Hotel Management System

System Design

Table of Contents

SYSTEM DESIGN DOCUMENT [1]

1. Introduction

1.1. Purpose of the System

The main purpose of the Hotel Management System is to make Hotel Staff comfortable to manage their employees and perform tasks, allow them to ensure their guests are pleased by their accommodation and experience. The system also creates a safe and fast booking and satisfying interface for the guests. By using our system to book a room, the user no longer needs to call the hotel and spend too much time on it. As well as being able to book a room in the hotel online safely and in a fast manner. Hotels can easily manage what their employees must do daily, get reports weekly about employee's work, feedbacks, and workloads, reach every customers and rooms details. Managers can access their employee's payrolls, and all financial expenses of their hotel.

1.2. Design Goals

We aimed to develop a user-friendly interface. For this, attention has been paid to the readability of the texts and the color harmony between the elements on the window. Colors, that are easy on the eyes and not distracting, have been chosen. The design has been improved with some tactics such as shading, border, border-radius and what is called "negative space" in design. Necessary spaces have been left in appropriate places on the page. At the same time, the size of the text and elements (cards, buttons, etc.) was carefully adjusted. Our system is designed so that the user does not waste too much time to perform an action and can take the desired action in a short time, we avoided overwhelming the user with everything placed on the main page, and instead placed everything under categories that feel intuitive and easy to access.

1.3. Definitions, Acronyms, and Abbreviations

User: any person who uses the system.
Guest: any person with the intention to spend time in the hotel and use its services.
Employee: any person working at the hotel in any capacity.
Admin: administrator of the hotel also known as General manager.
Manager: manages staff and receptionists and deals with events hosted by the hotel, deals with guests in cases where he is needed.
Receptionist: deals with guest related matters and helps guests when necessary.
Staff: Can be cleaners, bellboys, waiters, cooks, and chefs.

RAD: Requirement Analysis Document.
SDD: System design Document
GUI: A GUI or graphical user interface is a form of user interface that allows users to interact with electronic devices through a graphical interface.
HTML: Hyper Text Markup Language. It is the standard markup language for documents designed to be displayed in a web browser.

CSS: Cascading Style Sheets. CSS is a style sheet language used for describing the presentation of a document written in HTML.
UX: User Experience. UX abbreviation is used to define the design process to create products that provide meaningful and proper experiences to users.
MVC: Model, View, and Controller. MVC abbreviation is used to define a software design pattern commonly used for developing user interfaces that distribute the program logic into three elements.
JS: JavaScript. JavaScript is the main programming language of the Web.
IDE: Integrated Development Environment. And IDE is a software application that provides facilities to programmers for software development.
TCP/IP: Transmission Control Protocol and the Internet Protocol.

1.4. References

Currently there is no system in place to be replaced by the system we are building.

2. Current Software Architecture

As described in the section before, there is no system to be replaced by the one we are building.
Thus, here is **a survey of current architectures for similar systems:**
There are … architectures we considered when we looked at similar systems:

- **Layered architectural style**, because it our system is web based, this means we will have calls between the browser and the web server, which makes it a runtime dependency relationship between our layers. As We will show in our hardware software mapping (section 3.3), we assumed the general case where the client layer and server layer are running on different machines, which will be the normal case, but that does not mean both layers cannot be running on the same machine, which will be the case during testing. For the hierarchical decomposition both open and closed architectures for the layered architectural style did not fit our system as it doesn't provide interactivity for our users.
- **Repository Architectural Style,** we considered this style because it is used for database management systems and it's well suited for applications with constantly changing, complex data processing tasks, however the central repository can quickly become a bottleneck, both from a performance aspect and a modifiability aspect, plus it doesn't provide the level of interactivity we want in our system, which is provided by another Architectural Style.
- **Model View Controller architecture** (MVC): a special case of the repository where Model implements the central data structure and Control objects dictate the control flow. subsystems are classified into three different types: Model subsystems maintain domain knowledge, do not depend on any view or controller subsystem →Entity objects. View subsystems display it to the user →Boundary objects. Controller subsystems manage the sequence of interactions with the user →Control objects.
  MVC is well suited for interactive systems, especially when multiple views of the same model are needed. MVC can be used for maintaining consistency across distributed data, it does however introduce the same performance bottleneck as for other repository styles, but with the advantage of providing interactivity.

