he can check all details for a specific employee, where admin can modify or delete data • In the other case admin can add an employee
Precondition	Admin must be logged in
Alternatives	None
Post Condition	Admin will have detailed knowledge about hotel's employees and customers.

U4

Name	**Admin manages inventory**
Summary	Admin can add, modify or delete a product.
Actor	Admin
Description	Admin clicks Inventory button on the navigation bar a. Admin clicks Add Product button and will be redirected to a fillable empty form where he/she can add information about the new entered product (name, amount, type etc.) Admin clicks Submit button in order to add the new product in the inventory b. Admin clicks Edit Product button and is redirected to a form which contains the information for the current product from where he can update that information Admin clicks Save button to save the new entered information for the product c. Admin points to the product that he wants to remove and clicks Remove button from the products view page A confirmation dialog will be showed asking if he/she is sure to remove the specific product
Precondition	Admin must be logged in
Alternatives	None
Post Condition	The inventory page is refreshed, and the product change(added, modified or deleted) will be shown on the inventory's page with the new information

U5

Name	**Admin manages rooms**
Summary	Admin can view, edit, add, delete rooms
Actor	Admin
Description	Admin goes to the navigation bar, in the settings tab can view, edit, add, and delete rooms.
Precondition	Admin must be logged in
Alternatives	None
	Admin will have detailed knowledge about hotel's capacity.

U6

Name	**Admin views dashboard**
Summary	Admin should manage the financial report calculations, view guest details, guest review
Actor	Admin
Description	1. Admin clicks on the Statistics Tab in the navigation bar 2. Admin Clicks on Reports tab 3. Admin can choose two date (first date indicated from which date the report will be calculated and the second date indicates until what day the report will be calculated. 4. Admin can view guest details 5. Can view guest review
Precondition	Admin is logged in and has good knowledge on wages, expenses and budget of the hotel
Alternatives	None
	The report will be shown with different models (Line graph, Pie charts, Bar graph etc.) and with normal numbers as well.

U7

Name	**Notes for employees**
Summary	Admin can assign task to each of the employees.
Actor	Admin, receptionist, cleaner
Description	e. Admin clicks on assign tasks in the dashboard page f. Admin can assign task to each of the employees if he/she is not available to reach them in other ways g. The tasks are shown on the dashboard of each user
Precondition	Admin must be logged in
Alternatives	Admin can reach employees in other way (for example call them)
Post Condition	Employees are able to see the tasks in their dashboard.

U8

Name	**Cleaning process**
Summary	The cleaner has to check for the assigned room and after the work is done to change the status of the room cleaned.
Actor	Cleaner
Description	a. Cleaner clicks on the Rooms to be Cleaned panel b. Cleaner views which rooms are assigned to him/her for cleaning c. Cleaner clicks print report button to have a more transparent view of the room status
Precondition	Cleaner has to be logged in.
Alternatives	She can do the job manually.
Post Condition	After cleaning one of the assigned rooms he/she sets the status of the room to cleaned

U9

Name	**View booked rooms and availability**
Summary	Admin and receptionist are able to check if each room is booked or not in a specific day.
Actor	Admin, receptionist
Description	a. User clicks on Booked Rooms tab in the navigation bar b. User can view the availability of each room on the schedule located in the Booked Rooms page and in print their cleaness status. c. User can choose Booked Room list view from where he can see all rooms and their availability on specific days d. User searches for a specific room number or type to filter the rooms availability view
Precondition	User has to be logged in.
Alternatives	None.
Post Condition	User has a clear rooms availability view.

U10

Name	**Guest registration**
Summary	User can register into the system.
Actor	guest
Description	a. Guest is not registered yet on the system thus he opens the sign up page of the system b. Guest fills the registration form with all his/her information
Precondition	User has not been registered before and has to fill all text boxes information.
Alternatives	None
Post Condition	Guest have to receive a confirmation email.

U11

Name	**Guest messages**
Summary	System makes possible the communication between guest and receptionist.
Actor	Guest, Receptionist
Description	a. Guest clicks on send message button to open an text area where he/she can write his/her message b. Guest types the message with any issue or need that he/she has and clicks on the send button to send the message to the receptionist c. Receptionist sees that he/she has received a new message and clicks on the messages panel in the dashboard page d. Receptionist sees all the messages and clicks on the guest that he/she wants to reply e. Receptionist writes a reply message for the guest and clicks Send Message button
Precondition	user has to be logged in
Alternatives	Guest can call receptionist.
Post Condition	Guest is notified for the message from receptionist.

U12

Name	**Booking**
Summary	User can book a room after checking its availability.
Actor	Guest, receptionist
Description	a. User chooses two dates (check-in and check-out) which defines the period that he wants to stay in the hotel b. User will have a list of rooms that are available during these days with some information about the room c. User can click in one of the rooms available on these dates to view full details of that room d. User clicks book room within the dates that he/she has chosen
Precondition	User has to be logged in and room has to be available in order to be booked. If the guest make reservation by himself receptionist is just notified.
Alternatives	Guest can call receptionist.
Post Condition	Guest will be notified that the room is successfully booked

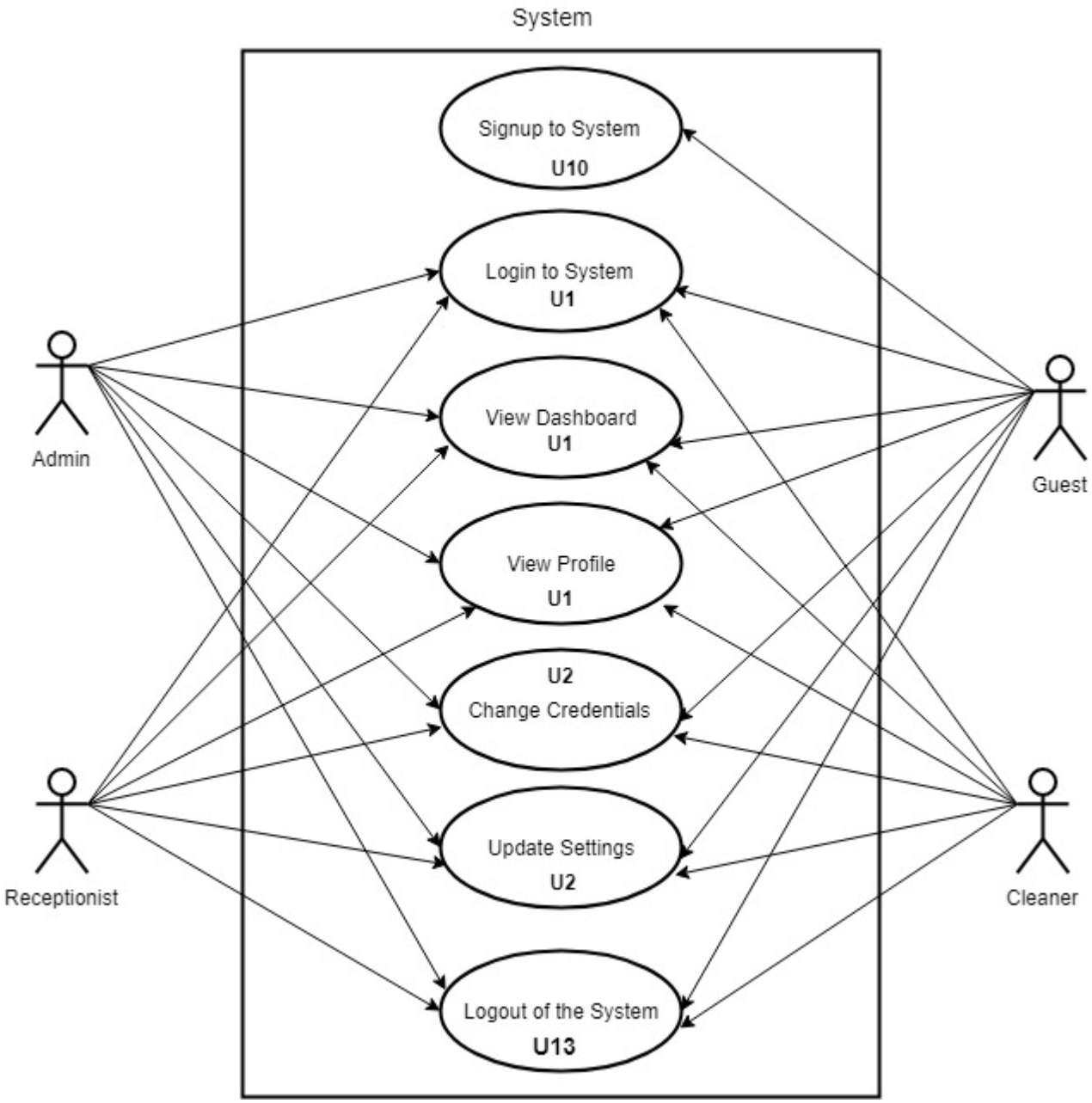
U13

Name	**Log out**
Summary	User can logout the system after his/her session is over.
Actor	Admin, receptionist, cleaner, guest
Description	User click logout under settings tab.
Precondition	User has to be logged in.
Alternatives	None
Post Condition	If the user wants to login again, he/she has to write his credentials again otherwise a successful login will not be executed.

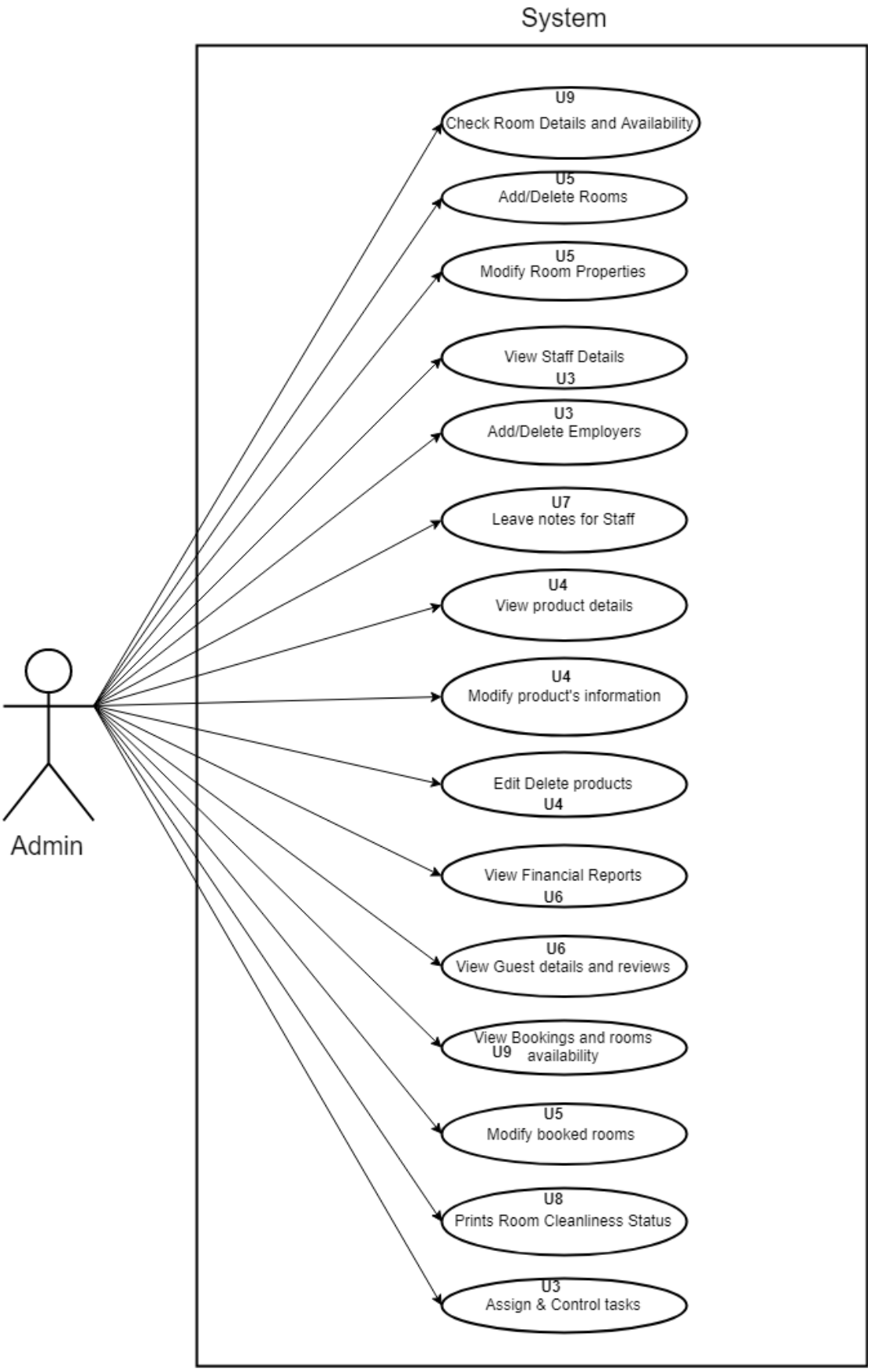
5. Behavioral Diagrams

5.1.1 Use Cases

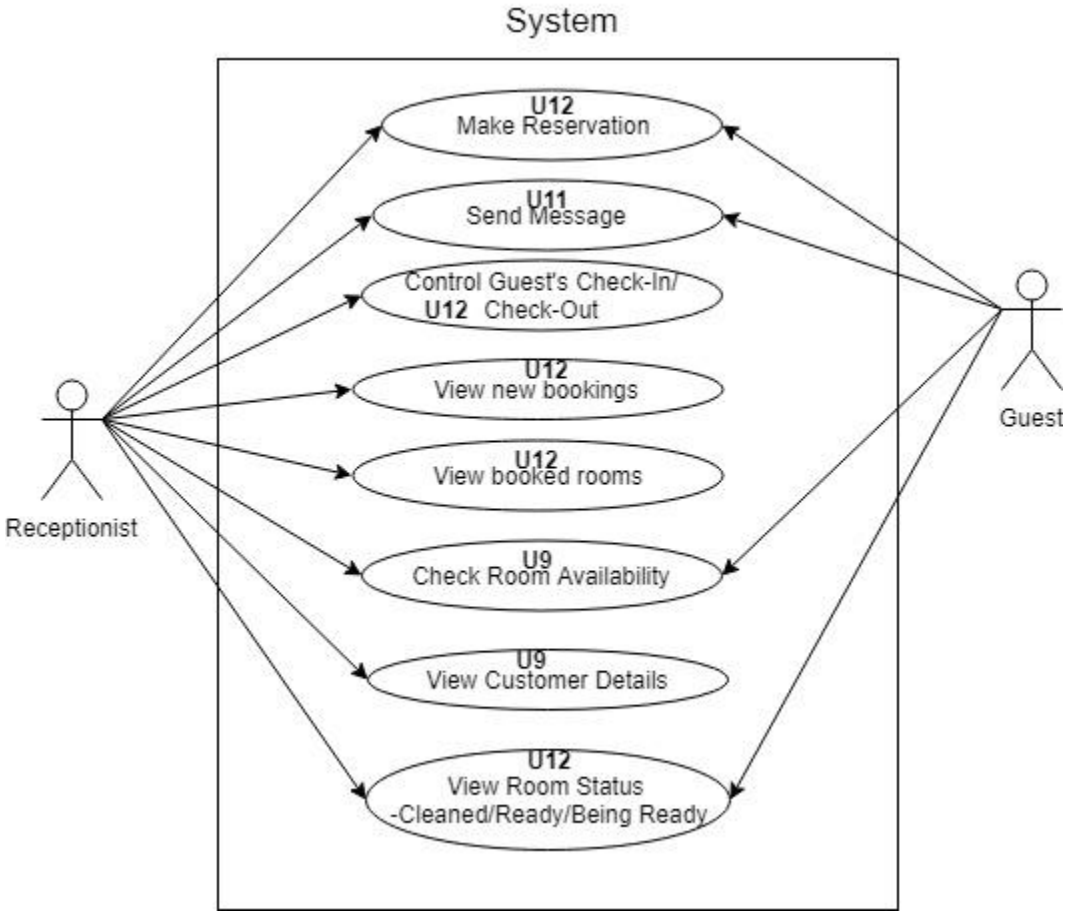
5.1.1.1 Use Case – Basic Operations



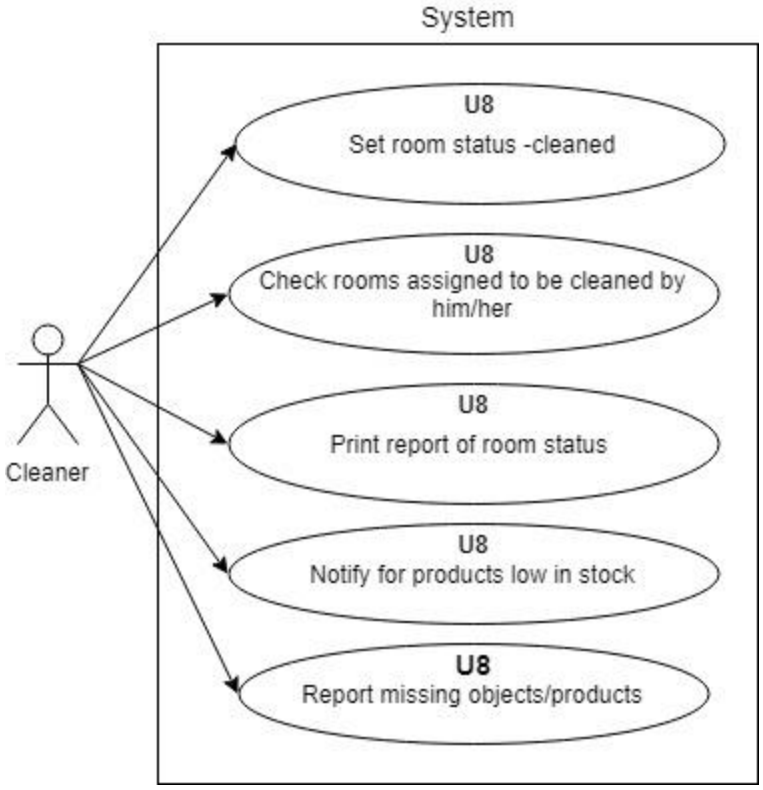
5.1.1.2 Use Case – Admin



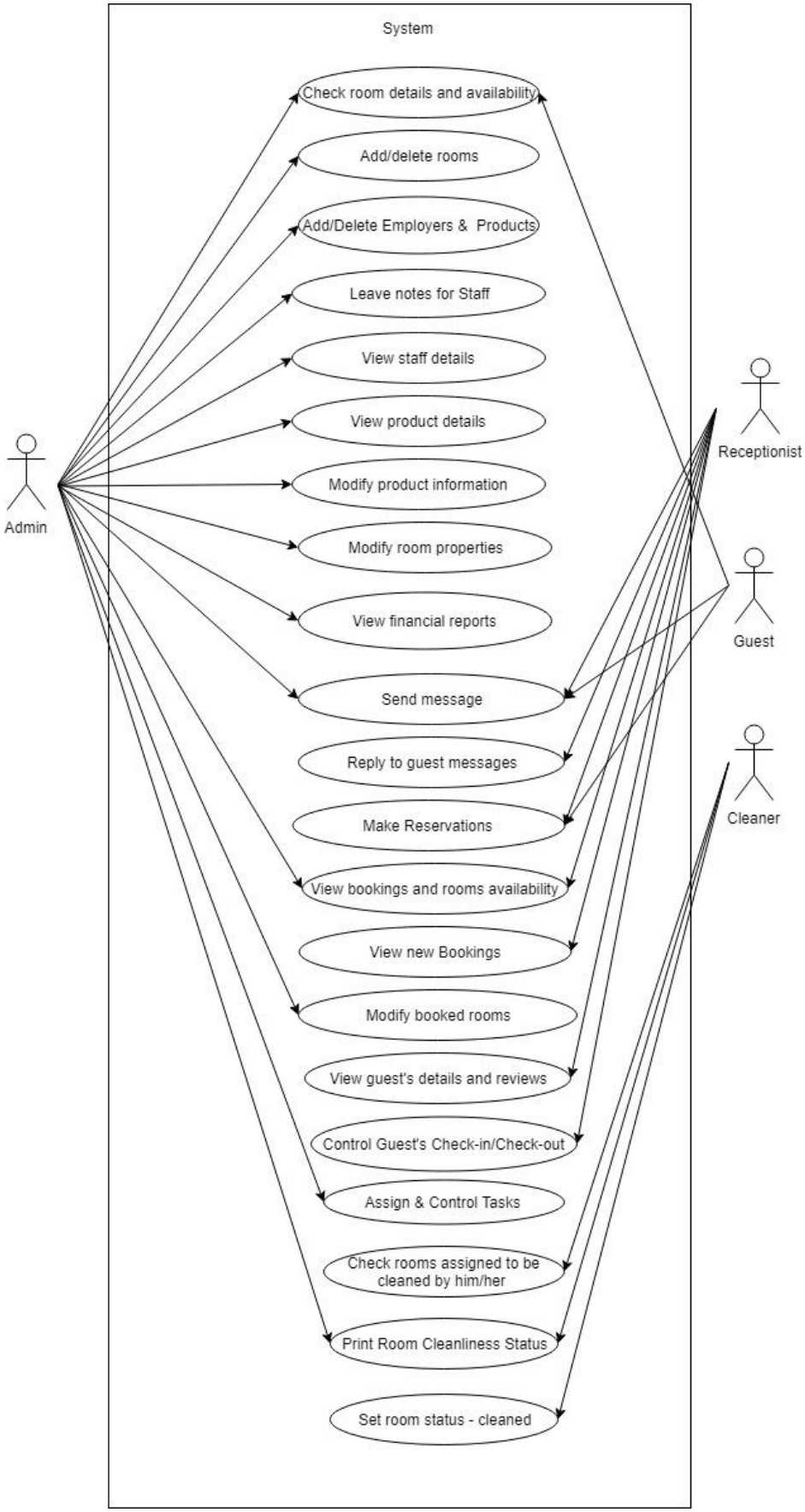
5.1.1.3 **Use Case – Receptionist & Guest**



