**Table of Contents**

1. [Introduction 3](#_bookmark0)
   1. [Abstract 3](#_bookmark1)
   2. [ISO 4-5](#_bookmark2)
   3. [HMS History 6](#_bookmark3)
2. [State of Art 7](#_bookmark6)
   1. [Related Systems 7-11](#_bookmark4)
   2. [Comparison 12](#_bookmark5)
3. [System Analysis 13](#_bookmark7)
   1. [Overview/introduction 13](#_bookmark8)
   2. [Context Diagram 14](#_bookmark9)
   3. [LVL 0 Context 15-17](#_bookmark10)
   4. [Data Dictionary 18-22](#_bookmark10)
   5. [Process 22-41](#_bookmark10)
4. [System Design 41](#_bookmark26)
   1. [Database ER-Table 41](#_bookmark10)
   2. [GUI 42-54](#_bookmark10)
5. Implementation
   1. [Soft and Hard Ware Used 55-56](#_bookmark10)
   2. [Database Implementation 57-](#_bookmark10)71
6. References and Conclusions
   1. [References 72](#_bookmark10)
   2. [Conclusions 72](#_bookmark10)
7. [Code 73](#_bookmark0)-101

Introduction

# **Abstract**

A **hotel reservation system**is the mechanism through which guests can create secure online reservations. While the process is similar to booking with an online travel agent (OTA), the difference is the hotel’s booking engine essentially links up to their own website so that there are no additional fees incurred for the property.

Through the hotel reservation system software, guests can choose how long they will stay, the type of room they want, get add-ons, and pay securely online through a payment platform. In this article we’ll go in depth about what a hotel reservation system is, how it works, and the benefits that it offers to the hospitality sector. So let’s get started.

## **What is a hotel reservation system?**

A hotel reservation system is a software application that **allows guests to book directly with the hotel online**, with no intermediaries necessary. The software essentially processes online reservations made via the hotel’s website and then passes this information to the hotel’s own backend so that the information can be easily accessed. Bookings are then managed by hotel staff.

With the boom of the Millennial traveler, now **more than 700 million people are expected to book primarily online by 2023**, so having an online reservation system is key to reaching a widespread audience. It is also key to generating a good first impression because guests are able to place bookings without having to navigate to another domain. Keeping the whole process internally prevents clients from navigating away from the page before making the final booking.

## **How does a hotel reservation system work?**

The [reservation management software](https://www.mews.com/en/products/reservation-management) essentially automates the booking process, syncing up with your hotel’s website and social media so that guests can reserve easily and conveniently without having to visit another site. The reservation system can also link up to the channel manager, allowing hotels to distribute their availability to both online agents in real time.

These types of booking systems are used in the hospitality industry in order to allow guests to book their own stays online through software. It helps facilitate a smoother booking process and easily gathers data into your system. Furthermore, it gives you an edge over your competitors and improves efficiency by automatically syncing across your system and channels.

This type of software is**key to having global appeal and to increased occupancy rates**. **It’s an essential tool to increase reservations, create enhanced guest experiences, leading to higher customer retention and increased revenue.**

## **What are the benefits of using a hotel reservation system?**

Nowadays, having software to manage bookings is without doubt fundamental, no matter the size of the property. So let’s take a look at why it’s necessary and the benefits that an online booking system brings.

### Improved efficiency

**A reservation system improves efficiency for hotels,** primarily because availability is updated immediately across all channels, including your own system. This reduces the amount of time needed for the front desk to dedicate to administrative tasks and also minimizes the risk of [overbooking](https://www.mews.com/en/blog/hotel-overbooking-strategy).

Furthermore, an online booking system has the ability to automate tasks like sending out booking confirmations so that staff can spend their time taking care of other important areas like providing top-notch customer service. Since guests are essentially in charge of generating their own reservations, bookings arrive into the system with all the necessary information already in place – all the reception staff have to do is get ready for their arrival. Automating processes is the key to having a successful and efficient operation during the pre and post trip process.

### Gives you a competitive edge

In the digital age, there’s no excuse for not being online. However, some smaller hotels don’t want to invest right away in software to manage their own bookings and thereby have to forfeit some of the profit gains to intermediaries.

Hoteliers can also customize the software to implement a clear branding message throughout the buyer journey, generating confidence, and therefore leading to conversions. The booking engine can reflect your [hotel branding](https://www.mews.com/en/blog/hotel-branding) within your own unique URL, giving a sense of security to the guests and making them want to book directly with you, an advantage competitors won’t have if they don’t have their own booking engine. Plus, in this way, booking experiences are more functional and user-friendly than through a third party.

Furthermore, since the vast majority of people are now booking online,**having the ability to book online gives you an edge over those businesses that have not yet invested in the software to make this possible**. When guests book directly through the hotel’s own reservation system, hotels don’t have to share the profit, so they can also offer better prices to guests.

### Reduces chances of human error

When you have your own system, guests are in charge of making their own reservations, so presumably, there will be no errors in guest details and contact information. However, should an error occur, the hotel will not be held accountable, hence keeping your reputation intact. Moreover, since the reservation system syncs up with channel managers, the likelihood of an overbooking due to human error is little to none. Reducing human error in bookings allows a better service and frees up time to take care of other tasks rather than cleaning up errors.

### Data collection is easier

Collecting customer data is key for marketing purposes, and an online booking system is a way to ensure that data collection is easy and that all guest data is in one place. You can easily understand guest demographics, preferences, what amenities they prefer, where they are coming from, and how many guests are traveling. **You can use this information to tailor your marketing strategies and improve the overall guest experience, thus boosting retention rates.**

### Easily manage rewards

A booking system allows you to keep track of your reward customer’s stays and allot them benefits accordingly. Guests can keep track of their rewards by having a unique login so that they input their information once and then from thereon out, it’s the system that monitors their rewards status, with little maintenance necessary from staff.

Since [hotel loyalty programs](https://www.mews.com/en/blog/hotel-loyalty-programs) are an important means of guest retention and keeping occupancy up in low season, the simpler it is to administer rewards, the easier it is for your staff to keep track of, and the more likely it is you’ll be able to keep the guests coming back thanks to rewards.

### Conclusion

We’ve looked at what a hotel reservation system is, how it works, and the benefits to the hospitality industry. The benefits of an online booking system range from reducing human error to improved efficiency, from making data collection and rewards management easier to giving you a competitive edge.

With the advent of technology,**having a hotel reservation system software is the one of the easiest changes a hotel can implement to reap the benefits** mentioned in this article. It allows you to keep occupancy up and ensure everything is running smoothly with little intervention from hotel staff. The more time that is freed up for staff to take care of administrative duties, the more time they can dedicate to offering the best service possible.

# Background history about hotels Booking systems

Technology advancements have had a dramatic impact on the hotel industry throughout history. In the 1950’s travel lodging really started to explode and shape what the industry looks like today.

—Justin De Rise

Part of a new blog ‘Mini-Series’ where we are looking at the evolution of hotel technology – both staff-facing and guest-facing – and its impact on the guest experience.

Technology has probably done more to shape our expectations and preferences of our hotel stay than any upgrade or freebie could offer; and if it wasn’t for these advancements in the 1950’s, our hotel experience might look and feel a bit different today. However, while hotels have evolved a great deal since and while some changes are for the better, some items make us nostalgic for those clichéd “good old days”.

Hotels have been around a long time, popping up in the US during the 18th century; but it wasn’t until the beginning of 1950s that the lodging industry started to explode. With the end of the Second World War, combined with affordable cars and a comprehensive Interstate system, the economies of the western world boomed and led to the start of a consumer-led economy. While many considered the 1950’s as one of the most conservative decades, it was a time when technology underwent a dramatic evolution.

## **Booking and checking-in**

Before the 1940’s, reservations were booked by mail, telegram, or telephone, and often resulted in lost or forgotten reservations. It was only in 1947 the first hotel reservation system ‘Hoteltype’ was established by Westin. The new machines enabled instantaneous confirmation of

reservation requests. Roughly ten years later in 1958 Sheraton advanced Westin with the introduction of ‘Reservatron’, the industry’s first automated electronic reservation system, and the first toll-free reservation number.

**Pricing**

The average room rate would have set you back approx. $5… per night! (Gasoline was a little less than 20 cents a gallon too – those were the days!).​​

Prior to the 50’s only a select amount of countries had television available; with massive growth throughout the decade, by the end of the 50’s almost the entire western hemisphere had television. If you were lucky to stay at the Hilton, they were the first chain to install television sets in

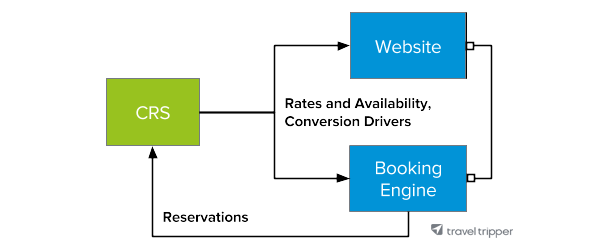
all their guest rooms back in 1951. In the Mid 1950’s the Atlas Hotels developed the first in-room coffee concept (where would we be without our in-room tea and coffee!) and in 1957 Hilton began offering a direct-dial telephone service.

**Payment**

With the explosion of hotel stays in the 1950’s (while at only $5/night) they needed to be paid for somehow, and while cash is always a viable option, it was the 50’s that saw the introduction of the very first credit card.

While The Diner’s Club Card was the first to launch, it was exclusively for restaurant dining and expected full payment each month. But in 1958, American Express’ was introduced and with it offered consumers staggered payments and “free-range shopping”.

Check us out next week when we’ll be looking at the hotel stay of the 1960’s and what additional “technological” advancements they had on the previous era and also how they compare to hotels of today!



A modern model for the hotel central reservation system.

In this model, the website is able to pull info directly from the CRS, allowing travelers to browse rates for specific dates without entering the booking engine. Travelers are also able to browse various rooms without having to re-enter dates.

State of the art

# Related Systems

# 

### **Vertical Booking (CRS) Overview**

Vertical Booking is a leading, global reservation technology provider with hotel clients in 107 countries worldwide, offering solutions translated into 29 languages and usable in all currencies. No matter the size of your property or group, Vertical Booking can provide a customizable solution that supports the entire guest booking cycle - including a booking engine, GDS distribution, channel manager, OTA rate comparison tool and a reservation call center application - all through one integrated CRS dashboard.

# 

### **Pegasus CRS (Cendyn) Overview**

Take full control of your hotel’s distribution strategy with the Pegasus CRS, an award-winning cloud-based reservations solution that offers unprecedented flexibility, scalability, security and support in managing your rates and inventory. We offer comprehensive connectivity to all major distribution channels, while prioritizing conversion on your most profitable channel: your website. Our five-star support team of revenue, marketing and sales experts works proactively to ensure success for your hotel.



### **Amadeus – iHotelier® Reservations & Booking Engine Overview**

Amadeus' market-leading reservations and booking engine solutions deliver unmatched flexibility and scale of distribution for independent hoteliers and mid-scale chains. Create new demand and increase revenue with our powerful iHotelier Central Reservations System by automating distribution across web, mobile, voice, travel agent, OTA, and metasearch channels. Leverage our booking engine’s data-driven design to engage more users and remove barriers to conversion. Centralize management of ARI to decrease the time, effort and cost of distribution. Work with one single vendor to meet all your distribution and booking requirements.



### **Revinate (RezForce Call Center) Overview**

Our highly trained, expert agents capture bookings when your team is busy or unavailable. We hire train and performance test only the best US-based agents who deliver an industry-leading 99.89% booking accuracy rate for our clients. Every call results in guest lead data to capture second chance bookings and grow your marketing database. With RezForce Lux, agents are trained with Forbes Travel Guide’s 5-Star certification standards, and consistently earn an overall satisfaction score of 99%. Increase profitability by balancing fixed and variable labor costs that flex with your business. Our pay-for-performance pricing model means you only pay us when we handle and/or book your reservation calls.



### **RateTiger CRS (by eRevMax) Overview**

RateTiger CRS is a complete distribution platform - offering fully customizable connectivity solution to hotels. It helps hotels automate room pricing, online connectivity, channel management, direct booking as well as all the aspects of online distribution, so that they can spend more time creating wonderful experiences for the guests. RateTiger CRS centralizes the core actions in a single sign-on platform providing the much-needed simplicity in daily operations. Product Benefits 1. Widens distribution of inventory at a lower cost. 2. Eliminates the need to manage individual channel extranet. 3. Distributes property data, rates and availability directly from hotel system. 4. Allows for creation of distinct rate strategies by channel. 5. Ability to build customized booking engine for hotel / brand website. Product Features 1. Centralized hotel management with call-to-actions from various providers, helping hotels make informed decisions. 2. Ability to build rates, policies, and Global Distribution System (GDS) access codes at Chain level & Individual hotel level and distribute to 650k+ Travel agents, Consortia, and Representation companies. 3. Increase hotel direct business through RateTiger Booking Engine that offers customizable, user-friendly interface for a great customer experience. It also offers the ability to apply promo-code and connection with loyalty programs & CRM to help increase direct bookings. 4. Metasearch Distribution - Ability to connect to multiple meta channels like GHA, TripAdvisor, Trivago to redirect more traffic to brand website and increase conversion. 5. Content Management - Ability to push room type, details, policies etc. directly from CRS to hotel Booking Engine and other OTAs like Booking.com.

# 

### **Windsurfer CRS by SHR Overview**

SHR–Keeping Hotels Competitive. SHR, Sceptre Hospitality Resources, provides advanced tools and services that help hotels execute their best distribution strategy while delighting guests and optimizing profitability. The technical maturity of SHR ”having built not one but two Central Reservations Systems (CRS)” is second to none. Windsurfer CRS offers advanced features that allow you to easily manage your rates and inventory across all distribution channels, providing not only one of the best booking engines on the market but also the strongest integrations and connections to wherever you sell your rooms. Year after year, Windsurfer continues to be a preferred distribution platform for a wide variety of hotel properties. Responsive and flexible, our Internet Booking Engine (IBE), lets you easily merchandise and sell your rooms, packages, and add-ons in virtually any way you choose. Offering a responsive design, shopping cart-style booking process, persuasive messaging, and other unique features, the IBE is fully optimized for single property independents, multi-property groups, and large hotels. In addition to its innovative technology, SHR also provides an outsourced Revenue Management for Hire service for properties of any size and brand affiliation. Dedicated to finding revenue solutions that work for your specific situation, we match you with an experienced revenue manager who will provide unique insight as you build and maintain a winning revenue management strategy. SHR brings hoteliers the best in nimble technology, intelligently supported by tested industry experts”keeping hotels competitive.

# 

### **GuestCentric CRS Overview**

Simplified and fully integrated solutions to generate direct reservations, manage inventory, and measure revenue. Keep all your reservations in one simplified interface, and watch the direct bookings pour in with our conversion-driven booking engine trusted. If it works for Ritz Paris, it can work for you too!



### **Hotel-Spider Overview**

Our Central Reservation System allows you to access the property data via an interface. Availabilities, prices, cancellation policies, pictures and descriptions are at your disposal. Reservations can be generated due to the integration directly in our system.

# 

### **Djubo (CRS) Overview**

Simply put, Djubo is the answer to all your hotel room sales management woes. Djubo is a cloud-based 360-degree hotel sales platform for hotels of all sizes. From managing room inventory on various online channels to simplifying everyday hotel management, Djubo does it all for you from a single user-friendly interface. A direct booking management engine, online channel manager and centralized reservation manager all rolled into one, Djubo is the hotel management software you have always wanted for your hotel. Improved analytics, automated channel management, co-ordination with the front office and sales desk, customizable website templates, and online booking engine are just some of the productive features that Djubo offers. Used by 1350+ hotels worldwide, the DJUBO Cloud Property Management System brings unprecedented efficiency to your hotel operations and brings the power of cloud connectivity to deliver an integrated hotel management experience. Visit www.djubo.com for more product features.

# 

### **GauVendi Overview**

GauVendi manages room inventory, availability and pricing in a granular way through digitalization of all dimensions of the room experience (location, design, bathroom etc.) and repackages inventory into Room Feature Combinations optimized for all sales channels & travel segments! This enables a truly personalized booking experience, novel e-commerce tactics supported in real time by Artificial Intelligence, and a differentiated distribution and brand strategy. For example, guest can pick their room feature preferences and select their best room match. Hoteliers can sell repackaged Room Products appealing to specific travel segments for example offering connecting rooms for friends and families traveling together. GauVendi was founded in 2020 by experienced hospitality and tourism leaders with more than 100 years of combined experience from around the globe.

# Comparison:

As you can see most of the HMS (Hotel Management Systems) only manage one hotel or two, it also not that simple for the user interact,

Our project not just about manage a single hotel system, our hotel can manage more than hotel with huge amount of data, also because we connect the hotel details with the room itself, we can easily add a future update that cover not just hotels but anyone how want to rent a room in his house and so on ...

Also, we have a simple GUI that make the booking process easy to do by any user,

With high quality admin/manager management, also the systems usually focus on user booking or admin management but we did it differently because we didn’t just focus on the booking process we focus on admin management, staff tasks, manager control, receptionist tasks and more, and that what makes us different

System Analysis

# Introduction

Systems are created to solve problems. One can think of the sys-tems approach as an organized way of dealing with a problem. Inthis dynamic world, the subject System Analysis and Design (SAD),mainly deals with the software development activities.

# 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.

# 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.

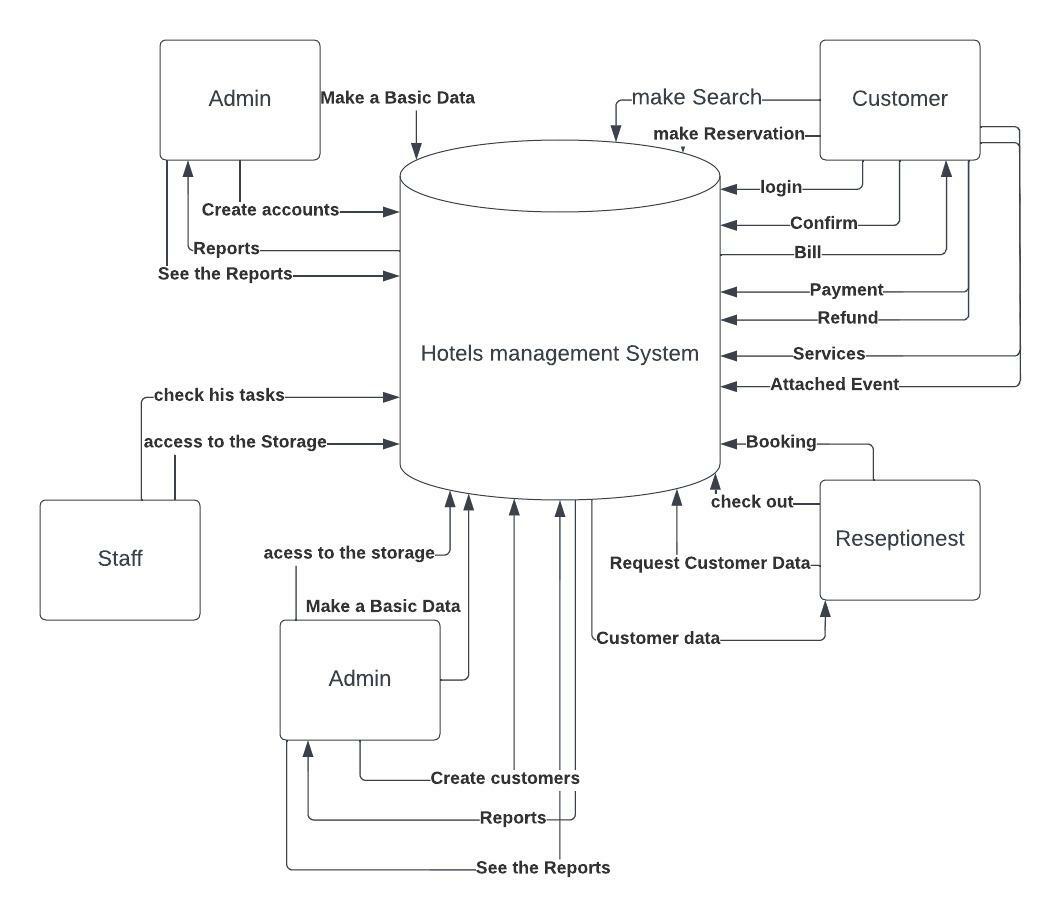
# Context Diagram

**A context diagram** is drawn in order to define and clarify the boundaries of the software system It identifies the flows of information between the system and external entities. The entire software system is shown as a single process

In order to produce the context diagram and agree on system scope, the following must be identified:

* external entities
* data-flows

in the following figure you’ll find the **Context Diagram** with its scope.