- **Client Server Architectural Style**: Often used in the design of database systems, Front-end: User application (client),
  Back end: Database access and manipulation (server)
  Functions performed by client: Input from the user (Customized user interface), Front-end processing of input data
  Functions performed by the database server: Centralized data management, Data integrity and database consistency, Database security.
  This option was dropped for the same reason as the others, lack of interactivity.

3. Proposed Software Architecture

Our system architecture can be classified as a **Model View Controller** architecture, we chose this architecture because it ensures interactivity as well as maintaining consistency across distributed data, the disadvantages discussed in the previous section are not a big factor that will affect our system. On top of that we value the interactivity this will bring to the users of our system.

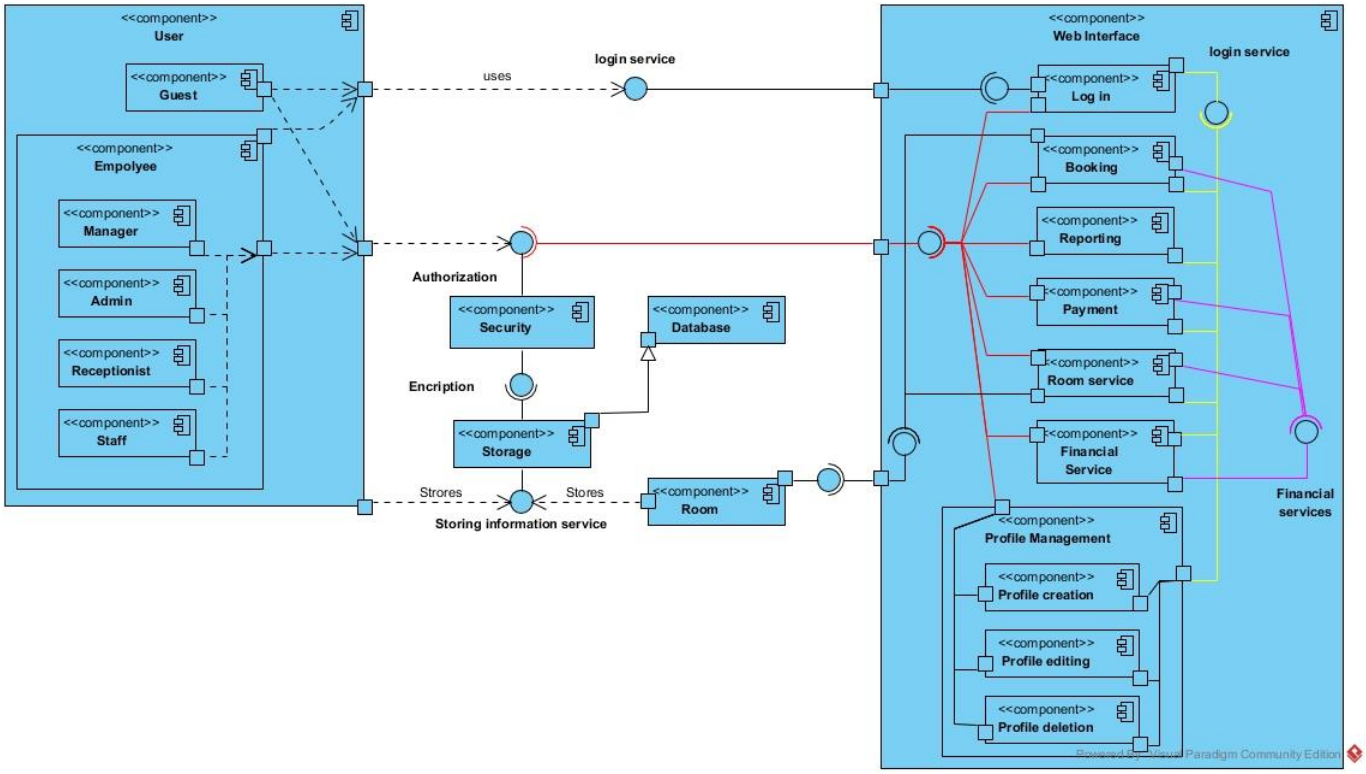
3.1. Overview

With log in subsystem, system users can access the system according to their roles. The system also has booking subsystem, which allows hotels guests to book their rooms. Reporting subsystem creates reports for our system's Manager/Admin users according to the Manager/Admin's requested information. The system uses payment subsystem with room service and booking subsystems, this subsystem deals with payment related matters, for both hotel employee side and guest side. Room service subsystem provides ease with room service requests. With that subsystem hotel guests can easily request room service on the system. The systems financial service subsystem provides