5.1.1.4 Use Case – Cleaner

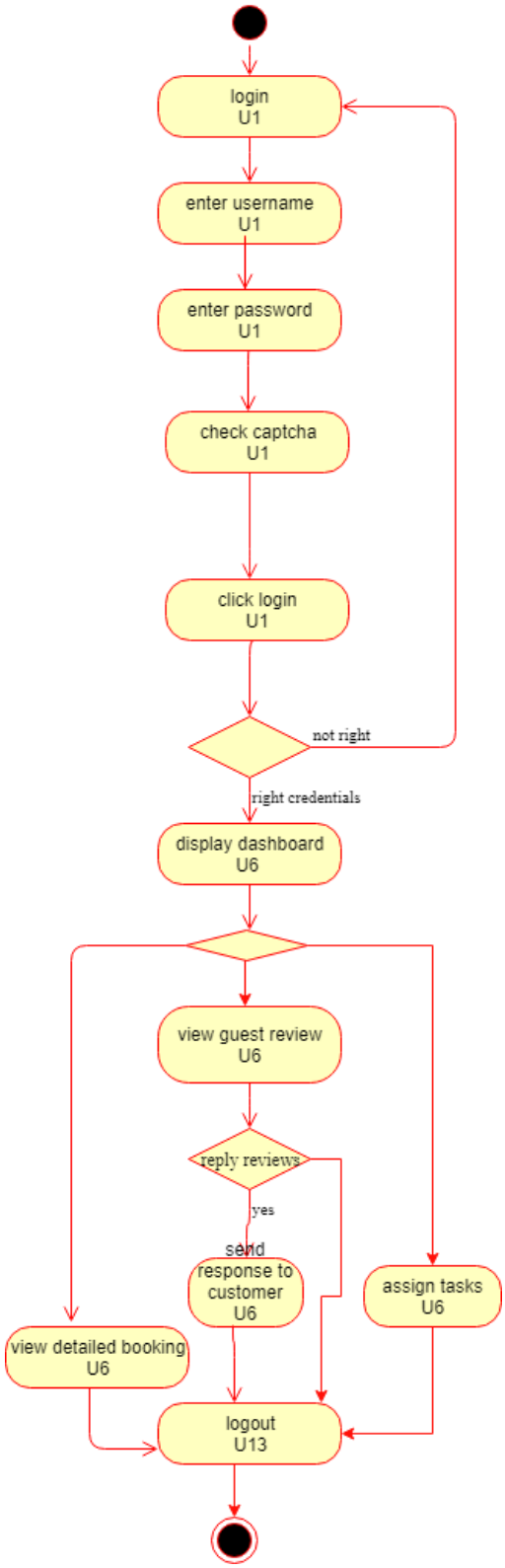


5.1.1.5 Use Case – General (All Users)

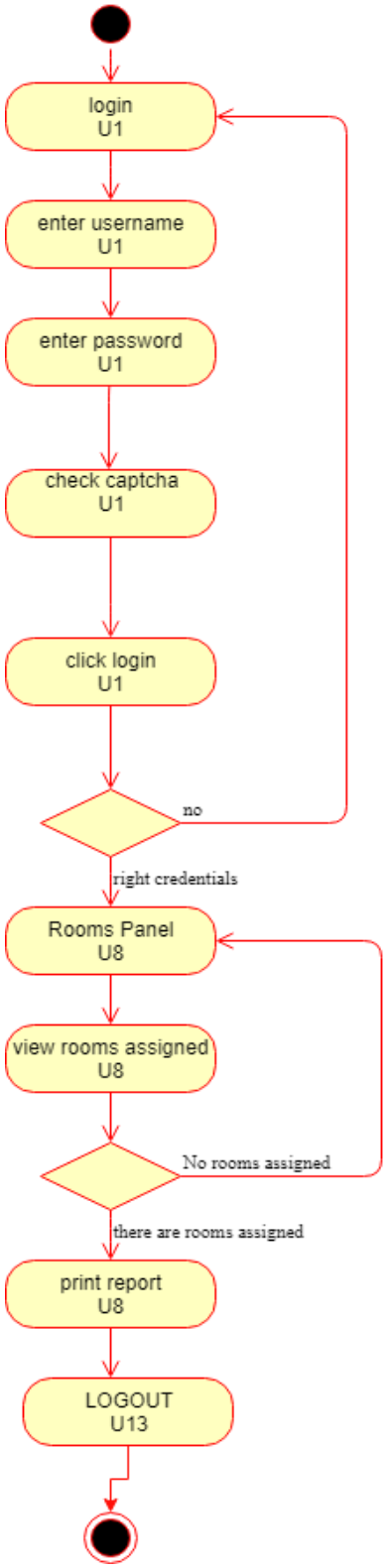


5.1.2 Activity Diagrams

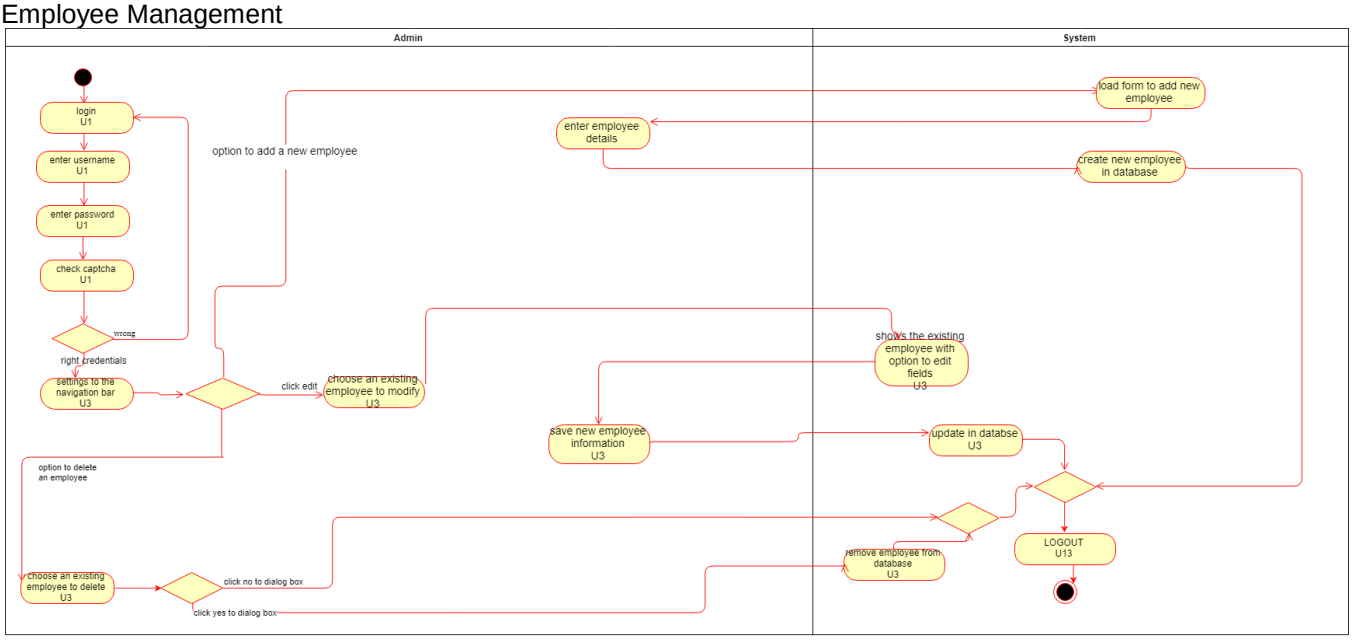
Admin Dashboard



Cleaner Activity

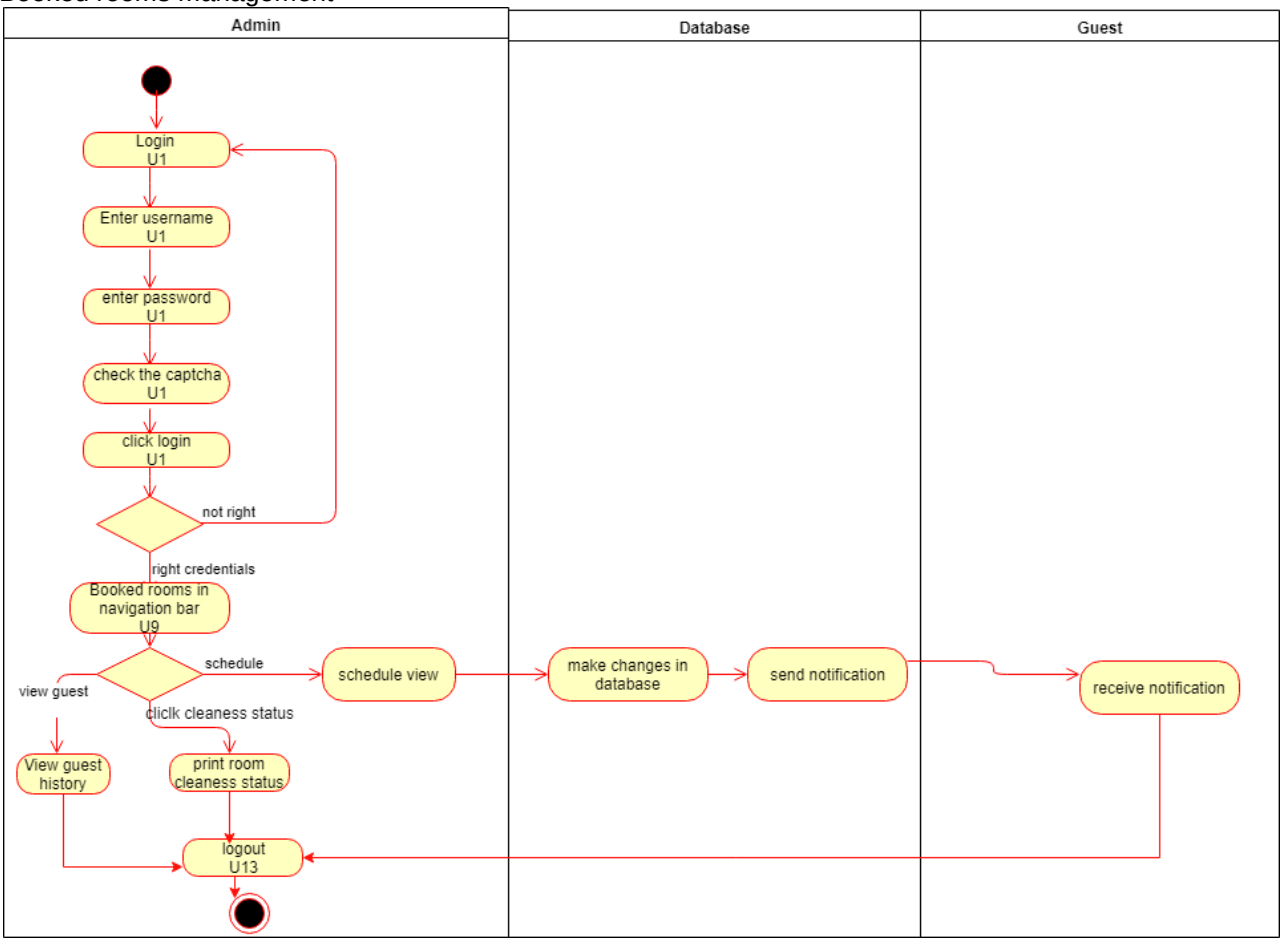


Employee Management

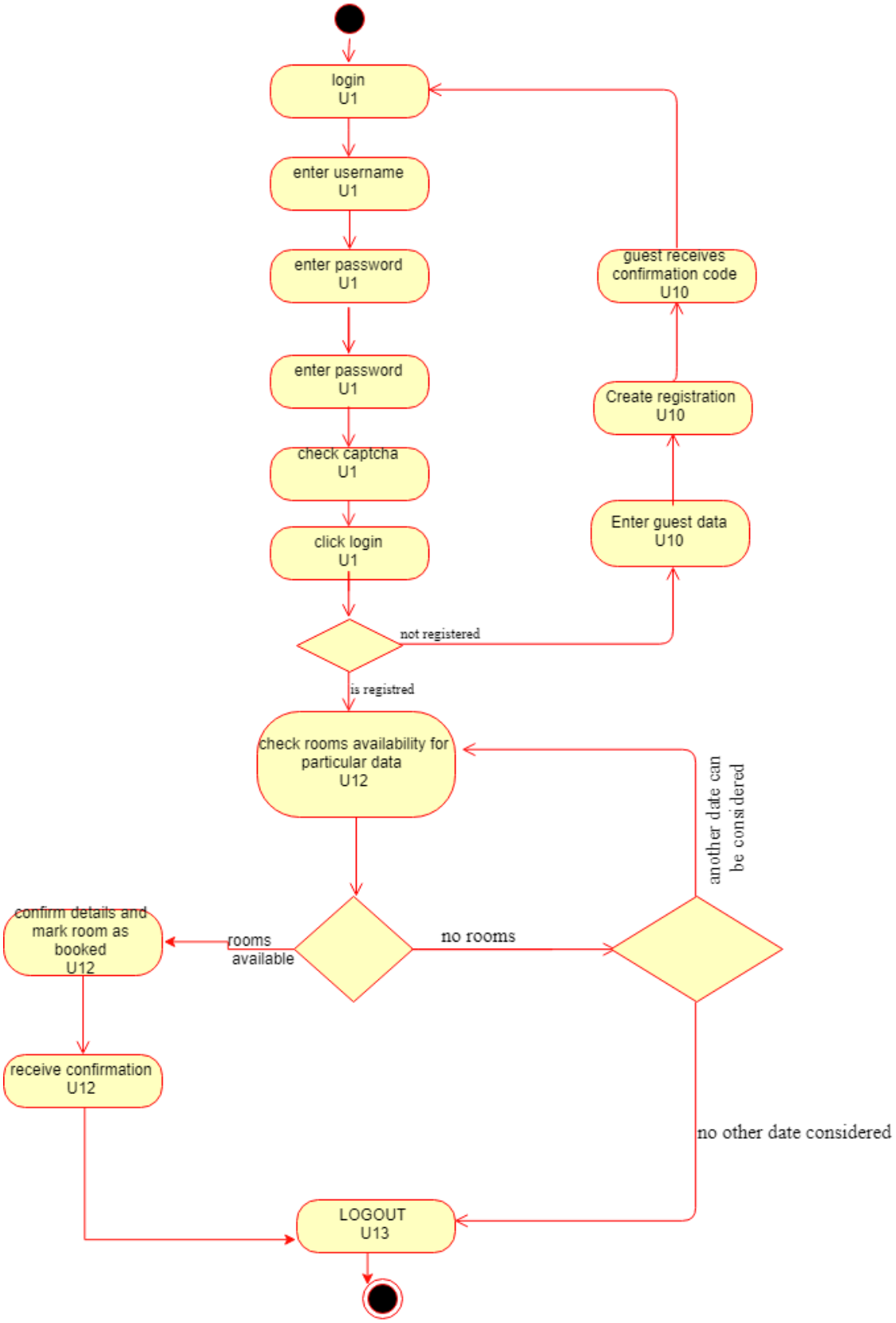


Text

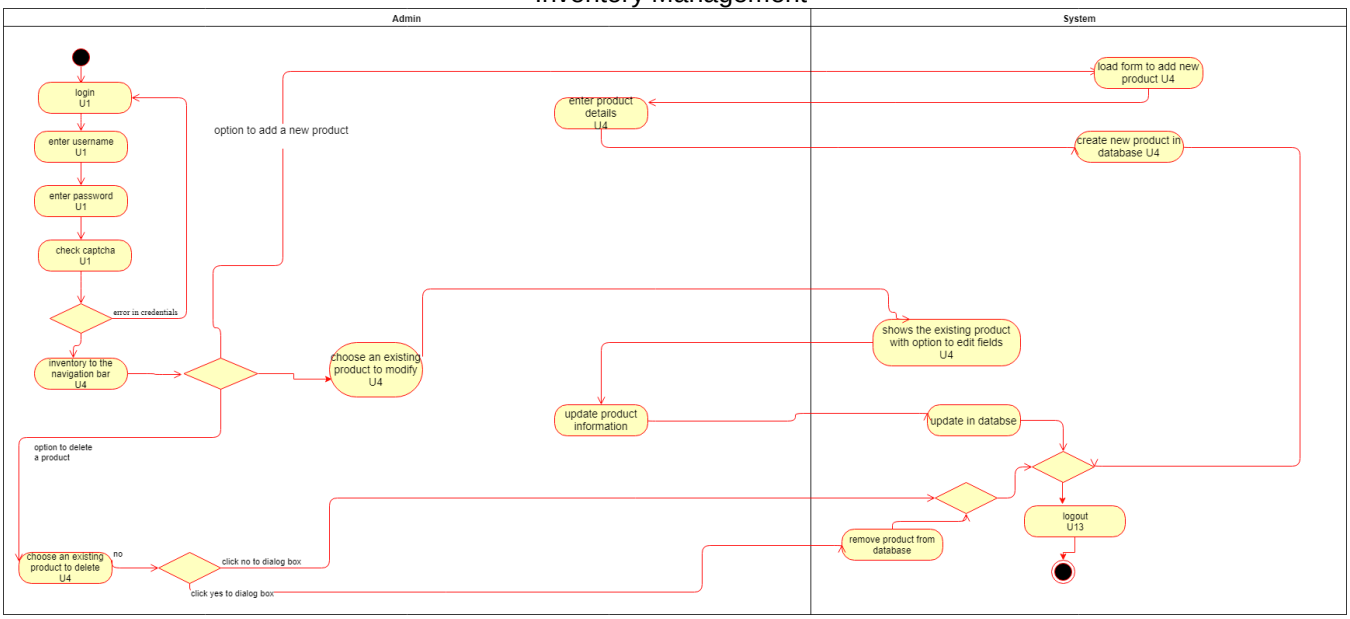
Booked rooms management



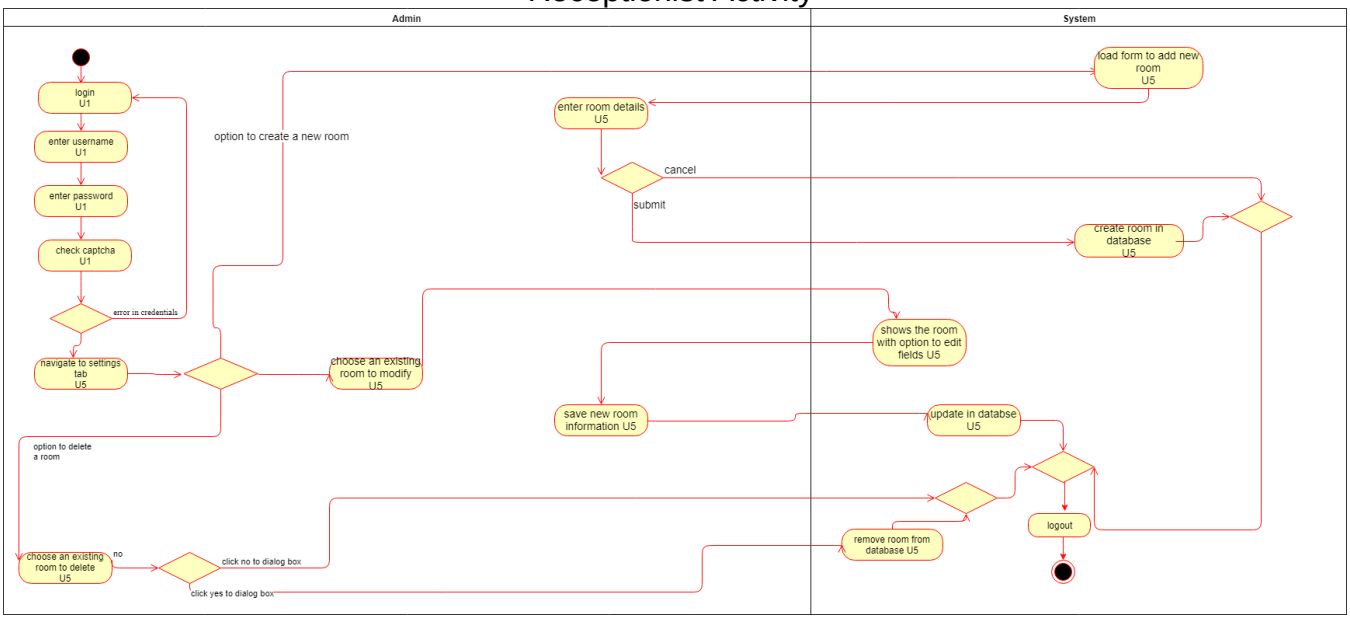
Guest Booking



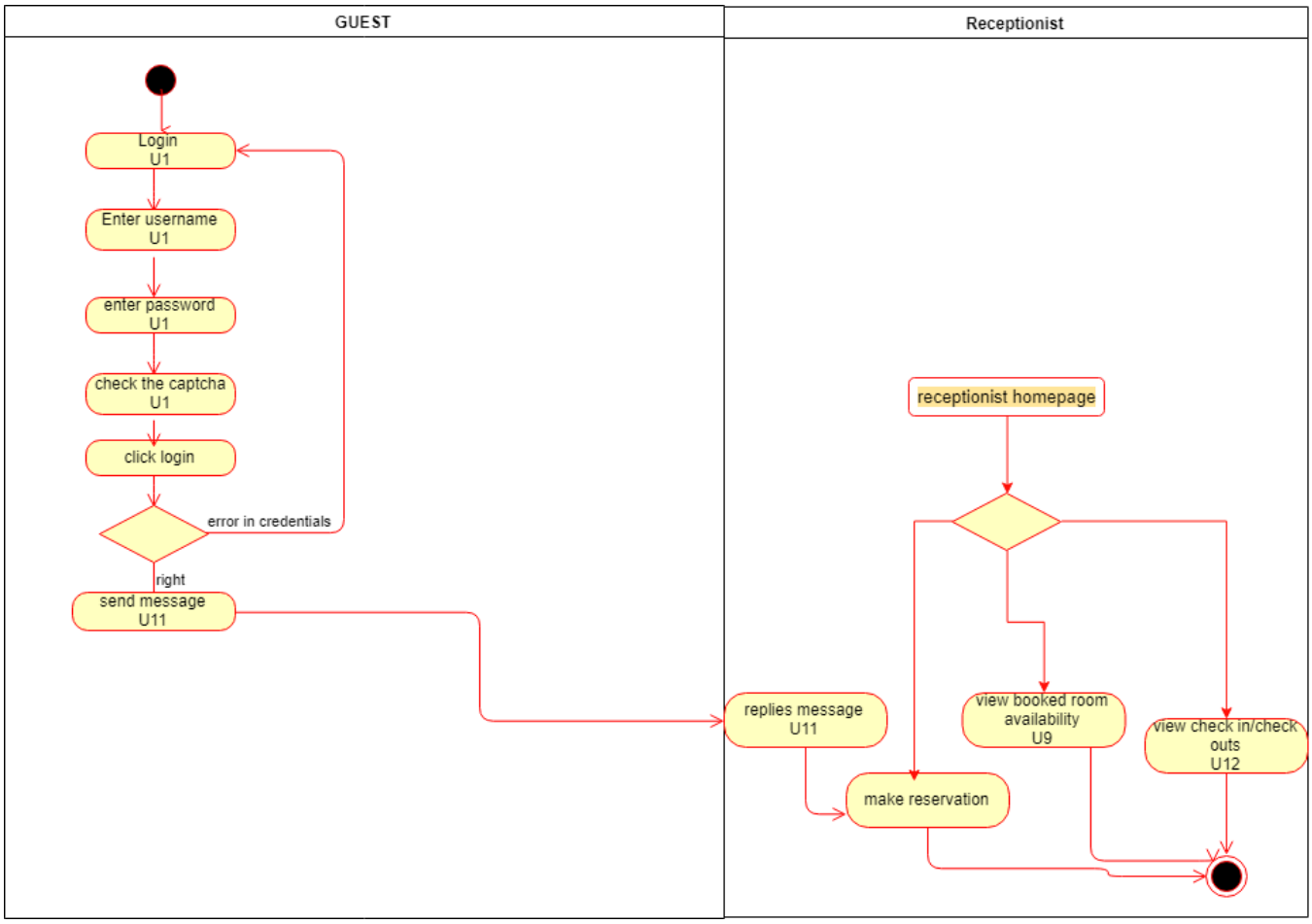
Inventory Management



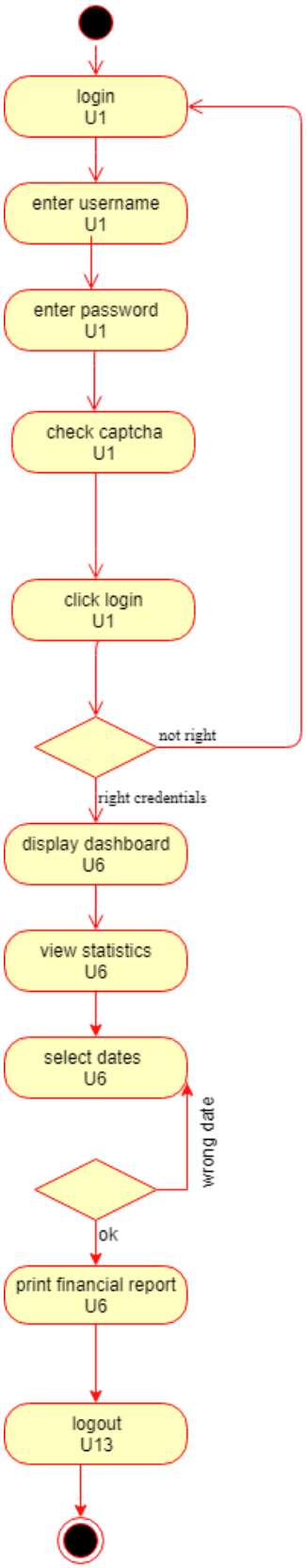
Receptionist Activity



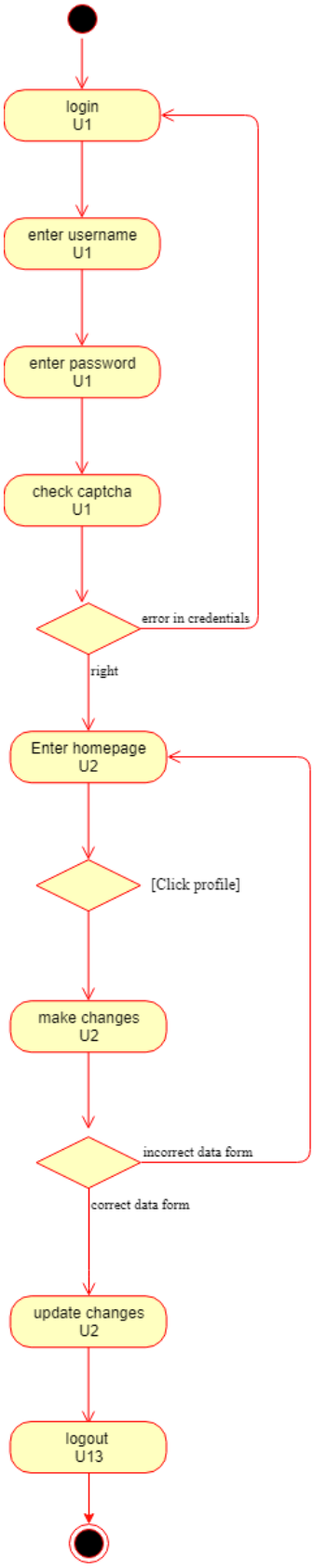
Guest –Receptionist Booking



Financial report

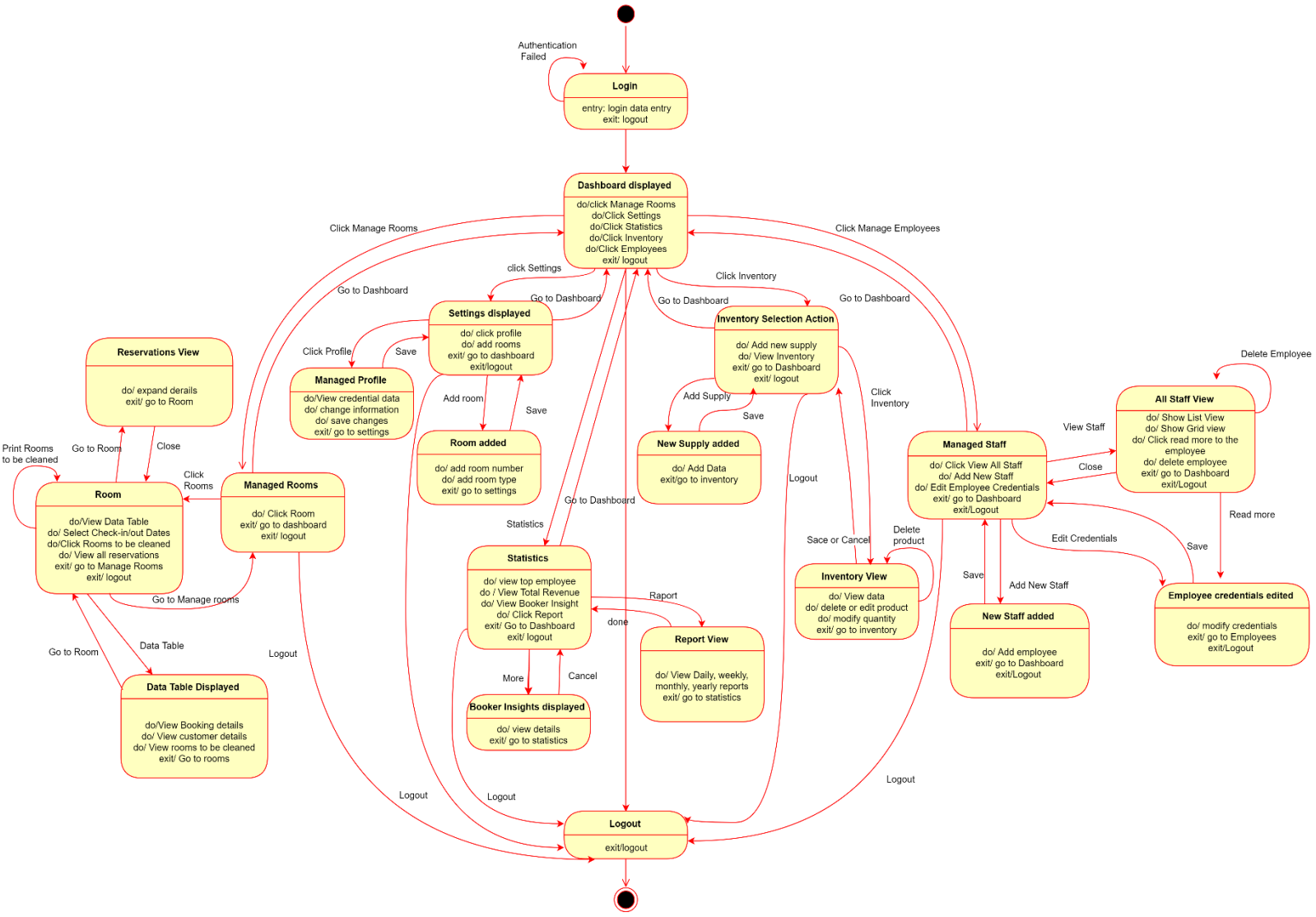