# LVL 0 Diagram

**Customer**

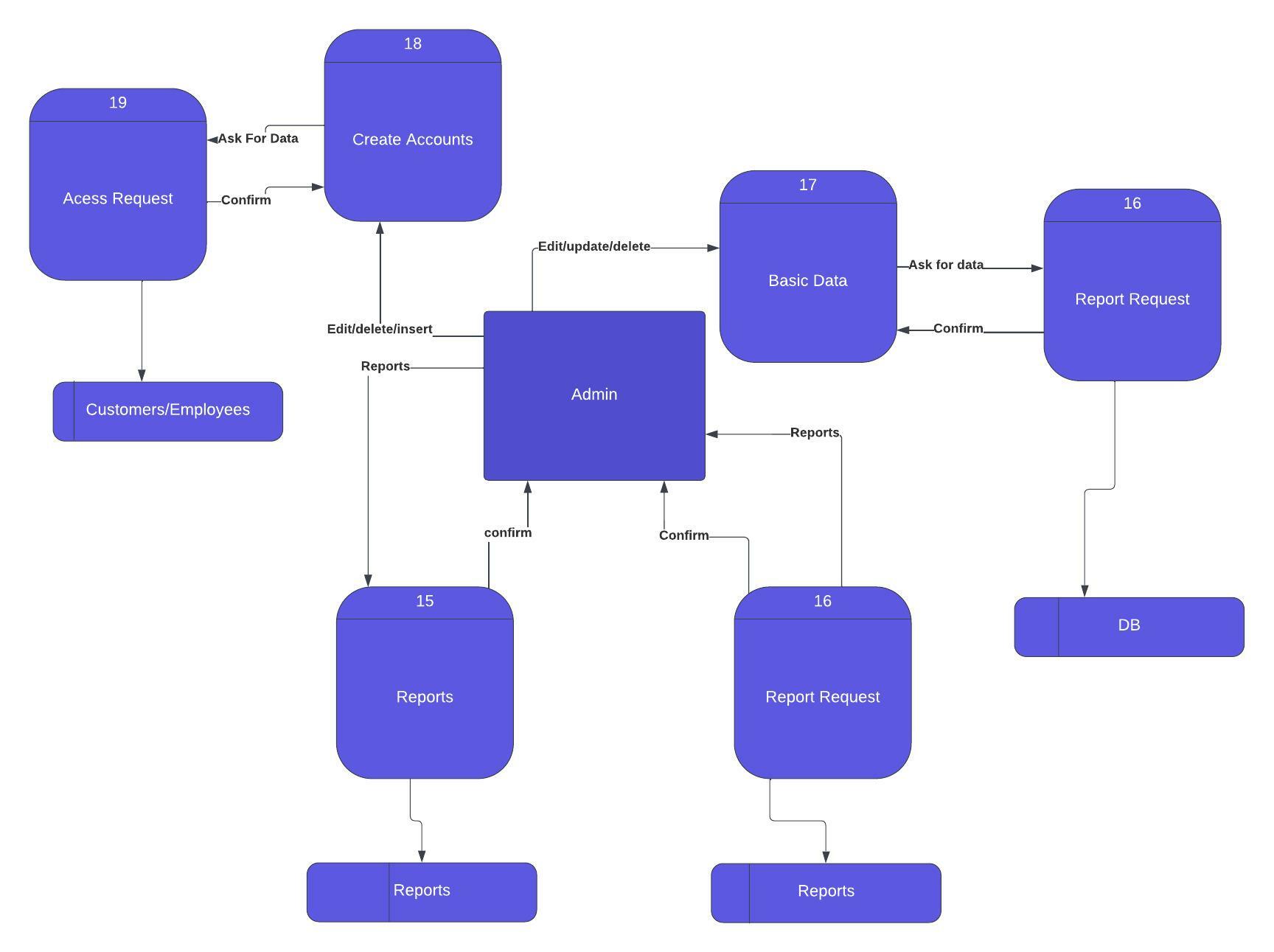




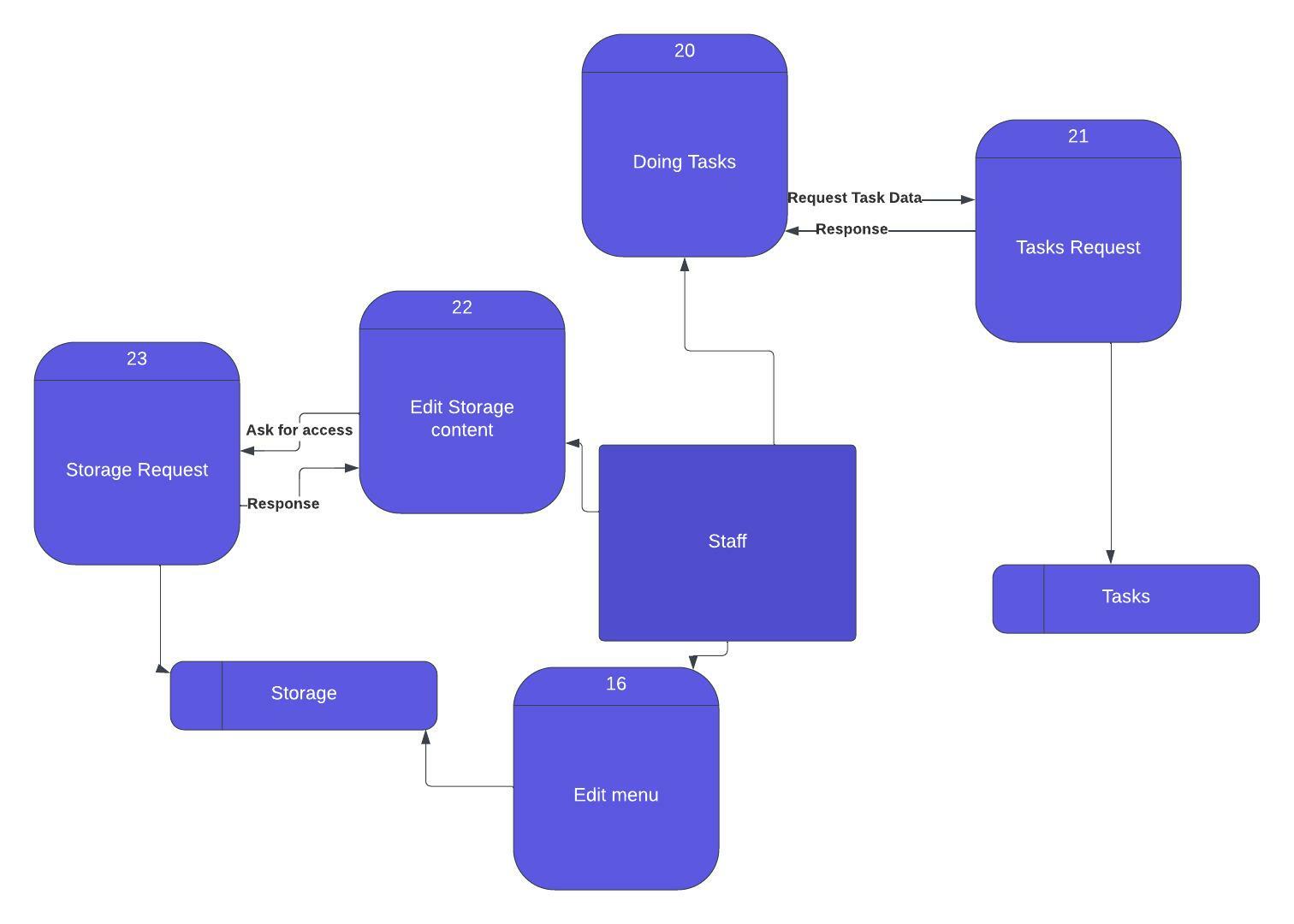
# **Manager**

# 

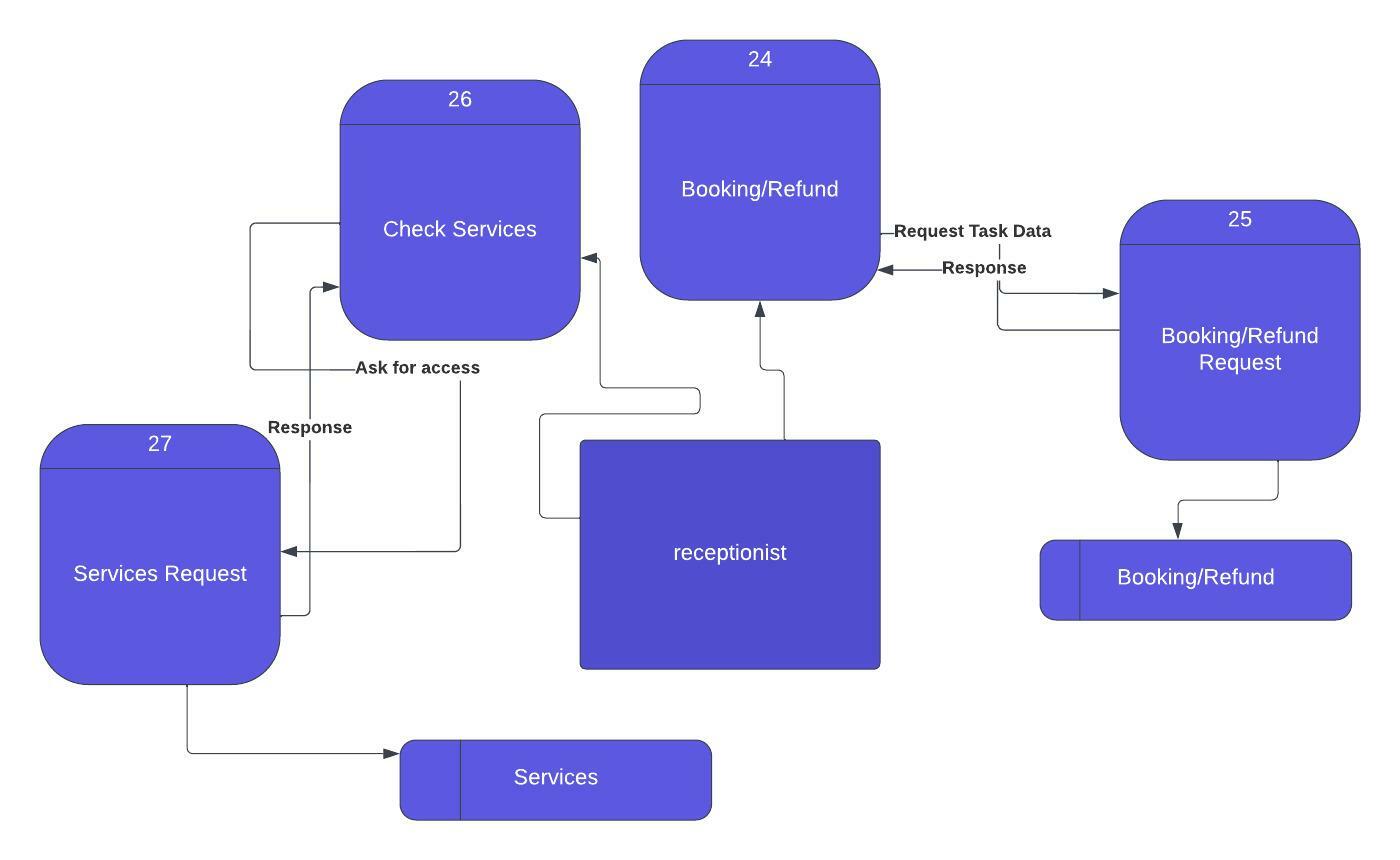
**Admin**



# **Staff**



# **Receptionist**



# Data dictionary

**Bookings**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| room Number | int | 20 | Foreign Key | Taken from room number Primary key | Room1.number = 5 |
| guest | char | 50 | Foreign Key | Taken from the guest’s name in guest table | Guest1.name= Heba |
| Date Of Reservation | Date Field | - | - | The time of reservation | 5/5/2022 |
| Start Date | Date Field | - | - | The visiting start time | 5/6/2022 |
| End Date | Date Field | - | - | The visiting end time | 8/7/2022 |

**Rooms**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| Payment method | Char Field | 20 | - | The room lock the payment method | Cash |
| number | Integer Field | - | Primary key | The number of the room in the hotel | 5 |
| capacity | Small Integer Field | - | - | The capacity of persons that can visit | 5 |
| Number Of Beds | Small Integer Field | - | - | Number of bids in the room | 4 |
| Room Type | Char Field | 20 | - | The type of the room | Luxury |
| price | Float Field | - | - | Price that the admin/manager can sign | 50$ |
| Status StartDate | Date Field | - | - | The start visiting date | 5/6/2022 |
| Status End Date | Date Field | - | - | The end visiting date | 8/7/2022 |
| Hotel Name | Char Field | 20 | - | The name of the hotel that contain that room | Havana |

**Room Services**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| Cur Booking | Char Field | 20 | Foreign Key | The customer’s name who books the room | Ahmed |
| room | Integer Field | - | Foreign Key | The number of the room in the hotel | 5 |
| Created Date | Date Field | - | - | The reservation create date | 5/6/2022 |
| Services Type | Char Field | 20 | - | The type of service that the user ask for | Food |
| price | Float Field | - | - | The price of that service | 15$ |

**Refunds**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| guest | Char Field | 20 | Foreign Key | The customer’s name who wants to refund | Ahmed |
| reservation | Date Field | - | Foreign Key | The reservation create date | 5 |
| reason | Text Field | - | - | Reason why he wants to refund | I got an emergency |

**events**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| Event Type | Char Field | 20 | - | The event type chosen from default common types | Live Song |
| location | Char Field | 100 | - | The location of the event | Cairo El Obour 6th street |
| Start Date | Date Field | - | - | The date that’s the even start on | 5/6/2022 |
| End Date | Date Field | - | - | The date that’s the even end on | 6/6/2022 |
| explanation | Text Field | - | - | More details about the event | Sia singer is coming |

**Bills**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| guest | Char Field | 20 | Foreign Key | The guest that booking | Live Song |
| Total Amount | Float Field | - | - | The total amount based on room price and how many days he’ll stay | Cairo El Obour 6th street |
| summary | Text Field | - | - | Some more details | 5/6/2022 |
| date | Date Time Field | - | - | The date of booking | 6/6/2022 |

**Announcements**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| content | Text Field | 20 | - | The content of the message that admin/manager want to send | There’re some problems in our hotel that we are going to fix immediately |
| sender | Char Field | - | Foreign Key | The admin/manager’s name | manager/Ahmed |
| date | Date Field | - | - | The date of sending the message | 5/6/2022 |

**Event Attendees**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| event | Text Field | - | Foreign Key | The event type name | Live song |
| Number Of Dependees | Char Field | - | - | Number of visitors that come with the guest | 5 |
| guest | Date Field | - | Foreign Key | Guest name | Ahmed |

**Employees**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| user | One To One Field | - | - | The Employee name | Live song |
| Phone Number | Phone Number Field | - | unique | The phone number | 0113449892 |
| salary | Float Field | - | - | The salary can be changed by the Admin | 500$ |

**Tasks**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Field Name | Data Type | Field Length | Constraints | Description | Example |
| employee | One To One Field | - | Foreign Key | The Employee name | Live Song |
| Start Time | Date Time Field | - | - | The date of task | 5/6/2022 |
| End Time | Date Time Field | - | - | deadline | 6/6/2022 |
| description | Text Field | - | - | More details | the Food is well done |

**Process**

|  |
| --- |
| Scenario Name: Listing empty rooms |
| Participating Actors: Bob: Receptionist, Ruby: Manager |
| Flow of Events:   1. Ruby accesses the system with her credentials. 2. Ruby clicks on get general report about all rooms button on the main page 3. The system shows Ruby general report about all rooms within a specified time interval. 4. Bob accesses the system with user Id and password. 5. Bob reaches the homepage of the system and clicks the get report button. 6. System redirects the receptionist to a page to fill the necessary information about the rooms he is looking for. 7. System creates the report about all the rooms and their general information. |

|  |
| --- |
| **Scenario Name**: Editing time interval for rooms. |
| **Participating Actors**: Bob: Receptionist, Alice: Guest |
| **Flow of Events**:   1. Alice accesses the system with her credentials. 2. Alice requests to edit the time interval for her room in accordance with the hotel’s policy. 3. The system creates a report for Alice’s request and sends it to Bob. 4. Bob accesses the system with his user ID and password. 5. Bob checks Alice’s request on the system. 6. Bob requests policy approval form the system for Alice. 7. System sends that policy to Alice. 8. After getting Alice’s approval system allows Bob to edit the room info. 9. Bob reaches to the system and changes time interval as Alice asked. |

|  |
| --- |
| **Scenario Name**: Create/cancel reservations |
| **Participating Actors**: Bob: Receptionist, Alice: Guest |
| **Flow of Events**:   1. Alice logs into the system with the required credentials. 2. Alice creates a request for creating/canceling a reservation. 3. The system sends the request to Bob. 4. Bob accesses the system with his user ID and password. 5. Bob accepts/denies the request from Alice. 6. The system informs Alice with an announcement. |

|  |
| --- |
| **Scenario Name**: Create a default Guest account |
| **Participating Actors**: Alice: Guest |

|  |
| --- |
| **Scenario Name**: Reaching Guest Information |
| **Participating Actors**: Bob: Receptionist |
| **Flow of Events**:   1. Bob accesses the system with user ID and password. 2. Bob clicks on show all guest information button on the main page. 3. The system shows a page with the list of guests and their information. |

|  |
| --- |
| Scenario Name: Editing Employees Information |

|  |
| --- |
| **Flow of Events**:   1. Alice clicks on Create account button. 2. The system gives Alice a form to fill in with the necessary information. 3. Alice fills the form and clicks on save button. 4. System automatically creates a guest account for Alice with given information. |

|  |
| --- |
| **Scenario Name:** Request room service / cleaning |
| **Participating Actors**: Alice: Guest, Mark: Employee |
| **Flow of Events**:   1. Alice accesses the system with her credentials. 2. Alice clicks on request room cleaning/service button. 3. The system creates the request and sends it to Mark. 4. Mark’s list of tasks is updated to match the new change. 5. Mark logs in to system with user ID and password and checks his tasks. |

|  |
| --- |
| **Scenario Name**: List the services or events |
| **Participating Actors**: Alice: Guest |
| **Flow of Events**:   1. Alice accesses the system with her credentials. 2. Alice clicks on List the services or events button. 3. The system shows Alice the list of services and events provided by the hotel. |

|  |
| --- |
| **Scenario Name**: Access all rooms information |
| **Participating Actors**: Clark: Admin |
| **Flow of Events**:   1. Clark logs in to the system with user ID and password. 2. Clark clicks on Access all rooms information button 3. The system shows the list of all the rooms of the hotel with their information displayed. |

|  |
| --- |
| **Scenario Name**: Access all the customers information |
| **Participating Actors**: Clark: Admin |

|  |
| --- |
| **Flow of Events**:   1. Clark logs in to the system with user ID and password. 2. Clark clicks on Access all the customers information button. 3. The system shows the list of all the customers information. |

|  |
| --- |
| **Scenario Name**: Access staff lists |
| **Participating Actors**: Clark: Admin |
| **Flow of Events**:   1. Clark logs in to the system with user ID and password. 2. Clark clicks on Access staff lists button. 3. The system shows the list of all the staff information. |

|  |
| --- |
| **Scenario Name**: Editing Room Details |
| **Participating Actors**: Clark: Admin |
| **Flow of Events:**   1. Clark logs in to the system with user ID and password. 2. Clark clicks on Edit room detail button. 3. The system shows the list of all the rooms. 4. Clark chooses a room to edit its details. 5. The system shows the details of that room. 6. Clark edits and clicks on save changes button. 7. The system updates the related information. |

|  |
| --- |
| **Scenario Name**: Listing and Editing Salaries of Employees |
| **Participating Actors**: Clark: Admin |
| **Flow of Events:**   1. Clark logs in to the system with user ID and password. 2. Clark clicks on Edit Salaries button. 3. The system shows the list of all the staff salaries. 4. Clark chooses the employee or group of employees to change their salaries. 5. The system opens the editing page. 6. Clark enters the changes he wants to make. 7. The system updates the information. |

|  |
| --- |
| **Scenario Name**: Creating Events |

|  |
| --- |
| **Participating Actors**: Ruby: Manager |
| **Flow of Events:**   1. Ruby logs in to the system with user ID and password. 2. Ruby clicks on Create Event button. 3. The system requests the necessary information to create an event. 4. Ruby enters the necessary information. 5. The system validates the information and creates the event. |

|  |
| --- |
| **Scenario Name**: The list of daily staff |
| **Participating Actors**: Ruby: Manager |
| **Flow of Events**:   1. Ruby accesses the system with her user ID and password. 2. The system redirects Ruby to the home page. 3. Ruby clicks on request the list of daily staff button on the page. 4. The system creates the list of the daily staff. 5. System brings that list to Ruby’s screen. |

|  |
| --- |
| **Scenario Name**: Editing Tasks of Employees |
| **Participating Actors**: Ruby: Manager, Rosalia: Employee |
| **Flow of Events:**   1. Ruby logs in to the system with user ID and password. 2. Ruby clicks on edit employee tasks button on the page. 3. The system shows the list of the employees. 4. Ruby chooses an employee to change the task for. 5. The system opens a page for Ruby with the employee’s tasks to change. 6. Ruby edits/change tasks for Rosalia from the system. 7. The system updates the task list for Rosalina. |

|  |
| --- |
| **Participating Actors**: Ruby: Manager, Rosalia: Employee |
| **Flow of Events:**   1. Ruby logs in to the system with user ID and password. 2. Ruby clicks on edit employee information button on the page. 3. The system lists all the employees and their respective information. 4. Ruby chooses an employee to change its information. 5. The system opens a page to edit the employee’s information. 6. Ruby edits/changes the employee’s information from the system. 7. The system updates the employee’s information. |

|  |
| --- |
| **Scenario Name**: Canceling tasks for Employees |
| **Participating Actors**: Ruby: Manager, Rosalia: Employee |
| **Flow of Events:**   1. Rosalina logs in to the system with user ID and password. 2. Rosalina clicks on cancel task button. 3. The system creates a request and sends it to Ruby. 4. Ruby logs in to the system with user ID and password. 5. The system shows Ruby the request created by Rosalina. 6. Ruby clicks on cancel employee tasks button on the page. 7. The system removes the canceled task from Rosalina’s list of tasks. |

|  |
| --- |
| **Scenario Name**: Listing Daily Tasks |
| **Participating Actors:** Rosalia: Employee |
| **Flow of Events:**   1. Rosalia accesses to the system with user ID and password. 2. Rosalina clicks on daily tasks button. 3. The system shows the list of daily tasks. |

|  |
| --- |
| **Scenario Name**: Accessing Information |
| **Participating Actors:** Rosalia: Employee |

|  |
| --- |
| **Flow of Events:**   1. Rosalia accesses to the system with user ID and password. 2. Rosalina clicks on show personal information button. 3. System shows Rosalia her own list of all information and tasks listed |

|  |
| --- |
| **Scenario Name**: Giving Feedbacks |
| **Participating Actors:** Rosalia: Employee |
| **Flow of Events**:   1. Rosalia accesses to the system with user ID and password. 2. Rosalina clicks on give feedback button. 3. The system shows a list of Rosalina’s daily tasks. 4. Rosalina selects a task to give feedback on. 5. The system shows the task’s details. 6. Rosalina inputs the feedback into the system. 7. The system updates the task. |

*Use case model*

|  |
| --- |
| **Use case name**: Booking Room |
| **Participating actors**: Receptionist, Guest |
| **Flow of Events**:   1. Guest logs into the system with the required credentials. 2. Guest creates a request for creating/canceling a reservation. 3. The system sends the request to Receptionist. 4. Receptionist accesses the system with his user ID and password. 5. Receptionist accepts/denies the request from Guest. 6. The system informs the guest with an announcement. 7. The guest gives their feedback. 8. The system updates according to the new changes and lets the user know. |
| **Entry condition**: Receptionist and Guest logged into system. |
| **Exit conditions**: Guest payment failed, or Guest payment succeeded, or Receptionist completed the booking procedure. |
| **Quality requirement:**   * Searches done by the receptionist should return results withing an acceptable delay. * The guest should be able to cancel a reservation withing a fixed period. |

|  |
| --- |
| **Use case name**: see Guest information |
| **participating actors**: Receptionist |
| **flow of events**:   1. Receptionist accesses the system with user ID and password. 2. Receptionist clicks on show all guest information button on the main page. 3. The system shows a page with the list of guests and their information. |
| **entry condition**: Receptionist logged into system |
| **exit condition**: The receptionist successfully reached the guest using the information found on the system, or the guest could not be reached/ wrong number. |
| **quality requirement:**   * The receptionist should be able to access the guest information in a timely manner. * The customer’s sensitive information should be protected without compromising his safety while looking for relevant information. |

|  |
| --- |
| **Use case name**: Check daily task |
| **participating actors**: Employee |
| **flow of events**:   1. The employee logs into the system. 2. The employee clicks the Daily Task Button. 3. System displays the list to employee. |
| **entry condition**: Employee clicks the Daily Task Button |
| **exit condition**: System displays the list to employee. |
| **quality requirement:**   * Employee should be able to access their information and task in a timely manner. * The employee should only be able to see his/her own task. |

|  |
| --- |
| **Use case name**: Check the events |
| **participating actors**: Guest |
| **flow of events**:   1. The guest logs into the system. 2. The guest clicks on the Check events button. 3. The system shows the guest a list of the events they can attend. |
| **entry condition**: The guest clicks on the Check events button. |
| **exit condition**: The system shows the events available. |
| **quality requirement:**   * The guest listed the events in a timely manner. * The guest did not have to compromise any personal information to perform this task. |

|  |
| --- |
| **Use case name**: Change salary |
| **participating actors**: Admin |
| **flow of events**:   1. The admin logs into the system. 2. The admin clicks on list of employees. 3. The system shows the list of the employees. 4. The admin selects an employee or group of employees. 5. The admin clicks Change salary button and changes the salary/salaries. 6. The system updates the employees’ information accordingly. |

|  |
| --- |
| **entry condition**: The admin clicks on list of employees. |
| **exit condition**: The system updates the employees’ information successfully. |
| **quality requirement:**   * The admin should be able to access the staff list in a timely manner * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Edit room details |
| **participating actors**: Admin |
| **flow of events**:   1. The admin logs into the system. 2. The admin clicks on list of rooms button. 3. The system shows the list of the rooms. 4. The admin selects a room from the list and clicks edit button. 5. The admin inputs the changes and clicks save button. 6. The system updates the room information. |
| **entry condition**: The admin clicks on list of rooms. |
| **exit condition**: The system updates the room information |
| **quality requirement:**   * The admin accessed the room details in a timely manner * The admin changed the information in question without compromising other sensitive data. |

|  |
| --- |
| **Use case name**: Cancel an event |
| **participating actors**: Manager |
| **flow of events**:   1. The Manager logs into the system. 2. The Manager clicks on list of events. |

|  |
| --- |
| 1. The system shows the list of the events available. 2. The Manager selects an event. 3. The Manager clicks cancel event button. 4. The system updates the list of events accordingly. 5. The system sends a notification to all attendees and staff related. |
| **entry condition**: The manager logged into the system. |
| **exit condition**: The system updates the list of events successfully, or the cancelation is not permitted for legal reasons. |
| **quality requirement:**   * The manager should be able to cancel the event withing the period agreed on with the attendees and guests. * System interface gives meaningful feedbacks |