financial services for booking, payment, and room service subsystems. Also, the system has profile management subsystem. This subsystem has 3 subsystems. First one is profile creation subsystem, which allows profile creation to the users. Second one is profile editing; this subsystem provides editing on the user's profile in case of any change needed. Third one is profile deletion; this subsystem provides deletion of a user from the system. With this subsystem, hotel manager or admin can delete the users from the system in case of need. The system has security subsystem, which provides service to the system itself. This subsystem provides authorization service to identify the user and form the web interface according to the user's role. Also, security provides services for systems financial needs, like booking or payment. And finally, security subsystem provides encryption service to storing side for needed information. Finally, we have storage and database subsystems. Storage is a subsystem that is created for concurrency issues. Basically, it gives second layer of storing to the system. Data is stored in here at first then the data is transferred to the real database subsystem.

3.2. System Decomposition

In our System we deal with the complexity of the project by dividing our system into subsystems like payment, room, booking services.

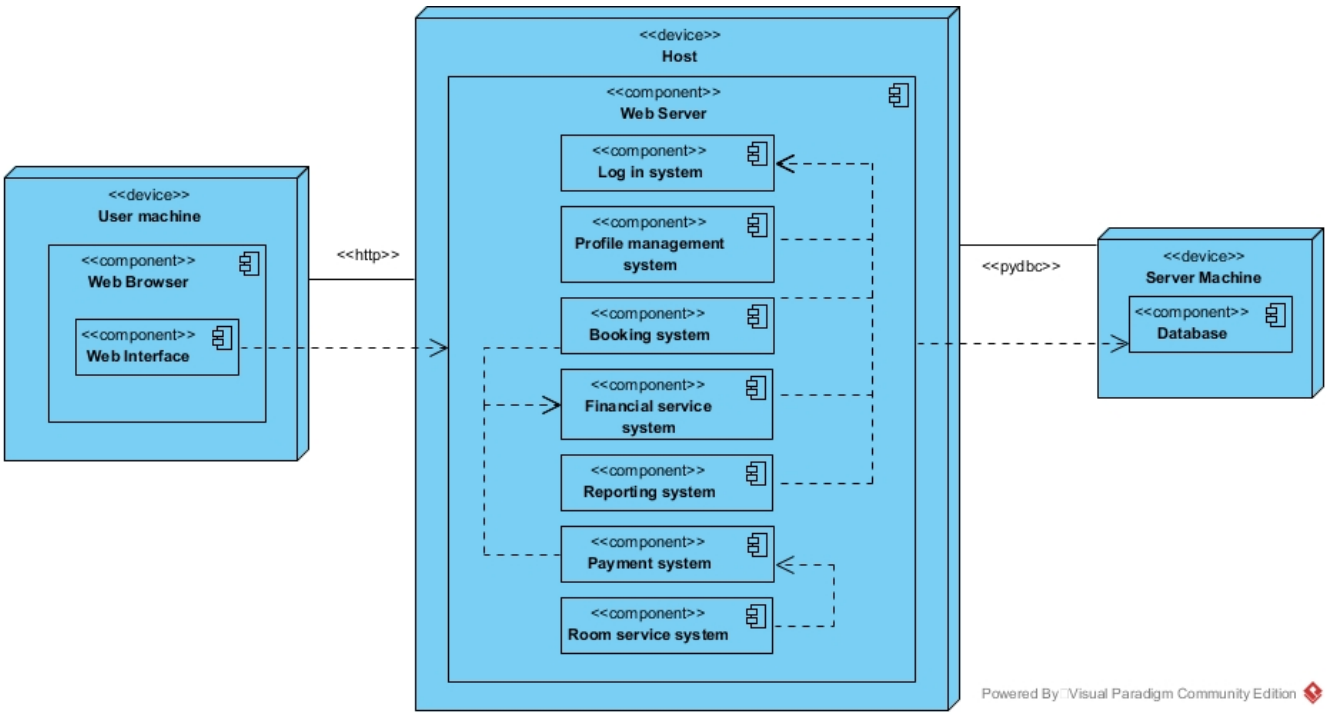
Also, with help of the MVC Design structure, our system can handle the requests easier because every subsystem has their own controller systems. This lets us make the project more organized, maintainable, and readable because the system is decomposed into smaller pieces.