Profile Management



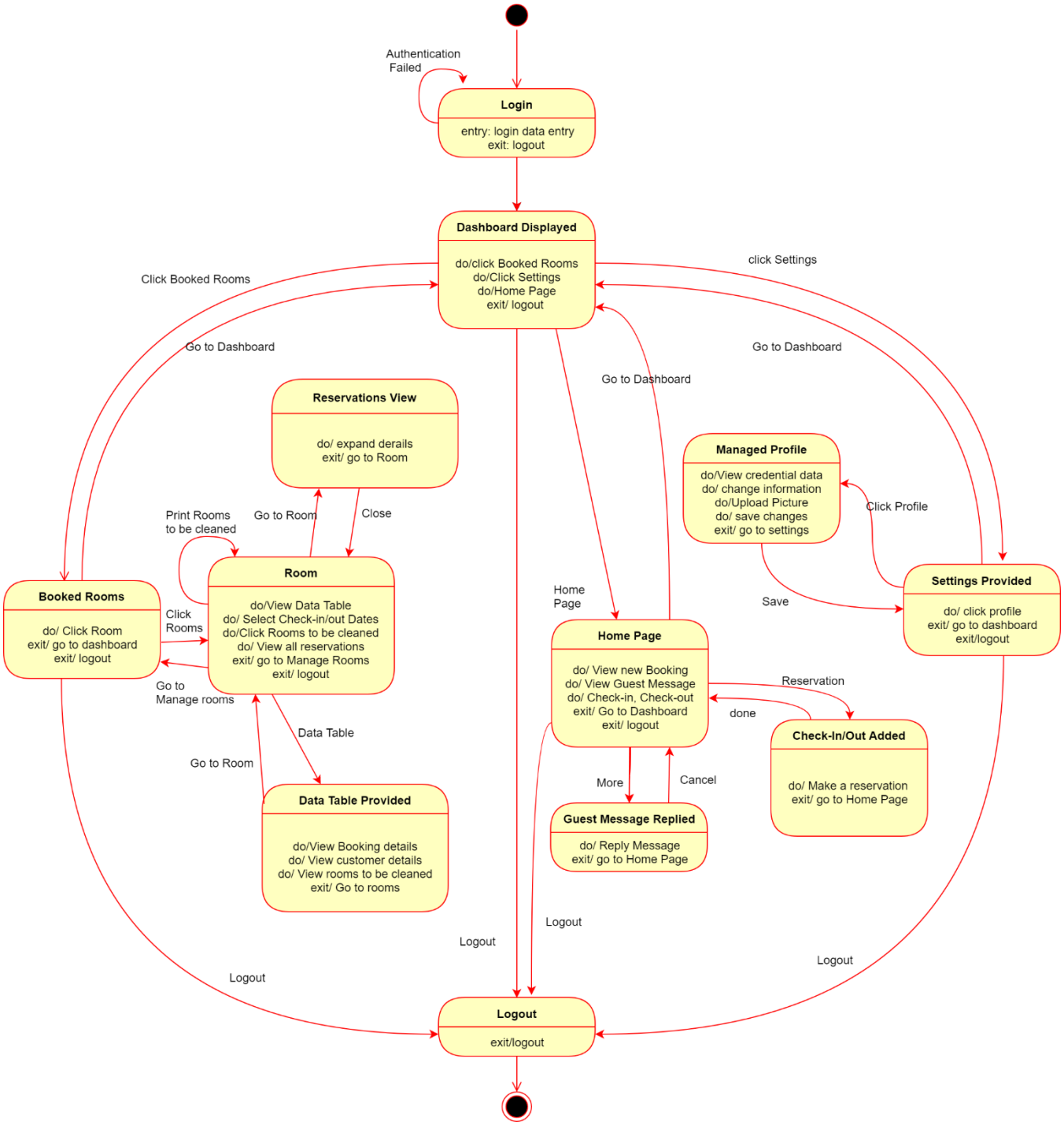
5.1.3 State Diagrams

5.1.3.1 State Diagram – Admin



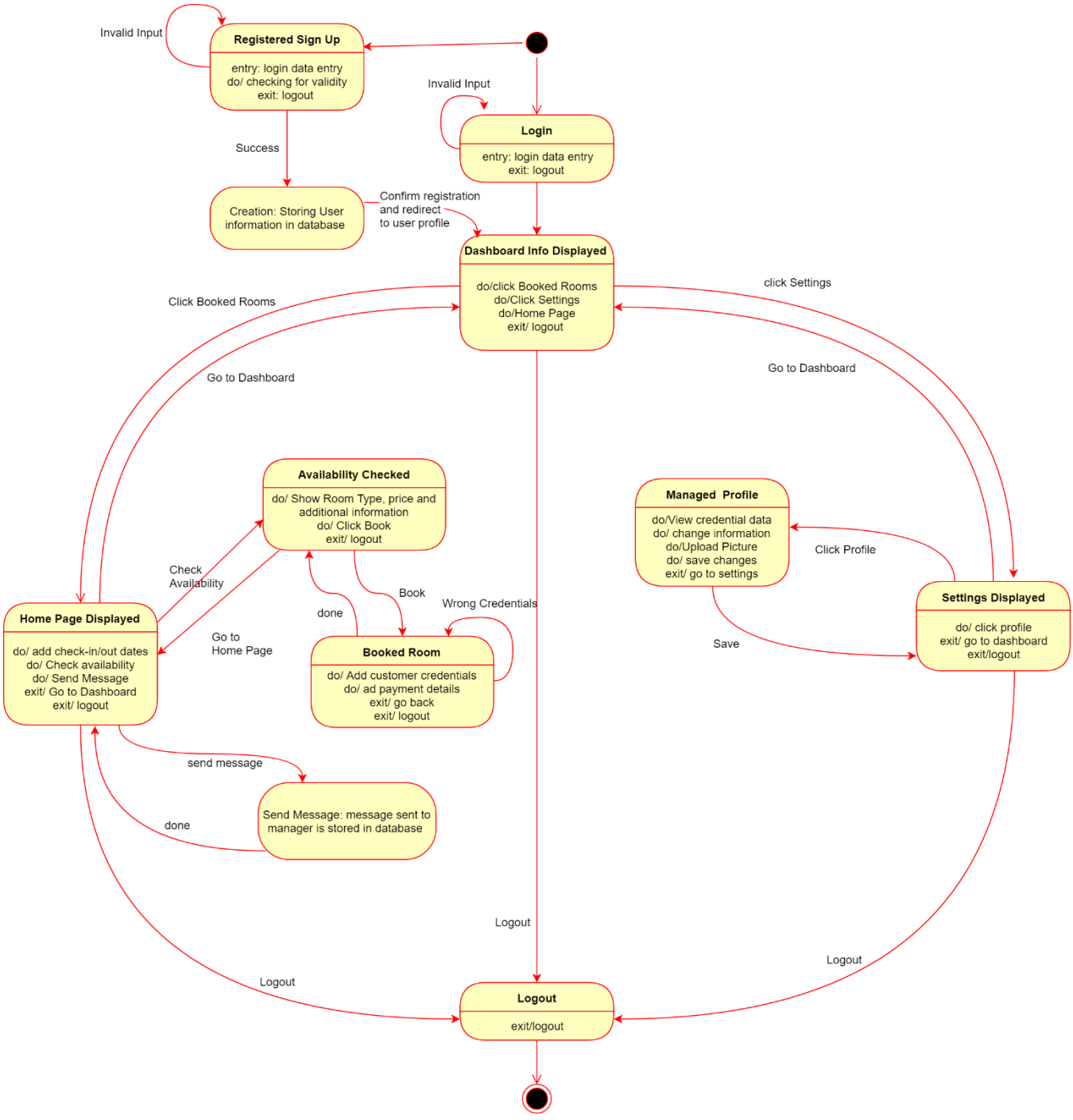
SD - UC 1, 2, 3, 4, 5, 6, 7, 9, 13

5.1.3.2 State Diagram – Receptionist



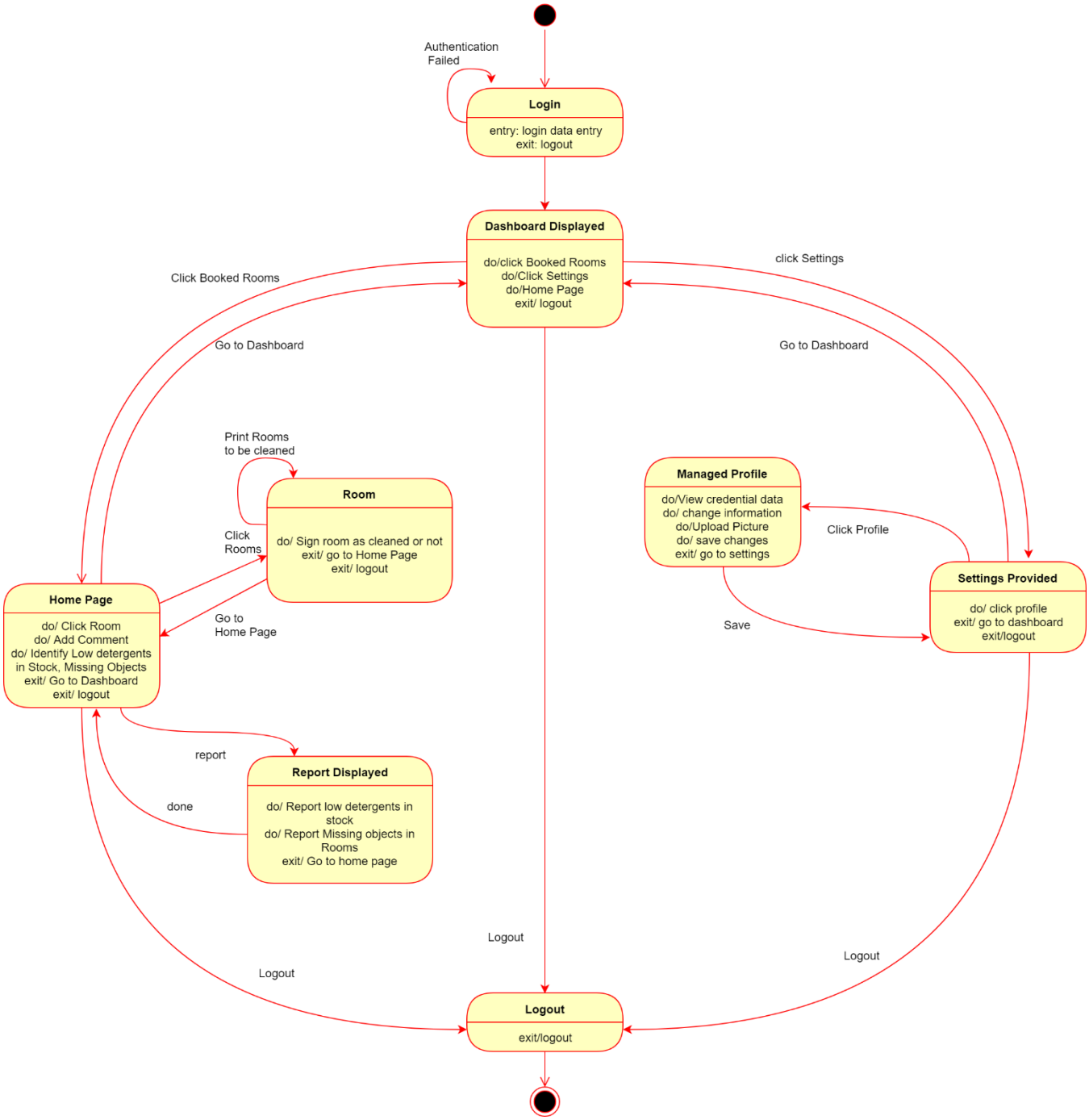
SD - UC 1, 2, 9, 13

5.1.3.3 State Diagram – Customer



SD - UC 1, 2, 10, 11, 12, 13

5.1.3.4 State Diagram – Cleaner

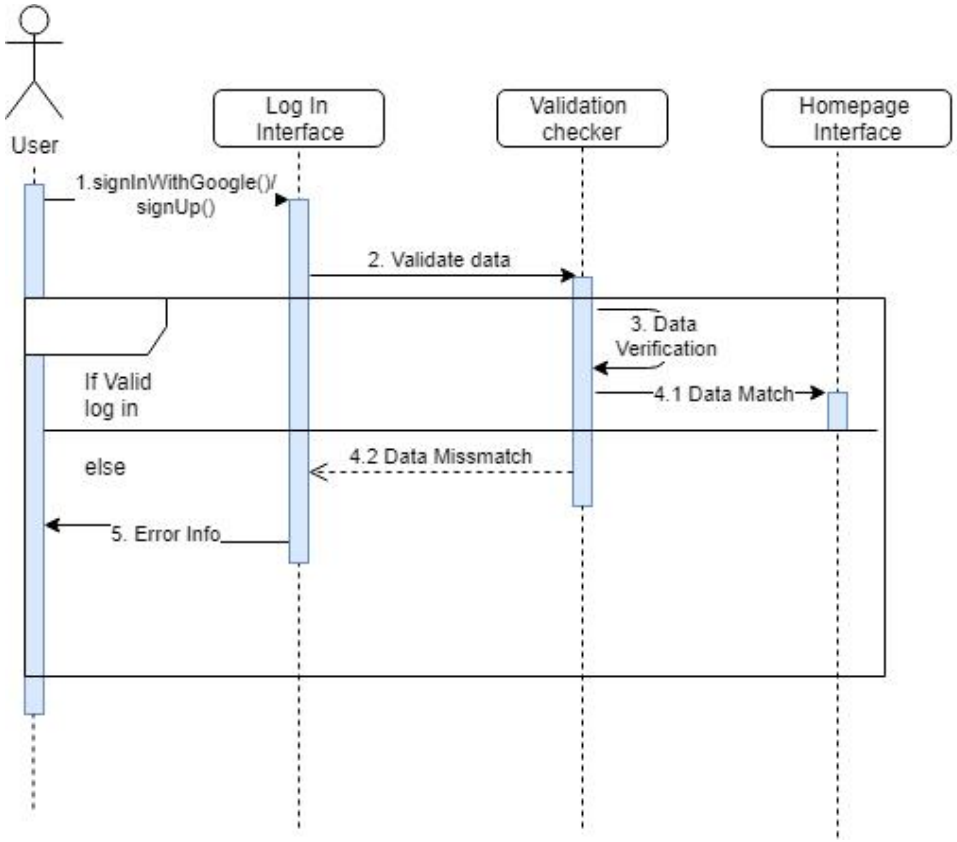


SD - UC 1, 2, 8, 13

5.1.4 Interaction Diagram

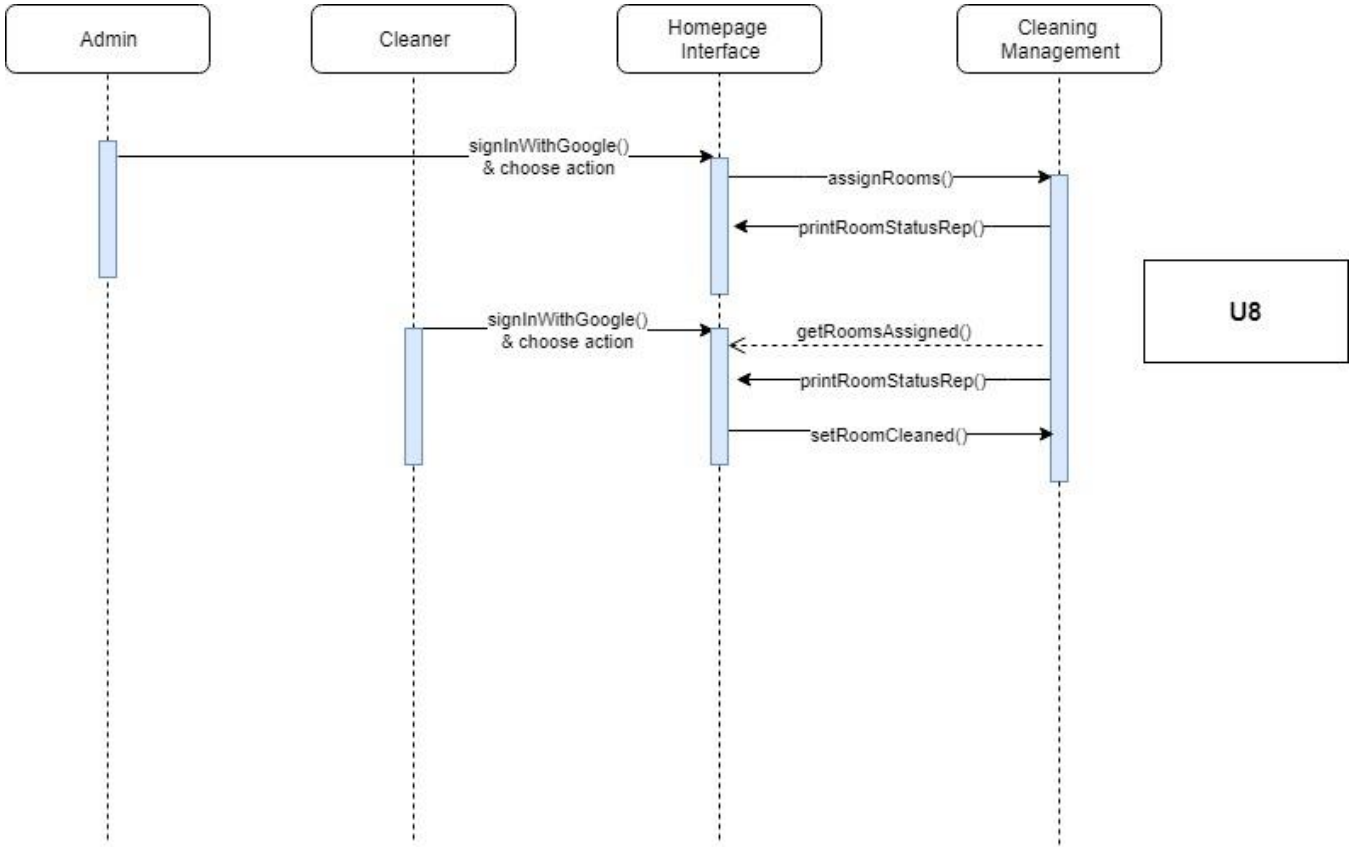
5.1.4.1 Sequence Diagram

5.1.4.1.1 Sequence Diagram – Authentication

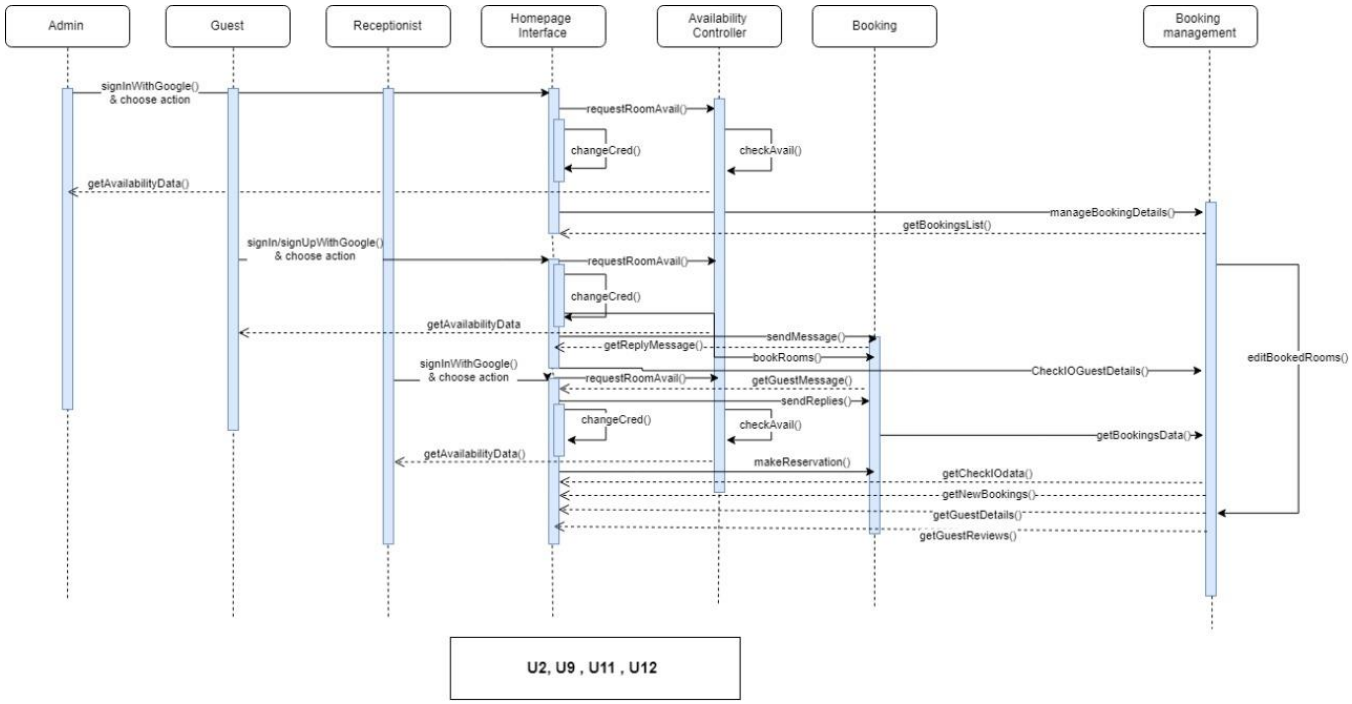


U1 , U10

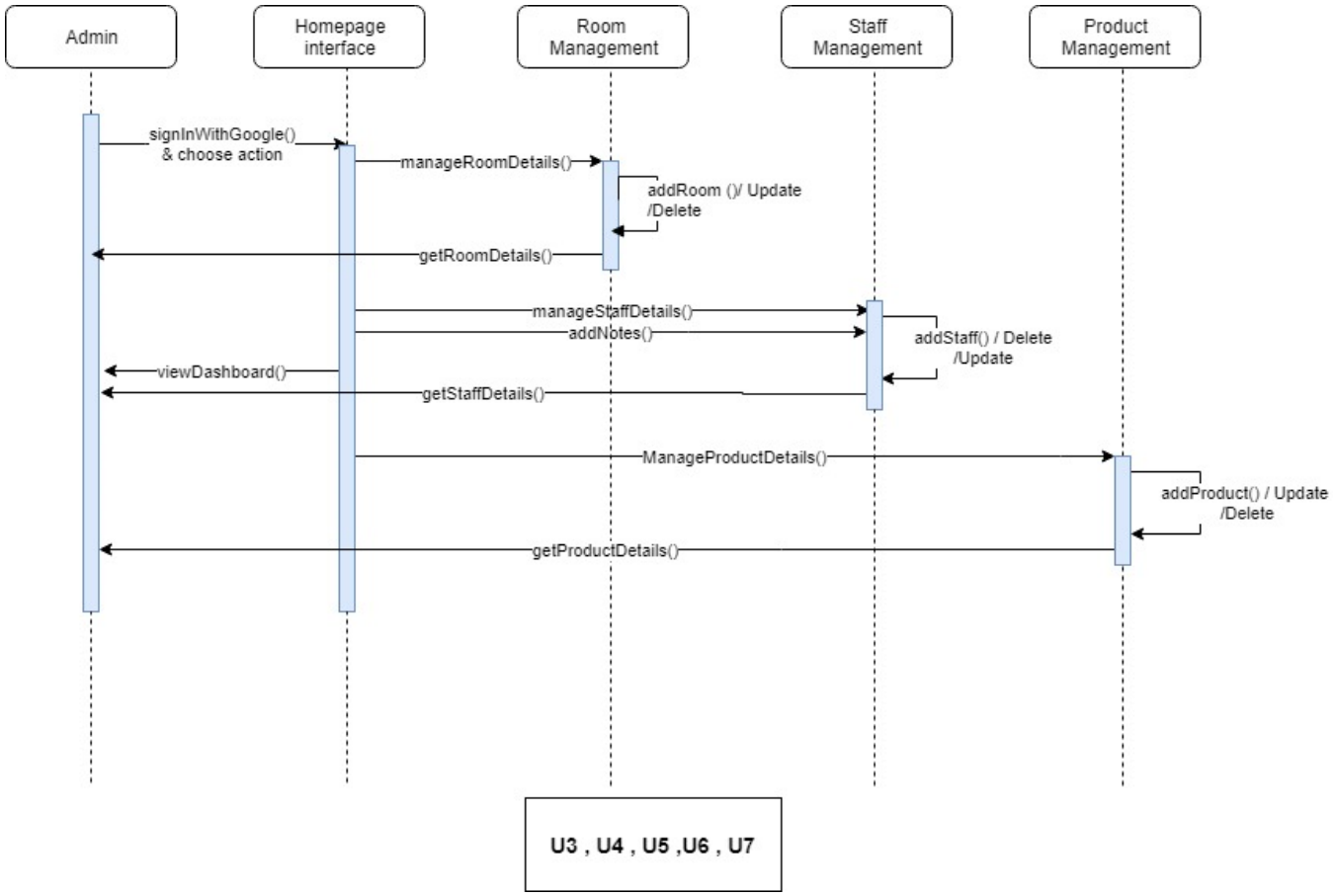
5.1.4.1.2 Sequence Diagram – Cleaning Process