|  |
| --- |
| **Use case name**: Edit employee’s tasks. |
| **participating actors**: Manager |
| **flow of events**:   1. Manager logs into the system. 2. Manager clicks on Edit employee’s task button 3. System shows the manager a list of all the employees. 4. The manager selects an employee to edit a task for. 5. The system shows the selected employee’s current tasks. 6. The manager chooses a task to edit. 7. The system updates the employee’s list of tasks. |
| **entry condition**: The manager logged into the system. |
| **exit condition**: The manager successfully edited the task for that employee. |
| **quality requirement:**   * The manager accessed the employee’s list of tasks in a timely manner. * The manager should be able to edit the employee’s task without interfering with other employees’ tasks. |

|  |
| --- |
| **Use case name**: log into the system |
| **Participating actors**: System User |

|  |
| --- |
| **Flow of Events**:   1. The user opens the website. 2. The user clicks the log in button. 3. System shows a form with username and password 4. User enters username and password and submit the form 5. System validates the inputs. 6. System authenticate the user and redirect him to the dashboard. |
| **Entry condition**: Click the log in button |
| **Exit conditions**: Submitting the wrong information to system or logging in successfully. |
| **Quality requirement:**   * System should respond in few seconds * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: See all Guests in a specific time interval |
| **Participating actors**: Manager/Admin |
| **Flow of Events**:   1. Manager/Admin logs into the system. 2. Manager/Admin clicks on List Guests. 3. The system shows Manager/Admin the list of all guests and input for time interval. 4. The Manager/Admin fill in the time interval. 5. The system updates the list of the guests that meet that requirement. |
| **Entry condition**: Clicking the list guest button |
| **Exit conditions**: updating the list of the guests successfully. |
| **Quality requirement:**   * System should respond in few seconds. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Get report of employee’s tasks |
| **Participating actors**: Manager/Admin |

|  |
| --- |
| **Flow of Events**:   1. Manager/Admin logs into the system 2. Manager/Admin clicks on get report of employee’s task button. 3. The system shows the Manager/Admin a list of employees to choose from. 4. Manager/Admin chooses an employee from the list and clicks get report button. 5. The system generates the employee’s report and displays the report. |
| **Entry condition**: Manager/Admin clicks on get report of employee’s task button. |
| **Exit conditions**: Successfully generating the report. |
| **Quality requirement:**   * System should respond in a timely manner. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: request off day |
| **Participating actors**: Employee, Admin/Manager. |
| **Flow of Events**:   1. Employee logs into the system. 2. Employee clicks on request off day. 3. The system asks about the date of the off day. 4. The Employee selects a date interval. 5. The system creates a request to be sent to the Admin/Manager. 6. The Admin/Manager gets the request and gives his/her feedback. 7. The system updates the Employee request. |
| **Entry condition**: Employee/Receptionist clicks on request off day |
| **Exit conditions**: The system updates the Employee/Receptionist request. |
| **Quality requirement:**   * System interface gives meaningful feedbacks. * Employee/Receptionist cannot request more than two weeks of break. * Employee/Receptionist cannot request an off day in the current week of the request. |

|  |  |
| --- | --- |
| **Use case name**: Create employee shift | |
|  | |
| **Participating actors: Manager** | |
| **Flow of Events:**   1. **Manager logs into the system.** 2. **Manager clicks on Create employee shift button.** 3. **System shows the manager a list of all the employees.** 4. **The manager selects an employee to create a shift for.** 5. **The system shows the selected employee’s working hours and days.** 6. **The manager chooses working hours for the shift.** 7. **The system updates the employee’s schedule.** | |
| **Entry condition: Manager clicks on Create employee shift button** | |
| **Exit conditions: The system updates the employee’s schedule, or if the employee does not have suitable hour for the shift.** | |
| **Quality requirement:**   * **System interface gives meaningful feedbacks.** * **The system returns the list of employees in seconds.** * **The manager should be able to create the employee’s shift without interfering with other employees’ shifts.** | |

|  |
| --- |
| **Use case name**: check employee’s shift |
| **Participating actors**: Manager |
| **Flow of Events**:   1. Manager logs into the system. 2. Manager clicks on check shift button 3. System shows the manager a list of all the employees. 4. The manager selects an employee to check his shift. 5. The system shows the selected employee’s working hours and days. |
| **Entry condition**: Manager clicks on check shift button |
| **Exit conditions**: The system shows the selected employee’s working hours and days. |
| **Quality requirement:**   * System interface gives meaningful feedbacks. * The system returns the list of employees in seconds. |

|  |
| --- |
| **Use case name**: Check weekly schedule. |
| **Participating actors**: Employee |

|  |
| --- |
| **Flow of Events**:   1. Employee logs into the system. 2. Employee clicks on weekly schedule. 3. The system shows the employee’s schedule. |
| **Entry condition**: Employee clicks on weekly schedule |
| **Exit conditions**: The system shows the employee’s schedule |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Create Guest account |
| **Participating actors**: Guest |
| **Flow of Events**:   1. The Guest opens the website. 2. Click the Create Account Button. 3. System shows a form with necessary information. 4. Guest enters necessary information and submits the form. 5. System validates if the guest already exists or not. 6. System saves guest to the database and redirects the guest to the login page. |
| **Entry condition**: The guest clicks on the register button. |
| **Exit conditions**: Trying to register an already existing guest or successfully registered. |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Create new user |
| **Participating actors**: System User |

|  |
| --- |
| **Flow of Events**:  1.The user opens the website. 2.Click the Register User Button.  3.System shows a form with necessary information. 4.User enters necessary information and submits the form. 5.System validates if the user already exists or not.  6.System saves user information and redirects him/her to the dashboard. |
| **Entry condition**: Clicking the register button. |
| **Exit conditions:** Trying to register an already existent user or successfully registered. |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Edit employee’s information |
| **Participating actors**: Manager |
| **Flow of Events**:   1. Manager logs in to the system. 2. Manager clicks on show employees’ list. 3. The system shows the list of all the employees. 4. Manager chooses the employee from the employee list. 5. System shows the chosen Employee’s detailed information. 6. Manager edits the information wanted on that page. 7. Manager clicks on the save button to save the changes on employee’s information. 8. System saves the changes on the screen and shows them to the Manager. |
| **Entry condition**: Clicking edit employee’s information button. |
| **Exit conditions:** Entering invalid format of information to different areas, deleting necessary information, or successfully editing and saving the information tab. |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedbacks. * System must take the compulsory information. |

|  |
| --- |
| **Use case name**: Refund payment |

|  |
| --- |
| **Participating actors**: Guest, Manager |
| **Flow of Events**:   1. Guest logs in to the system with his/her user ID and password. 2. Guest clicks on the refund button. 3. System opens the cancellation tab according to hotels policy. 4. System shows a form to guest about cancellation. 5. Guest fills the form and clicks on the save button. 6. System saves and sends this request form to the Manager. 7. According to hotel policy and hotel rules Manager can approve the form. 8. Once the system gets the approval from the manager, refunds the payment to the Guest, and deletes the reservation from the system. |
| **Entry condition**: Clicking cancel reservation button. |
| **Exit conditions:** Cancellation is not possible according to hotel policy and rules. Or request successfully sent to the Manager. |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedback * In case of refunding, system should send the money to Guest’s account within a reasonable time interval. |

|  |
| --- |
| **Use case name**: Request room service |
| **Participating actors**: Guest, Staff |
| **Flow of Events**:   1. Guest logs in to the system with his/her user ID and password. 2. Guest clicks on the request room service button. 3. System sends this request to the available Staff page. 4. System adds the task to the Staff’s task list. 5. Staff goes to the Guest’s room with necessary items according to the request. 6. Staff access their page and mark the new task as complete. 7. The guest’s bill is updated in accordance with the service provided. |
| **Entry condition**: Clicking request room service button. |
| **Exit conditions**: Request successfully added to the current task list. |
| **Quality requirement:**   * System should respond in a short time of seconds. * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Check task for specific time interval |
| **Participating actors**: Employee |

|  |
| --- |
| **Flow of Events**:   1. Employee clicks the see All Tasks Button. 2. System shows two inputs for starting and end-date. 3. Employee enters the starting date and end-date. 4. System creates a task list for specified time interval. 5. System displays the list to the employee. |
| **Entry condition**: Clicking see All Tasks button. |
| **Exit conditions**: System show the list for specific time interval, or no invalid date was given. |
| **Quality requirement:**   * System should respond in few seconds * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Request Room Cleaning |
| **Participating actors**: Guest, Staff |
| **Flow of Events**:   1. Guest logs in to the system with his/her user ID and password. 2. Guest clicks on the request room Cleaning button. 3. System sends this request to the Staff’s page. 4. System adds the task to the Staff ‘s task list. 5. Staff goes to the Guest’s room with necessary items according to the request. 6. Employee accesses to his/her page and marks the new task as complete. |
| **Entry condition**: Guest clicks on the request room Cleaning button |
| **Exit conditions**: The task is successfully added to the tasks list of the Staff. |
| **Quality requirement:**   * System should respond in a short time * System interface gives meaningful feedbacks. |

|  |
| --- |
| **Use case name**: Request Bill |
| **Participating actors**: Guest, Receptionist |

|  |
| --- |
| **Flow of Events**:   1. Guest logs into the system 2. The guest clicks on request bill button. 3. The system sends the request to the receptionist. 4. The receptionist accesses the guest’s request. 5. The receptionist approves the guest’s request. 6. The system sends a bill to the guest’s account. |
| **Entry condition**: Guest clicks request bill button |
| **Exit conditions**: Bill successfully sent to the Guest, there is no such a bill. |
| **Quality requirement:**   * System should respond in few seconds * System interface gives meaningful feedbacks. |

# **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.

# **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:**

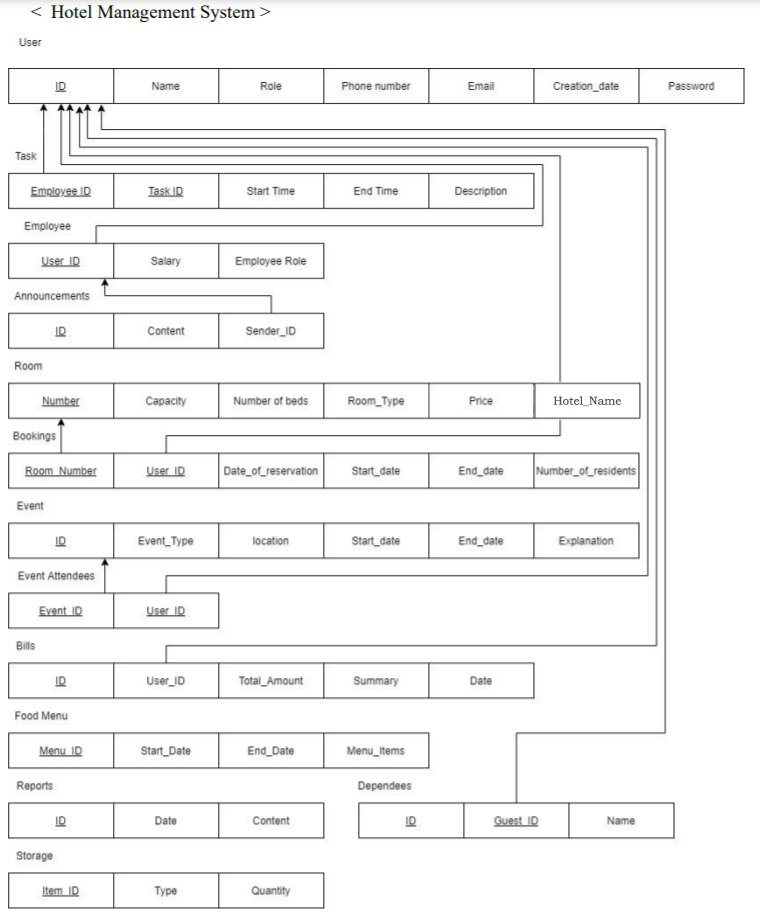
Provides security protocols for other components, such as two factor authentications for the payment component, and encryption for passwords stored in the database.

# **Room:**

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

System Design

# ER-Diagram



# GUI

**GUI overview**

A GUI includes GUI objects, like [icons](https://www.computerhope.com/jargon/i/icon.htm), [cursors](https://www.computerhope.com/jargon/c/cursor.htm), and [buttons](https://www.computerhope.com/jargon/b/button.htm). These graphical elements are sometimes enhanced with sounds, or visual effects like [transparency](https://www.computerhope.com/jargon/t/transpar.htm) and [drop shadows](https://www.computerhope.com/jargon/d/dropshad.htm). Using these objects, a user can use the computer without having to know commands.

Below is a picture of the Windows 7 [desktop](https://www.computerhope.com/jargon/d/desktop.htm) and an example of a GUI [operating system](https://www.computerhope.com/jargon/o/os.htm). In this example, you could use a [mouse](https://www.computerhope.com/jargon/m/mouse.htm) to move a [pointer](https://www.computerhope.com/jargon/m/mouspoin.htm) and [click](https://www.computerhope.com/jargon/c/click.htm) a program icon to start a program.

**What are the elements of a GUI?**

To make a GUI as [user-friendly](https://www.computerhope.com/jargon/u/userfrie.htm) as possible, there are different elements and objects that the user use to interact with the software. Below is a list of each of these with a brief description.

[Button](https://www.computerhope.com/jargon/p/pushbutt.htm) - A graphical representation of a button that performs an action in a program when pressed

[Dialog box](https://www.computerhope.com/jargon/d/dialogbo.htm) - A type of window that displays additional information, and asks a user for input.

[Icon](https://www.computerhope.com/jargon/i/icon.htm) - Small graphical representation of a program, feature, or file.

[Menu](https://www.computerhope.com/jargon/m/menu.htm) - List of commands or choices offered to the user through the menu bar.

[Menu bar](https://www.computerhope.com/jargon/m/menubar.htm) - Thin, horizontal bar containing the labels of menus.

[Ribbon](https://www.computerhope.com/jargon/r/ribbon.htm) - Replacement for the file menu and toolbar that groups programs activities together.

[Tab](https://www.computerhope.com/jargon/t/tab.htm) - Clickable area at the top of a window that shows another page or area.

[Toolbar](https://www.computerhope.com/jargon/t/toolbar.htm) - Row of buttons, often near the top of an application window, that controls software functions.

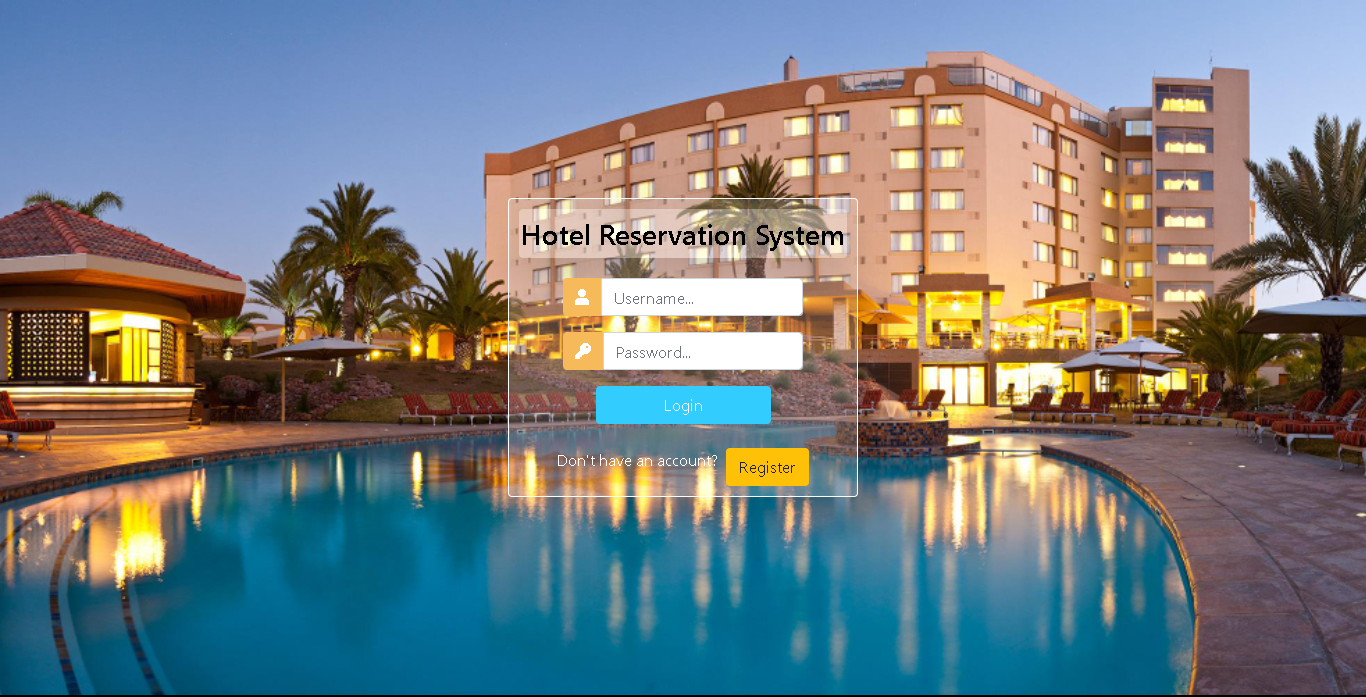
[Window](https://www.computerhope.com/jargon/w/window.htm) - Rectangular section of the computer's display that shows the program currently being used.

**What are the benefits of GUI?**

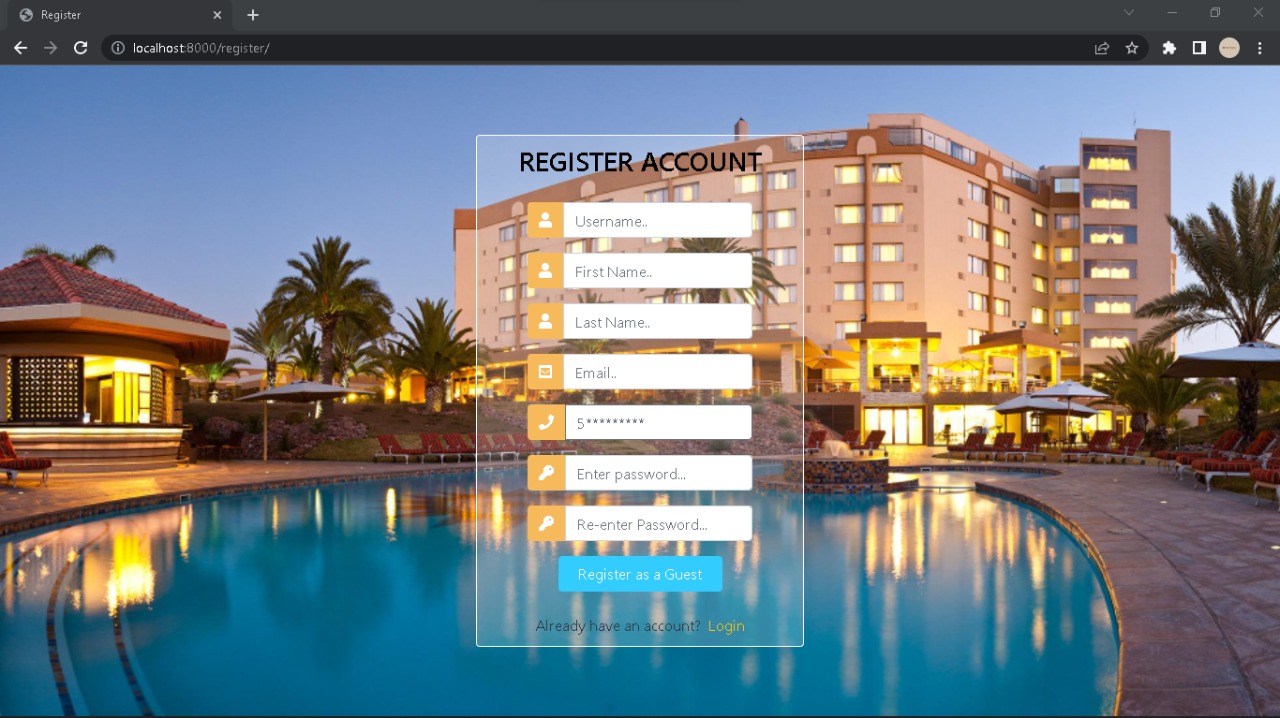
A GUI is considered to be more [user-friendly](https://www.computerhope.com/jargon/u/userfrie.htm) than a text-based [command-line interface](https://www.computerhope.com/jargon/c/commandi.htm), such as [MS-DOS](https://www.computerhope.com/jargon/m/msdos.htm), or the [shell](https://www.computerhope.com/jargon/s/shell.htm) of [Unix-like](https://www.computerhope.com/jargon/u/unix-like.htm) operating systems.

Unlike a [command-line operating system](https://www.computerhope.com/jargon/c/commandi.htm) or [CUI](https://www.computerhope.com/jargon/c/cui.htm), like [Unix](https://www.computerhope.com/jargon/u/unix.htm) or [MS-DOS](https://www.computerhope.com/jargon/m/msdos.htm), GUI operating systems are easier to learn and use because commands do not need to be memorized. Additionally, users do not need to know any [programming languages](https://www.computerhope.com/jargon/p/programming-language.htm). Because of their ease of use and more modern appearance, GUI operating systems have come to dominate today's market.