3.3. Hardware Software Mapping

Our system is a web-based solution and not on-premises solution. Our system makes it possible to access and manage your hotel from anywhere you have internet access. Web-based hotel solutions offer sleek and easy to learn interfaces that allow hotel staff to focus on putting the guest first instead of struggling with overbearingly complex, and sometimes redundant, software. The user will be able to access to the system, in its full capacity, from anywhere the user has access to a web browser. The user will always have the latest version.

< Hotel Management System >



3.4. Persistent Data Management

Persistent data:

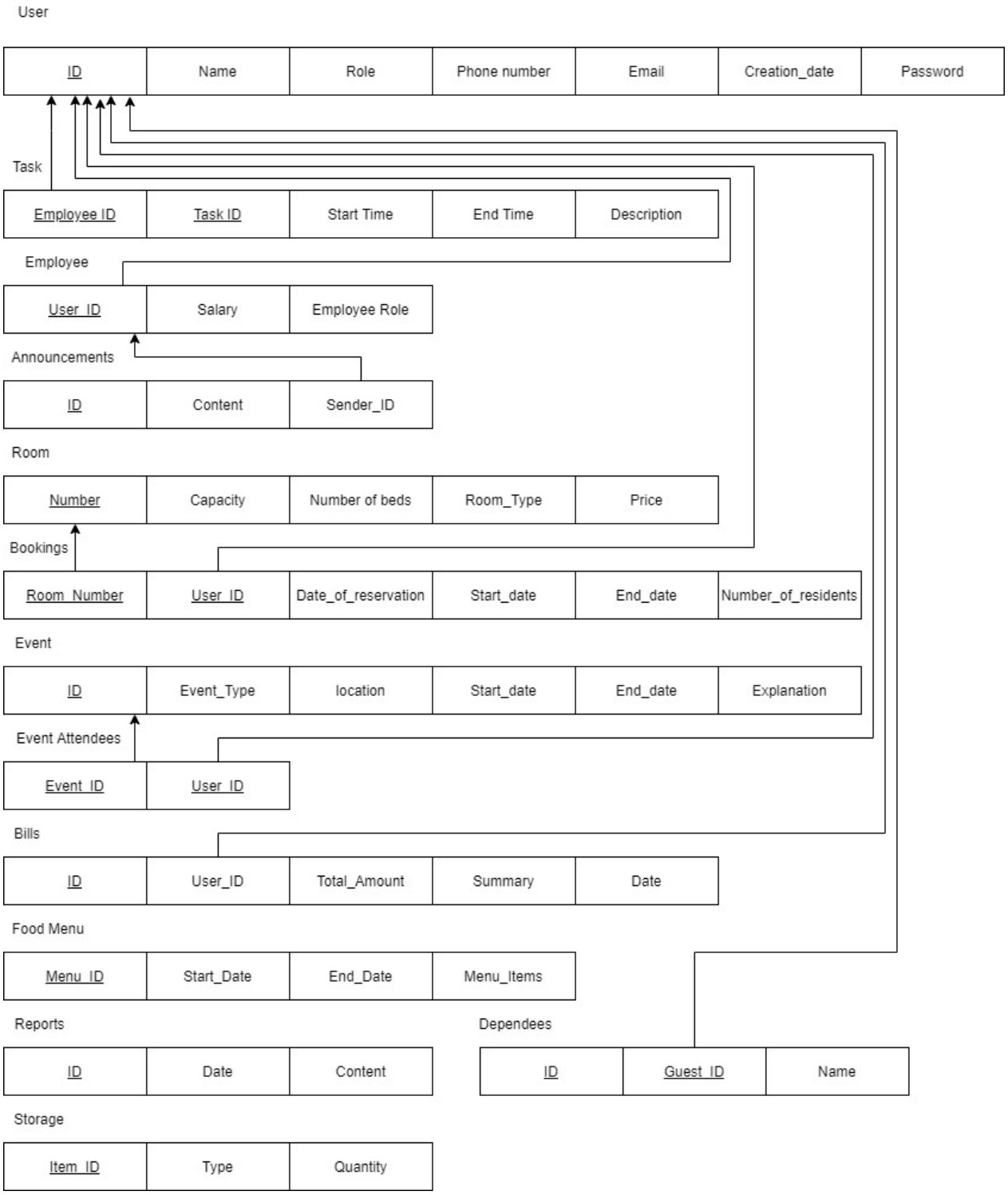
- User information: includes personal information, for employee's financial information and shifts.
- Room information: includes room details (number of beds, conditioning, TV)
- Event's information: includes dates, place, attendees, capacity, type of event.
- Announcements information: date, sender, recipients, description.
- Tasks information: employee name, place, date, time, status.