5.1.4.1.3 Sequence Diagram – Booking Process

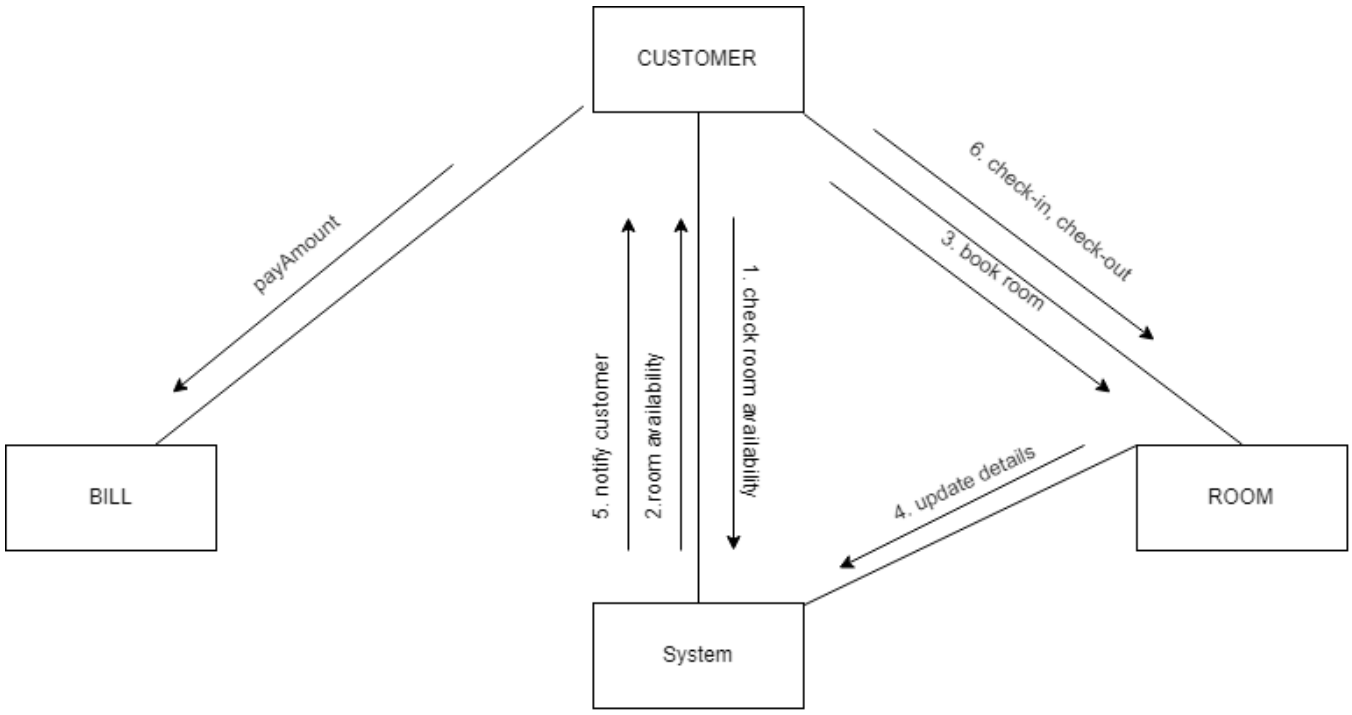


5.1.4.1.4 Sequence Diagram – Hotel Management

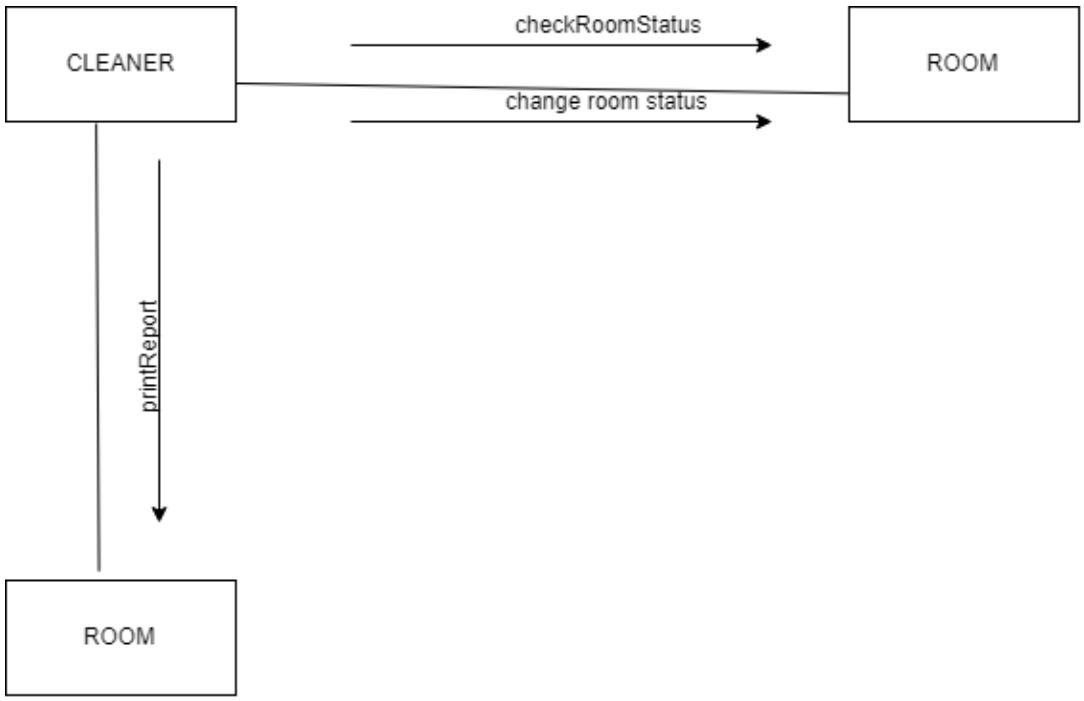


5.1.4.2 Collaboration Diagram

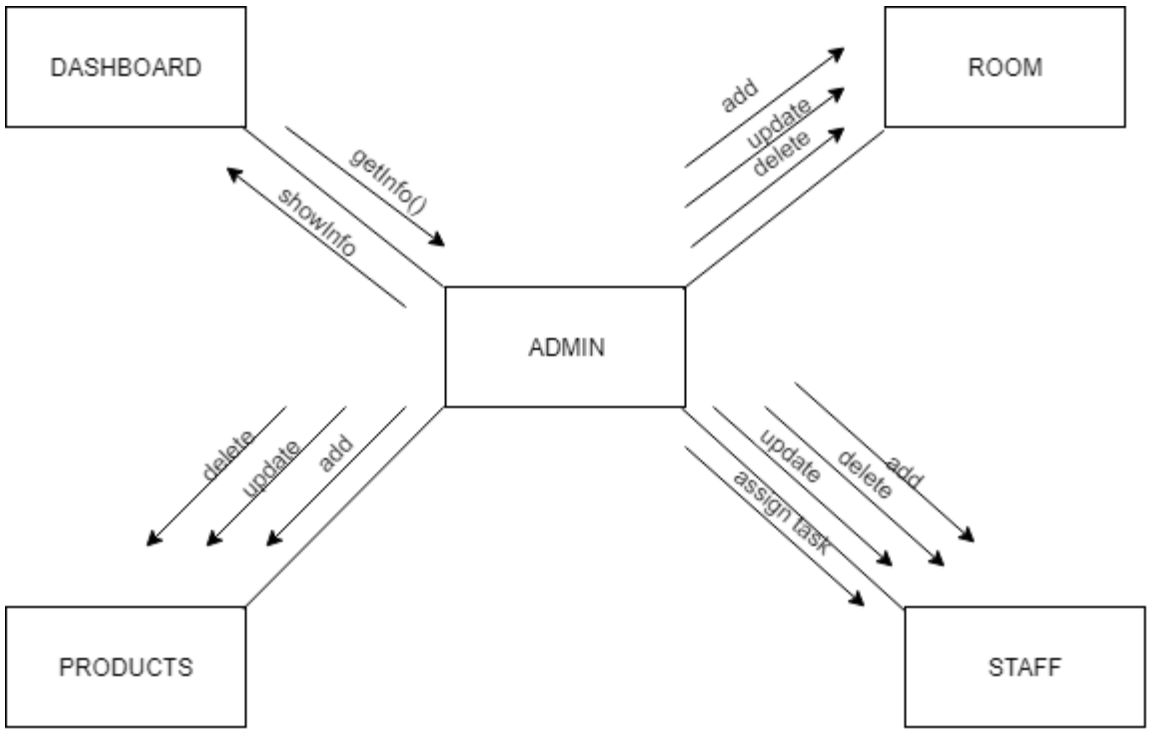
Bill-Customer-System



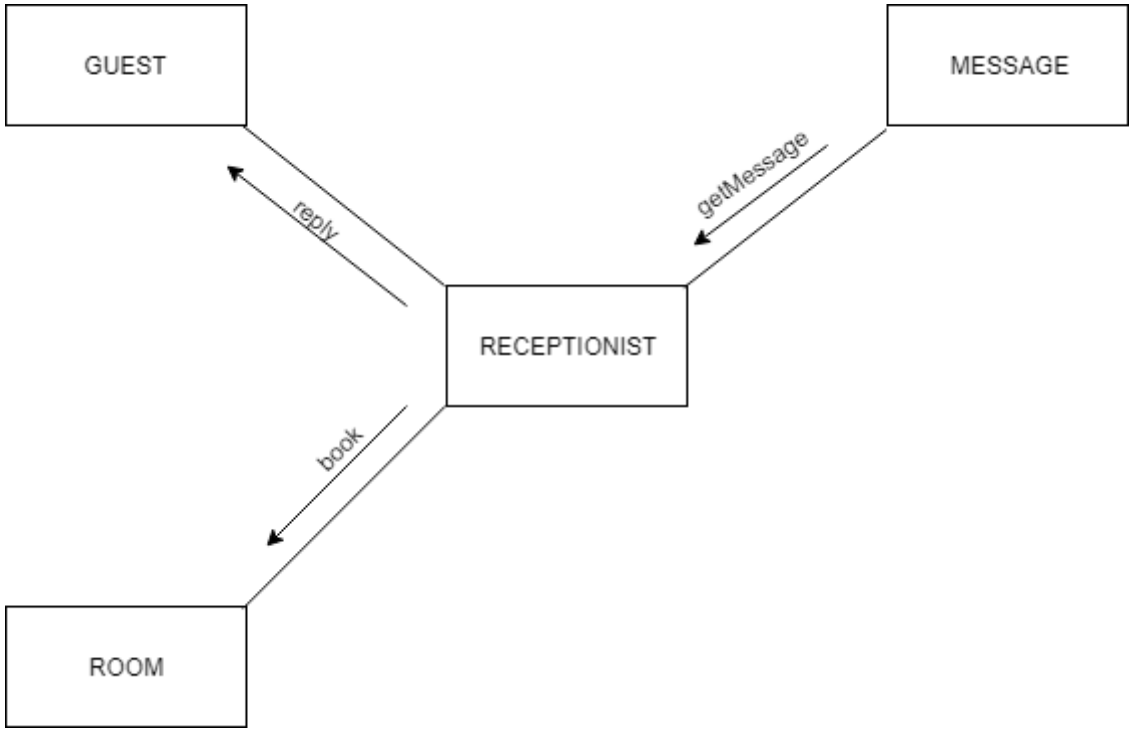
Cleaner Room Report



Dashboard – Admin – Room – Product – Staff



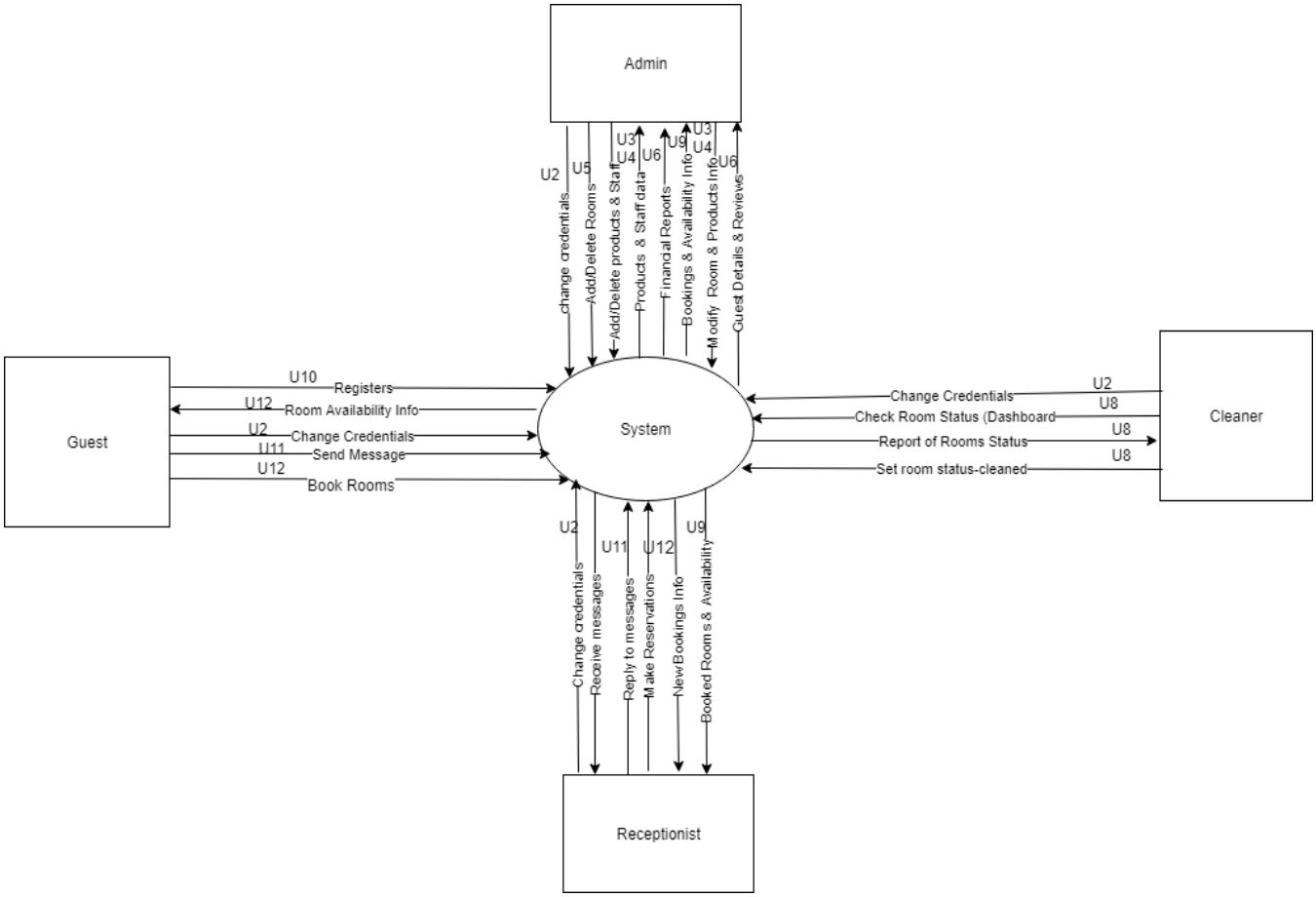
Guest – Message – Receptionist – Room



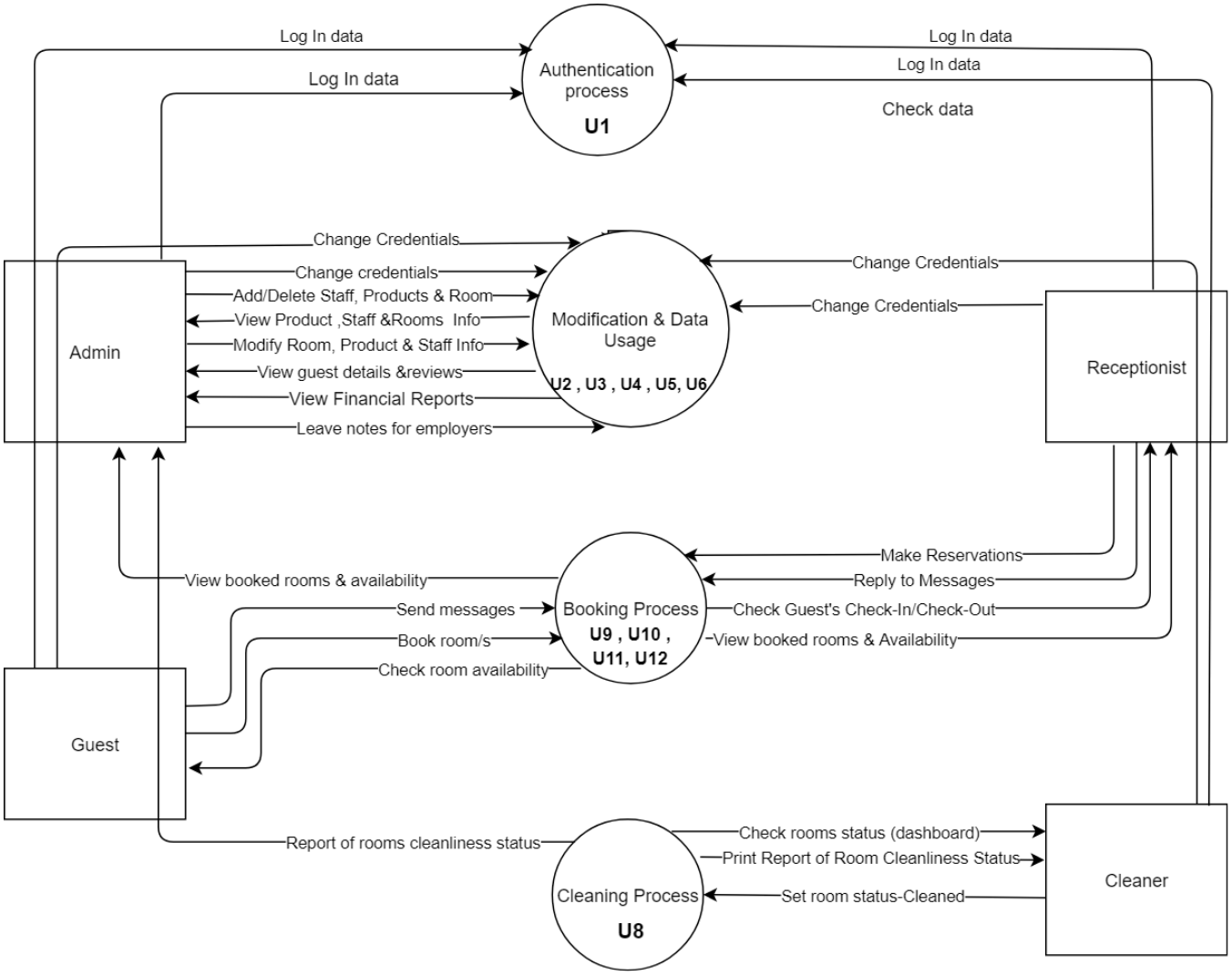
6. DFD & ERD

6.1.1 Data Flow Diagram (DFD)

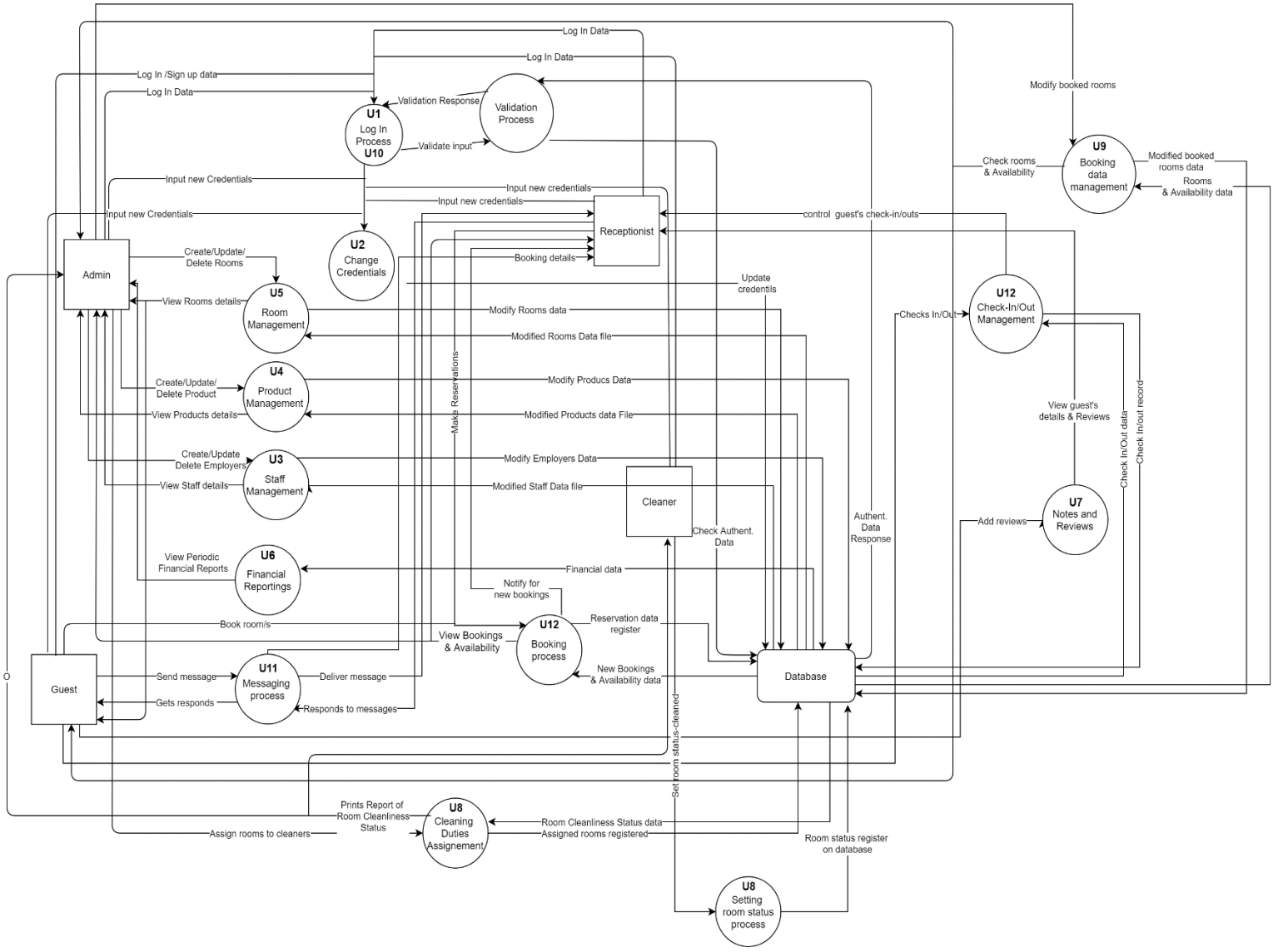
6.1.1.1 **DFD – Level 0**



6.1.1.2 **DFD – Level 1**

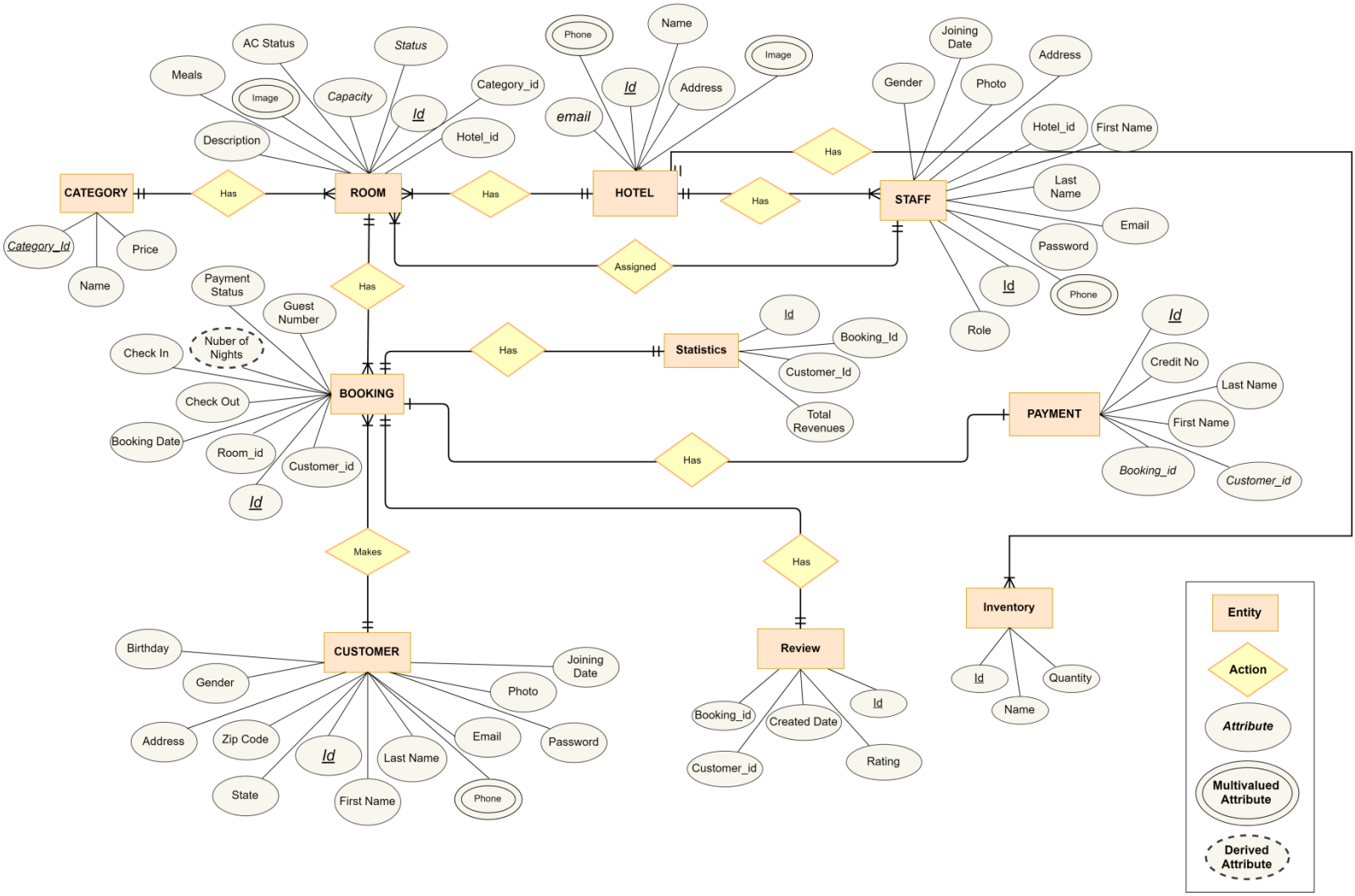


6.1.1.3 **DFD – Level 2**

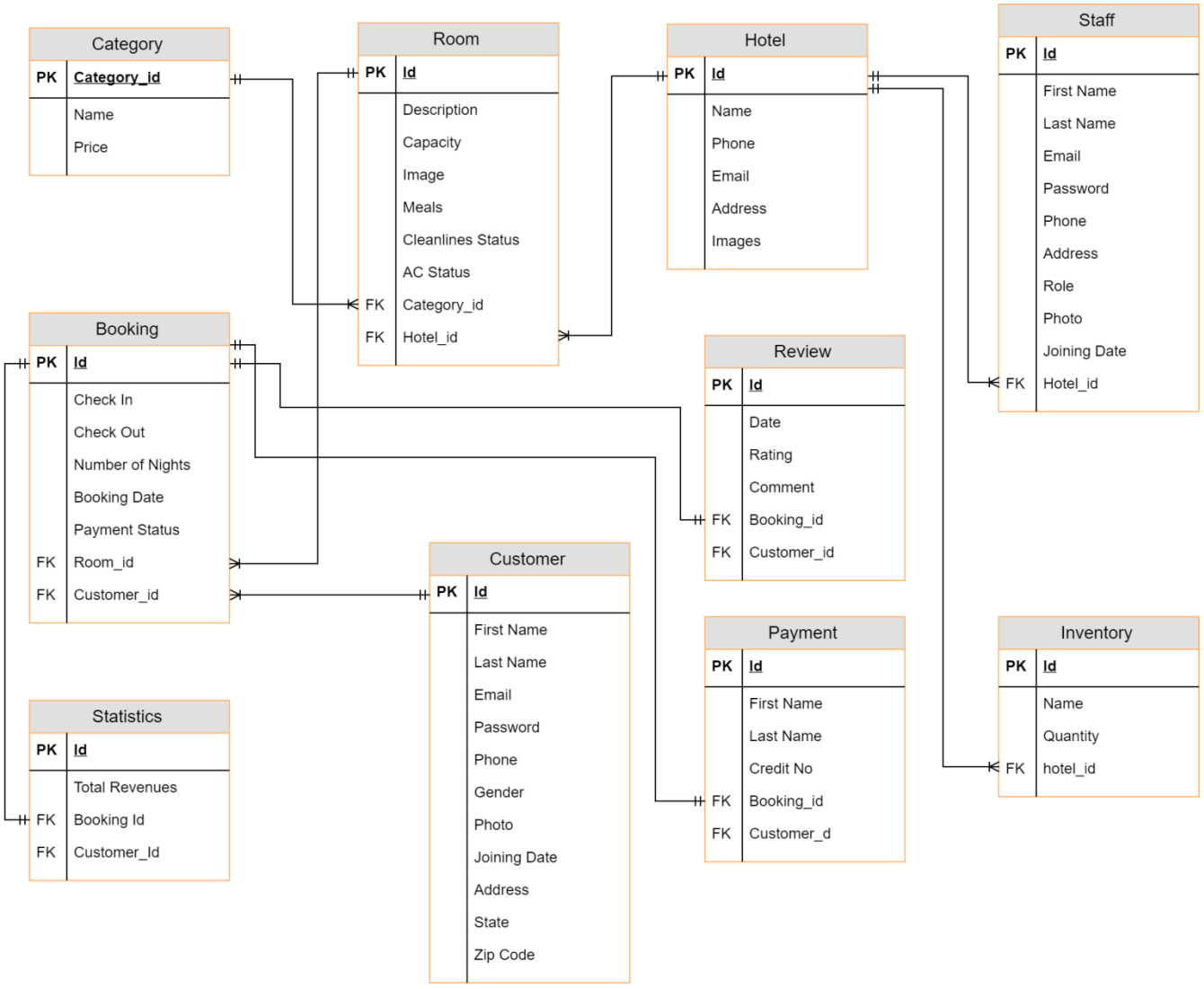


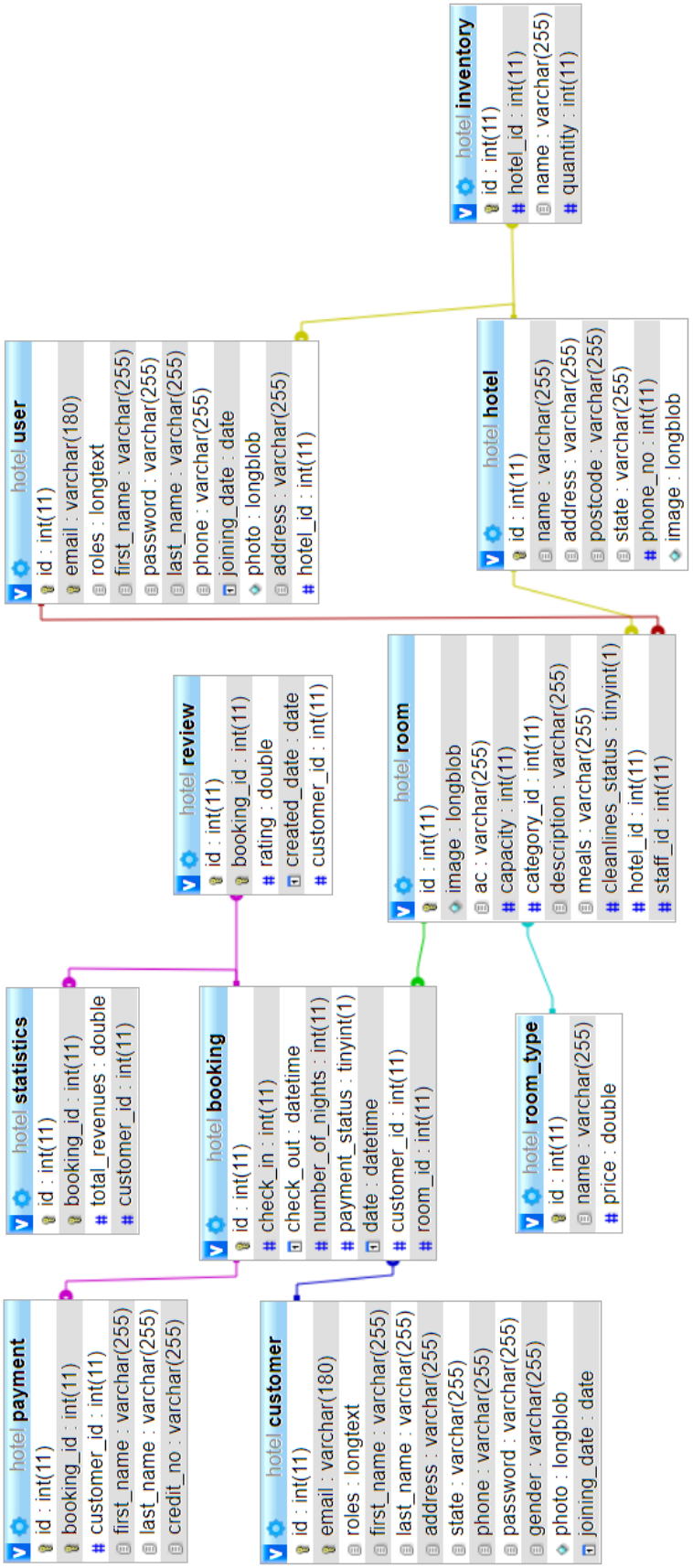
6.1.2 Entity Relation

6.1.2.1 ER - Diagram with Attributes



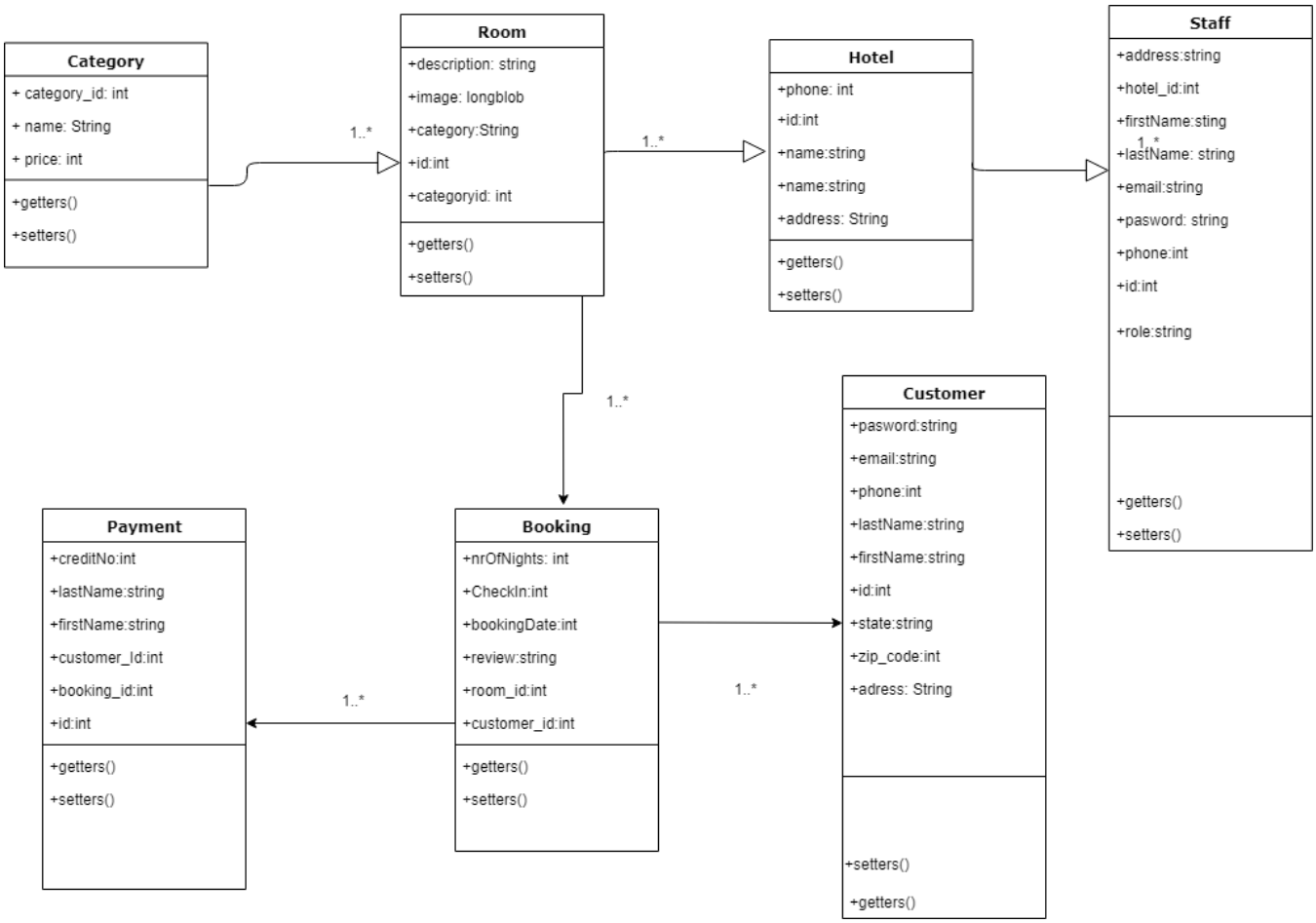
6.1.2.2 **Database Schema Design**



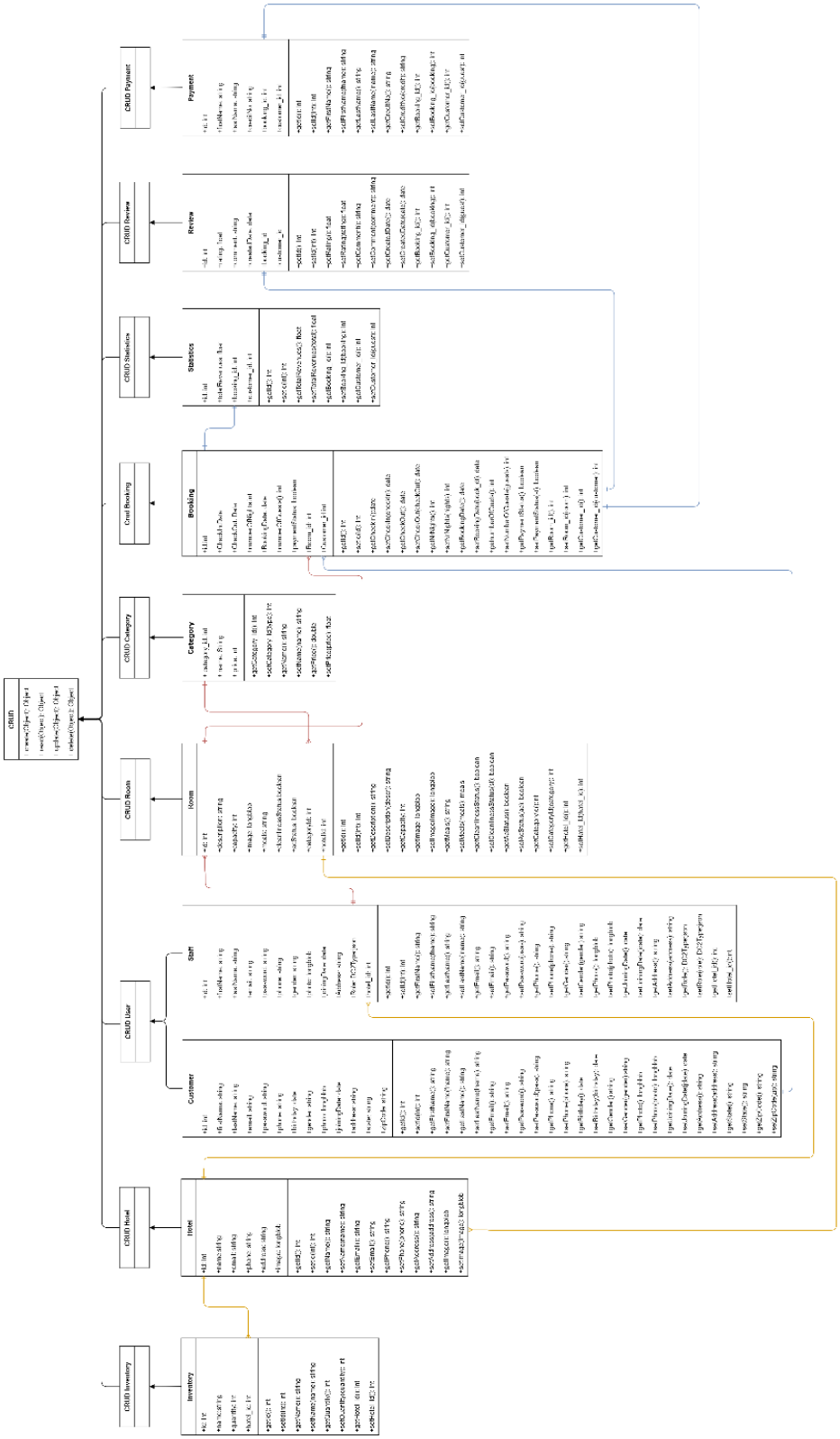

hotel inventory
id : int(11)
hotel_id : int(11)
name : varchar(255)
quantity : int(11)
hotel user
id : int(11)
email : varchar(180)
roles : longtext
first_name : varchar(255)
password : varchar(255)
last_name : varchar(255)
phone : varchar(255)
joining_date : date
photo : longblob
address : varchar(255)
hotel_id : int(11)
hotel hotel
id : int(11)
name : varchar(255)
address : varchar(255)
postcode : varchar(255)
state : varchar(255)
phone_no : int(11)
image : longblob
hotel review
id : int(11)
booking_id : int(11)
rating : double
created_date : date
customer_id : int(11)
hotel room
id : int(11)
image : longblob
ac : varchar(255)
capacity : int(11)
category_id : int(11)
description : varchar(255)
meals : varchar(255)
cleanlines_status : tinyint(1)
hotel_id : int(11)
staff_id : int(11)
hotel statistics
id : int(11)
booking_id : int(11)
total_revenues : double
customer_id : int(11)
hotel booking
id : int(11)
check_in : int(11)
check_out : datetime
number_of_nights : int(11)
payment_status : tinyint(1)
date : datetime
customer_id : int(11)
room_id : int(11)
hotel room_type
id : int(11)
name : varchar(255)
price : double
hotel payment
id : int(11)
booking_id : int(11)
customer_id : int(11)
first_name : varchar(255)
last_name : varchar(255)
credit_no : varchar(255)
hotel customer
id : int(11)
email : varchar(180)
roles : longtext
first_name : varchar(255)
last_name : varchar(255)
address : varchar(255)
state : varchar(255)
phone : varchar(255)
password : varchar(255)
gender : varchar(255)
photo : longblob
joining_date : date

7. Structural Diagrams

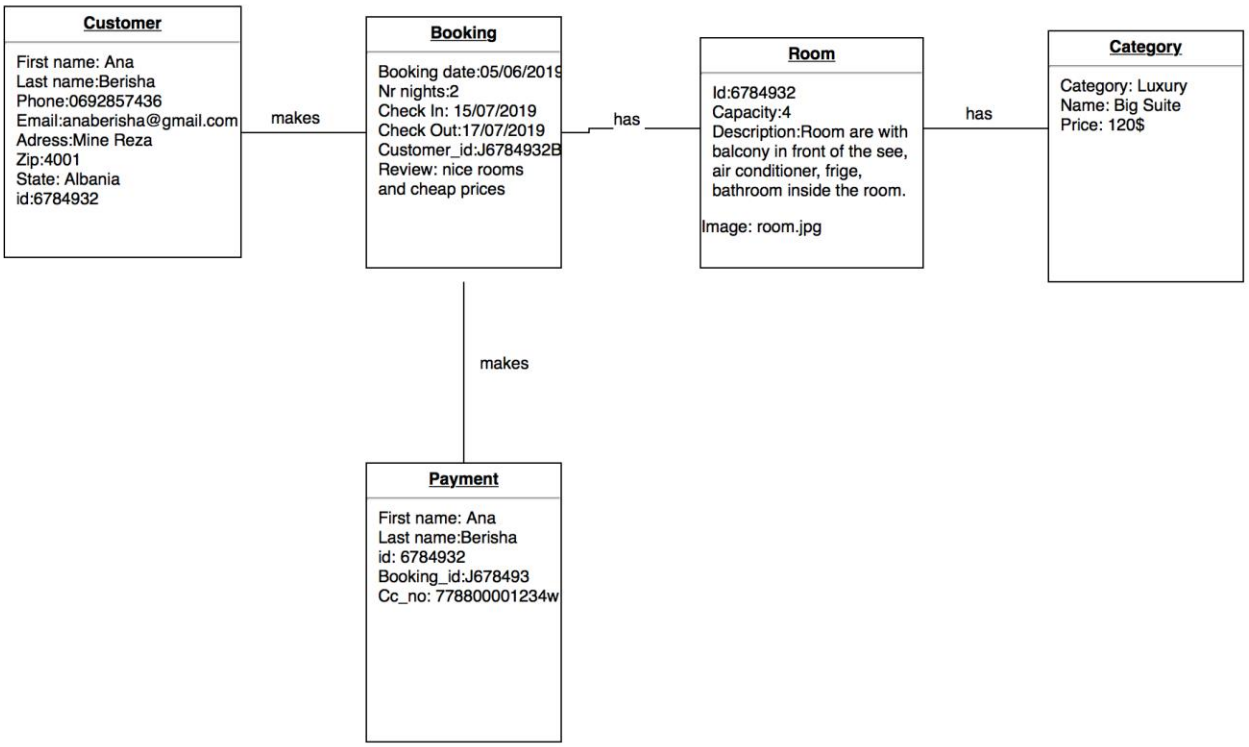
7.1.1 Class Diagram



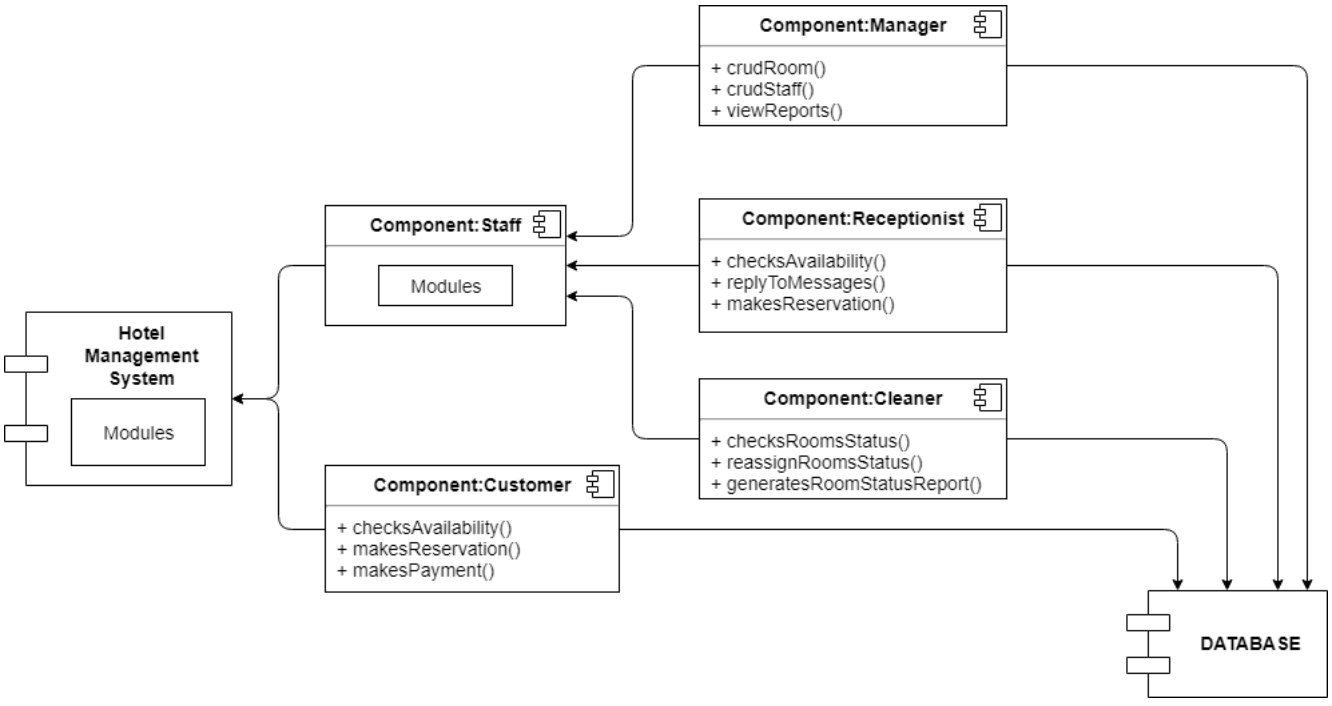
7.1.2 CRUD Class Diagram

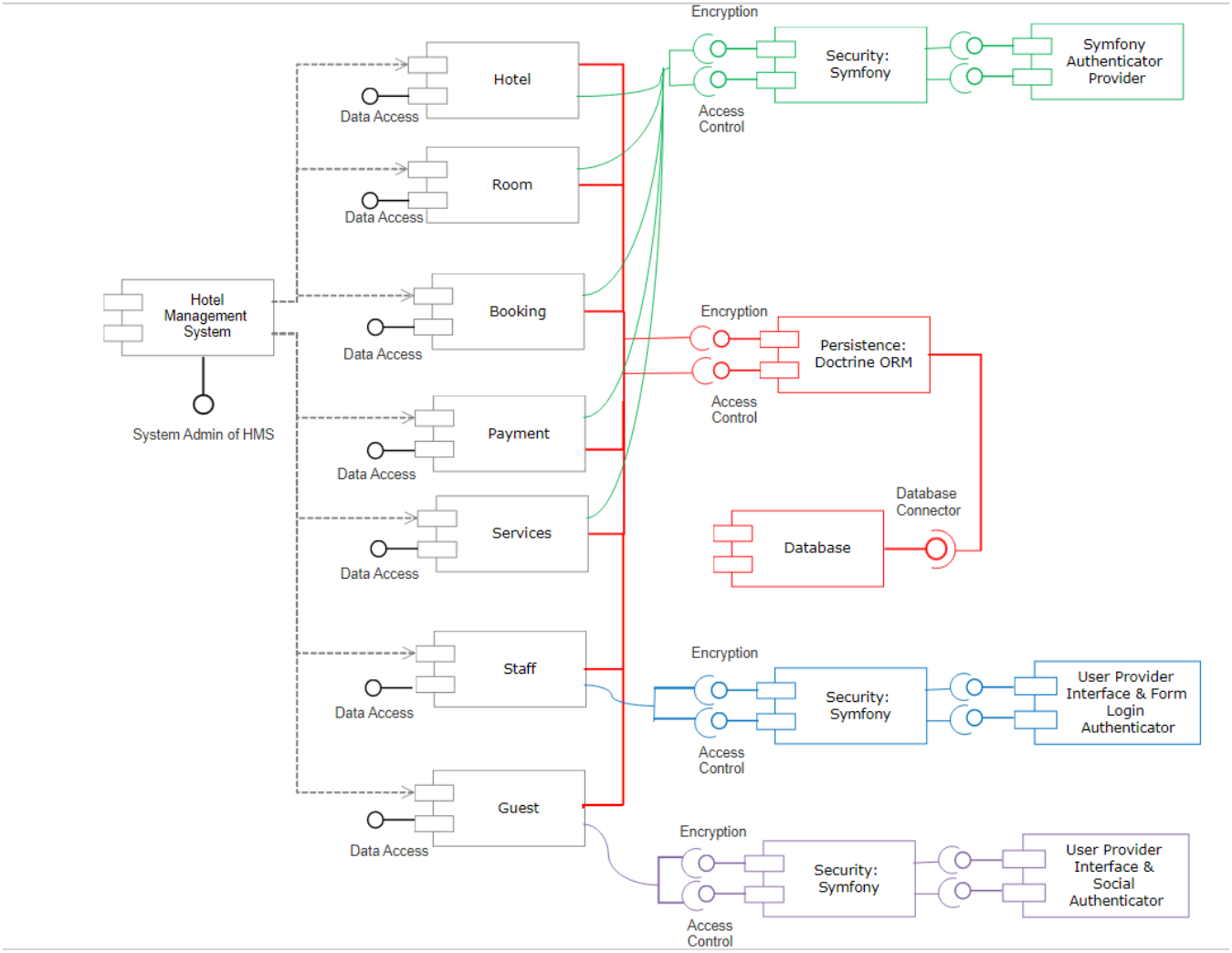


7.1.3 Object Diagram

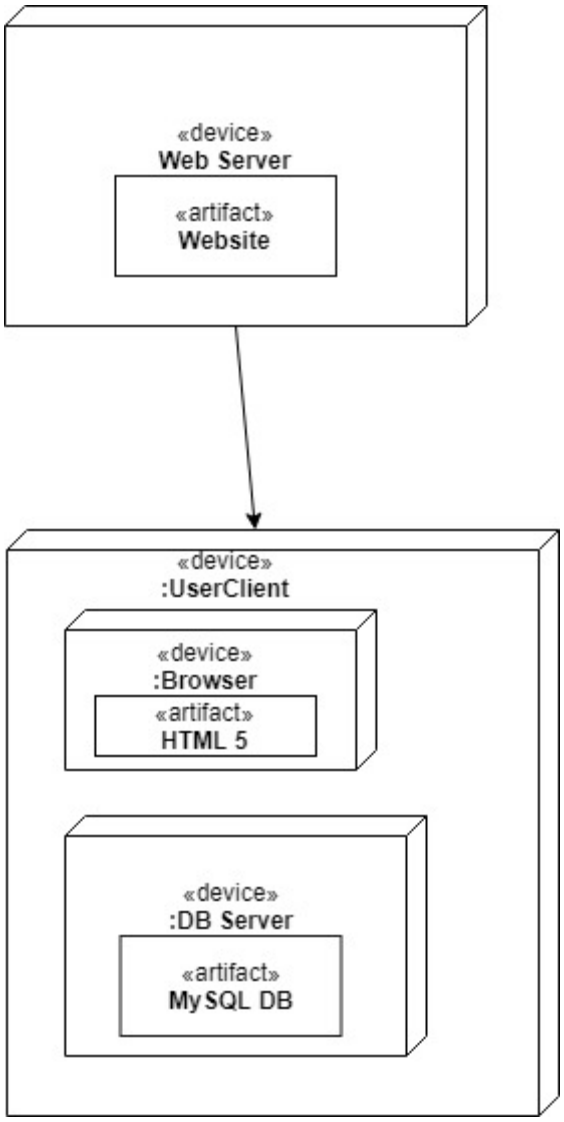


7.1.4 Component Diagram




Encryption
Security: Symfony
Symfony Authenticator Provider
Access Control
Hotel
Data Access
Room
Data Access
Hotel Management System
Booking
Data Access
Encryption
Persistence: Doctrine ORM
Access Control
System Admin of HMS
Payment
Data Access
Database Connector
Database
Services
Data Access
Staff
Data Access
Encryption
Security: Symfony
User Provider Interface & Form Login Authenticator
Access Control
Guest
Data Access
Encryption
Security: Symfony
User Provider Interface & Social Authenticator
Access Control

7.1.5 Deployment Diagram



8. Implementation

8.1 For Programmers

Our main objective since the beginning of this project was in building a web application which aims to facilitate the management system of a hotel. It will keep track of hotel reservations, inventory management, rooms to be cleaned and so much more. Using cutting-edge technologies which are user friendly in usage and robust in operation, aiming to facilitate the management system of an existing hotel.