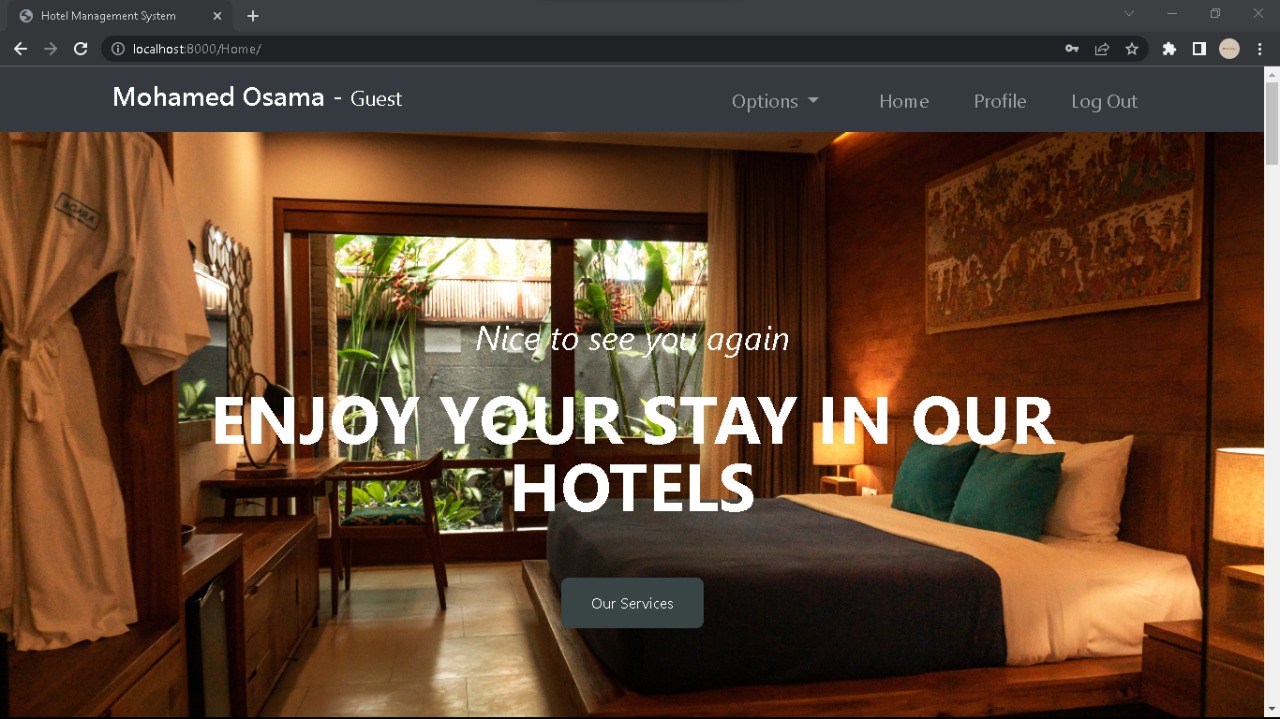
**Login**

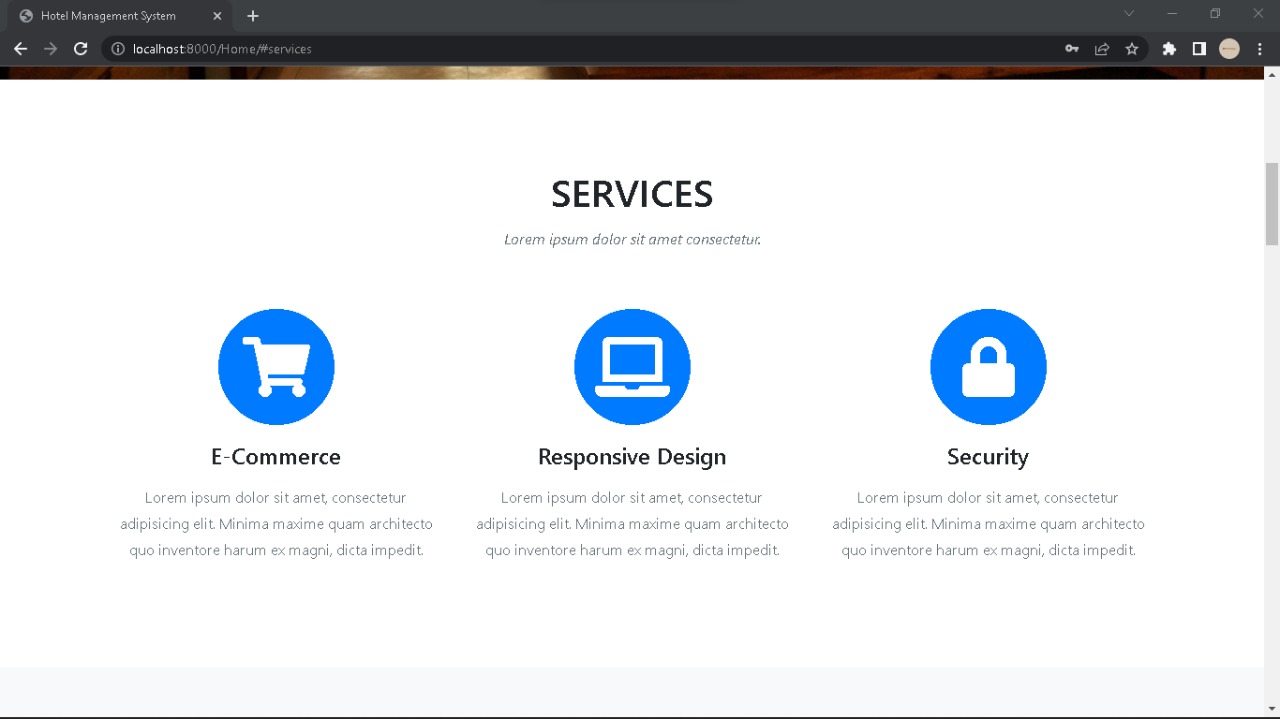
****

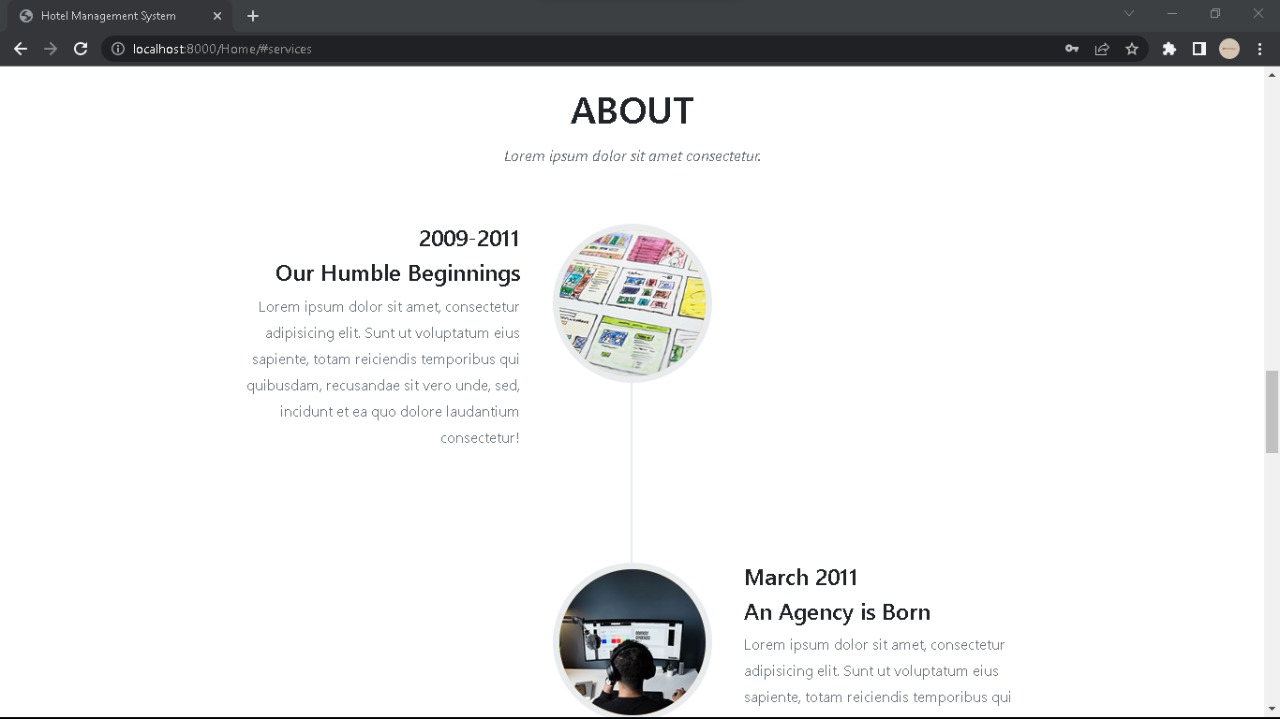
**Registration**

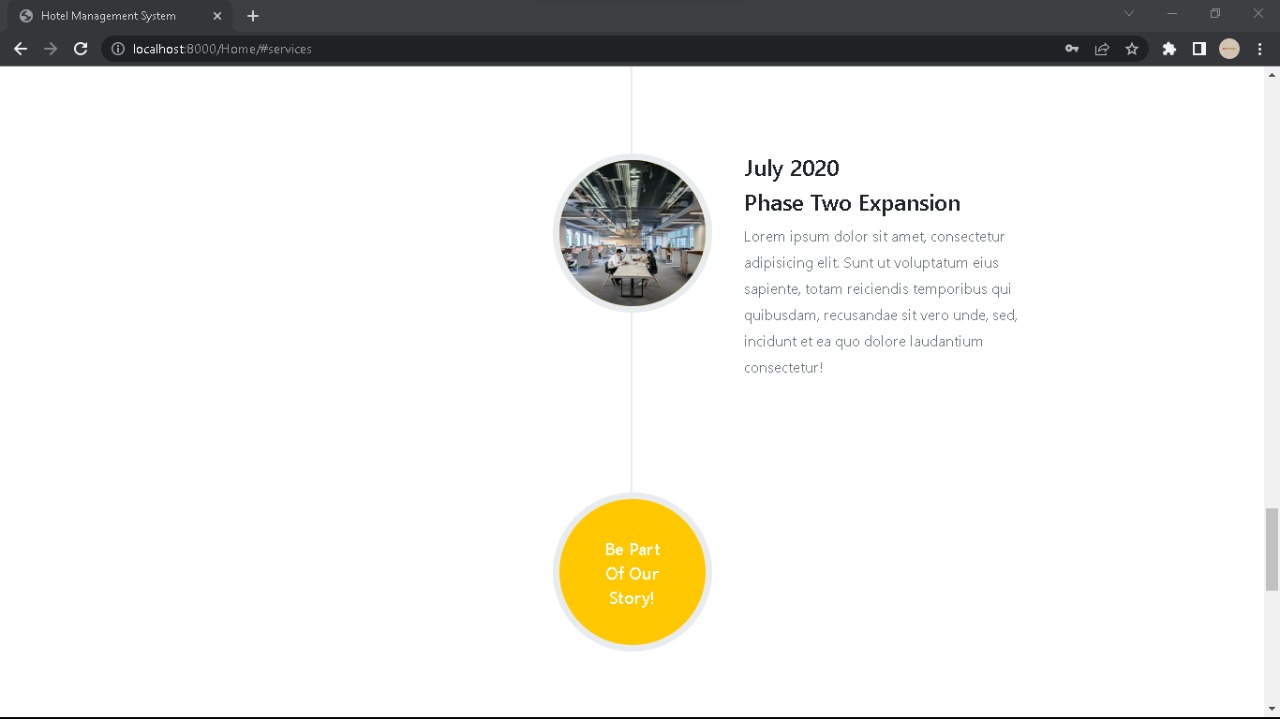
****

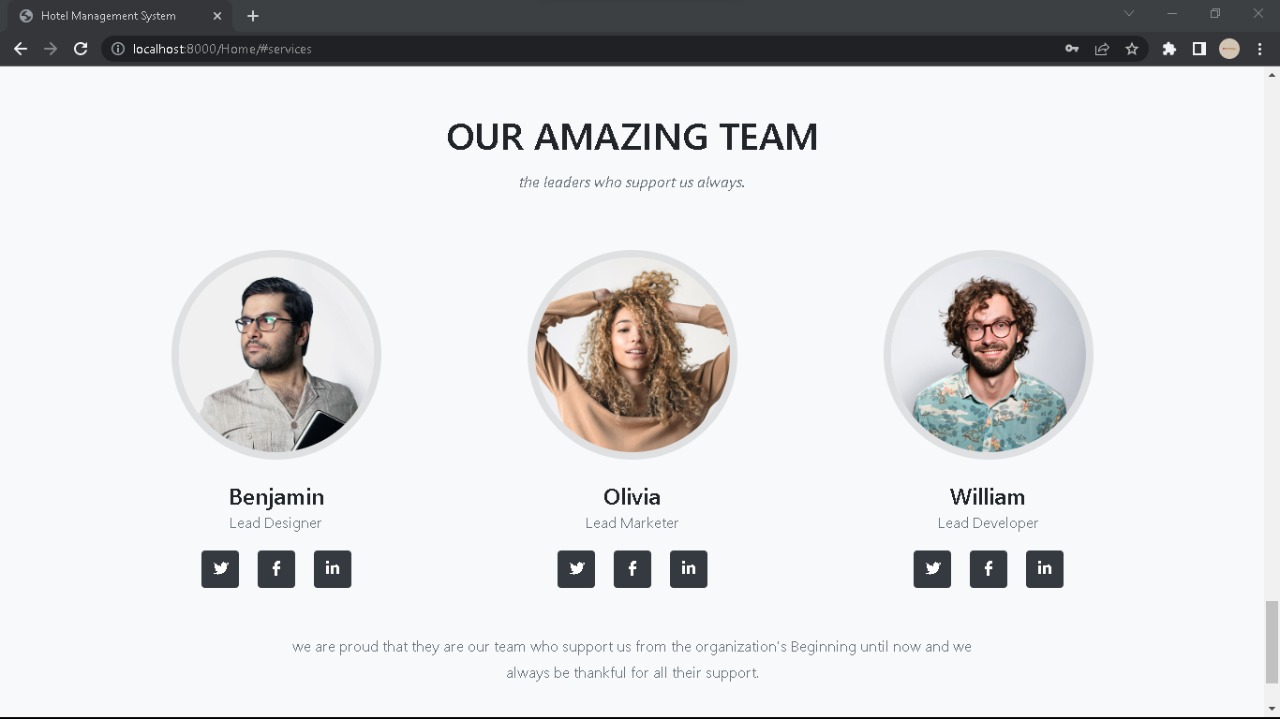
**Home**

****

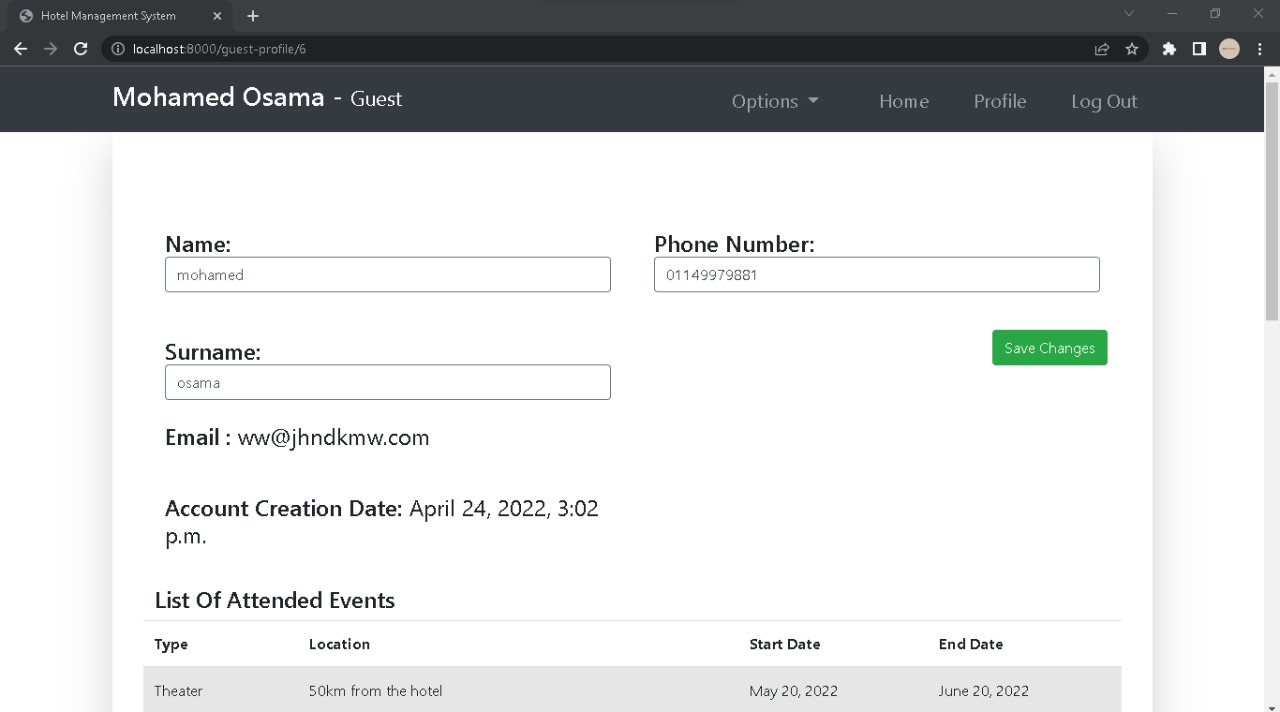
****

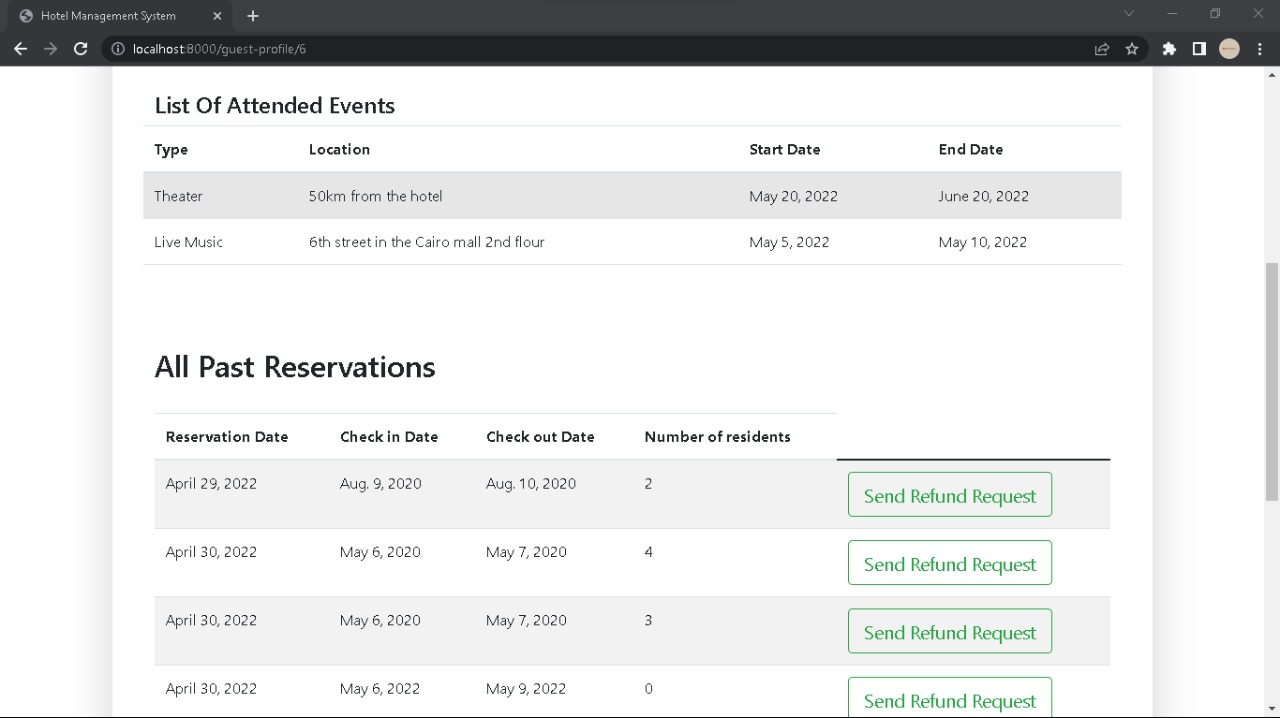
****

****

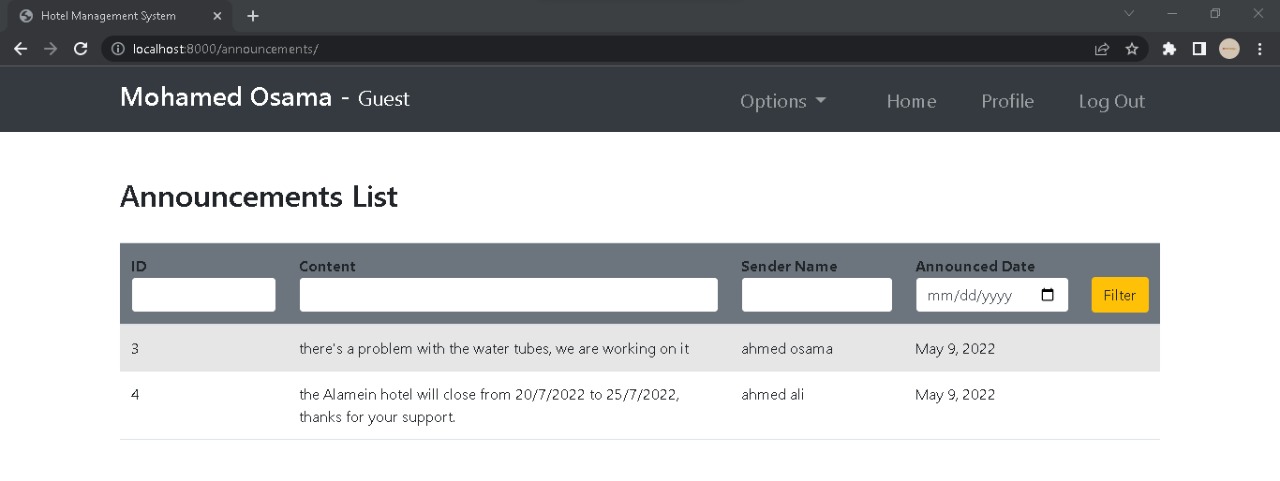
****

**Profile**

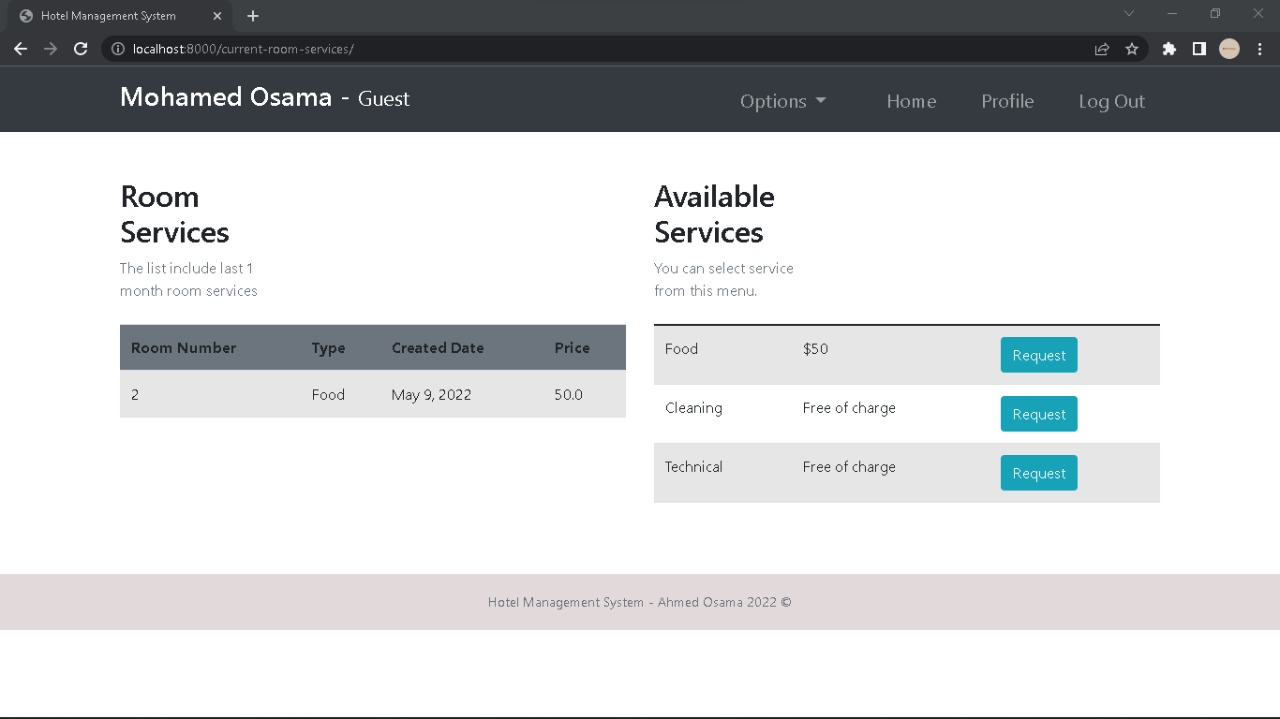
****

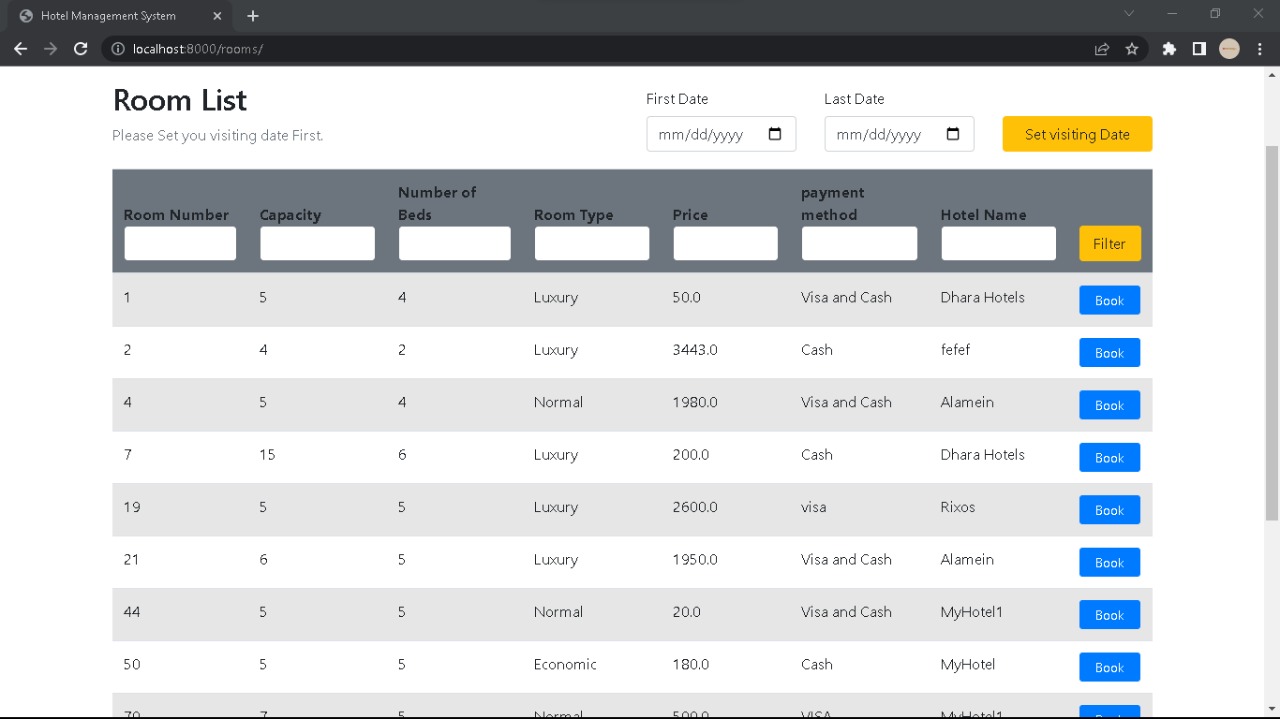
****

**Announcements List**

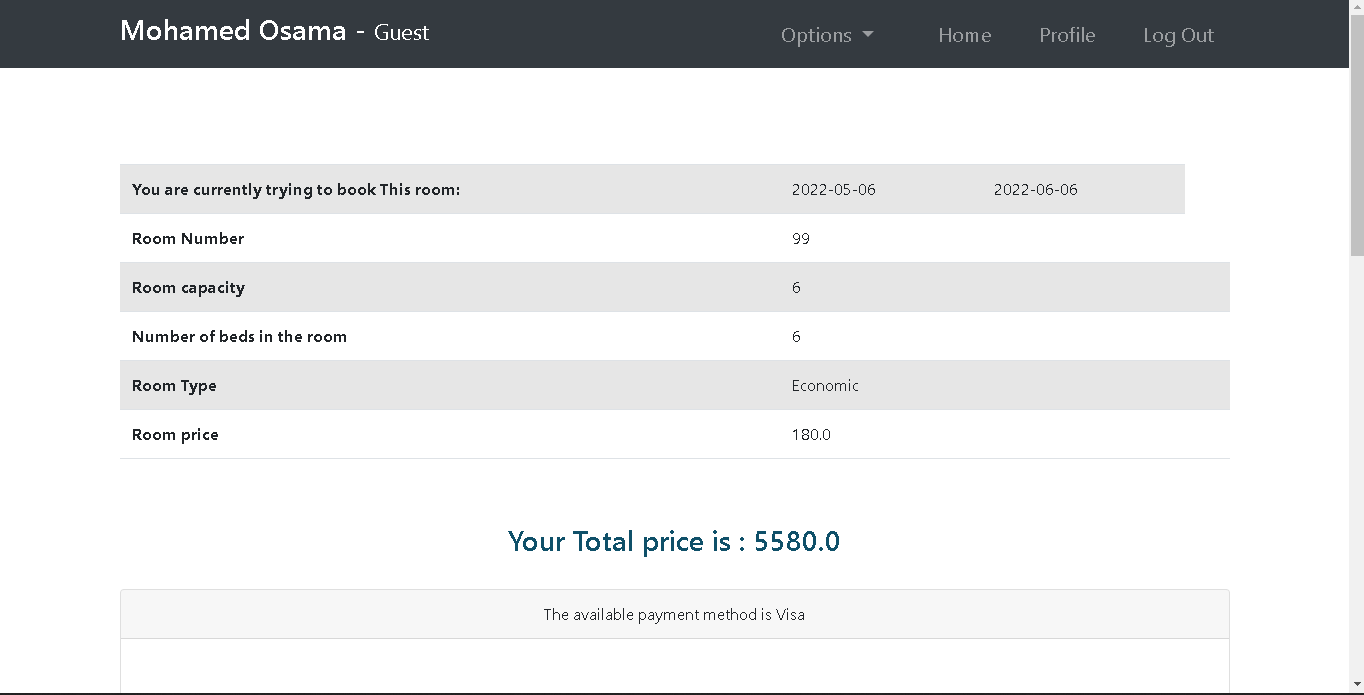
****

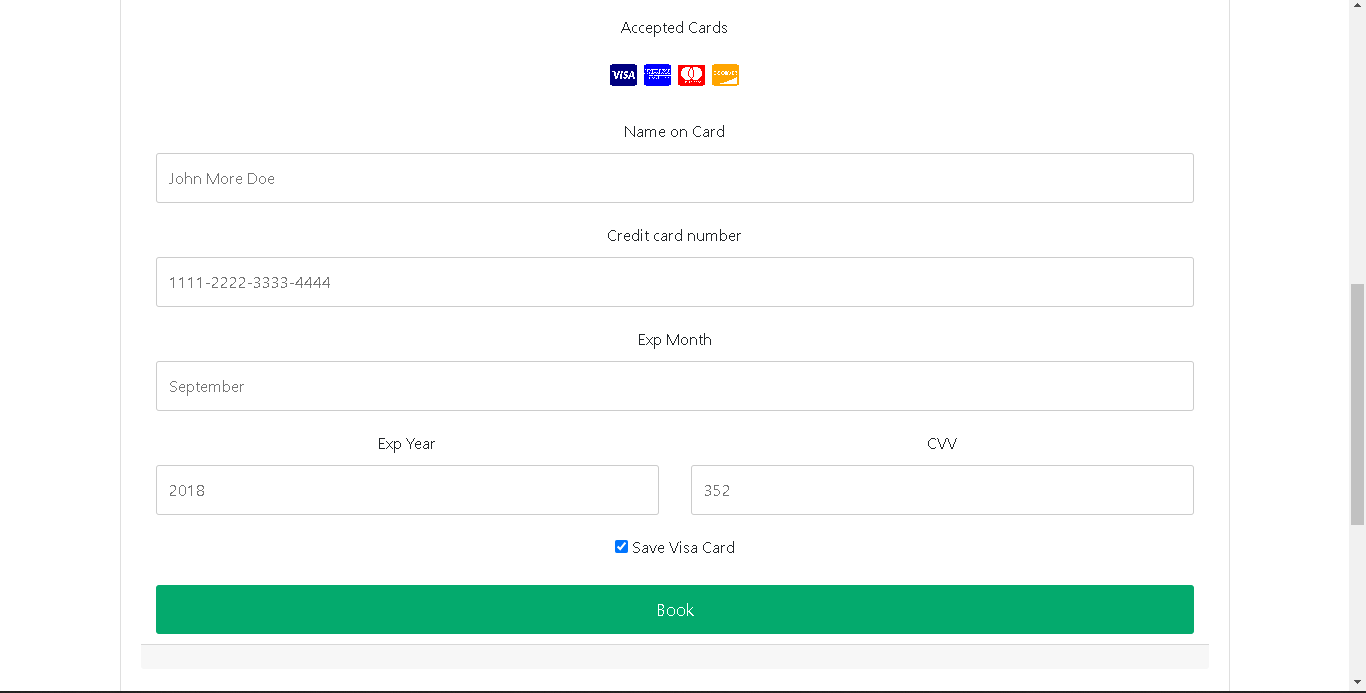
**Room Service**

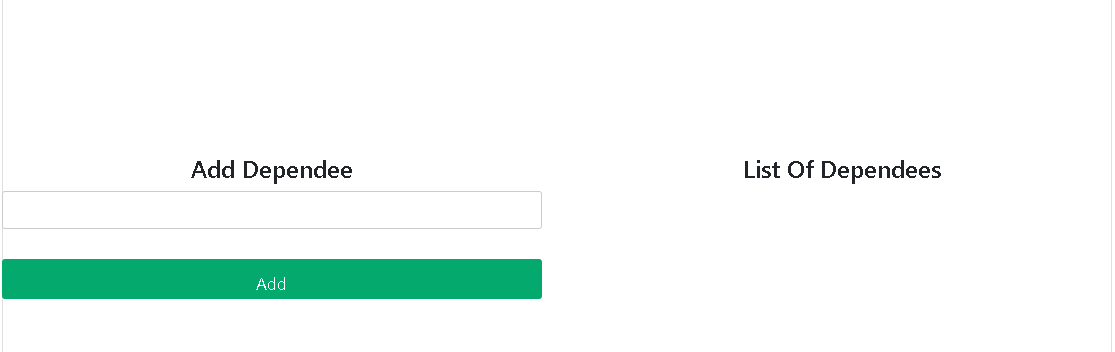
****

**Room List**

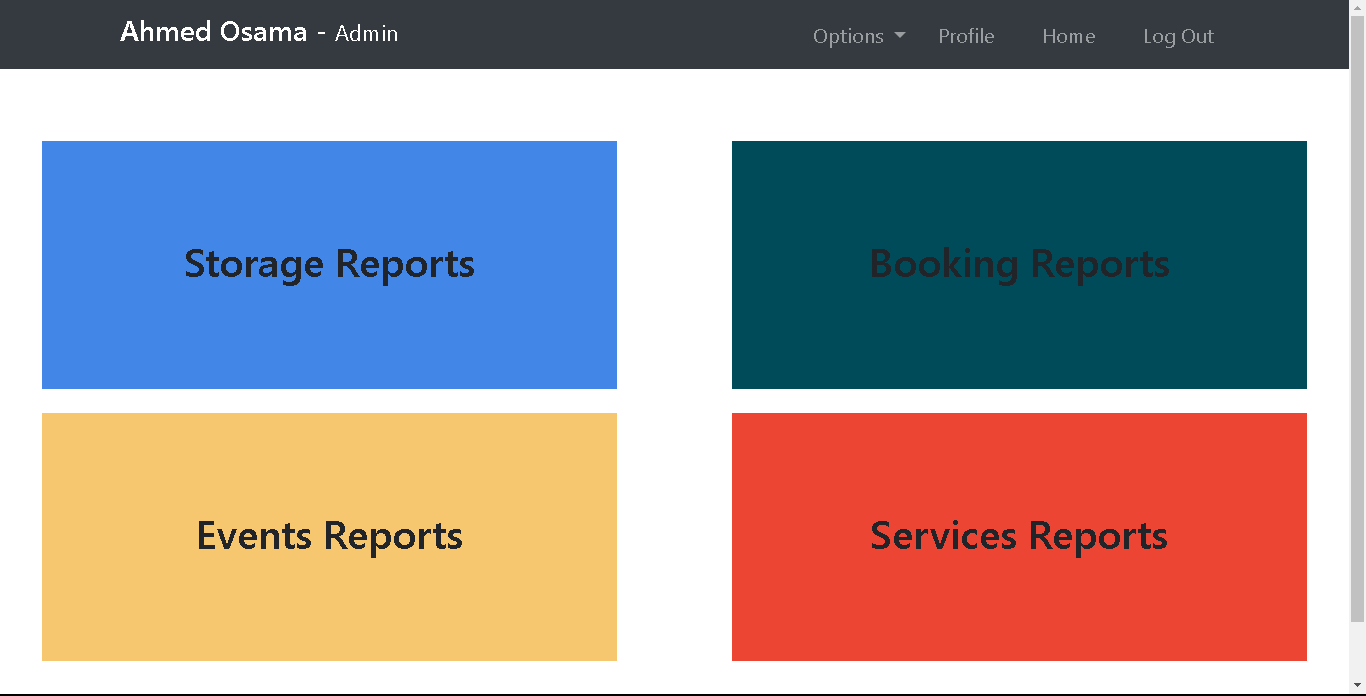
**Booking make**

****

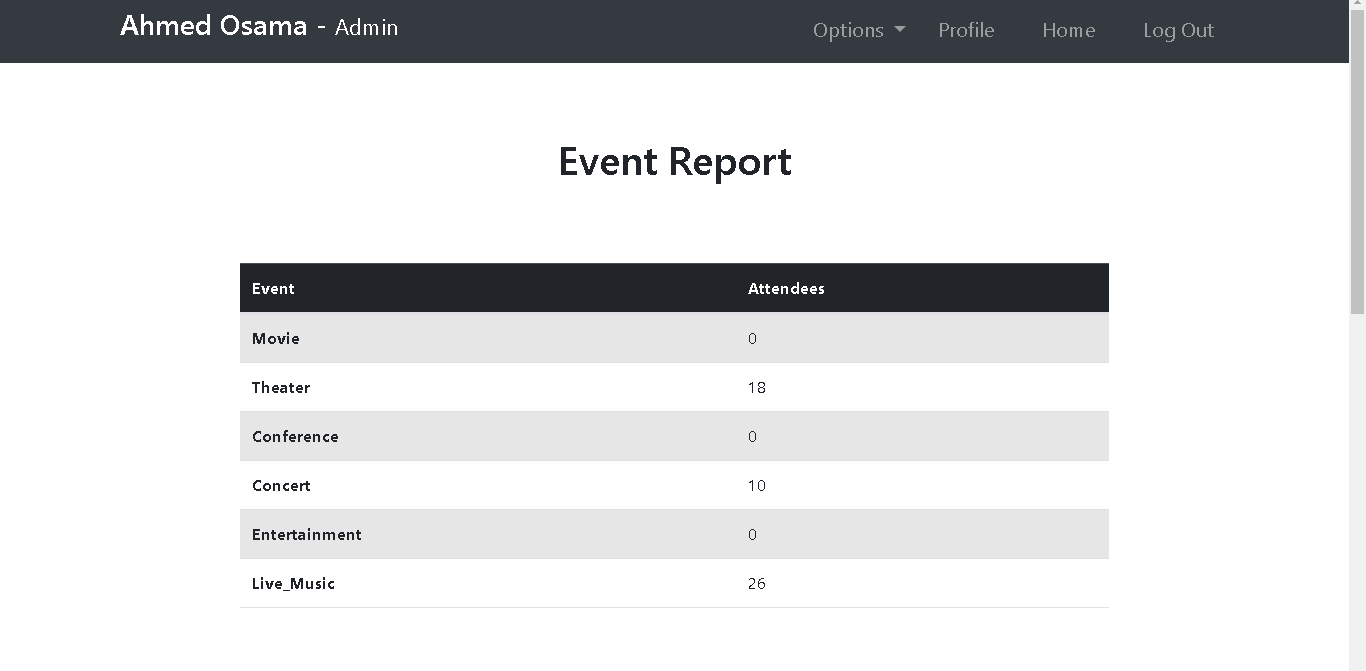
****

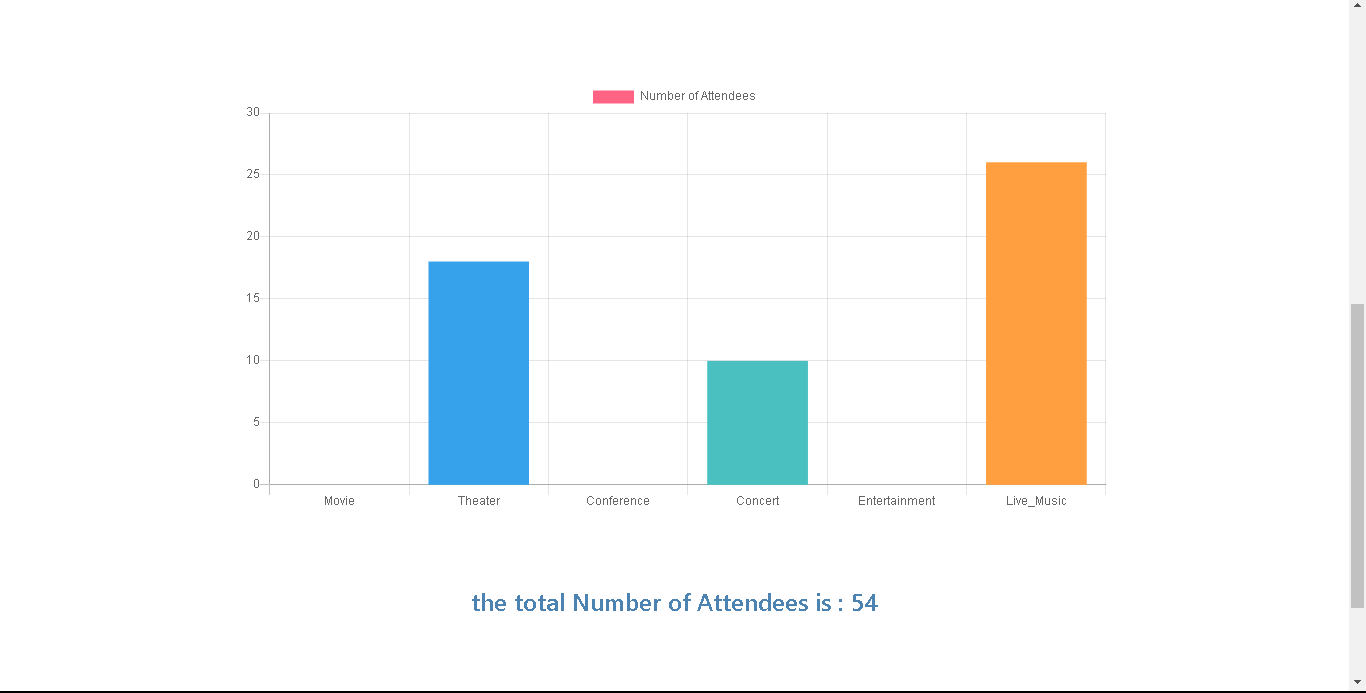
****

**Reports**

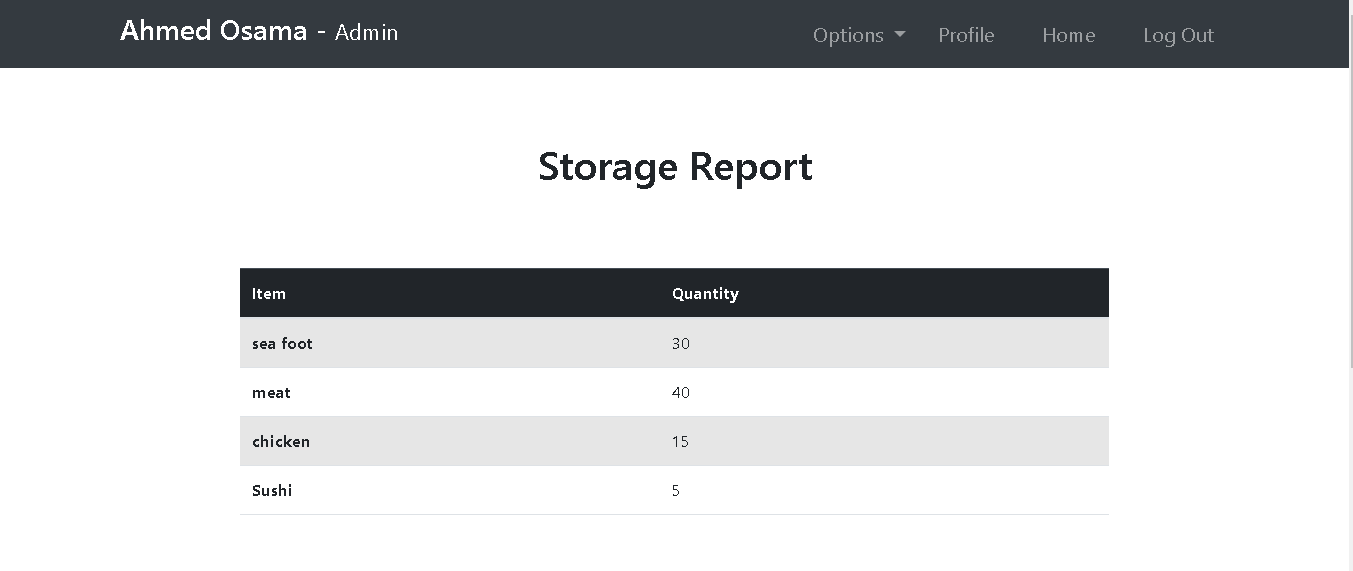
****

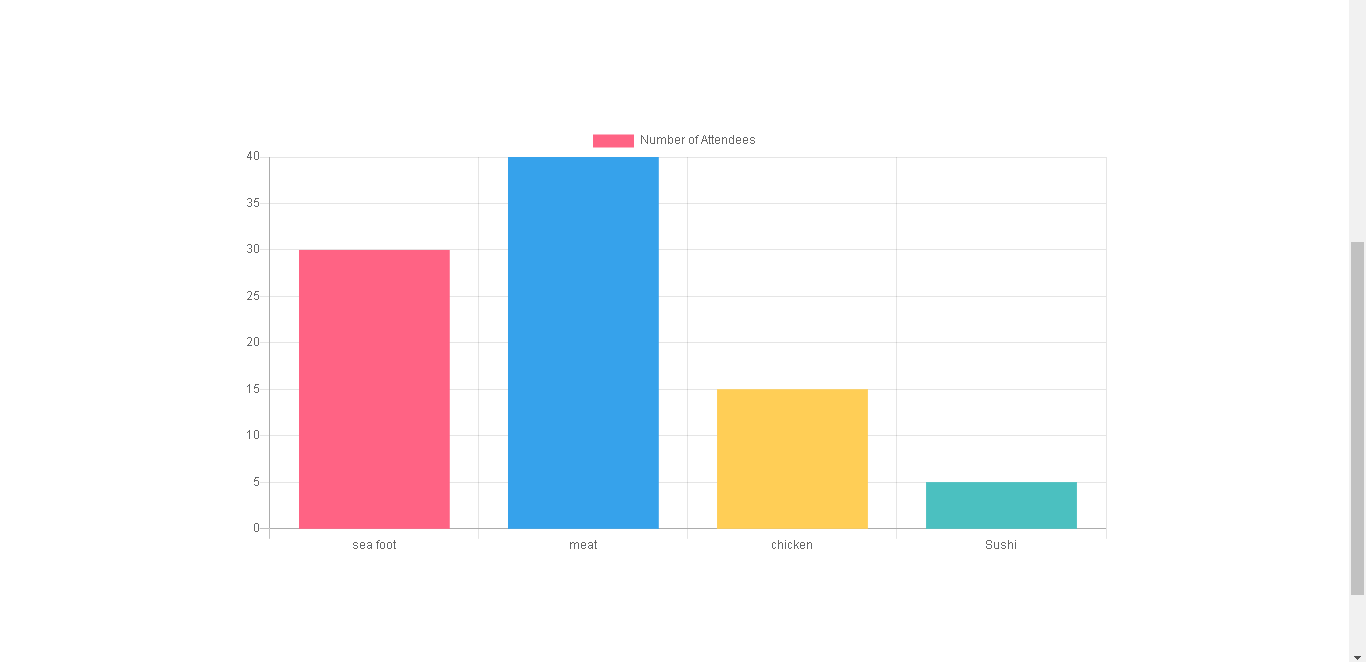
**Event Reports**

****

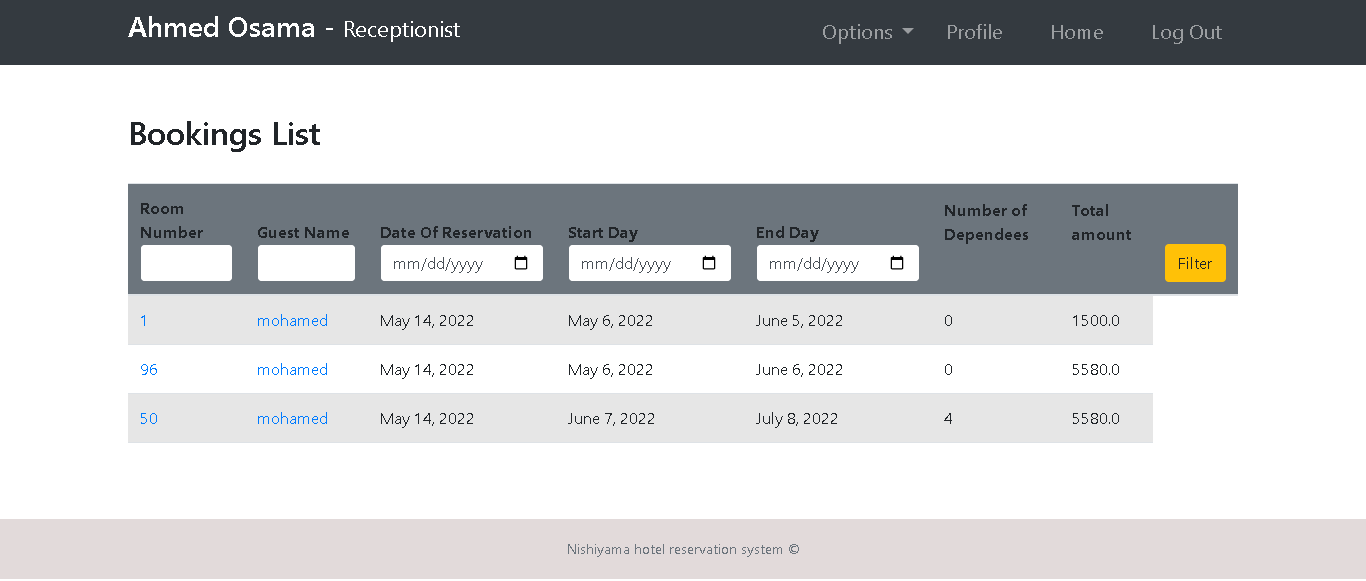
****

**Storage Report**

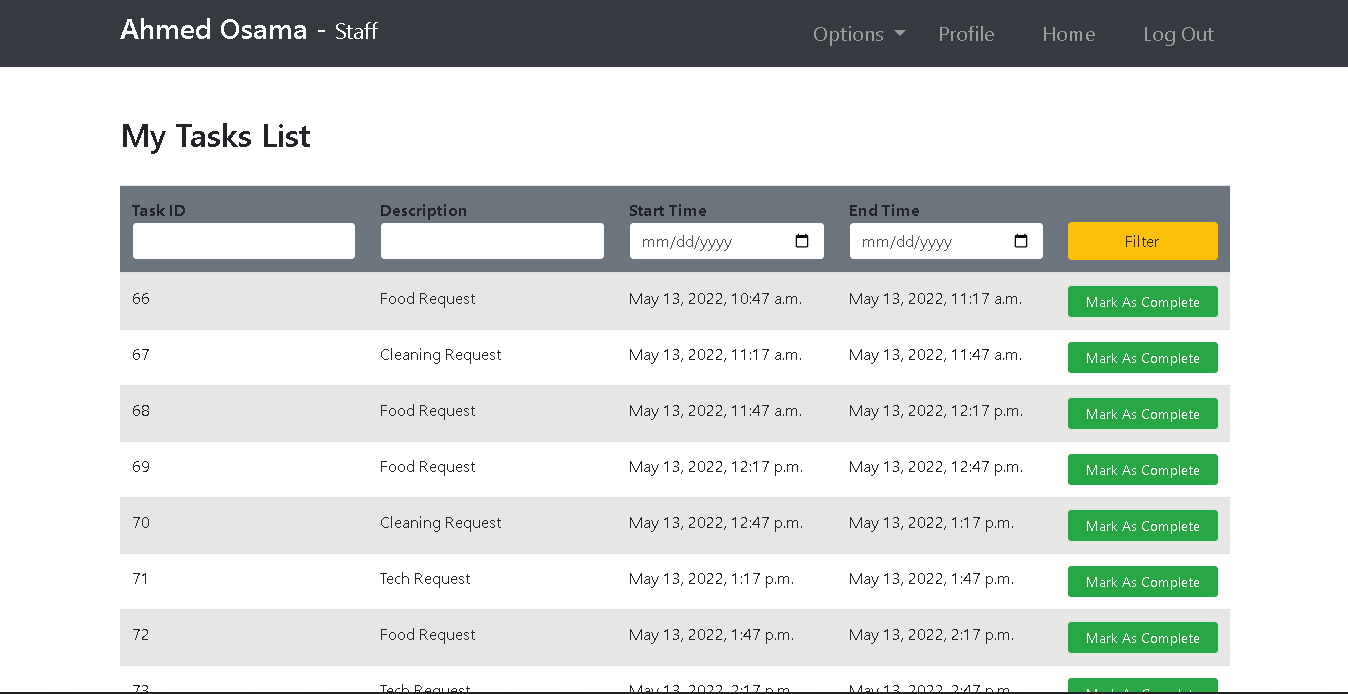
****

****

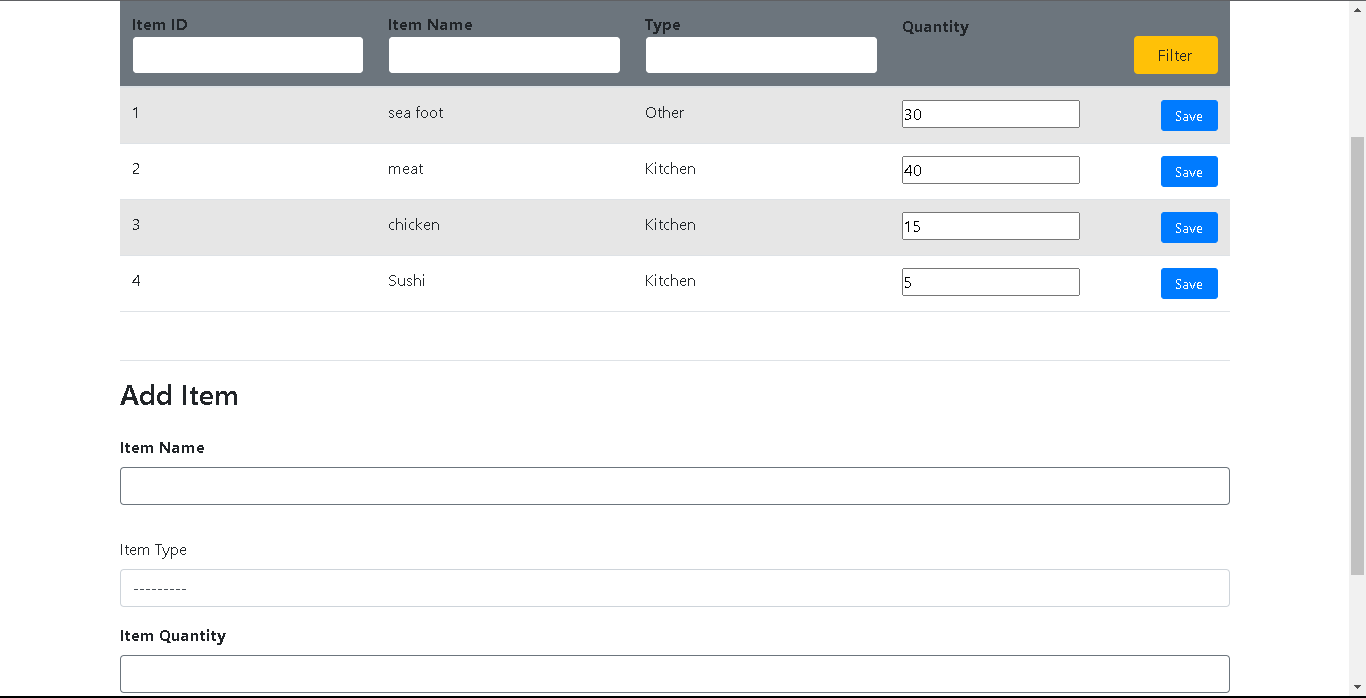
**Receptionist**

****

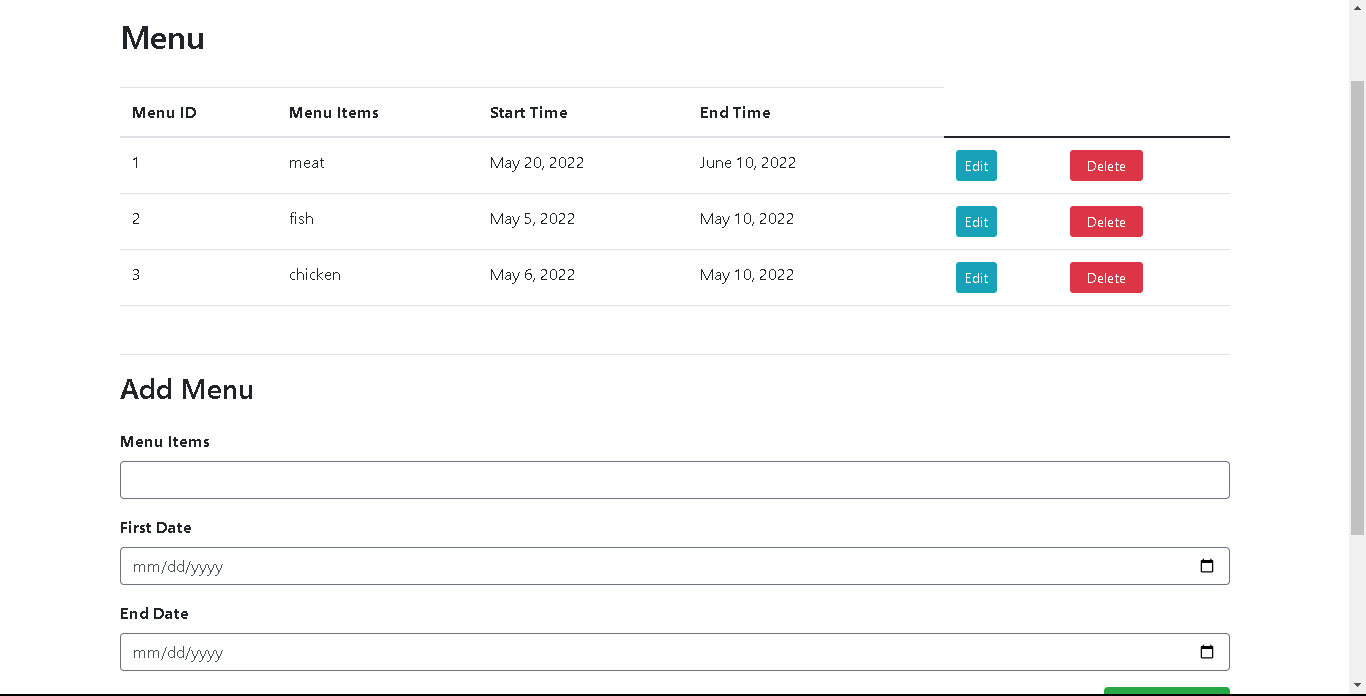
**Staff Tasks**

****

**Staff Storage**

****

**Staff menu**

****

Implementation

# Software and hardware

**Python**

is an object-oriented programming language. Almost everything in Python is considered as an object. An object has its own properties(attributes) and behavior(methods).

**Django Platform**

 is a high-level Python Web framework that encourages rapid development and clean, pragmatic design.

**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.

**JS**

JavaScript. JavaScript is the main programming language of the Web.

**SQLite**

SQLite is a C-language library that implements a [small](https://sqlite.org/footprint.html), [fast](https://sqlite.org/fasterthanfs.html), [self-contained](https://sqlite.org/selfcontained.html), [high-reliability](https://sqlite.org/hirely.html), [full-featured](https://sqlite.org/fullsql.html), SQL database engine.

**OS**

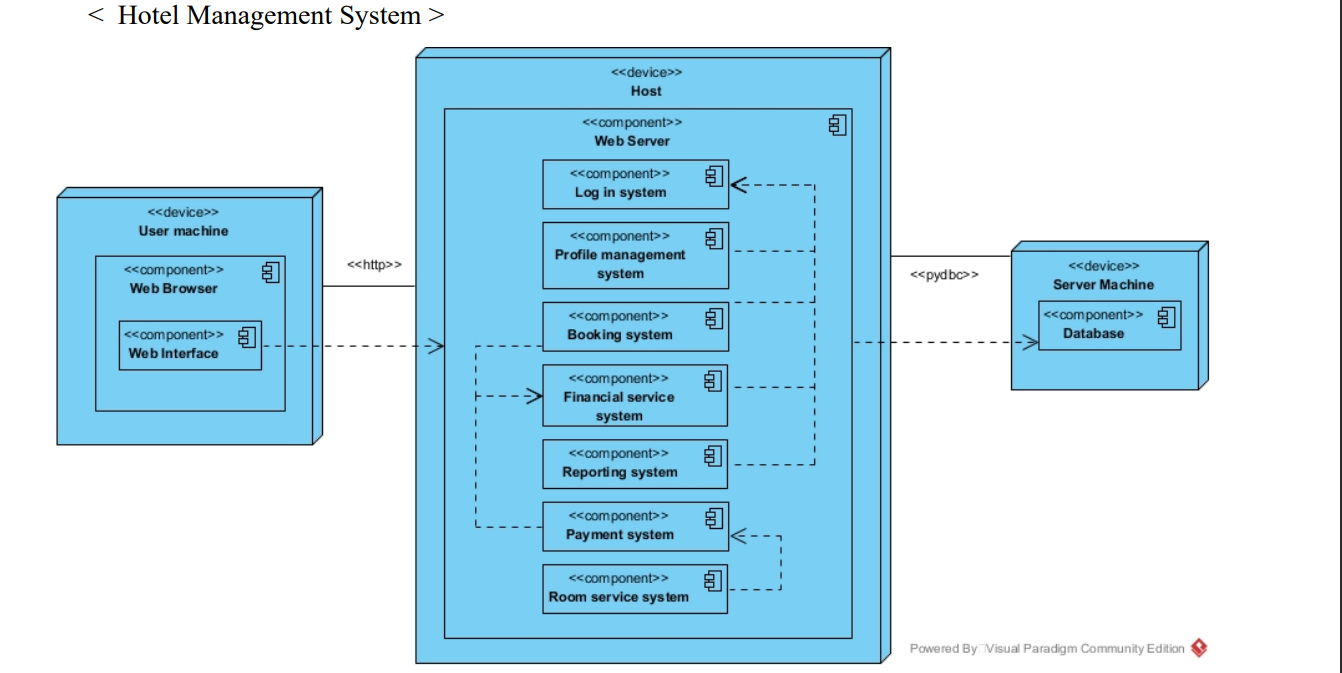
An operating system (OS) is [system software](https://en.wikipedia.org/wiki/System_software) that manages [computer hardware](https://en.wikipedia.org/wiki/Computer_hardware), [software](https://en.wikipedia.org/wiki/Computer_software) resources, and provides common [services](https://en.wikipedia.org/wiki/Daemon_(computing)) for [computer programs](https://en.wikipedia.org/wiki/Computer_program).

[Time-sharing](https://en.wikipedia.org/wiki/Time-sharing) operating systems [schedule tasks](https://en.wikipedia.org/wiki/Scheduler_(computing)) for efficient use of the system and may also include accounting software for cost allocation of [processor time](https://en.wikipedia.org/wiki/Scheduling_(computing)), [mass storage](https://en.wikipedia.org/wiki/Mass_storage), printing, and other resources.

**Photoshop**

Adobe Photoshop is the predominant photo editing and manipulation software on the market. Its uses range from the full-featured editing of large batches of photos to creating intricate digital paintings and drawings that mimic those done by hand.

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.

Database implementation

**What is database implementation?**

 Database implementation is the process of installing database software, performing configuration and customization, running and testing the database and then integrating it with applications. Finally, the implementation process involves training the users.