Data management infrastructure:

The system stores the persistent data related to the users and bookings, rooms, announcements, and events in tables as a database.

The system uses three different layers for encapsulation these are: models, views, and controllers. We get the data from the database with help from our models and controllers, then we get the results back to the user through the same steps.

< Hotel Management System >



3.5. Access Control and Security

3.5.1 Access Control

- System admin has access to all aspects of the system. (System admin represents the engineers that are responsible for deploying and maintaining the system.)
- Admin has access to all hotel management related except direct access to the database. (System can access the database).
- Manager has access to all staff and guest related subsystems except the salary and hotel's financial subsystems.
- Receptionist has access to guest related subsystems.
- Staff has access to service-related subsystems.
- Guest has access to booking and their own financial subsystems (requesting bill, requesting room service and refund.).

	Create account	Guest information	Room information	Employee information	Event information	Booking	Room service	Employee task	Announcement	Finance	Database
System admin	Create()	View() Edit()	View() Edit()	View() Edit()	View() Edit()	View() Edit() Book()	View() Edit()	View() Edit()	View() Edit() Create()	View() Edit()	Access() Update()
Administrator	Create()	View() Edit()	View() Edit()	View() Edit()	View() Edit()	View() Edit() Book()	View() Edit()	View() Edit()	View() Edit() Create()	View() Edit()	
Manager	Create()	View() Edit()	View() Edit()	View() Edit()	View() Edit()	View() Edit() Book()	View() Edit()	View() Edit()	View() Edit() Create()	View()	
Receptionist	Create()	View()	View()	View()	View()	View() Edit() Book()			View()	View()	
Staff	Create()			View()			View() Edit()	View() Update()	View()	View()	
Guest	Create()	View()			View()	Book()	Request() Edit()		View()		

3.5.2 Security:

- The system will not store password as text in our database, it will instead store a hashed version of the password to prevent exposing sensitive data in case of a breach, because storing passwords in text format will make it easy for anyone who gets access to the database all passwords, hashing is a security measure. We are using **bcypt** algorithm to manage the passwords and hashing them, hashing is the process of converting a given password into another value. A hash function (in this case **bcrypt**) is used to generate the new value according to a mathematical algorithm.

Example:
hashUsing**bcypt(**password123)=
$2y$12$vSKoOf8DJElTy4XHXd3U6eosrkiY5mW24UUs8RADAuL6EuUVokqOu

- The system uses two factor authentication (sending an email to the user) for all payments on the system to prevent accidental purchases or unintended actions. If a user clicks on purchase or pay button accidentally or not, we should verify if that action was intended or not by the user.

- The system has two-factor authentication for changing the password via e-mail to verify the identity of the user and only allow authorized users to perform such tasks. If a user clicks on change password button, or someone has his password and wants to change it, we should verify if that action was intended or not by the actual user, we assume only the user has access to his email address, and so s/he will be the only one permitted to change the password.