Enera Hotel Management System is developed using Symfony 4.2 Framework with MVC architecture. By using the two most outstanding technological benefits of symphony (Bundles and Components) it was easier than ever to eliminate strict dependencies in the architecture.

For the templating part we used the modern template engine for php (Twig). Even why PHP is a template engine itself, it did not have a very high evolution compared with the other template engines in recent years. To support this statement based on the actual facts as well PHP template lacks some features that the modern template must have:

- **Concise**: The php language is verbose and becomes ridiculously verbose when it comes to output escaping

```
<?php echo $var ?>
<?php echo htmlspecialchars($var, ENT_QUOTES, 'UTF-8') ?>
```

In comparison, Twig has a very concise syntax, which make templates more readable:

```
{{ var }}
{{ var|escape }}
{{ var|e }}          {# shortcut to escape a variable #}
```

- **Template oriented syntax**: Twig has shortcuts for common patterns, like having a default text displayed when you iterate over an empty array:

```
{% for user in users %}
    * {{ user.name }}
{% else %}
    No users have been found.
{% endfor %}
```

- **Full Featured**: Twig supports everything you need to build powerful templates with ease: multiple inheritance, blocks, automatic output-escaping, and much more:

```
{% extends "layout.html" %}
{% block content %}
    Content of the page...
{% endblock %}
```

- **Easy to learn**: The syntax is easy to learn and has been optimized to allow web designers to get their job done fast without getting in their way.

Of course, PHP is also the language for which you can find the more template engine projects. But most of them do not embrace web development best practices yet:

- **Extensibility**: Twig is flexible enough for all your needs, even the most complex ones. Thanks to an open architecture, you can implement your own language constructs (tags, filters, functions, and even operators) to create your very own DSL.
- **Unit tested**: Twig is fully unit-tested. The library is stable and ready to be used in large projects.
- **Documented**: Twig is fully documented, with a dedicated online book, and of course a full API documentation

- **Secure**: When it comes to security, Twig has some unique features:

- *Automatic output escaping*: To be on the safe side, you can enable automatic output escaping globally or for a block of code:

```
{% autoescape "html" %}
    {{ var }}
    {{ var|raw }}      {# var won't be escaped #}
    {{ var|escape }}   {# var won't be doubled-escaped #}
{% endautoescape %}
```

- *Sandboxing*: Twig can evaluate any template in a sandbox environment where the user has access to a limited set of tags, filters, and object methods defined by the developer. Sandboxing can be enabled globally or locally for just some templates:

```
{{ include('page.html', sandboxed = true) }}
```

- **Clean Error Messages**: Whenever you have a syntax problem within a template, Twig outputs a helpful message with the filename and the line number where the problem occurred. It eases the debugging a lot.

- **Fast**: One of the goals of Twig is to be as fast as possible. To achieve the best speed possible, Twig compiles templates down to plain optimized PHP code. The overhead compared to regular PHP code was reduced to the very minimum.

MySql Database is used for storing data such as Users, Reservations, Customers, Room details etc. We have also used Doctrine ORM for database storage and object mapping. It provides persistence services and related functionality. Below we have shown a simple example of how it is implemented:

```
$user = new User();
$user->name = "Jona";
$user->password = "Cara";
$entityManager->persist($user);
$entityManager->flush();
$output = "The user with id $user->id has been saved.";
```

Doctrine can generate object classes from an existing database, and the programmer can then specify relations and add custom functionality to the generated classes. There is no need to generate or maintain complex XML database schemas, as seen in many other frameworks.

These three are the main components of our project. We have used other bundles such as **Oauth2** for Google Authentication, or UserProviderInterface for login authentication. Other features such as Bcrypt password encoder or Cross-Site Request Forgery (CSRF) protection is being used.

8.2 For normal Users

A detailed description that can be understand by every normal user for the implementation of each feature is given below:

The Main page consists of basic information about the hotel such as about, features, Contact information, Login and Register redirecting. The change on the Home Page when redirecting in different tabs is done in real time.

Register: This part is only for the customers which want to reserve a room for a specific time in our hotel. They can get registered using their Google Account or provide their information manually and to Sign up. Every user detail is encrypted using the safest and latest protection technology and no data can be hacked from their account.

Login: Here every possible user including Admin, Receptionist, Cleaner, and Customer can be logged in using their corresponding credentials. Again the password is secured and the authentication is done using a highly secured password protection algorithm (Bcrypt).

After a successful login or registration the user will be redirected to their corresponding dashboard page based on their access. If they have already stored a picture for their account in the database their account will show that picture to the corresponding user. Otherwise if no picture is provided the system will automatically provide a profile picture based on the User Gender (Female profile picture for females and male profile picture for males). From the dashboard they can access every possible action that is allowed based on their roles in the system.
Example:

Cleaner will be allowed to check only for the rooms that are cleaned or not cleaned and after cleaning a specific room he/she can now check the room as cleaned from the system. The cleaner can also view and change his/her credentials.

Customer will be allowed to check Room availability for specific dates, book or cancel premade booked rooms, make payments and change his/her credentials.

Receptionist will be allowed to check Room availability for specific dates, book or cancel premade booked rooms for customers, set payment status, customer status (if booking is confirmed, customer is checked in or checked out etc.) check room status, change his/her credentials etc.

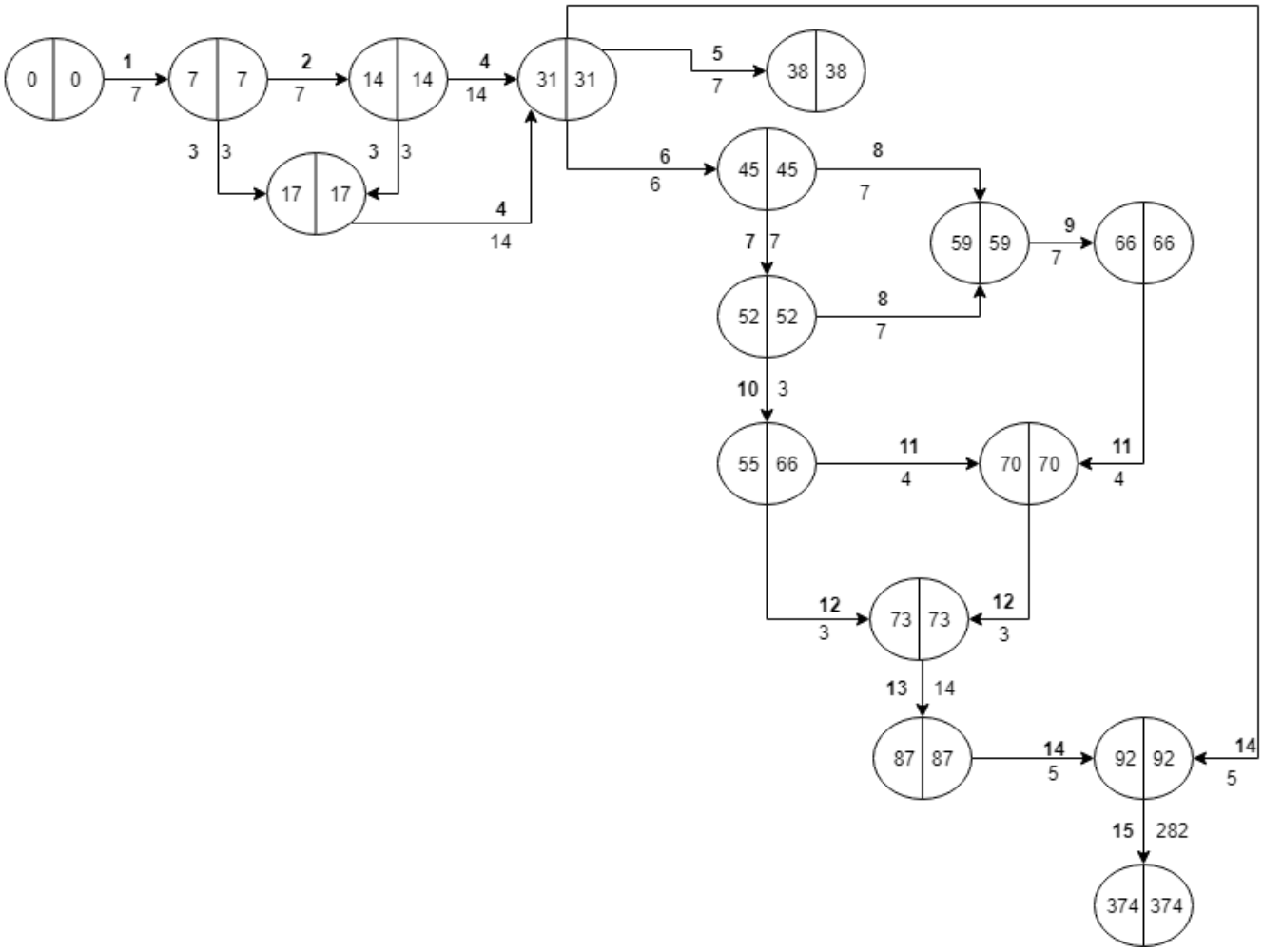
Admin will have access on everything that is provided by the system such as Creating, Reading, Updating, Deleting staff details, room details, product details etc. He can view statistics based on the bookings, earnings etc.

9. Project Planning

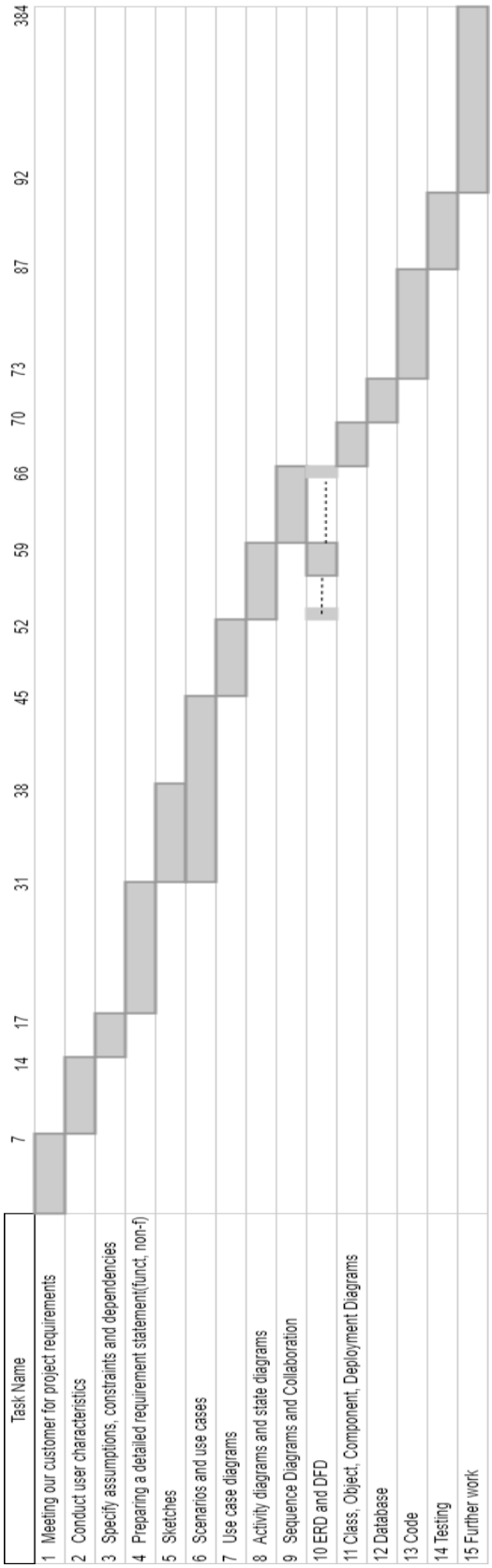
9.1 Task distribution

	ACTIVITY	DURATION (days)	DEPENDS ON	WORKED
1	Meeting our customer for project requirements	7		ALL MEMBERS
2	Conduct user characteristics	7	1	ALL MEMBERS
3	Specify assumptions, constraints and dependencies	3	1,2	ALL MEMBERS
4	Preparing a detailed requirement statement (functional, non-functional)	14	2,3	ALL MEMBERS
5	Sketches	7	4	JONA CARA
6	Scenarios and use cases	14	4	JONA CARA FIORI FIZI
7	Use Case Diagrams	7	6	HALISA QUKU
8	Activity Diagrams and State Diagrams	7	6,7	FIORI FIZI JONA CARA
9	Sequence Diagrams and Collaboration	7	8	HALISA QUKU FIORI FIZI
10	ERD and DFD	3	7	JONA CARA HALISA QUKU
11	Class, Object, Component, Deployment Diagrams	4	9,10	VALENTINA ROCI SHEJLA DOMNORI JONA CARA HALISA QUKU
12	Database	3	10,11	JONA CARA JORID SPAHO
13	Code	14	12	JONA CARA
14	Testing	5	13,4	JORID SPAHO
15	Further work	282	14	BY CUSTOMER CHOICE

9.2 Network diagram



9.3 Grant chart



Appendix

Appendix A

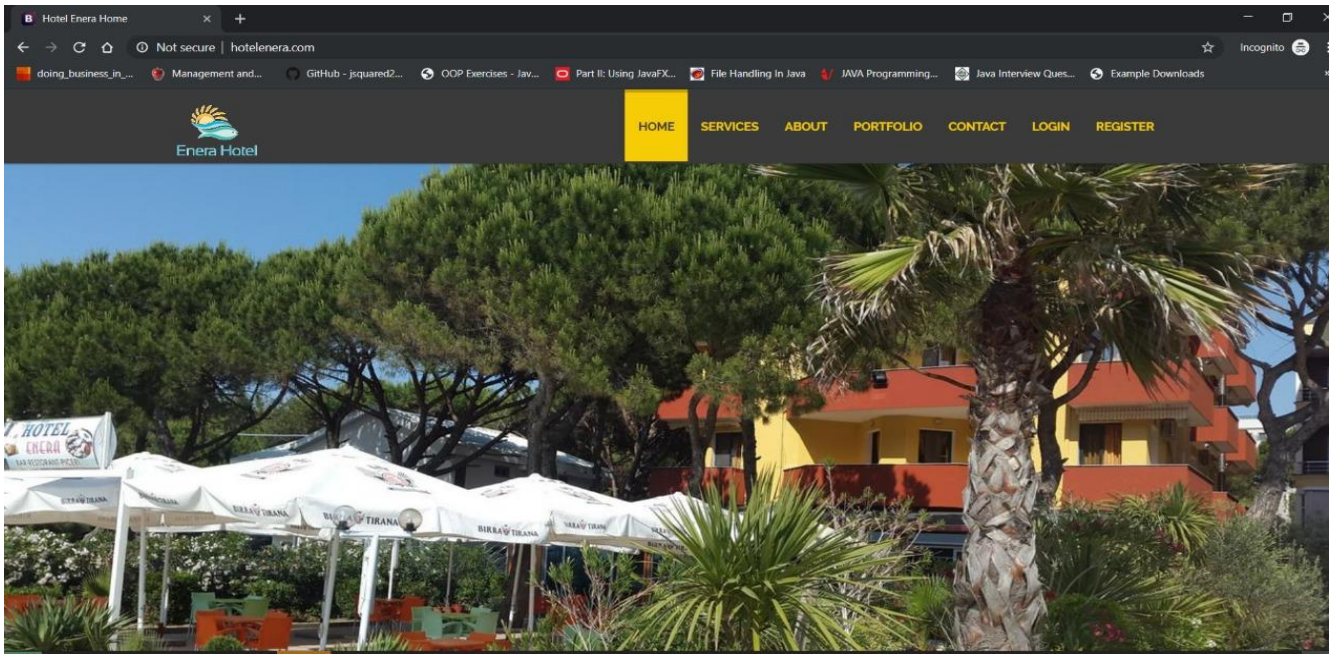
Definitions, Acronyms and Abbreviations

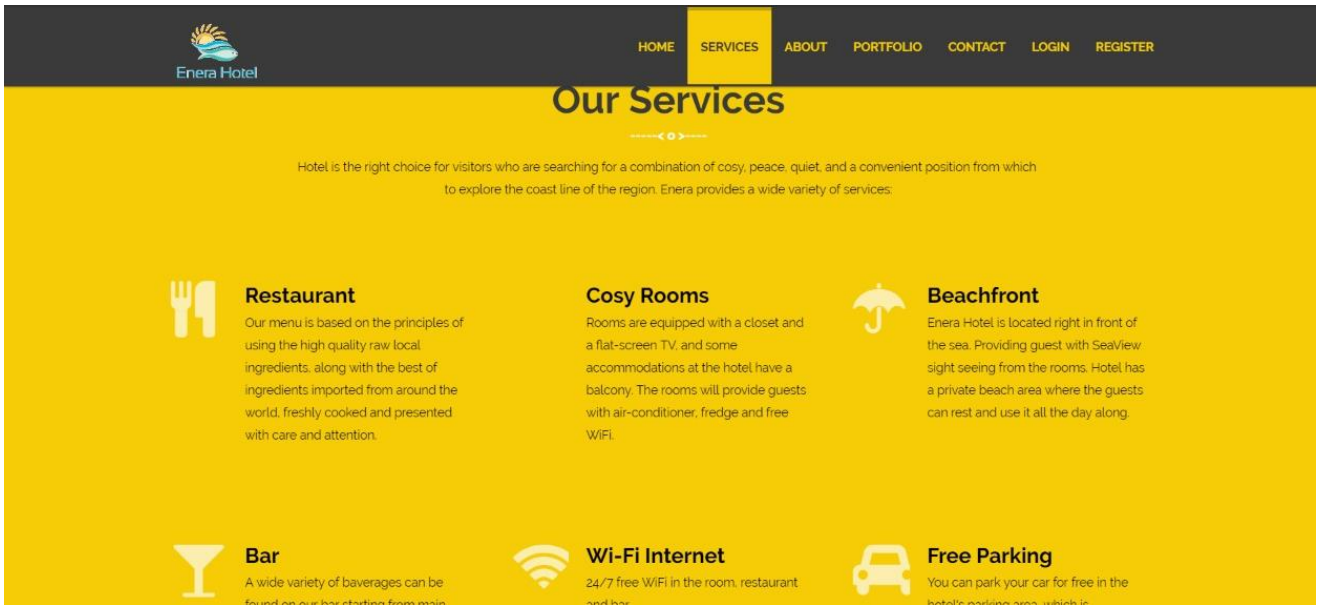
1. Easy Hotel Management System (EHMS)
2. Pc- personal computer
3. Req#- requirements
4. Date Rvwd- Date reviewed
5. CRUD- stands for create, read, update and delete. In this case CRUD refers to creating different instances of users.
6. SQL- means Structured Query Language which is the programming query language used to access update and remove database information through back-end programming
7. HTML- stands for Hyper Text Markup Language. HTML describes the structure of Web pages using markup. HTML elements are the building blocks of HTML pages. HTML elements are represented by tags.
8. HTTP- HTTP means HyperText Transfer Protocol. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.
9. CSRF-Cross-Site Request Forgery is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated.

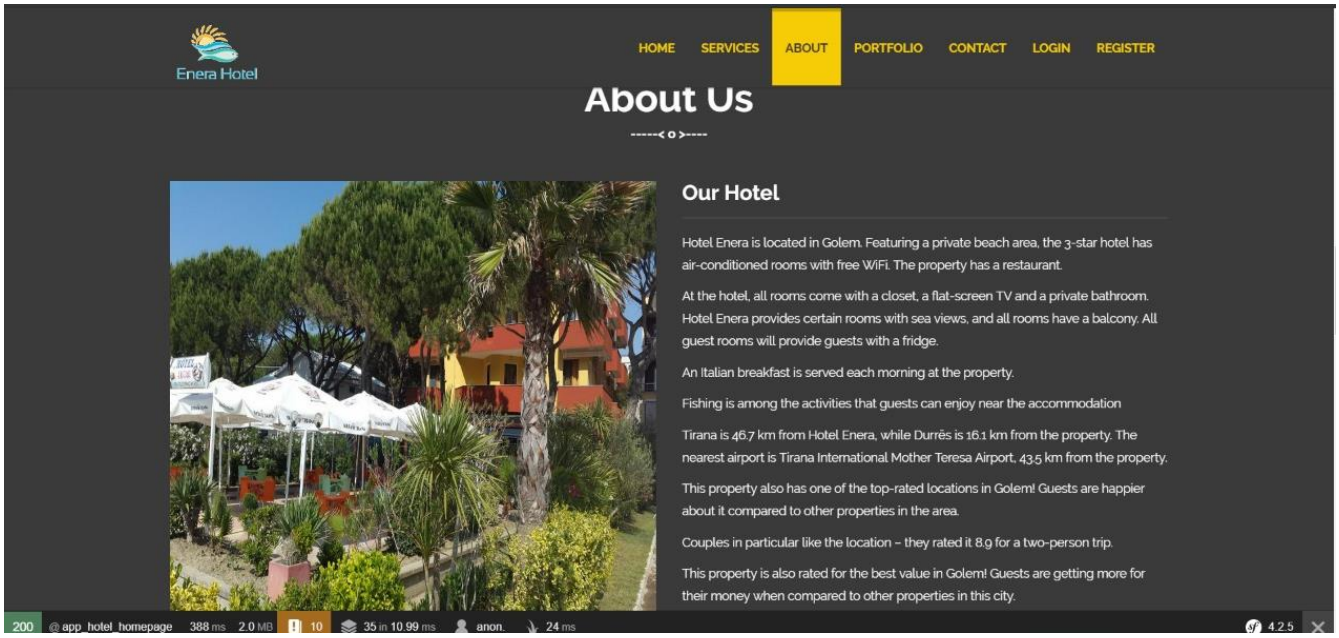
Appendix B

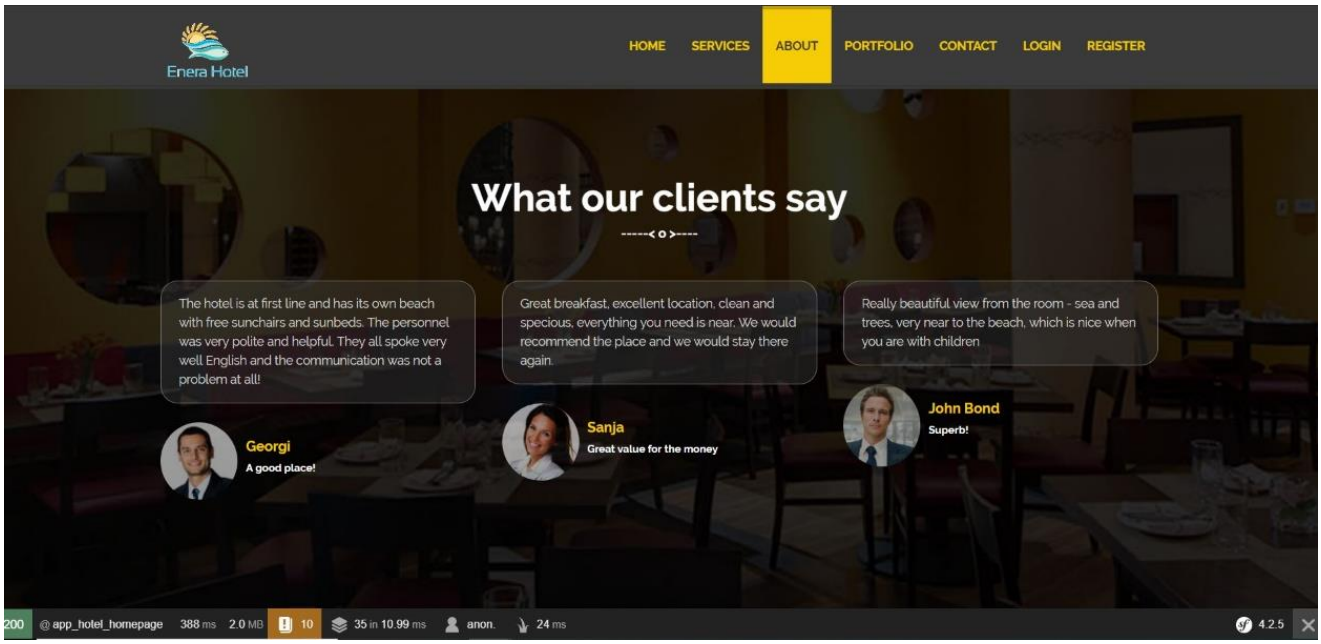
Screenshots & User Guide

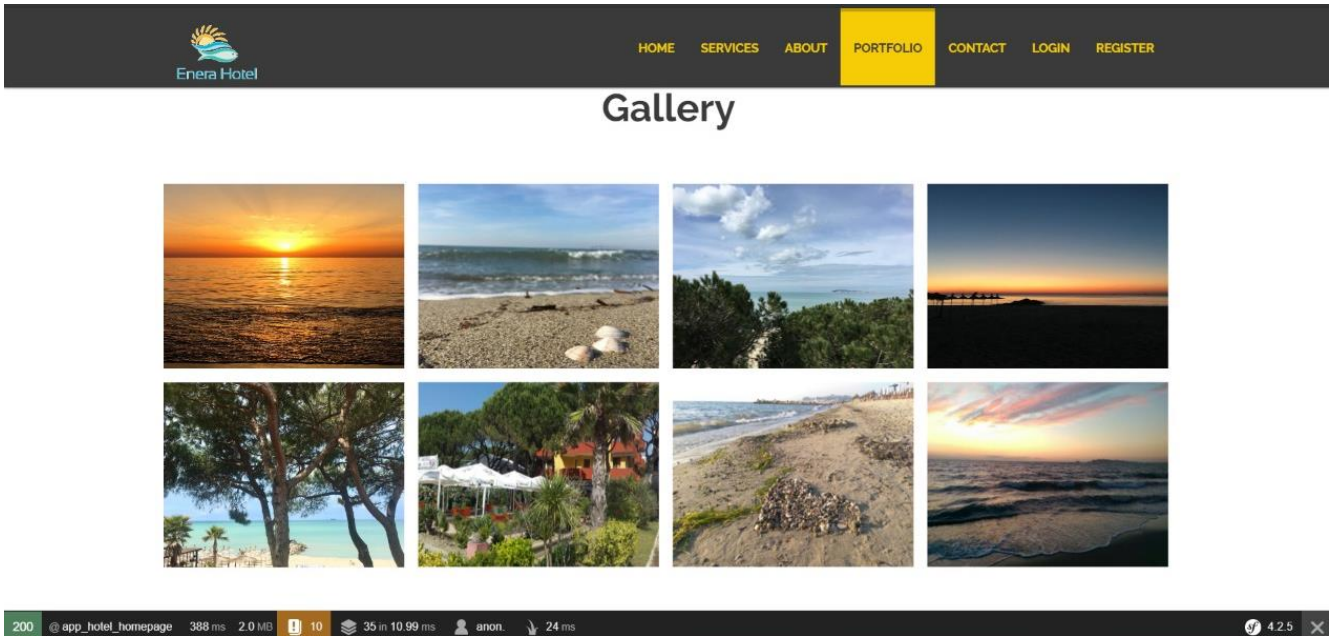
Enera Hotel Home Page where user can find information about the hotel and its services provided, images of the hotel, customer's feedback based on their experience and much more.

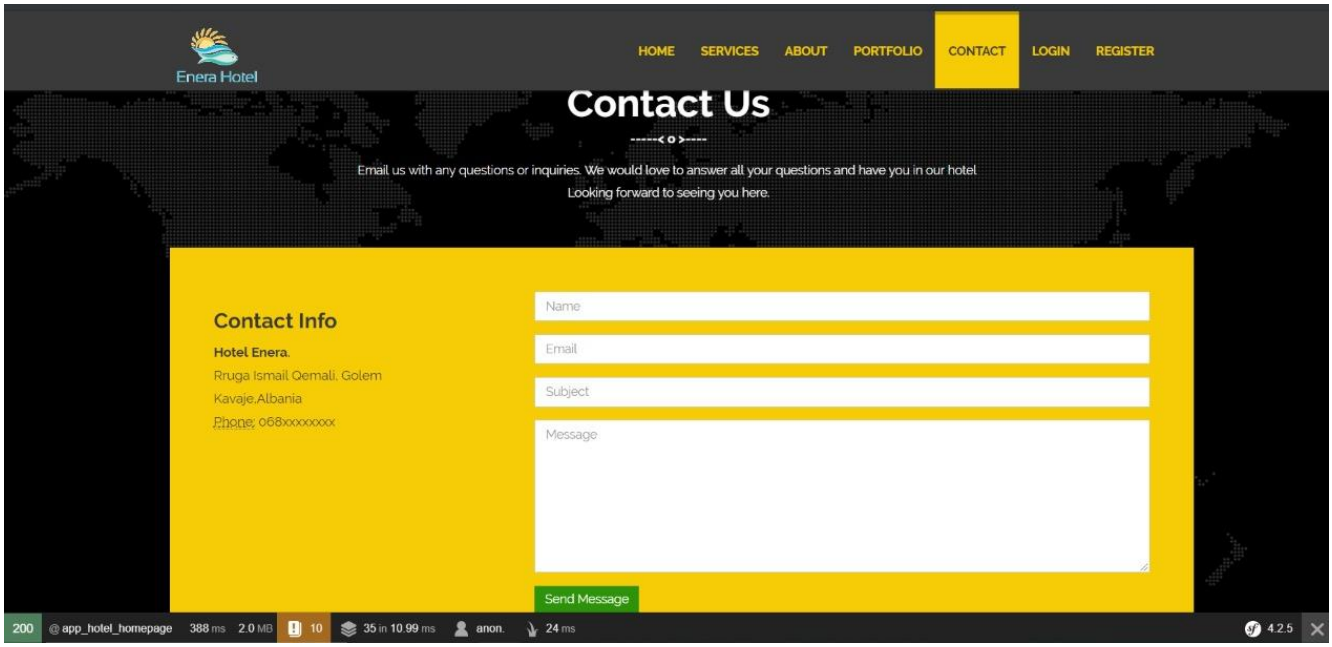




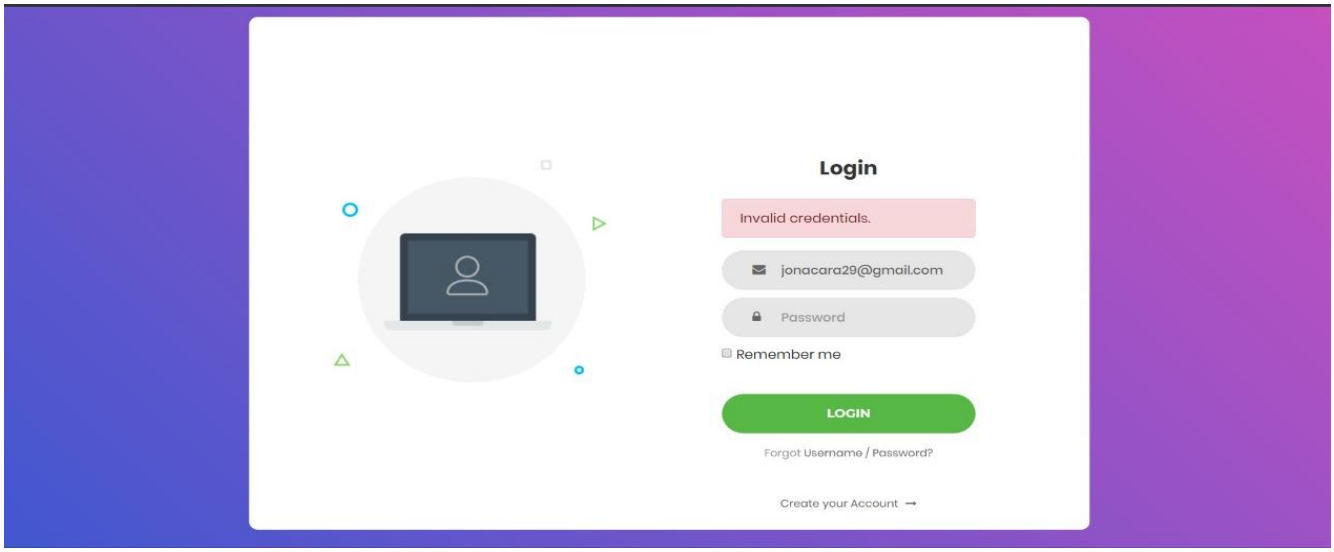

Enera Hotel
HOME
SERVICES
ABOUT
PORTFOLIO
CONTACT
LOGIN
REGISTER
About Us
Our Hotel
Hotel Enera is located in Golem. Featuring a private beach area, the 3-star hotel has air-conditioned rooms with free WiFi. The property has a restaurant.
At the hotel, all rooms come with a closet, a flat-screen TV and a private bathroom. Hotel Enera provides certain rooms with sea views, and all rooms have a balcony. All guest rooms will provide guests with a fridge.
An Italian breakfast is served each morning at the property.
Fishing is among the activities that guests can enjoy near the accommodation
Tirana is 46.7 km from Hotel Enera, while Durrës is 16.1 km from the property. The nearest airport is Tirana International Mother Teresa Airport, 43.5 km from the property.
This property also has one of the top-rated locations in Golem! Guests are happier about it compared to other properties in the area.
Couples in particular like the location – they rated it 8.9 for a two-person trip.
This property is also rated for the best value in Golem! Guests are getting more for their money when compared to other properties in this city.
200 @app_hotel_homepage 388 ms 2.0 MB 10 35 in 10.99 ms anon. 24 ms 4.2.5