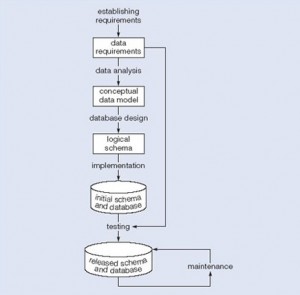
**Database Life Cycle**

We can use the waterfall cycle as the basis for a model of database development that incorporates three assumptions:

We can separate the development of a database – that is, specification and creation of a schema to define data in a database – from the user processes that make use of the database.

We can use the three-schema architecture as a basis for distinguishing the activities associated with a schema.

We can represent the constraints to enforce the semantics of the data once within a database, rather than within every user process that uses the data.

[](http://opentextbc.ca/dbdesign01/wp-content/uploads/sites/11/2013/12/SDLC-Database-300x295.jpg)

Using these assumptions and Figure 13.2, we can see that this diagram represents a model of the activities and their outputs for database development. It is applicable to any class of DBMS, not just a relational approach.

Database application development is the process of obtaining real-world requirements, analyzing requirements, designing the data and functions of the system, and then implementing the operations in the system.

**Analysis**

Data analysis begins with the statement of data requirements and then produces a conceptual data model. The aim of analysis is to obtain a detailed description of the data that will suit user requirements so that both high and low level properties of data and their use are dealt with. These include properties such as the possible range of values that can be permitted for attributes (e.g., in the school database example, the student course code, course title and credit points).

The conceptual data model provides a shared, formal representation of what is being communicated between clients and developers during database development – it is focused on the data in a database, irrespective of the eventual use of that data in user processes or implementation of the data in specific computer environments. Therefore, a conceptual data model is concerned with the meaning and structure of data, but not with the details affecting how they are implemented.

The conceptual data model then is a formal representation of what data a database should contain and the constraints the data must satisfy. This should be expressed in terms that are independent of how the model may be implemented. As a result, analysis focuses on the questions, “What is required?” not “How is it achieved?”

**What’s the Implementation?**

Implementation involves the construction of a database according to the specification of a logical schema. This will include the specification of an appropriate storage schema, security enforcement, external schema and so on. Implementation is heavily influenced by the choice of available DBMSs, database tools and operating environment. There are additional tasks beyond simply creating a database schema and implementing the constraints – data must be entered into the tables, issues relating to the users and user processes need to be addressed, and the management activities associated with wider aspects of corporate data management need to be supported. In keeping with the DBMS approach, we want as many of these concerns as possible to be addressed within the DBMS. We look at some of these concerns briefly now.