- The system automatically logs out the user if the user is inactive for more than 5 minutes to prevent others from accessing the user's machine while the user is not on it. If we by default allow long session timeouts, then the risk is on shared computers - future user accesses the session of the current user. If we try to blame the user for not logging out, then we will find your user base dwindle and we will have to deal with a lot of end user complaints.

3.6. Global Software Control

Our Hotel Management System is event-driven. Our system should fulfill many users' initiated requests at the same time. To deal with this situation our system uses event handlers. With these event handlers we can monitor the movement of the user, such as clicking or pressing a key. In other words, our system provides the users the ability to interact with the system at the same time and with different events. When a user interacts with our system the request will be sent only to the relevant subsystem. So other subsystems will not be busy, non-relevant subsystem will just ignore the request and will wait for another request from another or the same user.

For synchronization, the system will synchronize based on three factors: Startup, Authentication and Scheduling. On system startup the users will get a synchronized version of the page; same thing for users that just completed a successful authentication(login), and finally, a scheduled synchronization will be triggered by the system every few minutes (defined by the system admins).

3.7. Boundary Conditions

Start-up:
The data that needs to be accessed at startup time is database connection information.
The services that must be registered are domain name service (DNS). In our case we will be using a local host or Github hosting service.
The user interface will show the login page at startup time.

Shutdown:

No subsystem can terminate while the rest of the system is running unless it has no effect on the rest of the tunning subsystems.
Only in the case of complete system shutdown are all the subsystems allowed to shut down.
Other subsystems are notified in case one of the other subsystems terminates, if that subsystem was necessary for one of the running subsystems, they will notify the user of the current subsystem that is down and give meaningful feedback.
The updates are communicated to the database in real time, using the agreed-on protocols (MVC).
In case of intentional shut down, the users will be notified with a message.

Error behavior:

In the case of a communication link fail between server and web application, the system gives feedback to the user in the form of a dialog box. Or the system redirects the user to the previous page to the one they are working on or shows a 404-error page depending on the kind of error.
In case of failure (unintentional shut down) all the database information is automatically saved to our local machines(periodically), and upon restarting the system, the information is gathered from the server automatically and the failed communication link is reestablished.

4. Subsystem Services

Log in:
Provides the login service for all the users of the system, and acts as a gateway to access the other subsystems by giving authentication to the users.
This component is used by all the components of Web Interface and by **user** component.
This component uses **security** component.

Booking:
Provides the booking service for the guests and uses the financial service to bill the guest in accordance with his reservation and the **Room** service component to list the available rooms.
This component uses: **Financial Services, Login, Room, Security** components.

Reporting:
Provides reporting system to create reports related to all aspects of the system, depending on the user's permissions.
This component uses: **Login, Security** components.

Payment:
Provides the payment system for guests to pay for the services provided by the hotel, such as room reservations and bookings and rooms services. This subsystem is used when checking out of the hotel, it creates a bill for the user with the total amount to pay. And it uses the security subsystem to verify the payment information with tow factor authentication.
This component uses: **Financial Services, Login, Security** components.

Room service:
Provides available room services to guests such as food orders and room cleanings.
This subsystem uses the room subsystem to list the available services depending on the room. It uses the financial service subsystem to bill the user accordingly.
This subsystem uses: **Financial Services, Login, Room, Security** subsystems.

Financial service:
Provides all the financial related services used by the other subsystems. It provides the pricing of the rooms, room services, and other activities that require billing.
This component uses: **Login, Security** components.

Profile management:
Provides **account creation** component and **account editing** and **deletion** for users, it uses the **authorization** component to allow only certain users to access the profile management functionality. (System admins, Administrators, Managers).
This component (and it's subcomponents) uses: **Login, Security** components.

Security:
Provides security protocols for other components, such as two factor authentications for the payment component, and encryption for passwords stored in the database.

Room:
Provides the necessary information related to hotel rooms and is used to list the available rooms and occupied ones as well.

5. References

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.
2. Ian Sommerville (2015). Software Engineering, Pearson tenth edition.
3. http://www.cs.sjsu.edu/faculty/pearce/modules/lectures/ooa/requirements/IdentifyingURPS.htm