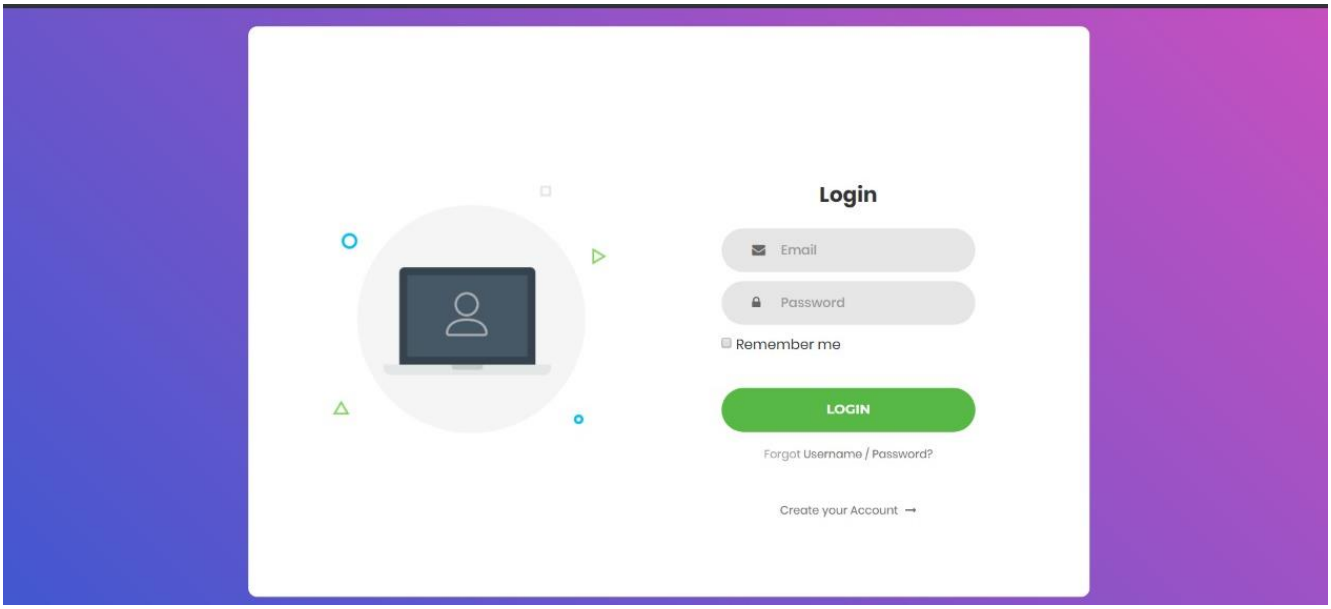

Enera Hotel
HOME
SERVICES
ABOUT
PORTFOLIO
CONTACT
LOGIN
REGISTER
What our clients say
The hotel is at first line and has its own beach with free sunchairs and sunbeds. The personnel was very polite and helpful. They all spoke very well English and the communication was not a problem at all!
Georgi
A good place!
Great breakfast, excellent location, clean and specious, everything you need is near. We would recommend the place and we would stay there again.
Sanja
Great value for the money
Really beautiful view from the room - sea and trees, very near to the beach, which is nice when you are with children
John Bond
Superb!
200 @app_hotel_homepage 388 ms 2.0 MB 10 35 in 10.99 ms anon. 24 ms 4.2.5


Enera Hotel
HOME
SERVICES
ABOUT
PORTFOLIO
CONTACT
LOGIN
REGISTER
Gallery
200 @ app_hotel_homepage 388 ms 2.0 MB 10 35 in 10.99 ms anon. 24 ms 4.2.5

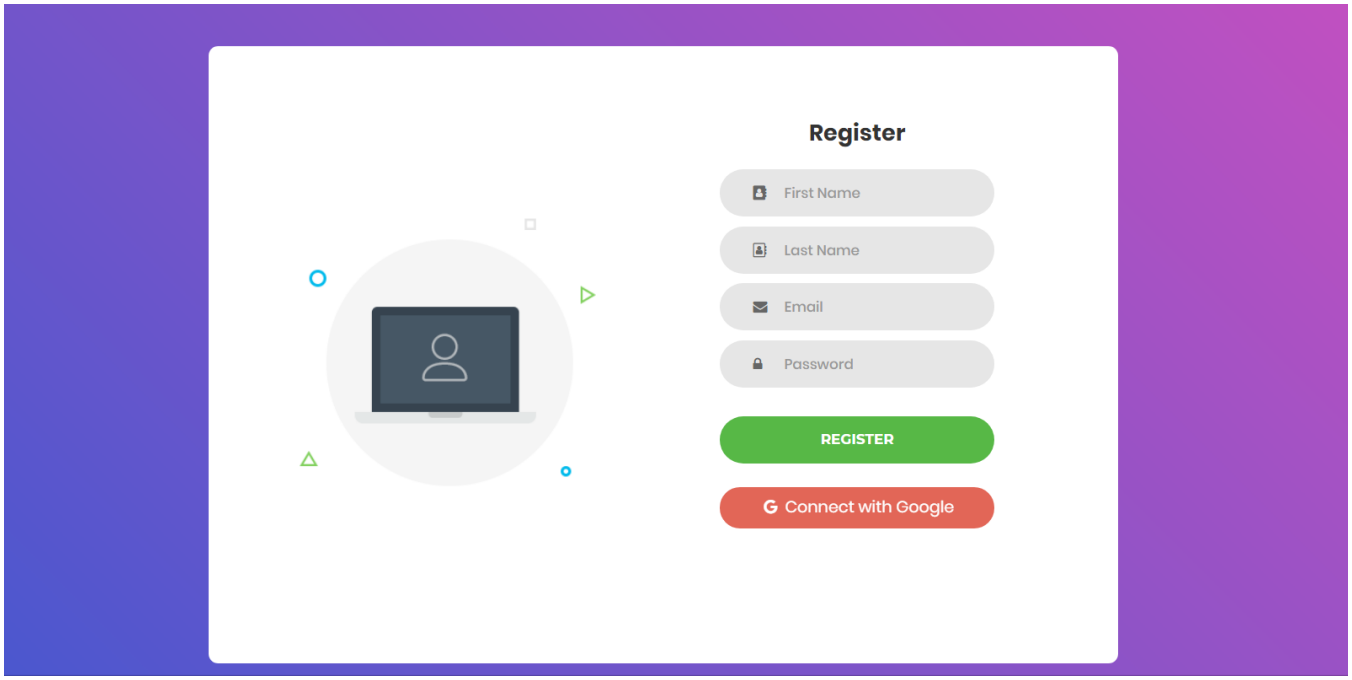

Enera Hotel
HOME
SERVICES
ABOUT
PORTFOLIO
CONTACT
LOGIN
REGISTER
Contact Us
Email us with any questions or inquiries. We would love to answer all your questions and have you in our hotel
Looking forward to seeing you here.
Contact Info
Hotel Enera.
Rruga Ismail Qemali, Golem
Kavaje,Albania
Phone: 068xxxxxxxx
Name
Email
Subject
Message
Send Message
200 @ app_hotel_homepage 388 ms 2.0 MB 10 35 in 10.99 ms anon. 24 ms 4.2.5

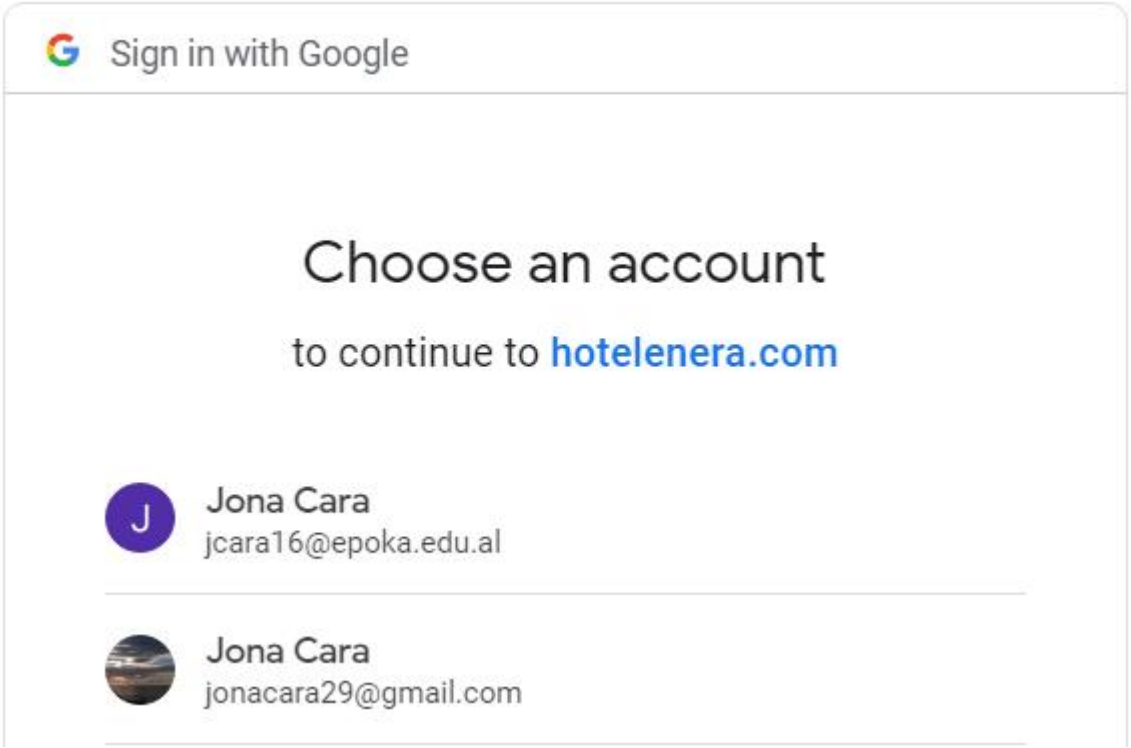
Login user interface where user can log into the system



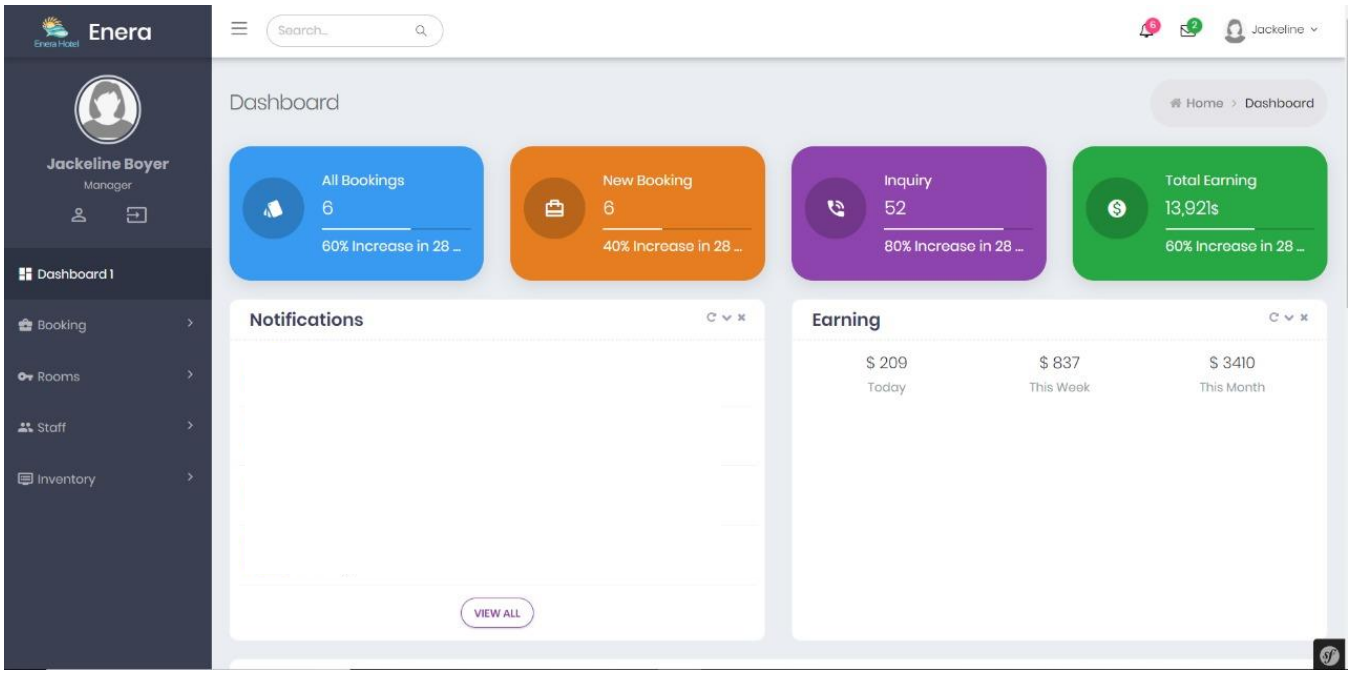


Register user interface; If customer does not have an account, it can register using its credentials or user can sign up using gmail.

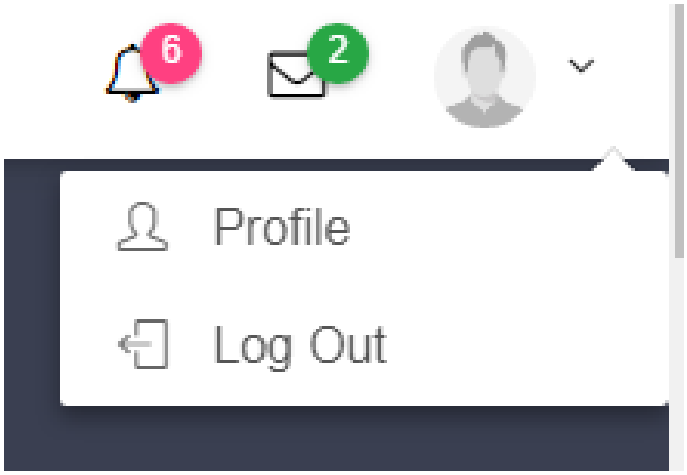




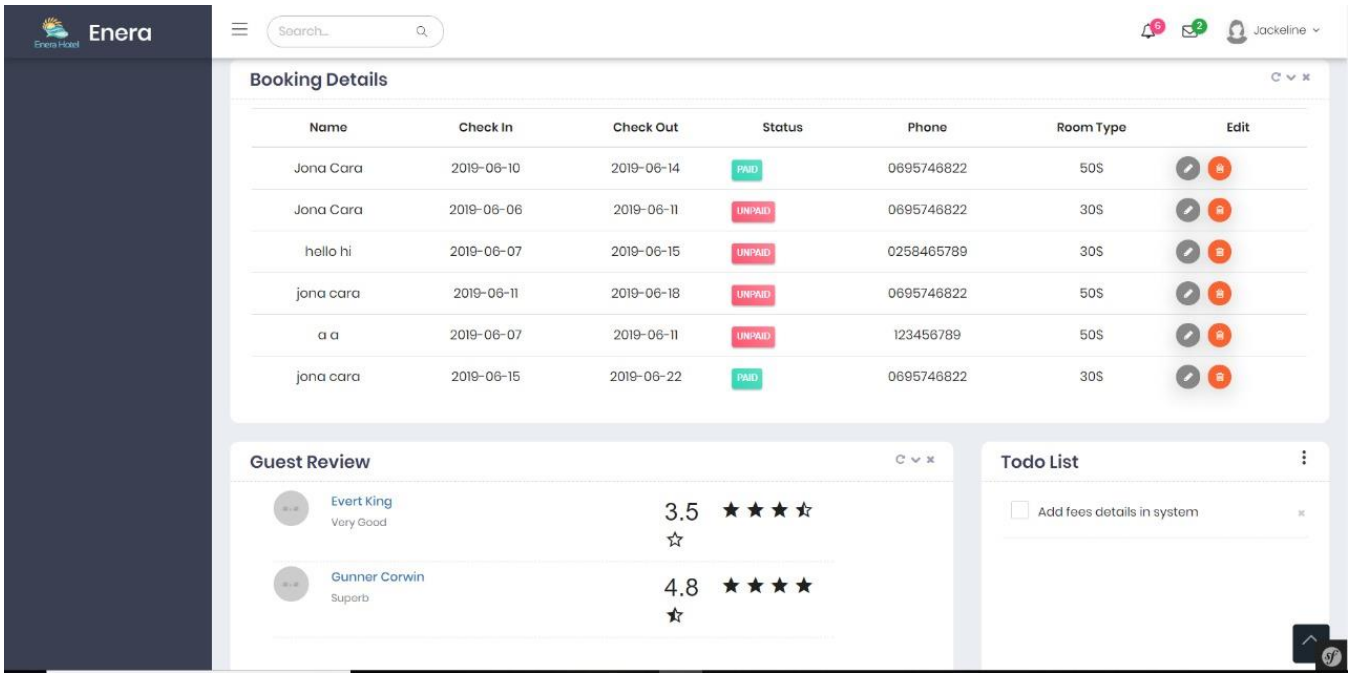
Admin Dashboard



Log Out

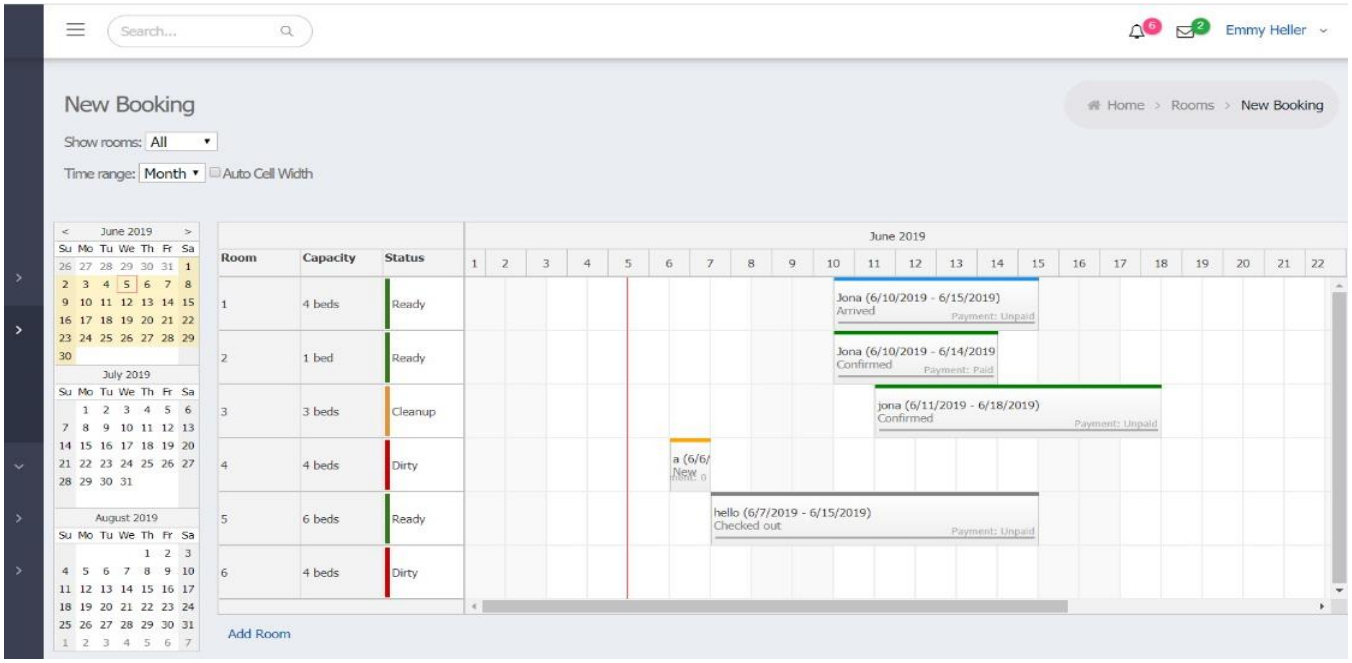


All bookings can be seen from Admin in a list view. Admin can edit a booking or delete it in case of booking cancellation by customer request.
Admin can also view guest reviews in the dashboard. Which will help him understand what customer like and dislike and make further improvements.



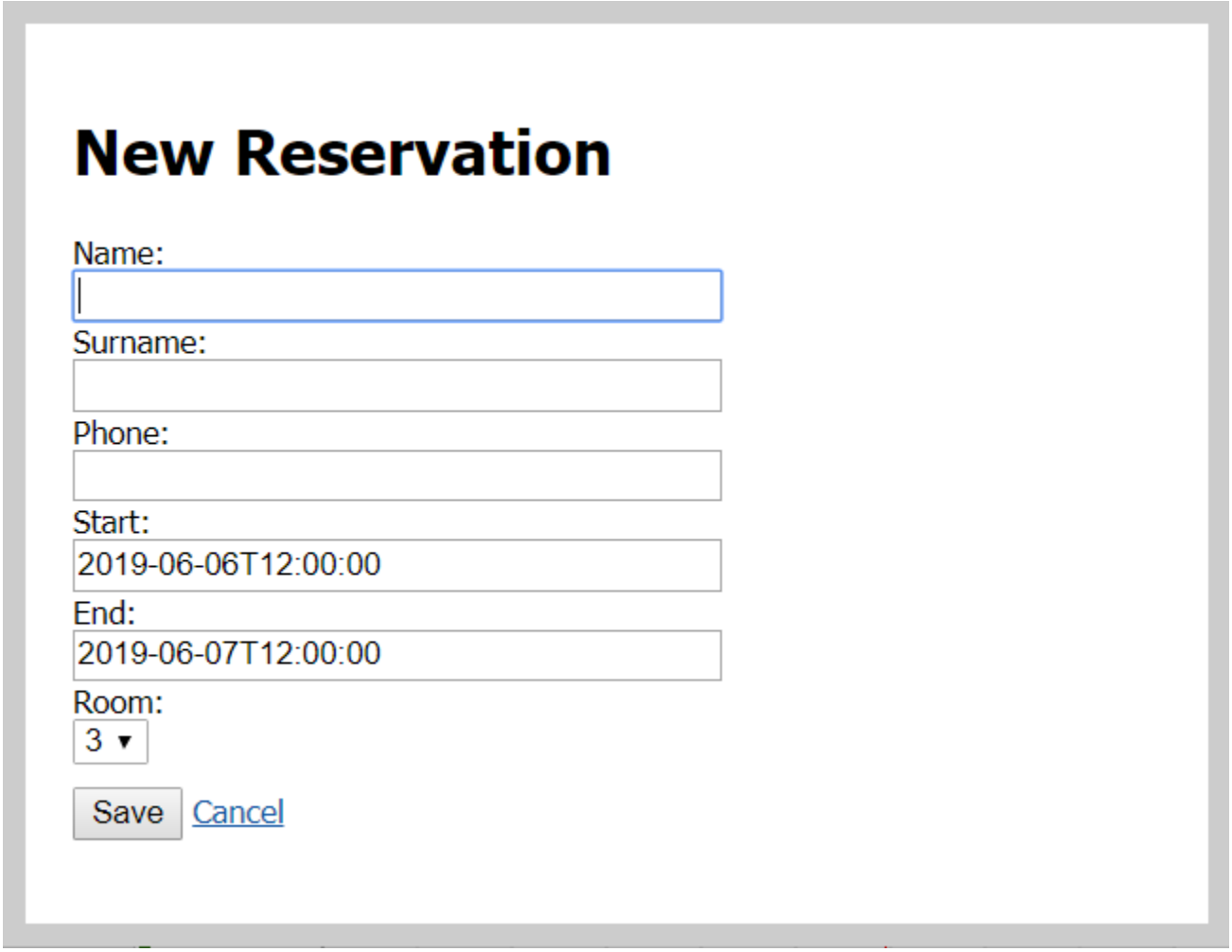
Booking interface.

- Used to add new booking
- Edit booking details
- Extend user stay by sliding in the calendar
- Set room as Ready, Cleanup or Dirty
- Set room to Confirmed/Arrived/New/Checked Out/Paid
- Select Dates
- Delete Bookings
- Real time Update
- View room and Customer details

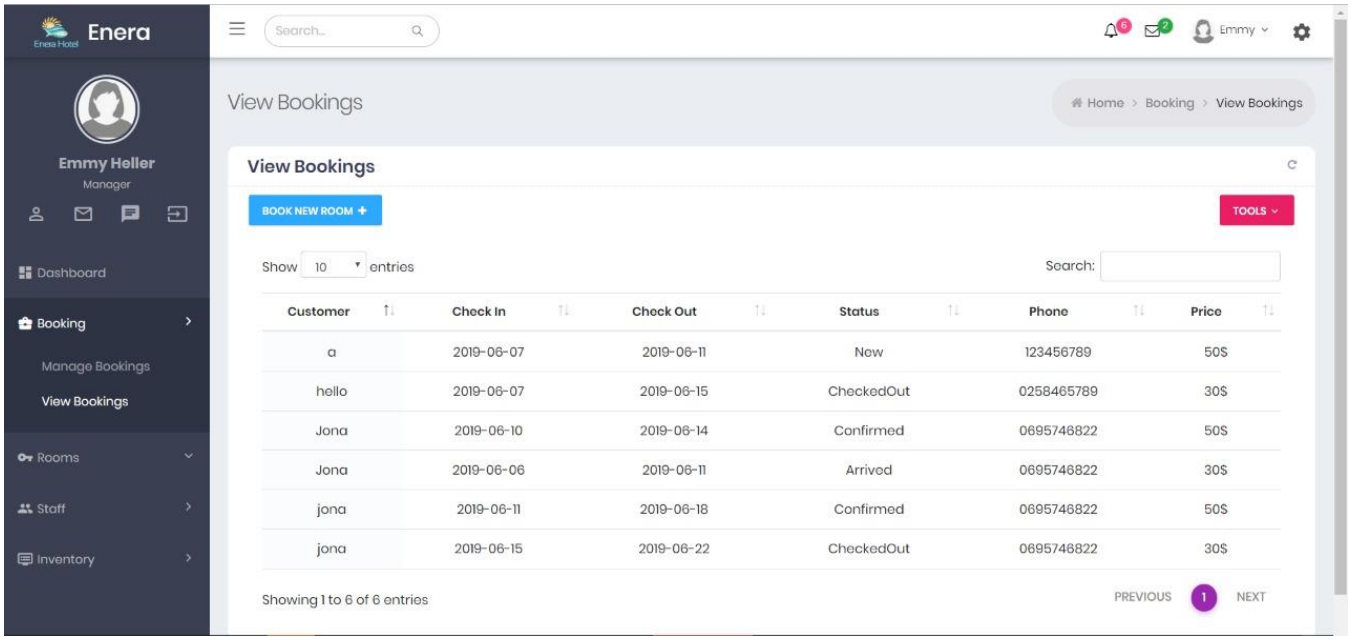


This interface is available to admin/manager, receptionist and customer in order to make bookings and check for availability.

By clicking in the calendar, a window about new reservation will be opened in order to register the new booking.

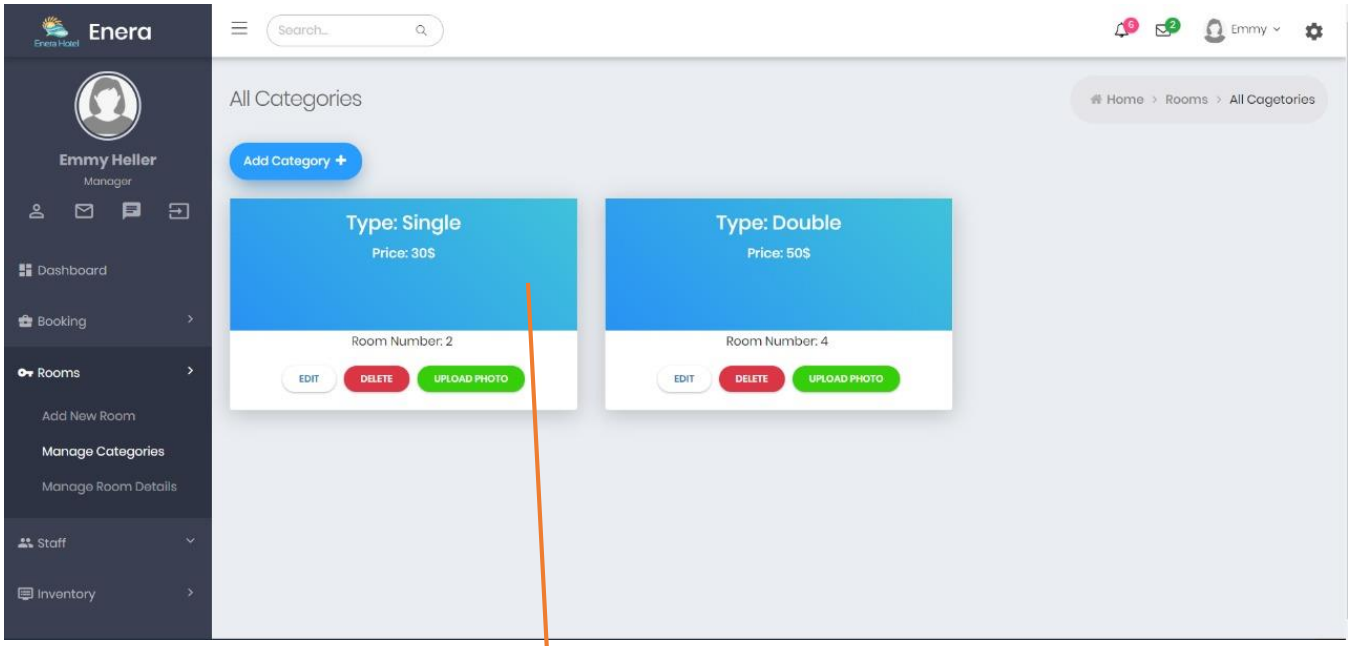


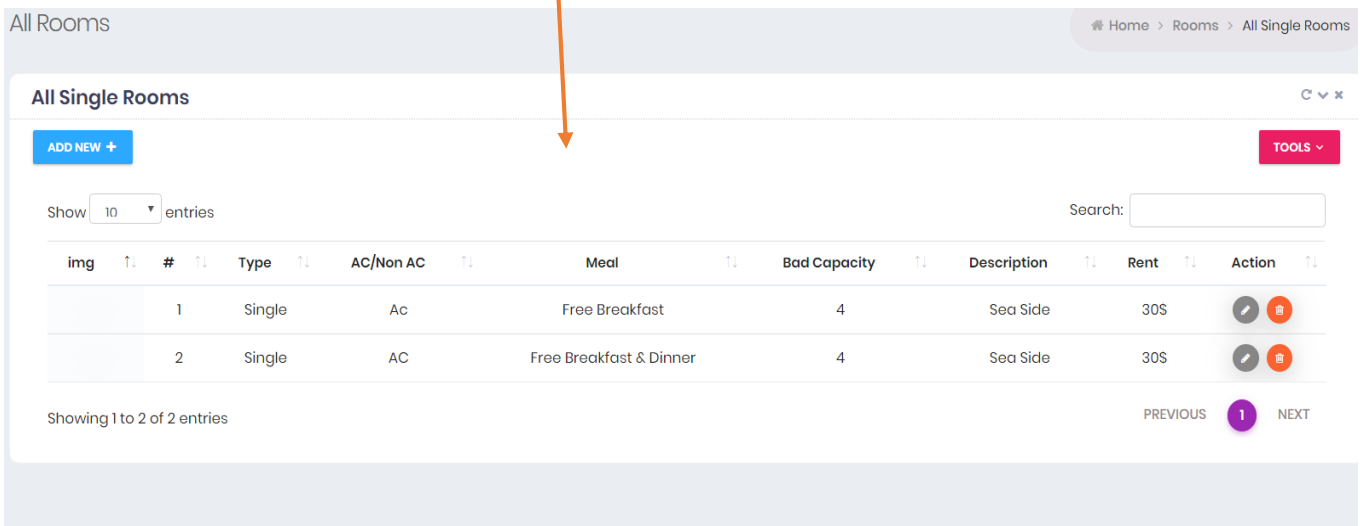
View booking – Admin can see all booking registered in the hotel



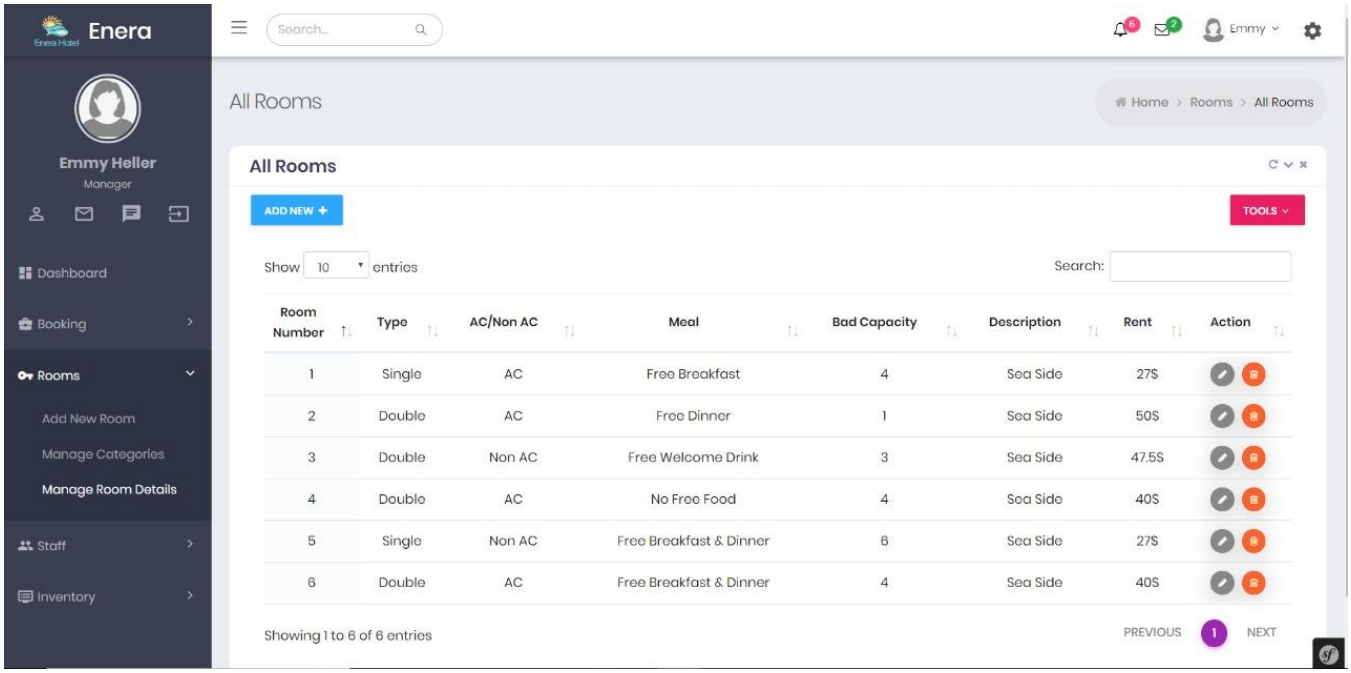
Room Categories are types of rooms that hotel offers, Add Category ads another room type which will be shown next to the rooms available. Delete button removes the room category, edit changes category details, and upload photos of a specific room.