In practice, implementation of the logical schema in a given DBMS requires a very detailed knowledge of the specific features and facilities that the DBMS has to offer. In an ideal world, and in keeping with good software engineering practice, the first stage of implementation would involve matching the design requirements with the best available implementing tools and then using those tools for the implementation. In database terms, this might involve choosing vendor products with DBMS and SQL variants most suited to the database we need to implement. However, we don’t live in an ideal world and more often than not, hardware choice and decisions regarding the DBMS will have been made well in advance of consideration of the database design. Consequently, implementation can involve additional flexing of the design to overcome any software or hardware limitations.

**Definitions, Acronyms, and Abbreviations ODD**: Object Design Document

**HMS:** Hotel Management System

**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.

**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.

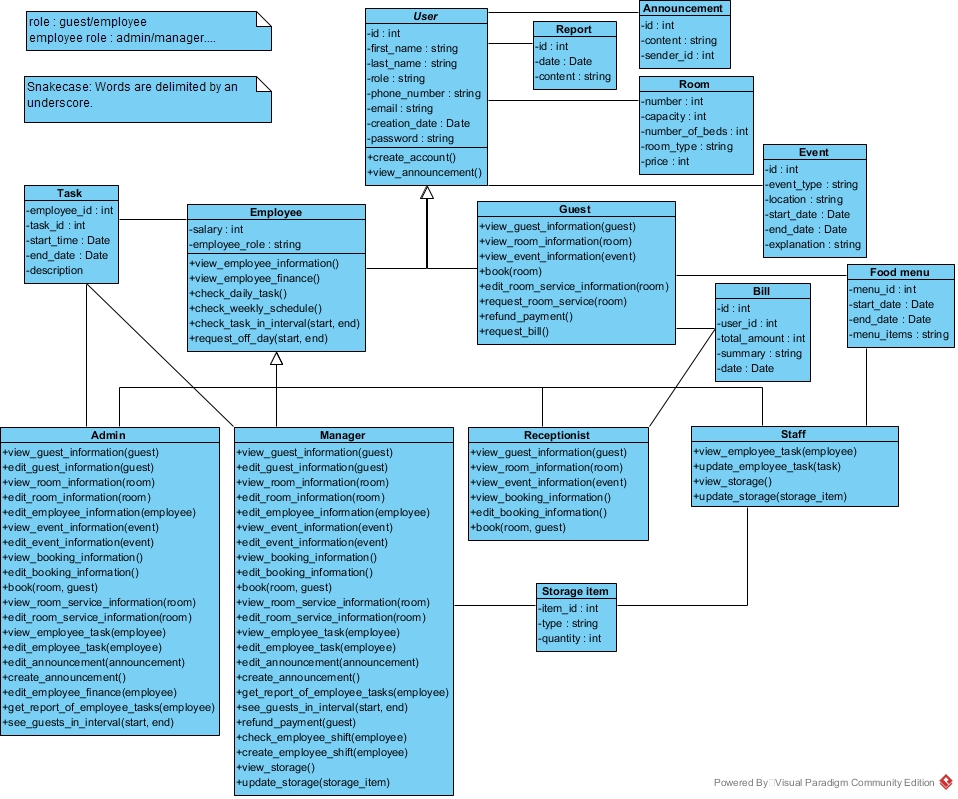
**UX:** User Experience. UX abbreviation is used to define the design process to create products that provide meaningful and proper experiences to users.

And now we are going to show you the Class Diagram for our program

But in the first **what’s the class diagram?**

**Class Diagram is**:

A class diagram is a type of diagram and part of a unified modeling language (UML) that defines and provides the overview and structure of a system in terms of classes, attributes and methods, and the relationships between different classes.



One of the important diagrams is Use Case

And now we are going to show you the Use Case Diagram for our program

But in the first **what’s the Use Case diagram?**

**Use-case diagrams:**

In UML, use-case diagrams model the behavior of a system and help to capture the requirements of the system.

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in use-case diagrams describe what the system does and how the actors use it, but not how the system operates internally.

Use-case diagrams illustrate and define the context and requirements of either an entire system or the important parts of the system. You can model a complex system with a single use-case diagram, or create many use-case diagrams to model the components of the system. You would typically develop use-case diagrams in the early phases of a project and refer to them throughout the development process.

Use-case diagrams are helpful in the following situations:

Before starting a project, you can create use-case diagrams to model a business so that all participants in the project share an understanding of the workers, customers, and activities of the business.

While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.

During the analysis and design phases, you can use the use cases and actors from your use-case diagrams to identify the classes that the system requires.

During the testing phase, you can use use-case diagrams to identify tests for the system.

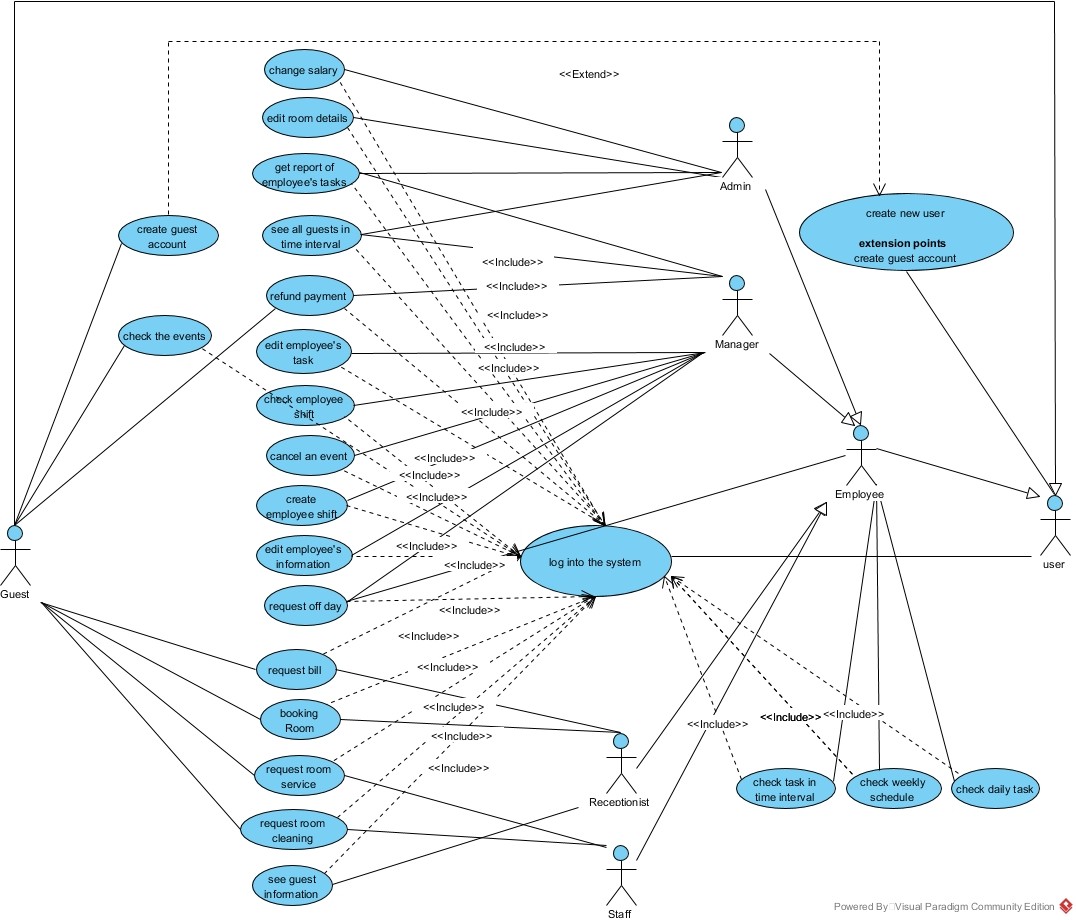
The following topics describe model elements in use-case diagrams:

[Use cases](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/cuc.html)  
A use case describes a function that a system performs to achieve the user’s goal. A use case must yield an observable result that is of value to the user of the system.