By clicking in the room type colored area, a list view of all rooms of that type will be shown

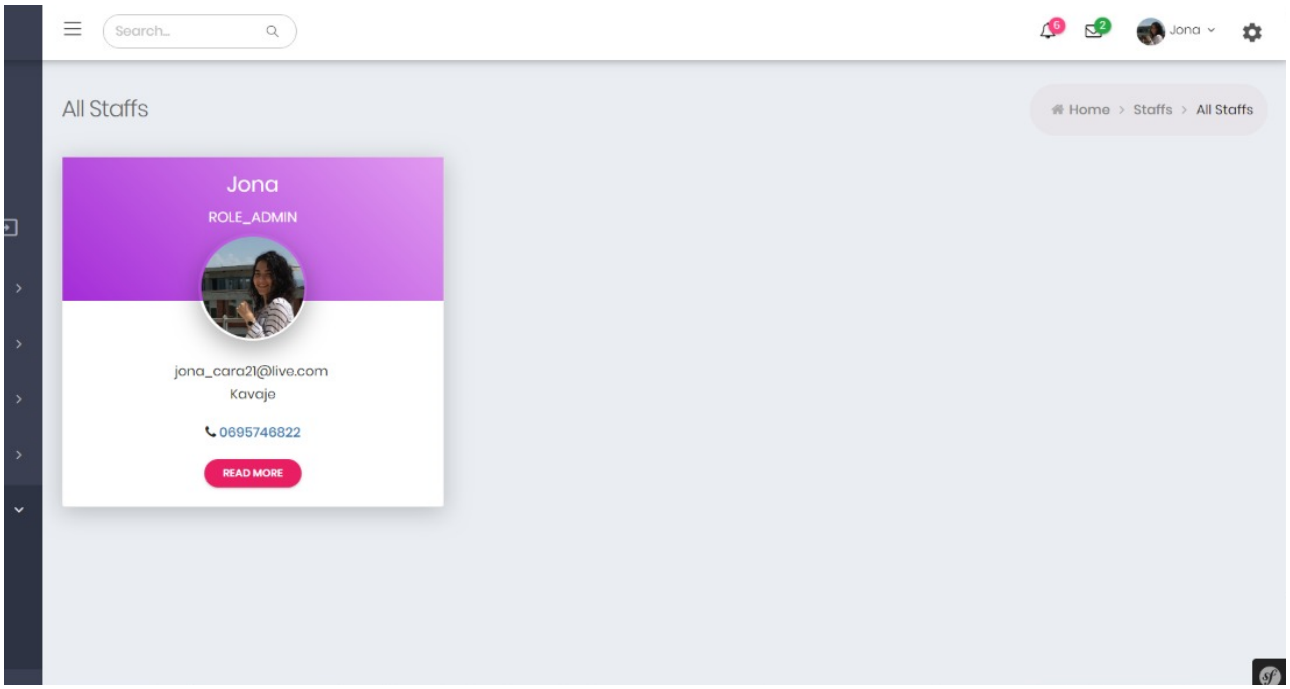




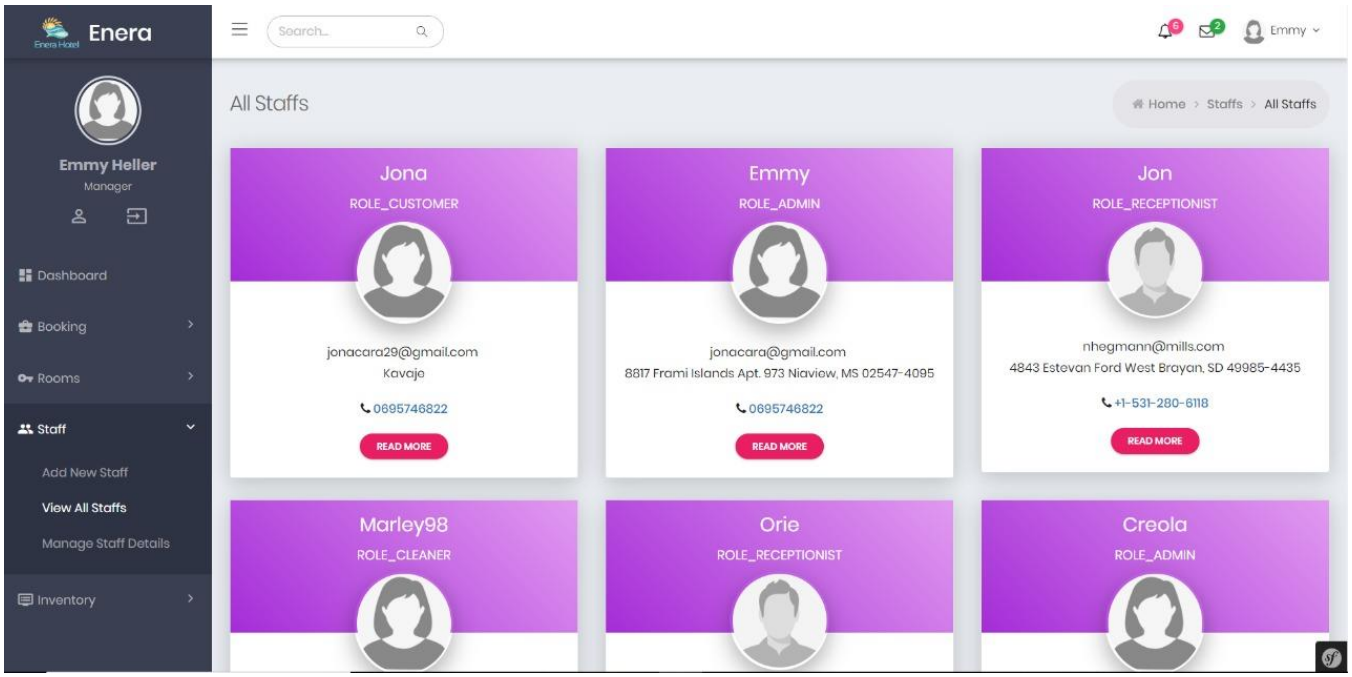
All rooms of all types in the hotel in a list view, where can be edited or deleted based on the manager desire and hotel requirements or availability.



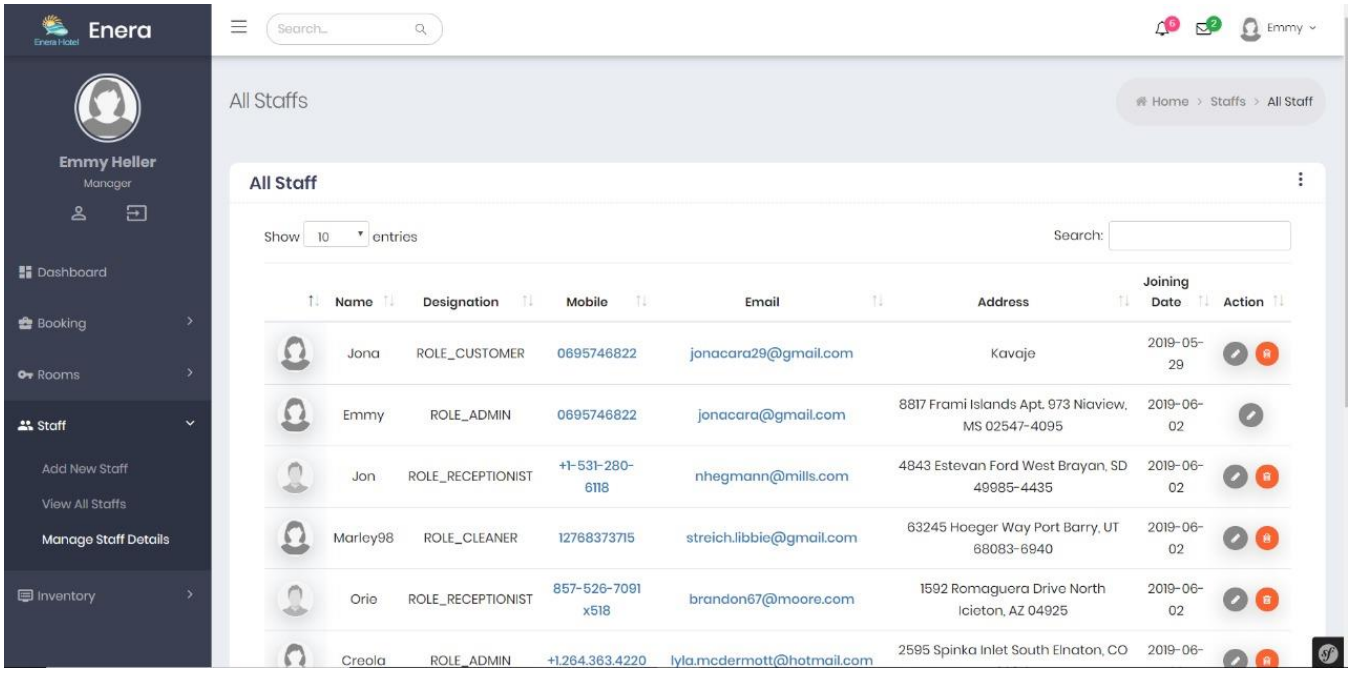
Hotel Staff in a Grid View



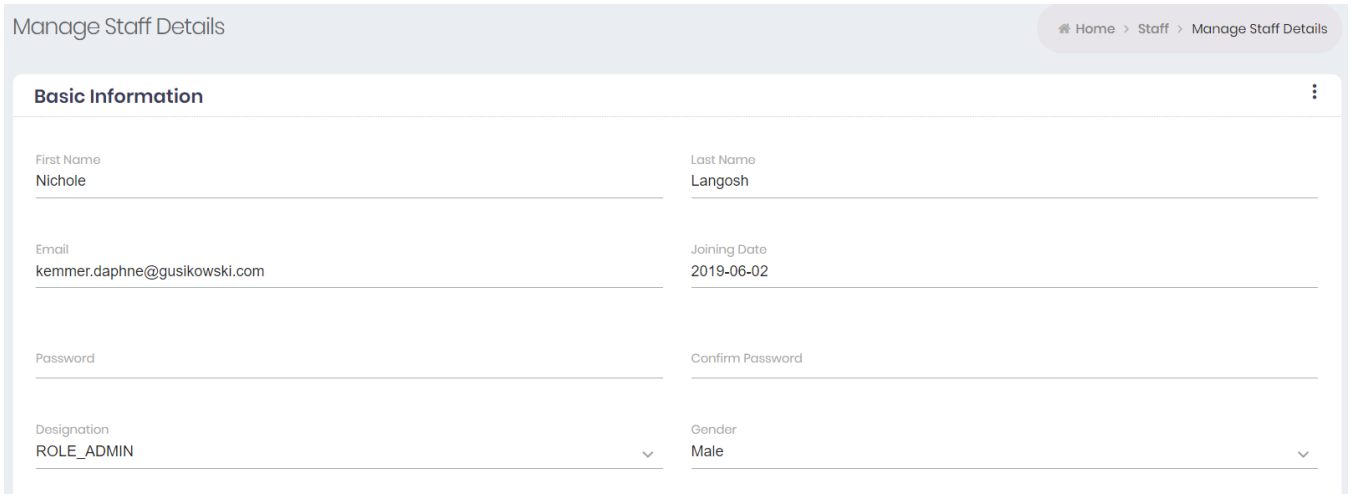
A full fist of employees generated using “fixtures” and faker bundle by Symfony only to provide a better visualization of this page.



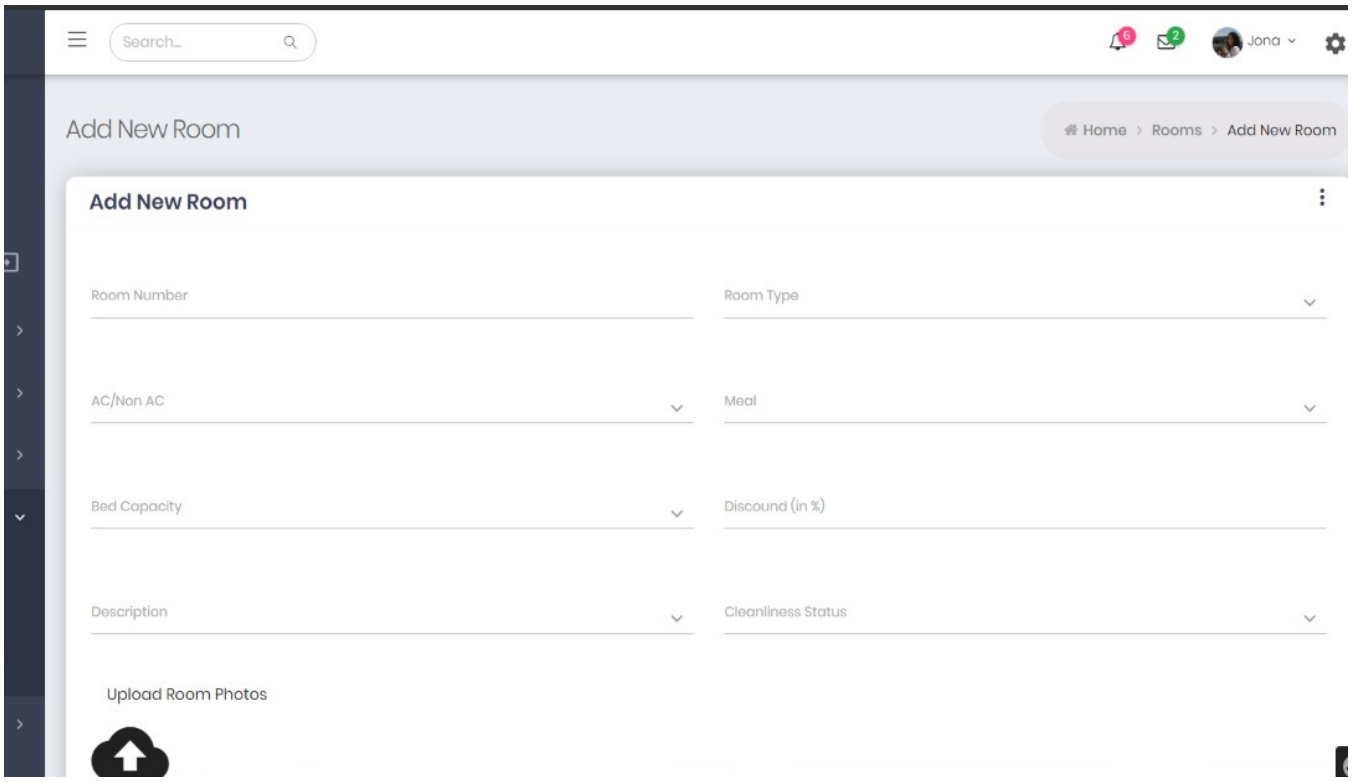
Staff details in a list view. A staff member can be edited or deleted. Except from the logged user.



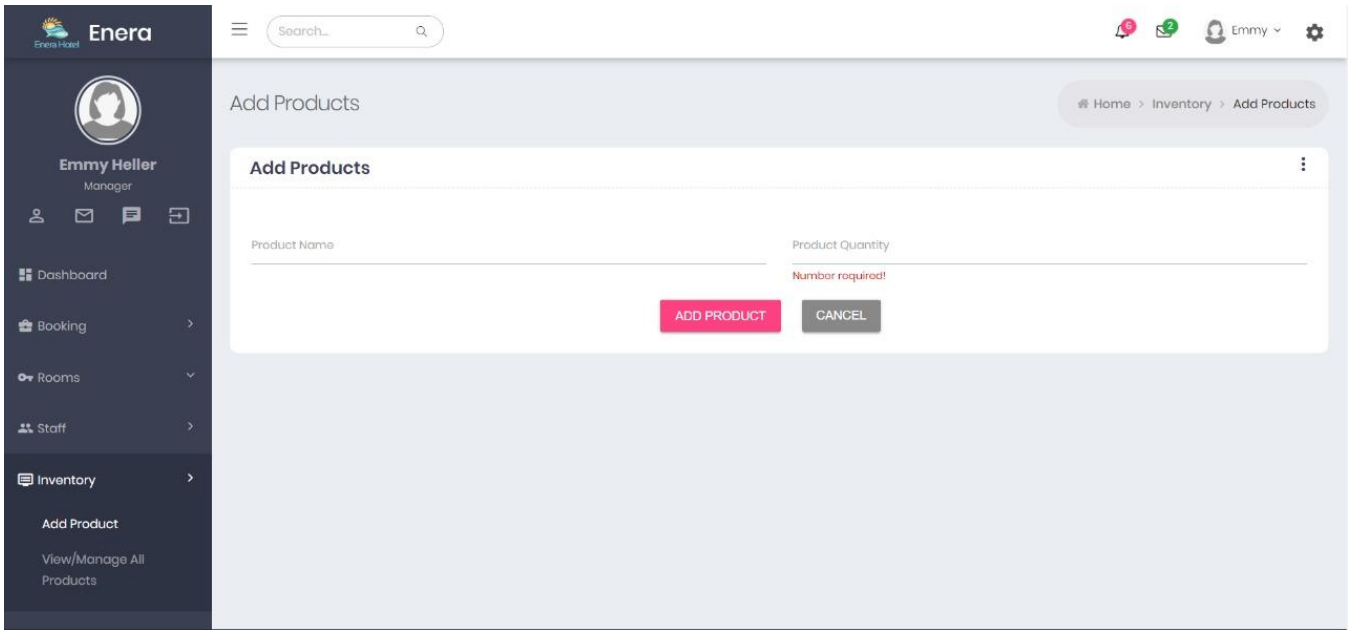
When clicking edit Button this interface with information taken from database will be shown for the selected user



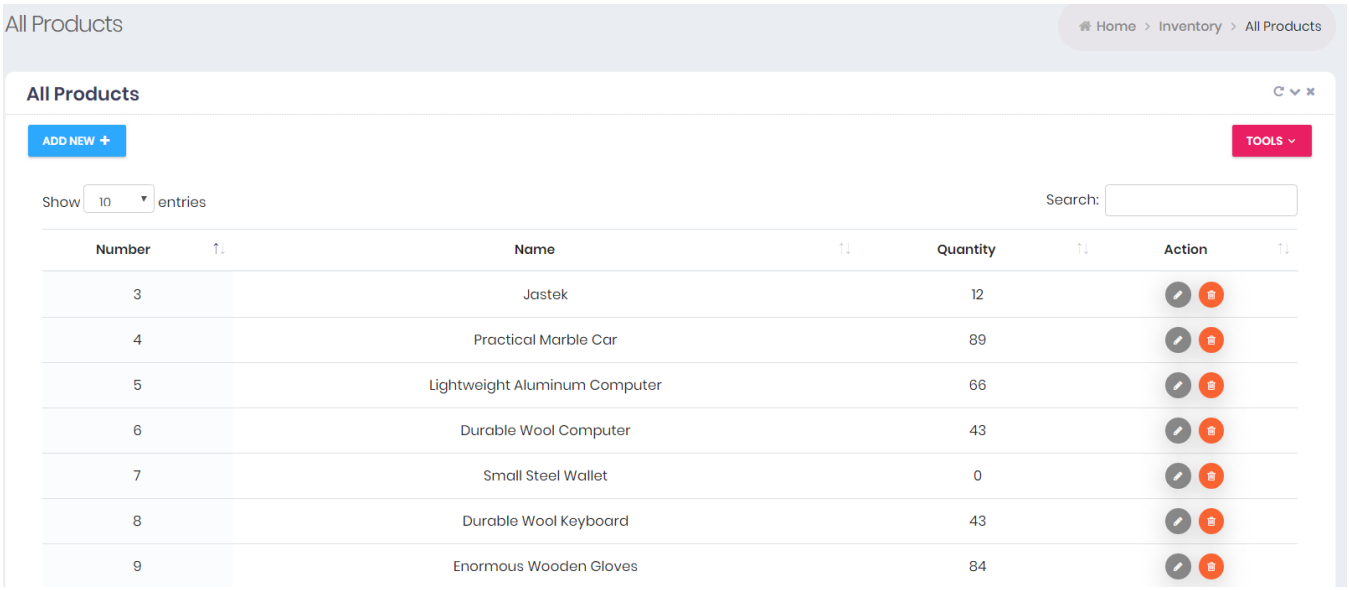
Add new room by entering information required below



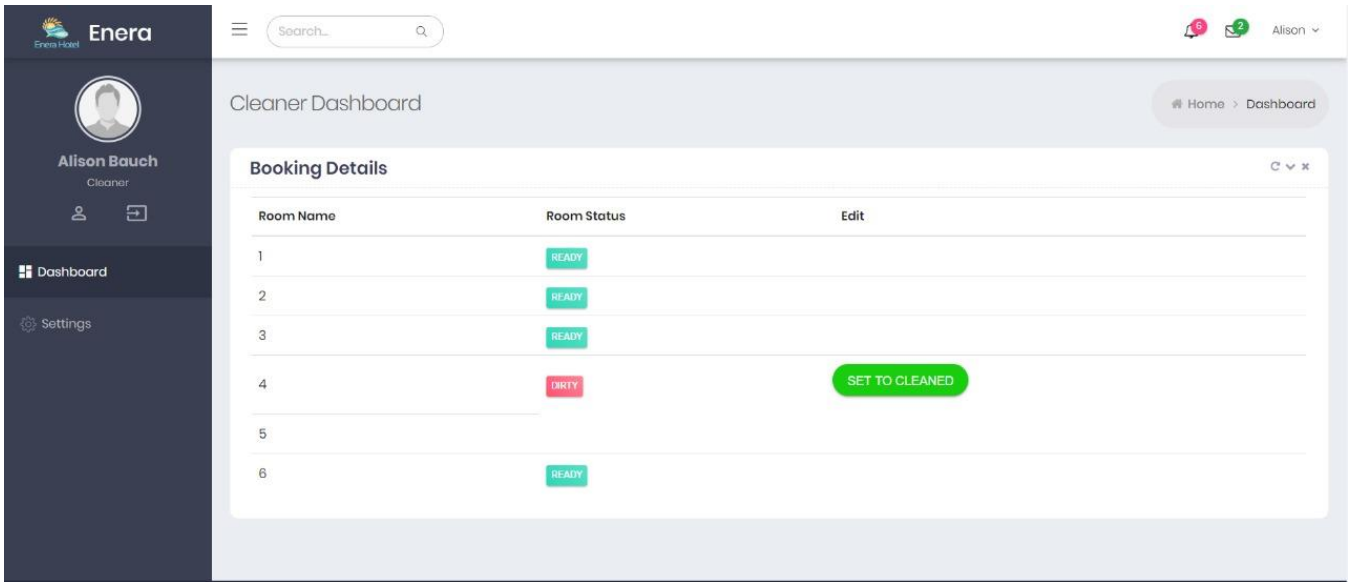
Add products used in the hotel



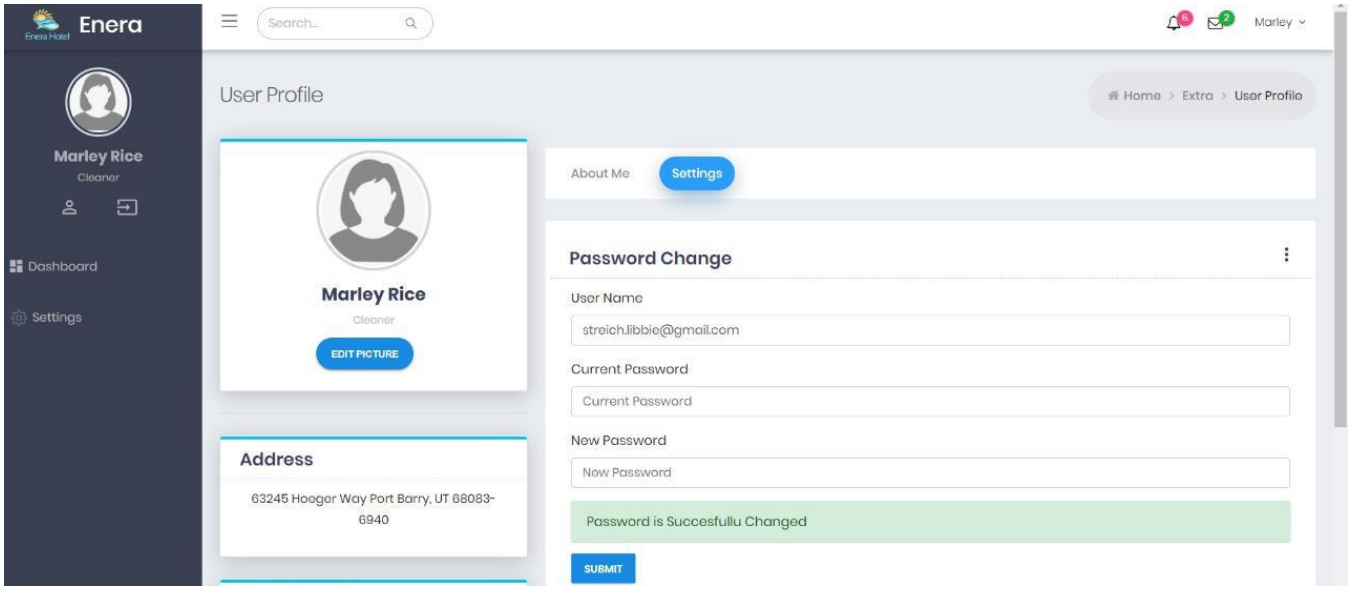
List view of products available



Cleaner Dashboard
Shows all rooms and room status. In the edit part a set to cleaned button will be shown if there is a need to take action in the room (e.g. Cleanup or Dirty room status). If Room is in “ready” status no button action is shown.



User profile shows user information, Ad in setting he can change his password and update the new one.



Appendix C

References

Symfony Casts. Retrieved from: https://symfonycasts.com/tracks/symfony

Symfony Documentation. Retrieved from: https://symfony.com/doc/current/index.html#gsc.tab=0

Roger S. Pressman, Ph.D. (2010) Software Engineering. A practitioner's Approach. ISBN 978–0–07–337597–7

Appendix D – Code

Hotel Controller

```
<?php

namespace App\Controller;

use App\Entity\Booking;
use App\Entity\Hotel;
use App\Entity\Review;
use App\Entity\User;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\ParamConverter;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;
use Doctrine\Common\Collections\Criteria;

class HotelController extends AbstractController
{
    /**
     * @Route("/")
     */
    public function homepage()
    {
        return $this->render('hotel/home.html.twig');
    }

    /**
     * @Route("/dashboard", name = "app_dashboard")
     * @IsGranted("ROLE_ADMIN")
     * @ParamConverter("current_date", options={"format": "Y-m-d"})
     */
    public function show()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $Bookings = $this->getDoctrine()->getRepository(Booking::class)->findAll();
        $criteria = new Criteria();
        $current_date = new \DateTime();
        $criteria->where($criteria->expr()->lte('Date',$current_date))
                 ->where($criteria->expr()->gte('Date',$current_date->modify('-7 days')));
        $new_booking = $this->getDoctrine()->getRepository(Booking::class)->matching($criteria);


        $Reviews = $this->getDoctrine()->getRepository(Review::class)->findAll();
        $hotel = $this->getDoctrine()->getRepository(Hotel::class)->findOneBy(['name' => 'Enera']);
        $hotel_image = $hotel->getImage();
        $images = array();
        foreach ($Users as $key => $user) {
            if($user->getPhoto()!=null)
            $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";
            }
        }
        return $this->render('hotel/view.html.twig',
            [
             'users' =>$Users,
             'images' => $images,
             'hotel_image' => $hotel_image,
             'bookings' => $Bookings,
             'new_booking' => $new_booking,
             'reviews' => $Reviews
            ]);
    }


    /**
     * @Route("/cleaner/dashboard", name = "app_cleaner_dashboard")
     * @IsGranted("ROLE_CLEANER")
     */
    public function show_cleaner()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $Bookings = $this->getDoctrine()->getRepository(Booking::class)->findAll();
```

```
        $Reviews = $this->getDoctrine()->getRepository(Review::class)->findAll();
        $hotel = $this->getDoctrine()->getRepository(Hotel::class)->findOneBy(['name' => 'Enera']);
        $hotel_image = $hotel->getImage();
        $images = array();
        foreach ($Users as $key => $user) {
            if($user->getPhoto()!=null)
            $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";
            }
        }
        return $this->render('hotel/view_cleaner.html.twig',
            [
             'users' =>$Users,
             'images' => $images,
             'hotel_image' => $hotel_image,
             'bookings' => $Bookings,
             'reviews' => $Reviews
            ]);
    }
    /**
     * @Route("/receptionist/dashboard", name = "app_receptionist_dashboard")
     * @IsGranted("ROLE_RECEPTIONIST")
     */
    public function show_receptionist()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $Bookings = $this->getDoctrine()->getRepository(Booking::class)->findAll();
        $Reviews = $this->getDoctrine()->getRepository(Review::class)->findAll();
        $hotel = $this->getDoctrine()->getRepository(Hotel::class)->findOneBy(['name' => 'Enera']);
        $hotel_image = $hotel->getImage();
        $images = array();
        foreach ($Users as $key => $user) {
            if($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";
            }
        }
        return $this->render('hotel/view_receptionist.html.twig',
            [
                'users' =>$Users,
                'images' => $images,
                'hotel_image' => $hotel_image,
                'bookings' => $Bookings,
                'reviews' => $Reviews
            ]);
    }
    /**
     * @Route("/customer/dashboard", name = "app_customer_dashboard")
     * @IsGranted("ROLE_CUSTOMER")
     */
    public function show_customer()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $Bookings = $this->getDoctrine()->getRepository(Booking::class)->findAll();
        $Reviews = $this->getDoctrine()->getRepository(Review::class)->findAll();
        $hotel = $this->getDoctrine()->getRepository(Hotel::class)->findOneBy(['name' => 'Enera']);
        $hotel_image = $hotel->getImage();
        $images = array();
        foreach ($Users as $key => $user) {
            if($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";
            }
        }
        return $this->render('hotel/view_customer.html.twig',
            [
                'users' =>$Users,
                'images' => $images,
                'hotel_image' => $hotel_image,
                'bookings' => $Bookings,
                'reviews' => $Reviews
```

```
            ]);
    }
}
```

Inventory Controller

```
<?php

namespace App\Controller;


use App\Entity\Hotel;
use App\Entity\Inventory;
use App\Entity\User;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class InventoryController extends AbstractController
{
    /**
     * @Route("/all_products", name = "app_inventory")
     * @IsGranted("ROLE_ADMIN")
     */

    public function view_inventory()
    {
        $Products = $this->getDoctrine()->getRepository(Inventory::class)->findAll();
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("inventory/inventory.html.twig",
            [
                'products' =>$Products,
                'users' =>$Users,
                'images' => $images,
            ]);
    }

    /**
     * @Route("/add_product", name = "app_add_inventory")
     */
    public function add_product(Request $request)
    {
        if ($request->attributes->get('_route') === 'app_add_inventory'
            && $request->isMethod('POST'))
        {
            $Product = new Inventory();
            $Product->setName($request->request->get("product_name"));
            $Product->setQuantity($request->request->get("product_quantity"));
            $Hotel = $this->getDoctrine()->getRepository(Hotel::class)->find(1);
            $Product->setHotel($Hotel);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($Product);
            $entityManager->flush();
            return $this->redirectToRoute('app_inventory');
        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";
```

```
            }
        }
        return $this->render('inventory/add_inventory.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }

    /**
     * @Route("/edit_product/{id}", name = "app_edit_product")
     * @IsGranted("ROLE_ADMIN")
     */
    public function edit_product(Request $request,$id)
    {
        $Product = $this->getDoctrine()->getRepository(Inventory::class)->find($id);
        if($request->attributes->get('_route') === "app_edit_product"
            && $request->isMethod("POST"))
        {
            $product_name = $request->request->get('product_name');
            $product_quantity = $request->request->get('product_quantity');

            $Product->setName($product_name);
            $Product->setQuantity($product_quantity);

            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_inventory');

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("inventory/edit_inventory.html.twig",
            [
                'product' => $Product,
                'users' =>$Users,
                'images' => $images,
            ]);
    }


    /**
     * @Route("/delete_product/{id}");
     * @Method({"DELETE"})
     */
    public function delete_product(Request $request, $id)
    {
        $product = $this->getDoctrine()->getRepository(Inventory::class)->find($id);
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->remove($product);
        $entityManager->flush();
        $response = new Response();
        $response->send();
    }
}
```