[Actors](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/cactor.html)  
An actor represents a role of a user that interacts with the system that you are modeling. The user can be a human user, an organization, a machine, or another external system.

[Subsystems](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/csubsys.html)  
In UML models, subsystems are a type of stereotyped component that represent independent, behavioral units in a system. Subsystems are used in class, component, and use-case diagrams to represent large-scale components in the system that you are modeling.

[Relationships in use-case diagrams](https://www.ibm.com/docs/en/SS8PJ7_9.6.1/com.ibm.xtools.modeler.doc/topics/crelsme_ucd.html)  
In UML, a relationship is a connection between model elements. A UML relationship is a type of model element that adds semantics to a model by defining the structure and behavior between the model elements.



**Object Model**

In the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML), an object diagram focuses on some particular set of [objects](https://en.wikipedia.org/wiki/Object_(computer_science)) and [attributes](https://en.wikipedia.org/wiki/Attribute_(computing)), and the links between these instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. In early UML specifications the object diagram is described as:

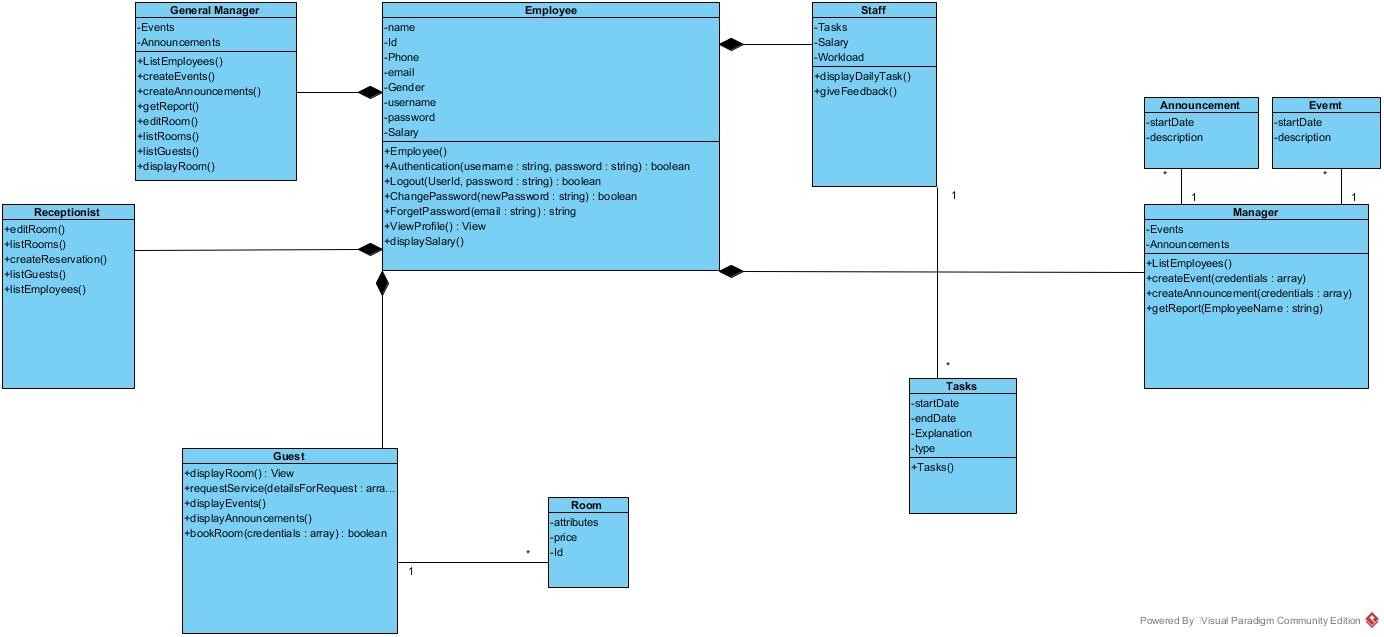
"*An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, namely to show examples of data structure*."[[1]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-1)[[2]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-2)

The latest UML 2.5 specification does not explicitly define object diagrams,[[3]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-3) but provides a notation for instances of classifiers.[[4]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-4)

Object diagrams and class diagrams are closely related[[5]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-5) and use almost identical notation.[[6]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-6) Both diagrams are meant to visualize static structure of a system. While class diagrams show [classes](https://en.wikipedia.org/wiki/Class_(computer_programming)), object diagrams display instances of classes ([objects](https://en.wikipedia.org/wiki/Object_(computer_science))).[[7]](https://en.wikipedia.org/wiki/Object_diagram#cite_note-7) Object diagrams are more concrete than [class diagrams](https://en.wikipedia.org/wiki/Class_diagram). They are often used to provide examples or act as test cases for class diagrams. Only aspects of current interest in a model are typically shown on an object diagram.

### **Object Diagram Usage**

If you are using a [UML modeling tool](https://en.wikipedia.org/wiki/UML_tool), you will typically draw object diagrams using some other diagram type, such as on a [class diagram](https://en.wikipedia.org/wiki/Class_diagram). An object instance may be called an *instance specification* or just an *instance*. A link between instances is generally referred to as a *link*. Other UML entities, such as an [aggregation](https://en.wikipedia.org/wiki/Aggregation_(object-oriented_programming)) or [composition](https://en.wikipedia.org/wiki/Object_composition) symbol (a diamond) may also appear on an object diagram.



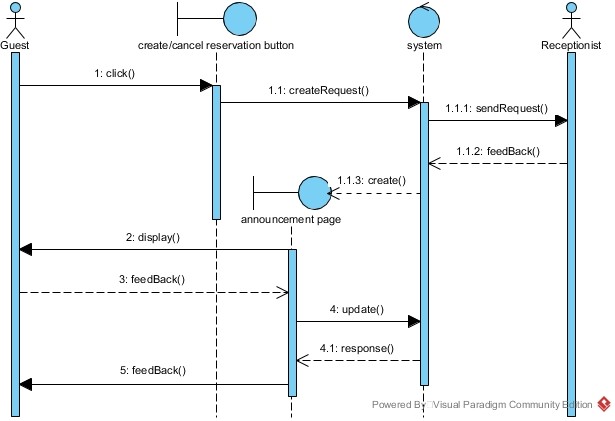
But to get a clear understand of our program sequence we have to show you the sequence diagram of our HMS

But in the First **What’s the Sequence Diagram**

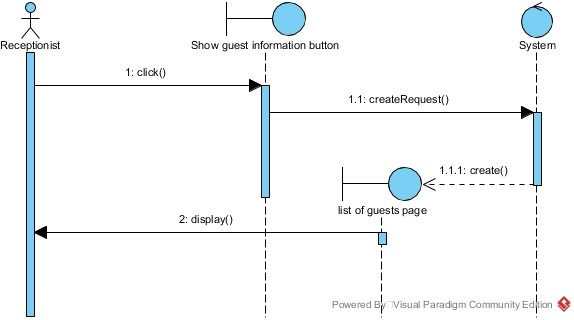
**A sequence diagram is**

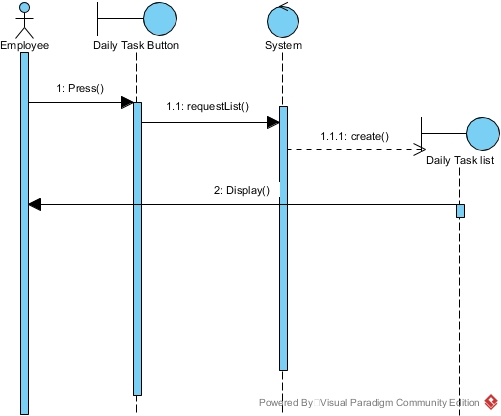
a Unified Modeling Language (UML) diagram that illustrates the sequence of messages between objects in an interaction. A sequence diagram consists of a group of objects that are represented by lifelines, and the messages that they exchange over time during the interaction.

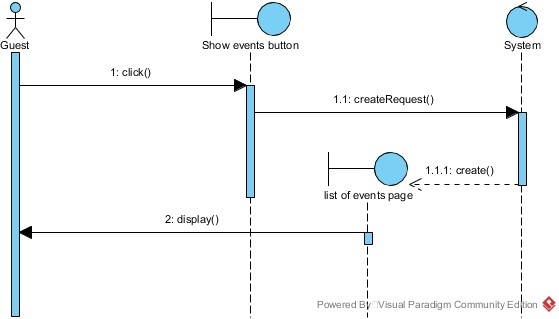
**Booking room**



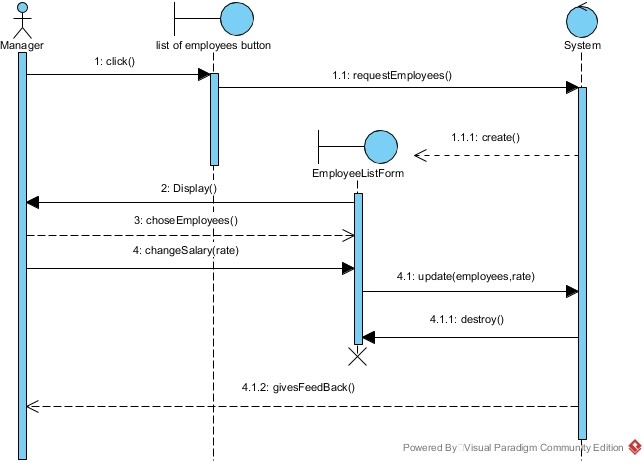
**See guest information**

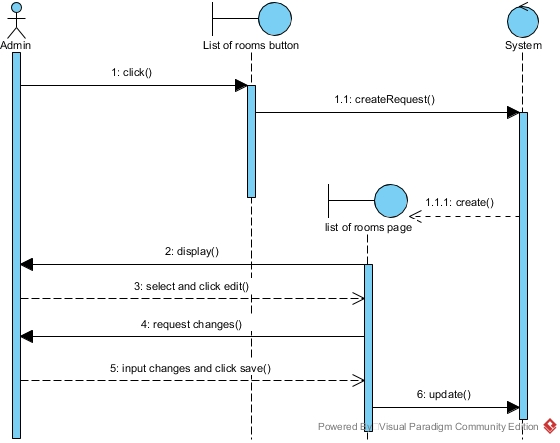


check daily tasks

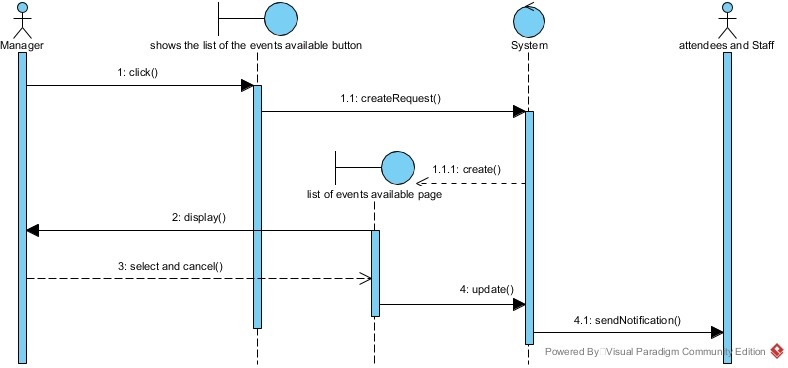
**check**

**change**



**edit room details**

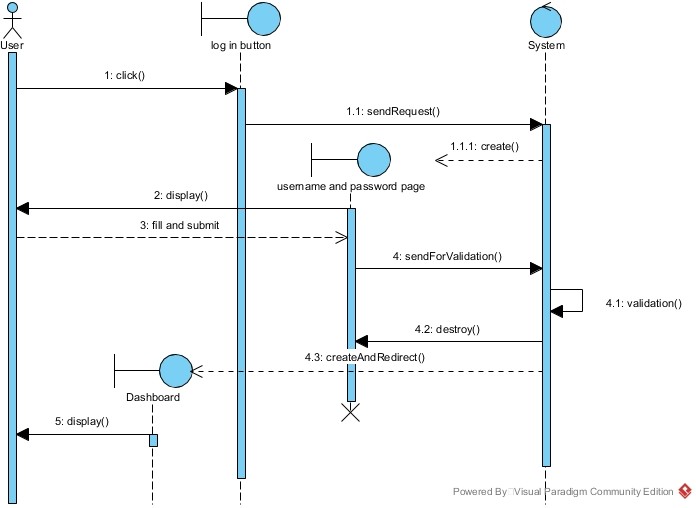
**cancel**



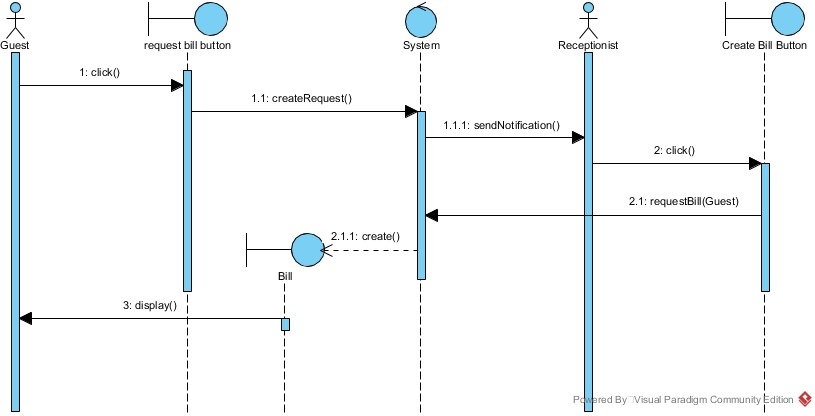
**edit**



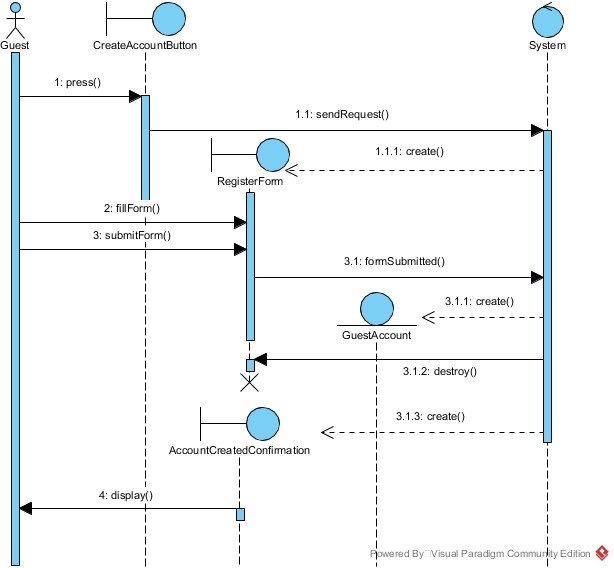
**log into the system**

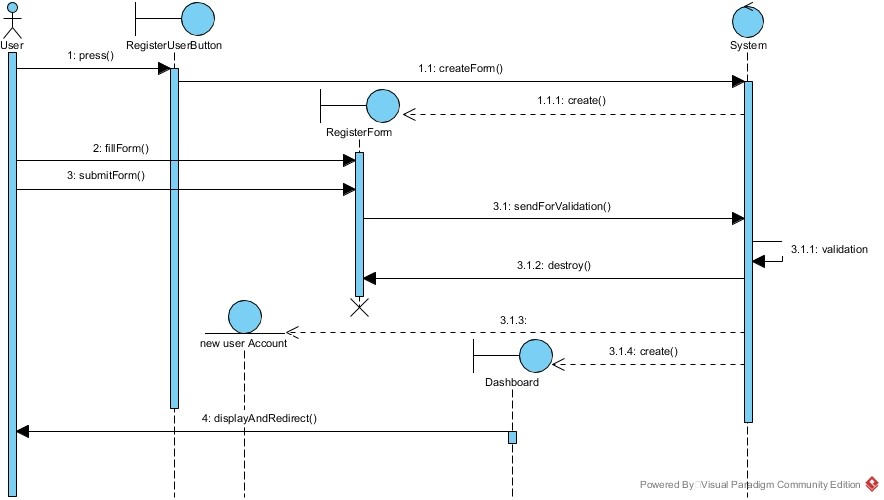


**request bill**

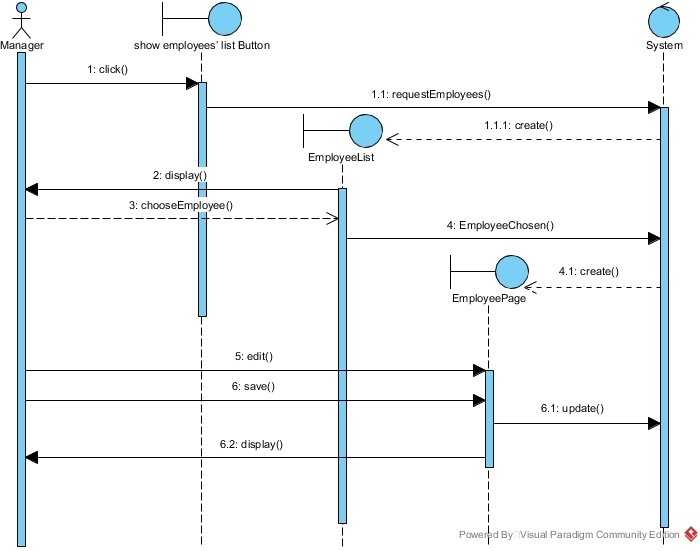


**create guest account**

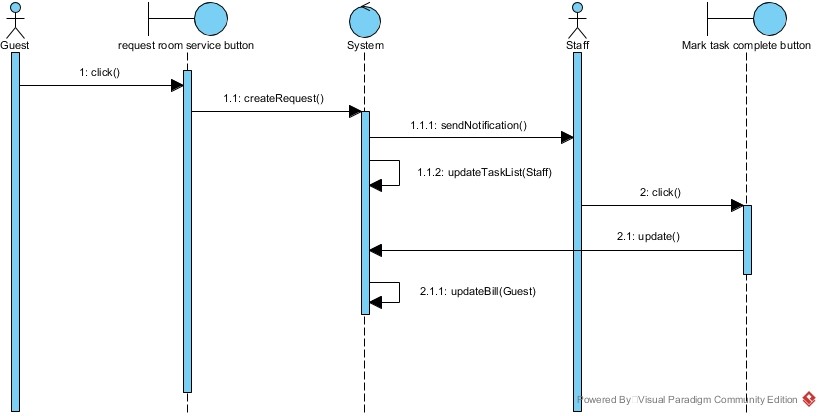


**create new user**

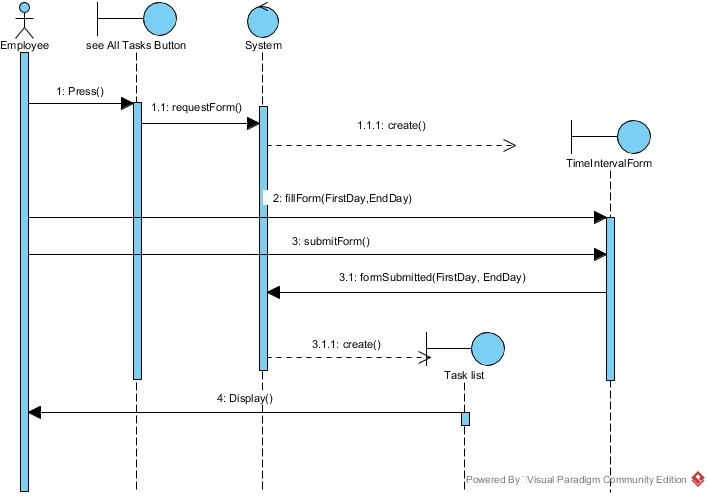
**edit**



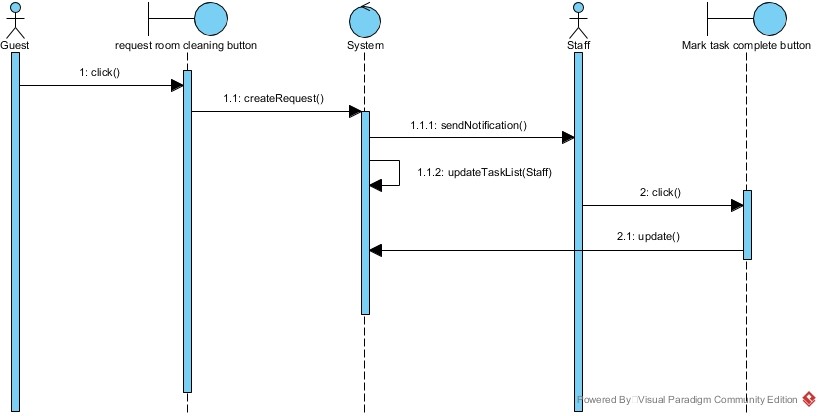
**Request room Service**



**check tasks for specific time interval**



**Request room cleaning**



References & Conclusion

# conclusions:

The entire project has been developed as per the requirements stated by the

user, it is found to be bug free as per the testing standards that are implemented. Any

specification untraced errors will be concentrated in the coming versions, which are planned to

be developed in near future.

The system at present does not take care of the money payment methods, as the

consolidated constructs need SSL standards and are critically to be initiated in the first face, the

application of the credit card transactions is applied as a developmental phase in the coming

days. The system needs more elaborative technicality for its inception and evolution.

# References:

1. Bruegge B. & Dutoit A.H.. (2010). Object-Oriented Software Engineering Using UML, Patterns, and Java, Prentice Hall, 3rd ed.

2. [http://www.cs.sjsu.edu/faculty/pearce/modules/lectures/ooa/requirements/IdentifyingURP S.htm](http://www.cs.sjsu.edu/faculty/pearce/modules/lectures/ooa/requirements/IdentifyingURP%20S.htm)

3. Ian Sommerville (2015). Software Engineering, Pearson tenth edition.

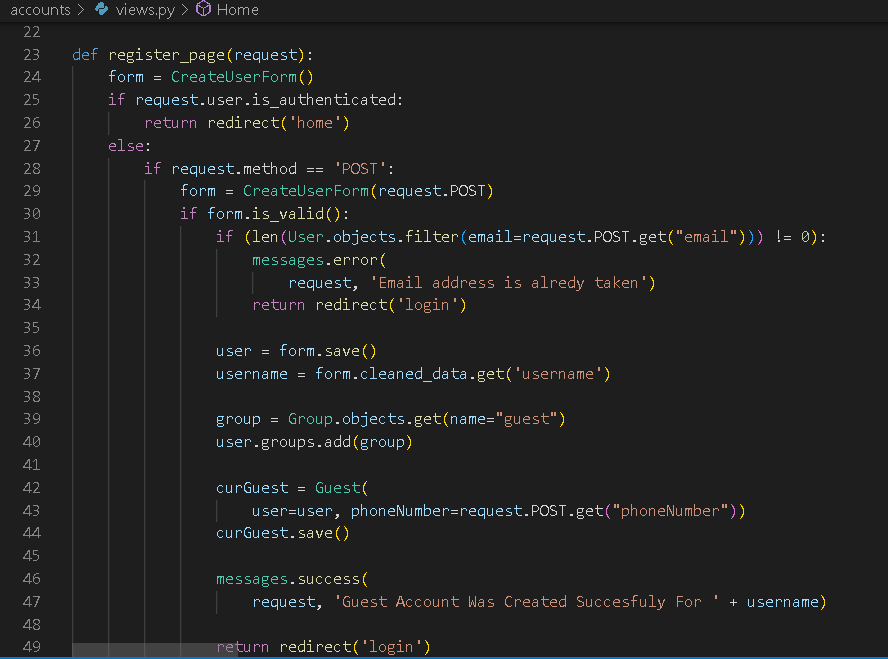
4. Currently there is no system in place to be replaced by the system we are building.