Room Controller

```
<?php

namespace App\Controller;


use App\Entity\Room;
use App\Entity\RoomType;
use App\Entity\User;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
use Symfony\Bundle\FrameworkBundle\Tests\Fixtures\Validation\Category;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Flex\Response;

class RoomController extends AbstractController
{

    /**
     * @Route("/edit_room", name = "app_all_rooms")
     * @IsGranted("ROLE_ADMIN")
     */
    public function view_room()
    {
        $Rooms = $this->getDoctrine()->getRepository(Room::class)->findAll();
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("rooms/view_edit_rooms.html.twig",
            [
                'rooms' =>$Rooms,
                'users' =>$Users,
                'images' => $images,
            ]);
    }

    /**
     * @Route("/add_room", name = "app_add_room")
     * @IsGranted("ROLE_ADMIN")
     */
    public function add_room(Request $request)
    {
        $Category = $this->getDoctrine()->getRepository(RoomType::class)->findAll();
        if ($request->attributes->get('_route') === 'app_add_room'
            && $request->isMethod('POST'))
        {
            $Room = new Room();
            $room_number = $request->request->get("room_number");
            $room_category = $request->request->get('room_category');
            $room_ac = $request->request->get('room_ac');
            $room_meals = $request->request->get('room_meals');
            $room_capacity = $request->request->get('room_capacity');
            $room_discount = $request->request->get('room_discount');
            $room_description = $request->request->get('room_description');
            $room_clean = $request->request->get('room_clean');

            $Category = $this->getDoctrine()->getRepository(RoomType::class)->findOneBy(['name' => $room_category]);
            $Room->setRoomNumber($room number);
            $Room->setCategory($Category);
            $Room->setAC($room ac);
            $Room->setMeals($room_meals);
            $Room->setCapacity($room_capacity);
            $Room->setDescription($room_description);
            $Room->setDiscount($room_discount);
```

```
            if($room_clean=="Cleaned")
                $Room->setCleanlinesStatus(1);
            else if ($room_clean == "Not Cleaned")
                $Room->setCleanlinesStatus(0);

            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($Room);
            $entityManager->flush();
            return $this->redirectToRoute('app_all_rooms');

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('rooms/add_room.html.twig',
            [
                'users' =>$Users,
                'images' => $images,
                'categories' => $Category,
            ]);
    }

    /**
     * @Route("/edit_room/{id}", name = "app_edit_room")
     * @IsGranted("ROLE_ADMIN")
     */
    public function edit_room(Request $request,$id)
    {
        $Room = $this->getDoctrine()->getRepository(Room::class)->find($id);
        if($request->attributes->get('_route') === "app_edit_room"
           && $request->isMethod("POST"))
        {
            $room_number = $request->request->get('room_number');
            $room_category = $request->request->get('room_category');
            $room_ac = $request->request->get('room_ac');
            $room_meals = $request->request->get('room_meals');
            $room_capacity = $request->request->get('room_capacity');
            $room_price_discount = $request->request->get('room_discount');
            $room_description = $request->request->get('room_description');
            $room_clean = $request->request->get('room_clean');
            $Category = $this->getDoctrine()->getRepository(RoomType::class)->findOneBy(['name' => $room_category]);

            $Room->setRoomNumber($room_number);
            $Room->setCategory($Category);
            $Room->setAC($room ac);
            $Room->setMeals($room meals);
            $Room->setCapacity($room capacity);
            $Room->setDescription($room_description);
            $Room->setDiscount($room_price_discount);
            if($room_clean=="Cleaned")
            $Room->setCleanlinesStatus(1);
            else if ($room_clean == "Not Cleaned")
                $Room->setCleanlinesStatus(0);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_all_rooms');

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("rooms/edit_room.html.twig",
```

```
            [
                'room' => $Room,
                'users' =>$Users,
                'images' => $images,
            ]);
    }


    /**
     * @Route("/all_categories", name = "app_all_categories")
     * @IsGranted("ROLE_ADMIN")
     */
    public function view_category()
    {
        $Categories = $this->getDoctrine()->getRepository(RoomType::class)->findAll();
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("rooms/all_categories.html.twig",
            [
                'category' =>$Categories,
                'users' =>$Users,
                'images' => $images,
            ]);
    }

    /**
     * @Route("/all_categories/{category}", name = "app_rooms_by_categories")
     * @IsGranted("ROLE_ADMIN")
     */
    public function view_room_by_category($category)
    {
        $Categories = $this->getDoctrine()->getRepository(RoomType::class)->findOneBy(['name'=>$category]);
        $Rooms = $this->getDoctrine()->getRepository(Room::class)->findBy(['category'=>$Categories]);
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render("rooms/rooms_by_categories.html.twig",
            [
                'category' =>$Categories,
                'users' =>$Users,
                'images' => $images,
                'rooms' => $Rooms,
            ]);
    }

    /**
     * @Route("/add_category", name = "app_add_category")
     * @IsGranted("ROLE_ADMIN")
     */
    public function add_category(Request $request)
    {
        if ($request->attributes->get('_route') === 'app_add_category'
            && $request->isMethod('POST'))
        {
            $Category = new RoomType();
            $category_name = $request->request->get("category_name");
            $category_price = $request->request->get('category_price');

            $Category->setName($category_name);
            $Category->setPrice($category price);
```

```
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($Category);
            $entityManager->flush();
            return $this->redirectToRoute('app_all_categories');

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('rooms/add_category.html.twig',
            [
                'users' =>$Users,
                'images' => $images,
            ]);
    }

    /**
     * @Route("/edit_category/{category}", name = "app_edit_category")
     * @IsGranted("ROLE_ADMIN")
     */
    public function edit_category(Request $request,$category)
    {
        $Category = $this->getDoctrine()->getRepository(RoomType::class)->findOneBy(['name'=>$category]);
        if ($request->attributes->get('_route') === 'app_edit_category'
            && $request->isMethod('POST'))
        {
            $category_name = $request->request->get("category_name");
            $category_price = $request->request->get('category_price');

            $Category->setName($category_name);
            $Category->setPrice($category_price);

            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_all_categories');

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('rooms/edit_category.html.twig',
            [
                'users' =>$Users,
                'images' => $images,
                'category' => $Category,
            ]);
    }

    /**
     * @Route("/delete_category/{id}");
     * @Method({"DELETE"})
     */
    public function delete_category(Request $request, $id)
    {
        $category = $this->getDoctrine()->getRepository
        (RoomType::class)->find($id);
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->remove($category);
        $entityManager->flush();
        $response = new Response();
        $response->send();
    }
```

```
}
```

Security Controller

```
<?php
namespace App\Controller;
use App\Entity\User;
use mysql_xdevapi\Exception;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Security\Http\Authentication\AuthenticationUtils;
class SecurityController extends AbstractController
{
    /**
     * @Route("/login", name="app_login")
     */
    public function login(AuthenticationUtils $authenticationUtils)
    {
        // get the login error if there is one
        $error = $authenticationUtils->getLastAuthenticationError();
        // last username entered by the user
        $lastUsername = $authenticationUtils->getLastUsername();
        $securityContext = $this->container->get('security.authorization_checker');
        if ($securityContext->isGranted('ROLE_ADMIN')) {
            return $this->redirectToRoute('app_dashboard');
        }

        else if ($securityContext->isGranted('ROLE_RECEPTIONIST')) {
            return $this->redirectToRoute('app_receptionist_dashboard');
        }
        else if ($securityContext->isGranted('ROLE_CLEANER')) {
            return $this->redirectToRoute('app_cleaner_dashboard');
        }
        else if ($securityContext->isGranted('ROLE_CUSTOMER')) {
            return $this->redirectToRoute('app_customer_dashboard');
        }

        else
        {
            return $this->render('security/login.html.twig', [
                'last_username' => $lastUsername,
                'error' => $error
            ]);
        }
    }
    /**
     * @Route("/register", name="app_register")
     */
    public function register(AuthenticationUtils $authenticationUtils)
    {
        $error = $authenticationUtils->getLastAuthenticationError();
        $lastUsername = $authenticationUtils->getLastUsername();
        return $this->render('security/register.html.twig',
            [
                'last_username' => $lastUsername,
                'error' => $error]);
    }
    /**
     * @Route ("/logout", name = "app_logout")
     */
    public function logout()
    {
        throw new \Exception("Logged Out Failed");
    }
}
```

Staff Controller

```php
<?php
namespace App\Controller;
use App\Entity\Hotel;
use App\Entity\User;
use http\Env\Response;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Method;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
use Symfony\Component\Security\Core\User\UserInterface;

/**
 * Class StaffController
 * @package App\Controller
 */
class StaffController extends AbstractController
{
    /**
     * @var UserPasswordEncoderInterface
     */
    private $passwordEncoder;
    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }
    /**
     * @Route("/all_staff", name="app_all_staff")
     * @IsGranted("ROLE_ADMIN")
     */
    public function view_staff()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('staff/all_staff.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }

    /**
     * @Route("/add_staff", name = "app_add_staff")
     * @IsGranted("ROLE_ADMIN")
     */
    public function add_staff(Request $request)
    {
        if ($request->attributes->get('_route') === 'app_add_staff'
            && $request->isMethod('POST'))
        {
            $User = new User();
            $User->setFirstName($request->request->get('first_name'));
            $User->setLastName($request->request->get('last_name'));
            $User->setEmail($request->request->get('email'));
            $joiningData =\DateTime::createFromFormat('Y-m-d',$request->request->get('joining_date'));
            $User->setJoiningDate($joiningData);
            $encodedPassword = $this->passwordEncoder->encodepassword($User,$request->request->get('password'));
            $User->setPassword($encodedPassword);
            $Designation  = $request->request->get('designation');
            $confpassword = $request->request->get('conf_password');
            $Role = [""];
            if($Designation=="Admin")
                $Role = ["ROLE_ADMIN"];
            else if($Designation=="Receptionist")
                $Role = ["ROLE_RECEPTIONIST"];
            else if($Designation=="Cleaner")
                $Role = ["ROLE_CLEANER"];
```

```
            $User->setRoles($Role);
            $designation  = $request->request->get('designation');
            $User->setPhone($request->request->get('phone'));
            $gender = $request->request->get('gender');
            $User->setGender($gender);
            $birthday = \DateTime::createFromFormat('Y-m-d',$request->request->get('birthday'));
            $User->setBirthday($birthday);
            $User->setAddress($request->request->get('address'));
            $hotel_id = 1;
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->persist($User);
            $entityManager->flush();
            return $this->redirectToRoute('app_all_staff');
        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('staff/add_staff.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }



    /**
     * @Route("/edit_staff", name = "app_edit_staff")
     * @IsGranted("ROLE_ADMIN")
     */
    public function edit_staff()
    {
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('staff/view_edit_staff.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }


    /**
     * @Route("/edit_credentials", name = "app_edit_credentials")
     * @IsGranted("ROLE_RECEPTIONIST")
     */
    public function edit_credentials(Request $request,UserInterface $loggedUser)
    {
        if ($request->attributes->get('_route') === 'app_edit_credentials'
            && $request->isMethod('POST'))
        {
            $FirstName = $request->request->get('first_name');
            $LastName = $request->request->get('last_name');
            $Email = $request->request->get('email');
            $Designation  = $request->request->get('designation');

            $PhoneNumber = $request->request->get('phone');
            $Gender =  $request->request->get('gender');
            $Birthday = \DateTime::createFromFormat('Y-m-d',$request->request->get('birthday'));
            $Address =  $request->request->get('address');
            $hotelId = new Hotel();
            $loggedUser->setFirstName($FirstName);
            $loggedUser->setLastName($LastName);
            $loggedUser->setEmail($Email);
```

```
            $loggedUser->setPhone($PhoneNumber);
            $loggedUser->setGender($Gender);
            $loggedUser->setBirthday($Birthday);
            $loggedUser->setAddress($Address);
            $Education = $request->request->get('education');
            $About= $request->request->get('about');
            $Experience = $request->request->get('experience');
            $loggedUser->setAddress($request->request->get('address'));
            $loggedUser->setExperience($Experience);
            $loggedUser->setEducation($Education);
            $loggedUser->setAbout($About);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_receptionist_dashboard');
        }


        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('staff/edit_receptionist.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }
    /**
     * @Route("/edit_cleaner_credentials", name = "app_edit_cleaner_credentials")
     * @IsGranted("ROLE_CLEANER")
     */
    public function edit_cleaner_credentials(Request $request,UserInterface $loggedUser)
    {
        if ($request->attributes->get('_route') === 'app_edit_cleaner_credentials'
            && $request->isMethod('POST'))
        {
            $FirstName = $request->request->get('first_name');
            $LastName = $request->request->get('last_name');
            $Email = $request->request->get('email');
            $JoiningDate = \DateTime::createFromFormat('Y-m-d',$request->request->get('joining_date'));
            $EncodedPassword = $this->passwordEncoder->encodepassword($loggedUser,$request->request->get('password'));
            $ConfirmPassword =  $this->passwordEncoder->encodepassword($loggedUser,$request->request-
>get('conf_password'));
            $Designation  = $request->request->get('designation');


            $PhoneNumber = $request->request->get('phone');
            $Gender =  $request->request->get('gender');
            $Birthday = \DateTime::createFromFormat('Y-m-d',$request->request->get('birthday'));
            $Address =  $request->request->get('address');
            $Education = $request->request->get('education');
            $About= $request->request->get('about');
            $Experience = $request->request->get('experience');
            $hotelId = new Hotel();
            $loggedUser->setFirstName($FirstName);
            $loggedUser->setLastName($LastName);
            $loggedUser->setEmail($Email);
            $loggedUser->setJoiningDate($JoiningDate);
            $loggedUser->setPassword($EncodedPassword);
            $loggedUser->setPhone($PhoneNumber);
            $loggedUser->setGender($Gender);
            $loggedUser->setBirthday($Birthday);
            $loggedUser->setAddress($Address);
            $loggedUser->setAddress($request->request->get('address'));
            $loggedUser->setExperience($Experience);
            $loggedUser->setEducation($Education);
            $loggedUser->setAbout($About);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_cleaner_dashboard');
        }


        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
```

```
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('staff/jona.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }


    /**
     * @Route("/edit_staff/{id}", name = "app_edit_one_staff");
     * @IsGranted("ROLE_ADMIN")
     */
    public function edit_one_staff(Request $request, $id)
    {
        $User = $this->getDoctrine()->getRepository
        (User::class)->find($id);
        if ($request->attributes->get('_route') === 'app_edit_one_staff'
            && $request->isMethod('POST'))
        {
            $FirstName = $request->request->get('first_name');
            $LastName = $request->request->get('last_name');
            $Email = $request->request->get('email');
            $JoiningDate = \DateTime::createFromFormat('Y-m-d',$request->request->get('joining_date'));
            $EncodedPassword = $this->passwordEncoder->encodepassword($User,$request->request->get('password'));
            $ConfirmPassword =  $this->passwordEncoder->encodepassword($User,$request->request->get('conf_password'));
            $Education = $request->request->get('education');
            $About= $request->request->get('about');
            $Experience = $request->request->get('experience');
            $Designation  = $request->request->get('designation');
            $Role = [""];
            if($Designation=="Admin")
                $Role = ["ROLE_ADMIN"];
            else if($Designation=="Receptionist")
                $Role = ["ROLE_RECEPTIONIST"];
            else if($Designation=="Cleaner")
                $Role = ["ROLE_CLEANER"];

            $PhoneNumber = $request->request->get('phone');
            $Gender =  $request->request->get('gender');
            $Birthday = \DateTime::createFromFormat('Y-m-d',$request->request->get('birthday'));
            $Address =  $request->request->get('address');
            $hotelId = new Hotel();
            $User->setFirstName($FirstName);
            $User->setLastName($LastName);
            $User->setEmail($Email);
            $User->setJoiningDate($JoiningDate);
            $User->setPassword($EncodedPassword);
            $User->setRoles($Role);
            $User->setPhone($PhoneNumber);
            $User->setGender($Gender);
            $User->setBirthday($Birthday);
            $User->setAddress($Address);
            $User->setExperience($Experience);
            $User->setEducation($Education);
            $User->setAbout($About);
            $User->setAddress($request->request->get('address'));
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_all_staff');
        }

        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "/images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "/images/emtygirl.png";

            }
        }

        return $this->render('staff/edit_staff.html.twig',
```

```
            ['user'=>$User,
                'users' =>$Users,
                'images' => $images,
                ]);
    }



    /**
     * @Route("/delete_staff/{id}");
     * @IsGranted("ROLE_ADMIN")
     * @Method({"DELETE"})
     */
    public function delete_staff(Request $request, $id)
    {
        $user = $this->getDoctrine()->getRepository
        (User::class)->find($id);
        $entityManager = $this->getDoctrine()->getManager();
        $entityManager->remove($user);
        $entityManager->flush();
        $response = new Response();
        $response->send();
    }

    /**
     * @Route("/profile", name="app_profile")
     */
    public function profile(Request $request, UserInterface $loggedUser){
        $jona = "";

        if ($request->attributes->get('_route') === 'app_profile'
            && $request->isMethod('POST'))
        {
            $User = new User();
            $username = $request->request->get('username');
            $password = $request->request->get('password');
            $new_password = $request->request->get('new_password');
            $jona="Jona";

            if(!$this->passwordEncoder->isPasswordValid($loggedUser,$password))
                $this->get('session')->getFlashBag()->add(
                    'error',
                    'Your Password is Incorrect. Please type the right password'
            );

            else
            {
                $logged = $this->getDoctrine()->getRepository(User::class)->findOneBy(['email'=>$username]);
                $logged->setPassword($this->passwordEncoder->encodePassword($logged,$new_password));
                $entityManager = $this->getDoctrine()->getManager();
                $entityManager->flush();
                $this->get('session')->getFlashBag()->add(
                    'success',
                    'Password is Succesfullu Changed');
            }

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }

        return $this->render('staff/user_profile.html.twig',
            ['users' =>$Users,
                'images' => $images,
                'jona'=>$jona]);
    }
    /**
     * @Route("/receptionist/profile", name="app_receptionist_profile")
     */
    public function profile_receptionist(Request $request, UserInterface $loggedUser){
        $jona = "";

        if ($request->attributes->get('_route') === 'app_receptionist_profile'
            && $request->isMethod('POST'))
        {
```

```
            $User = new User();
            $username = $request->request->get('username');
            $password = $request->request->get('password');
            $new_password = $request->request->get('new_password');
            $jona="Jona";

            if(!$this->passwordEncoder->isPasswordValid($loggedUser,$password))
                $this->get('session')->getFlashBag()->add(
                    'error',
                    'Your Password is Incorrect. Please type the right password'
                );

            else
            {
                $logged = $this->getDoctrine()->getRepository(User::class)->findOneBy(['email'=>$username]);
                $logged->setPassword($this->passwordEncoder->encodePassword($logged,$new_password));
                $entityManager = $this->getDoctrine()->getManager();
                $entityManager->flush();
                $this->get('session')->getFlashBag()->add(
                    'success',
                    'Password is Succesfullu Changed');
            }

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }

        return $this->render('staff/receptionist_profile.html.twig',
            ['users' =>$Users,
                'images' => $images,
                'jona'=>$jona]);
    }
    /**
     * @Route("/cleaner/profile", name="app_cleaner_profile")
     */
    public function profile_cleaner(Request $request, UserInterface $loggedUser){
        $jona = "";

        if ($request->attributes->get('_route') === 'app_cleaner_profile'
            && $request->isMethod('POST'))
        {
            $User = new User();
            $username = $request->request->get('username');
            $password = $request->request->get('password');
            $new_password = $request->request->get('new_password');
            $jona="Jona";

            if(!$this->passwordEncoder->isPasswordValid($loggedUser,$password))
                $this->get('session')->getFlashBag()->add(
                    'error',
                    'Your Password is Incorrect. Please type the right password'
                );

            else
            {
                $logged = $this->getDoctrine()->getRepository(User::class)->findOneBy(['email'=>$username]);
                $logged->setPassword($this->passwordEncoder->encodePassword($logged,$new_password));
                $entityManager = $this->getDoctrine()->getManager();
                $entityManager->flush();
                $this->get('session')->getFlashBag()->add(
                    'success',
                    'Password is Succesfullu Changed');
            }

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
```

```
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }

        return $this->render('staff/cleaner_profile.html.twig',
            ['users' =>$Users,
                'images' => $images,
                'jona'=>$jona]);
    }
}
```

User Controller

```
<?php


namespace App\Controller;


use App\Entity\Hotel;
use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;
use Sensio\Bundle\FrameworkExtraBundle\Configuration\IsGranted;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
use Symfony\Component\Security\Core\User\UserInterface;

class UserController extends AbstractController
{

    /**
     * @var UserPasswordEncoderInterface
     */
    private $passwordEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }



    /**
     * @Route("/customer/profile", name="app_customer_profile")
     */
    public function profile_customer(Request $request, UserInterface $loggedUser){
        $jona = "";

        if ($request->attributes->get('_route') === 'app_customer_profile'
            && $request->isMethod('POST'))
        {
            $User = new User();
            $username = $request->request->get('username');
            $password = $request->request->get('password');
            $new_password = $request->request->get('new_password');
            $jona="Jona";

            if(!$this->passwordEncoder->isPasswordValid($loggedUser,$password))
                $this->get('session')->getFlashBag()->add(
                    'error',
                    'Your Password is Incorrect. Please type the right password'
                );

            else
            {
                $logged = $this->getDoctrine()->getRepository(User::class)->findOneBy(['email'=>$username]);
                $logged->setPassword($this->passwordEncoder->encodePassword($logged,$new_password));
                $entityManager = $this->getDoctrine()->getManager();
                $entityManager->flush();
                $this->get('session')->getFlashBag()->add(
                    'success',
                    'Password is Succesfullu Changed');
            }

        }
        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user) {
            if ($user->getPhoto()!=null)
```

```
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }

        return $this->render('customer/customer_profile.html.twig',
            ['users' =>$Users,
                'images' => $images,
                'jona'=>$jona]);
    }

    /**
     * @Route("/edit_customer_credentials", name = "app_edit_customer_credentials")
     * @IsGranted("ROLE_CUSTOMER")
     */
    public function edit_customer_credentials(Request $request,UserInterface $loggedUser)
    {
        if ($request->attributes->get('_route') === 'app_edit_customer_credentials'
            && $request->isMethod('POST'))
        {
            $FirstName = $request->request->get('first_name');
            $LastName = $request->request->get('last_name');
            $Email = $request->request->get('email');
            $Designation  = $request->request->get('designation');

            $PhoneNumber = $request->request->get('phone');
            $Gender =  $request->request->get('gender');
            $Birthday = \DateTime::createFromFormat('Y-m-d',$request->request->get('birthday'));
            $Address =  $request->request->get('address');
            $hotelId = new Hotel();
            $loggedUser->setFirstName($FirstName);
            $loggedUser->setLastName($LastName);
            $loggedUser->setEmail($Email);
            $loggedUser->setPhone($PhoneNumber);
            $loggedUser->setGender($Gender);
            $loggedUser->setBirthday($Birthday);
            $loggedUser->setAddress($Address);
            $Education = $request->request->get('education');
            $About= $request->request->get('about');
            $Experience = $request->request->get('experience');
            $loggedUser->setAddress($request->request->get('address'));
            $loggedUser->setExperience($Experience);
            $loggedUser->setEducation($Education);
            $loggedUser->setAbout($About);
            $entityManager = $this->getDoctrine()->getManager();
            $entityManager->flush();
            return $this->redirectToRoute('app_customer_dashboard');
        }


        $Users = $this->getDoctrine()->getRepository(User::class)->findAll();
        $images = array();
        foreach ($Users as $key => $user)
        {
            if ($user->getPhoto()!=null)
                $images[$key] = "".base64_encode(stream_get_contents($user->getPhoto()));
            else
            {
                if($user->getGender()=="Male")
                    $images[$key] = "images/Emptyprofile.png";
                else if($user->getGender()=="Female")
                    $images[$key] = "images/emtygirl.png";

            }
        }
        return $this->render('customer/customer_information.html.twig',
            ['users' =>$Users,
                'images' => $images,]);
    }

}
```

Google Controller

```
<?php

namespace App\Controller;

use KnpU\OAuth2ClientBundle\Client\ClientRegistry;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\Annotation\Route;

class GoogleController extends AbstractController
{
    /**
     * Link to this controller to start the "connect" process
     *
     * @Route("/connect/google", name="connect_google")
     * @param ClientRegistry $clientRegistry
     * @return \Symfony\Component\HttpFoundation\RedirectResponse
     */
    public function connectAction(ClientRegistry $clientRegistry)
    {
        return $clientRegistry
            ->getClient('google')
            ->redirect();
    }

    /**
     * Facebook redirects to back here afterwards
     *
     * @Route("/connect/google/check", name="connect_google_check")
     * @param Request $request
     * @return JsonResponse|\Symfony\Component\HttpFoundation\RedirectResponse
     */
    public function connectCheckAction(Request $request)
    {
        if (!$this->getUser()) {
            return new JsonResponse(array('status' => false, 'message' => "User not found!"));
        } else {
            return $this->render('hotel/view.html.twig')
;
        }

    }

    /**
     * @Route("/additional_info", name = "register_info")
     */
    public function setAdditionalInformation()
    {
        return $this->render('customer/customer_information.html.twig');
    }

}
```

User Fixtures – Faker

```php
<?php

namespace App\DataFixtures;

use App\Entity\User;
use Doctrine\Common\Persistence\ObjectManager;
use Doctrine\Bundle\FixturesBundle\ORMFixtureInterface;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
use Faker\Factory;
use Bezhanov\Faker\Provider\Educator;

class UserFixture implements ORMFixtureInterface
{
    private $passwordEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }

    protected $faker;
    public function load(ObjectManager $manager)
    {
        $this->faker = Factory::create();
        $this->faker->addProvider(new \Bezhanov\Faker\Provider\Educator($this->faker));

        for ($i = 0; $i < 10; $i++) {
            $user = new User();
            $user->setEmail($this->faker->email)
                 ->setFirstName($this->faker->firstName)
                 ->setLastName($this->faker->lastName)
                 ->setPhone($this->faker->phoneNumber)
                 ->setJoiningDate(new \DateTime())
                 ->setAddress($this->faker->address)
                 ->setEducation($this->faker->course)
                 ->setExperience($this->faker->jobTitle)
                 ->setAbout($this->faker->paragraph)
                 ->setGender($this->faker->randomElement($array = array ('Male', 'Female')))
                 ->setRoles($this->faker->randomElement($array = array (['ROLE_ADMIN'],
['ROLE_RECEPTIONIST'],['ROLE_CLEANER'])))
                 ->setPassword($this->passwordEncoder->encodePassword(
                    $user,
                    "jona123"
                ));

            $manager->persist($user);
        }
        $manager->flush();
    }
}
```

Inventory Fixtures – Faker

```php
<?php

namespace App\DataFixtures;

use App\Entity\Hotel;
use App\Entity\Inventory;
use Bezhanov\Faker\Provider\Commerce;
use Doctrine\Common\Persistence\ObjectManager;
use Doctrine\Bundle\FixturesBundle\ORMFixtureInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
use Faker\Factory;
use Bezhanov\Faker\Provider;
class ProductFixture extends AbstractController implements ORMFixtureInterface
{
    private $passwordEncoder;

    public function __construct(UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->passwordEncoder = $passwordEncoder;
    }

    protected $faker;

    public function load(ObjectManager $manager)
    {
        $this->faker = Factory::create();
        $this->faker->addProvider(new Commerce($this->faker));

        for ($i = 0; $i < 10; $i++) {
            $product = new Inventory();
            $product->setQuantity($this->faker->randomNumber(2));
            $product->setName($this->faker->productName);
            $hotel = $this->getDoctrine()->getRepository(Hotel::class)->find(1);
            $product->setHotel($hotel);
            $manager->persist($product);
        }
        $manager->flush();
    }
}
```

Google Authenticator

```
<?php


namespace App\Security;

use App\Entity\Customer;
use Doctrine\ORM\EntityManagerInterface;
use KnpU\OAuth2ClientBundle\Client\ClientRegistry;
use KnpU\OAuth2ClientBundle\Security\Authenticator\SocialAuthenticator;
use League\OAuth2\Client\Provider\GoogleUser;
use Symfony\Component\HttpFoundation\RedirectResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Routing\RouterInterface;
use Symfony\Component\Security\Core\Encoder\UserPasswordEncoderInterface;
use Symfony\Component\Security\Core\User\UserProviderInterface;

class GoogleAuthenticator extends SocialAuthenticator
{
    private $clientRegistry;
    private $em;
    private $router;
    private $passwordEncoder;

    public function __construct(ClientRegistry $clientRegistry, EntityManagerInterface $em, RouterInterface $router,
UserPasswordEncoderInterface $passwordEncoder)
    {
        $this->clientRegistry = $clientRegistry;
        $this->em = $em;
        $this->router = $router;
        $this->passwordEncoder = $passwordEncoder;

    }


    public function supports(Request $request)
    {
        return $request->getPathInfo() == '/connect/google/check' && $request->isMethod('GET');
    }

    public function getCredentials(Request $request)
    {
        return $this->fetchAccessToken($this->getGoogleClient());
    }

    public function getUser($credentials, UserProviderInterface $userProvider)
    {
        /** @var GoogleUser $googleUser */
        $googleUser = $this->getGoogleClient()
```

```
            ->fetchUserFromToken($credentials);

        $email = $googleUser->getEmail();

        $customer = $this->em->getRepository('App:Customer')
            ->findOneBy(['email' => $email]);
        if (!$customer) {
            $customer = new Customer();
            $customer->setEmail($googleUser->getEmail());
            $customer->setFirstName($googleUser->getFirstName());
            $customer->setLastName($googleUser->getLastName());
            $customer->setJoiningDate(new \DateTime());
            $customer->setPassword($this->passwordEncoder->encodePassword(
                $customer,
                "jona123"
            ));
            $customer->setState("Albania");
            //$user->setC(new \DateTime(date('Y-m-d H:i:s')));
            $this->em->persist($customer);
            $this->em->flush();



        }

        return $customer;
    }

    /**
     * @return \KnpU\OAuth2ClientBundle\Client\OAuth2Client
     */
    private function getGoogleClient()
    {
        return $this->clientRegistry
            ->getClient('google');
    }

    /**
     * Returns a response that directs the user to authenticate.
     *
     * This is called when an anonymous request accesses a resource that
     * requires authentication. The job of this method is to return some
     * response that "helps" the user start into the authentication process.
     *
     * Examples:
     *  A) For a form login, you might redirect to the login page
     *      return new RedirectResponse('/login');
     *  B) For an API token authentication system, you return a 401 response
     *      return new Response('Auth header required', 401);
     *
     * @param Request $request The request that resulted in an AuthenticationException
     * @param \Symfony\Component\Security\Core\Exception\AuthenticationException $authException The exception that
started the authentication process
     *
     * @return \Symfony\Component\HttpFoundation\Response
     */
    public function start(Request $request, \Symfony\Component\Security\Core\Exception\AuthenticationException
$authException = null)
```

```
    {
        return new RedirectResponse('/login');
    }

    /**
     * Called when authentication executed, but failed (e.g. wrong username password).
     *
     * This should return the Response sent back to the user, like a
     * RedirectResponse to the login page or a 403 response.
     *
     * If you return null, the request will continue, but the user will
     * not be authenticated. This is probably not what you want to do.
     *
     * @param Request $request
     * @param \Symfony\Component\Security\Core\Exception\AuthenticationException $exception
     *
     * @return \Symfony\Component\HttpFoundation\Response|null
     */
    public function onAuthenticationFailure(Request $request,
\Symfony\Component\Security\Core\Exception\AuthenticationException $exception)
    {
        return new RedirectResponse($this->router->generate('app_login'));
    }

    /**
     * Called when authentication executed and was successful!
     *
     * This should return the Response sent back to the user, like a
     * RedirectResponse to the last page they visited.
     *
     * If you return null, the current request will continue, and the user
     * will be authenticated. This makes sense, for example, with an API.
     *
     * @param Request $request
     * @param \Symfony\Component\Security\Core\Authentication\Token\TokenInterface $token
     * @param string $providerKey The provider (i.e. firewall) key
     *
     * @return void
     */
    public function onAuthenticationSuccess(Request $request,
\Symfony\Component\Security\Core\Authentication\Token\TokenInterface $token, $providerKey)
    {
        return new RedirectResponse($this->router->generate('app_login'));
    }
}
```