5. hotel Tec report

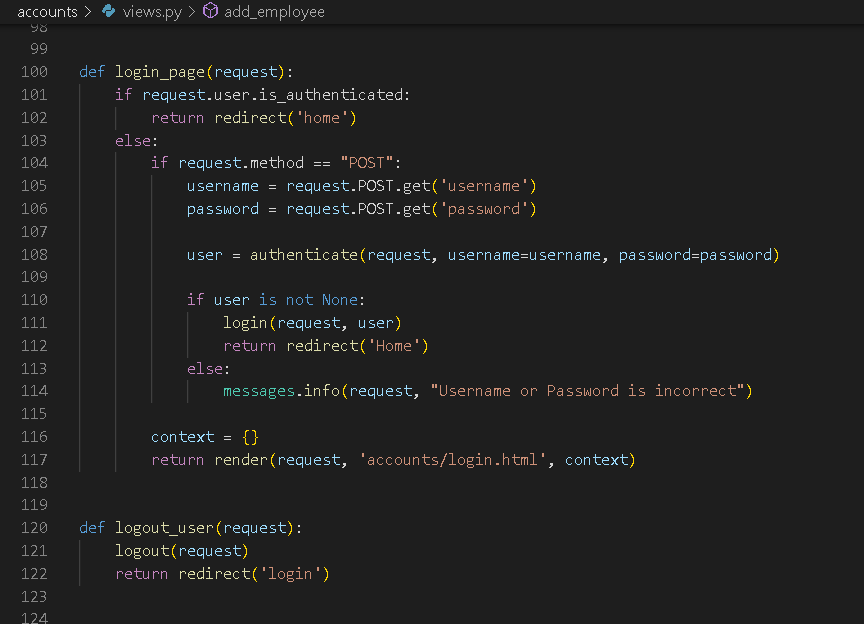
6. Currently there is no system in place to be replaced by the system we are building.

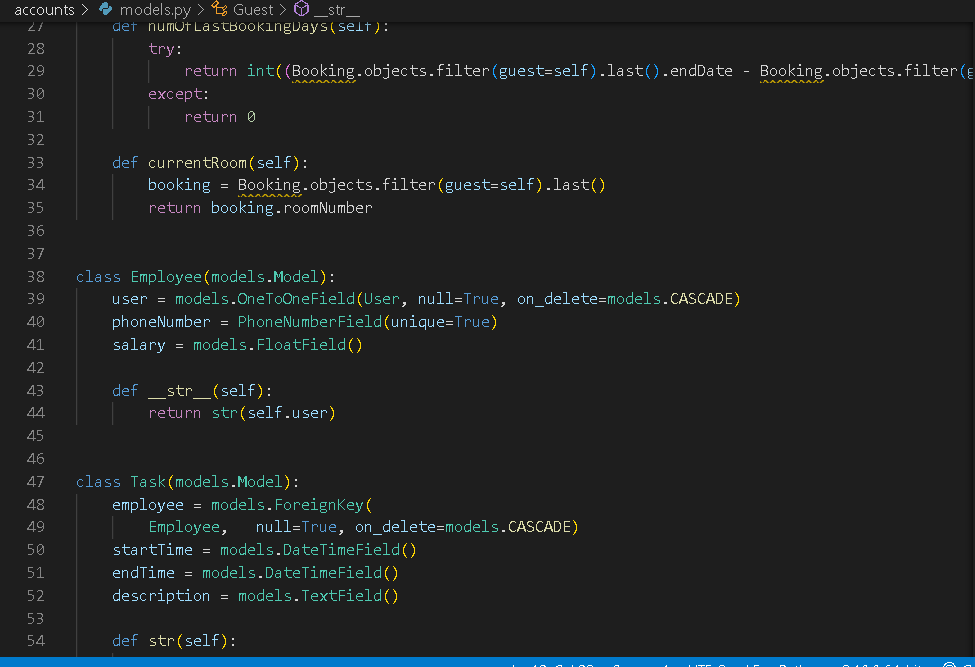
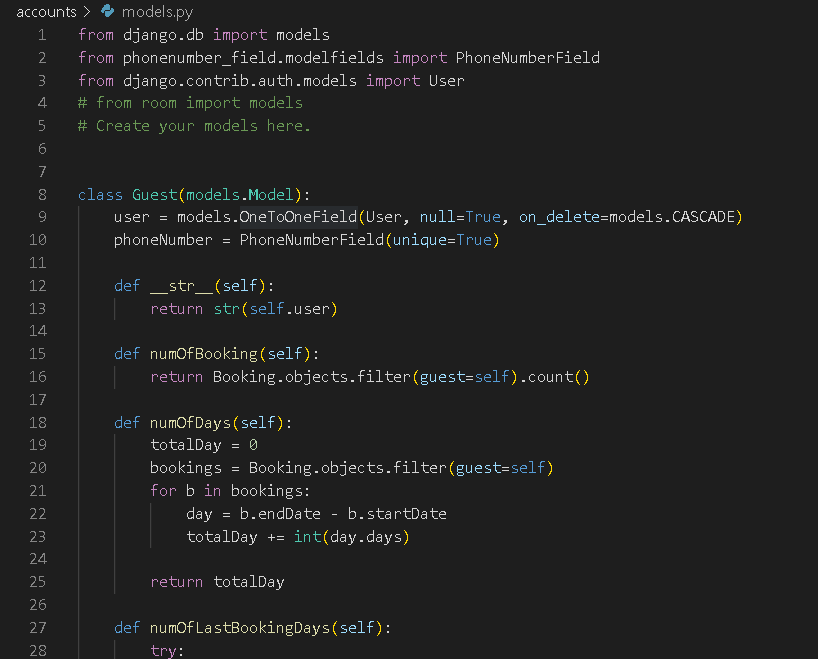
7. [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language)

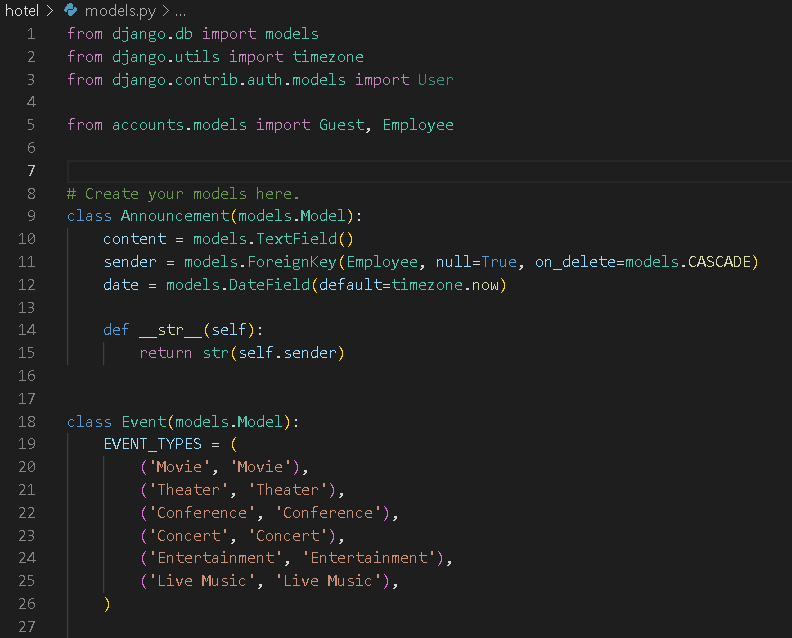
Code

**Register Back End**

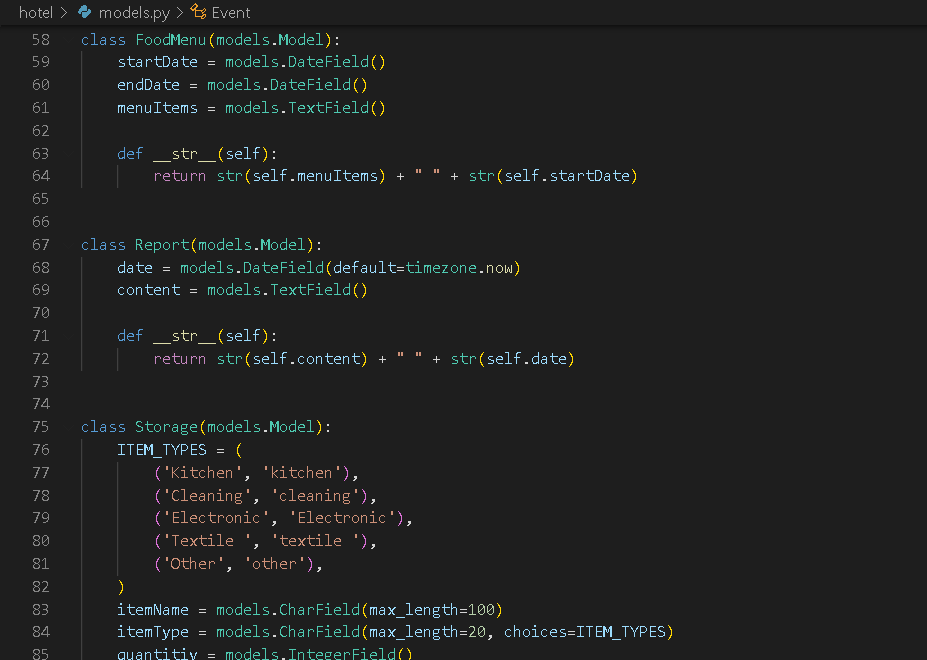
**Login Back End**

****

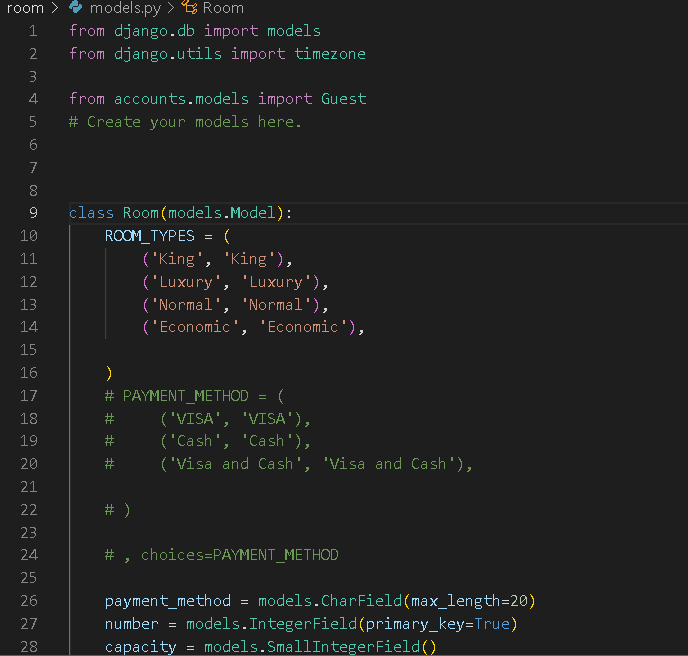
**Accounts Data Base**

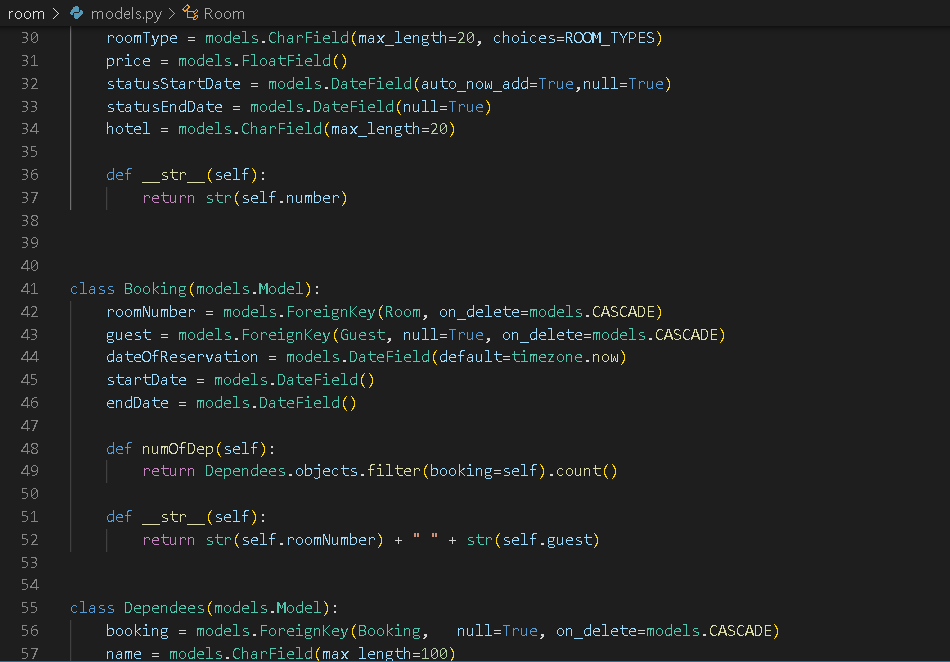
**Hotel Data Base**

****

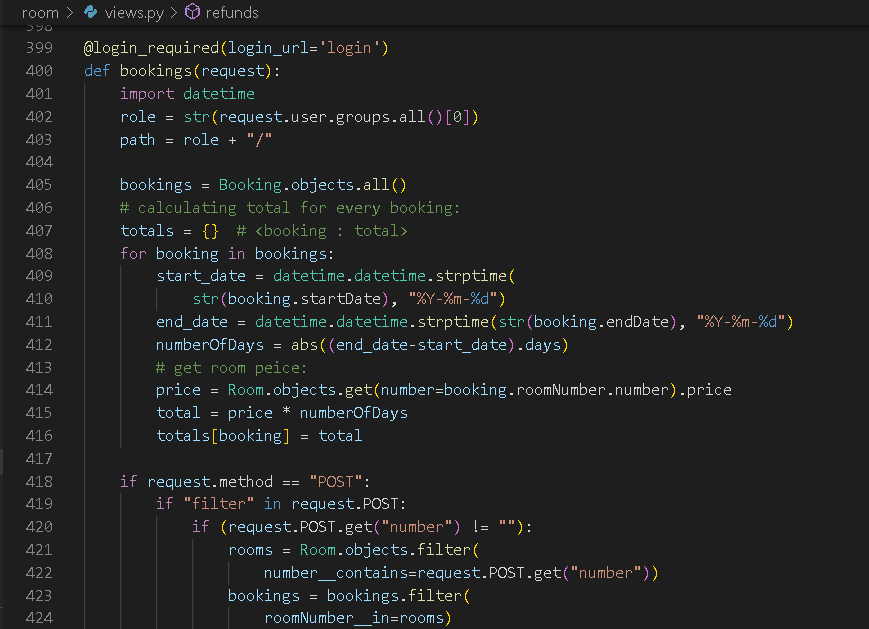
****

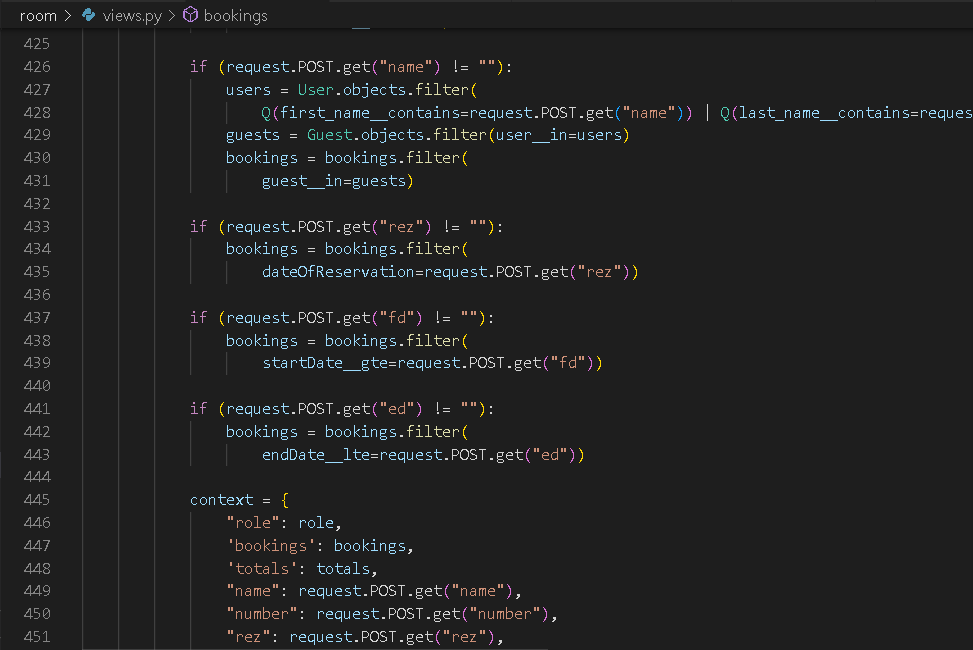
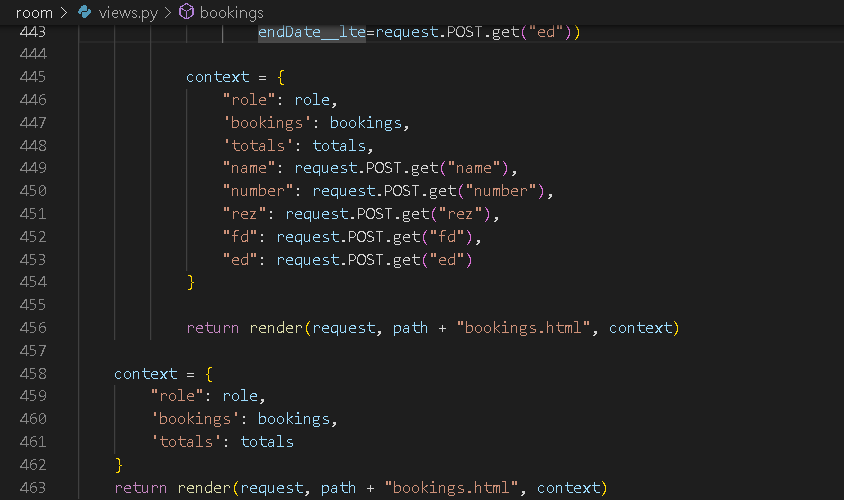
**Room Data Base**

****

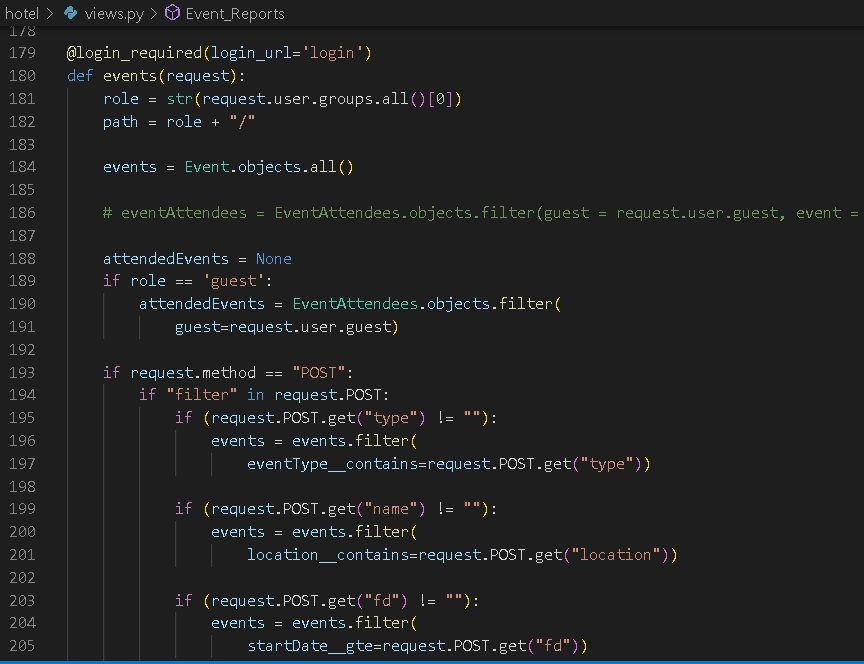
****

****

**Booing Back End**

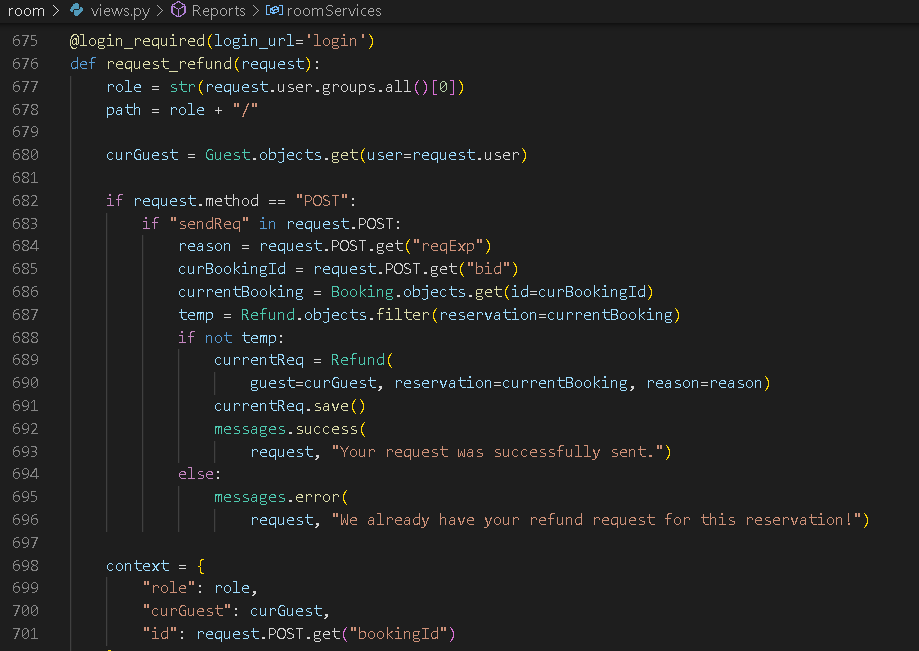
****

**Event Back End**

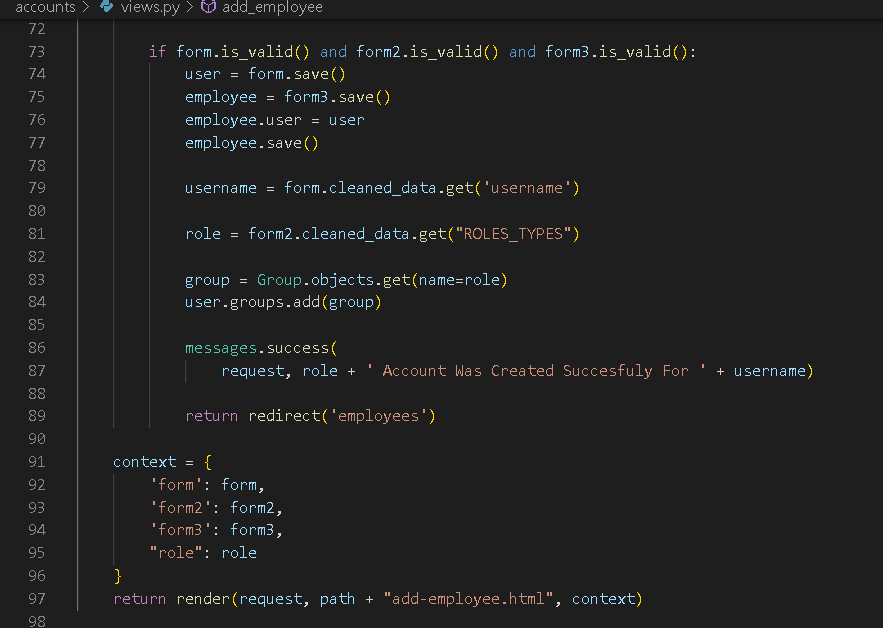
****

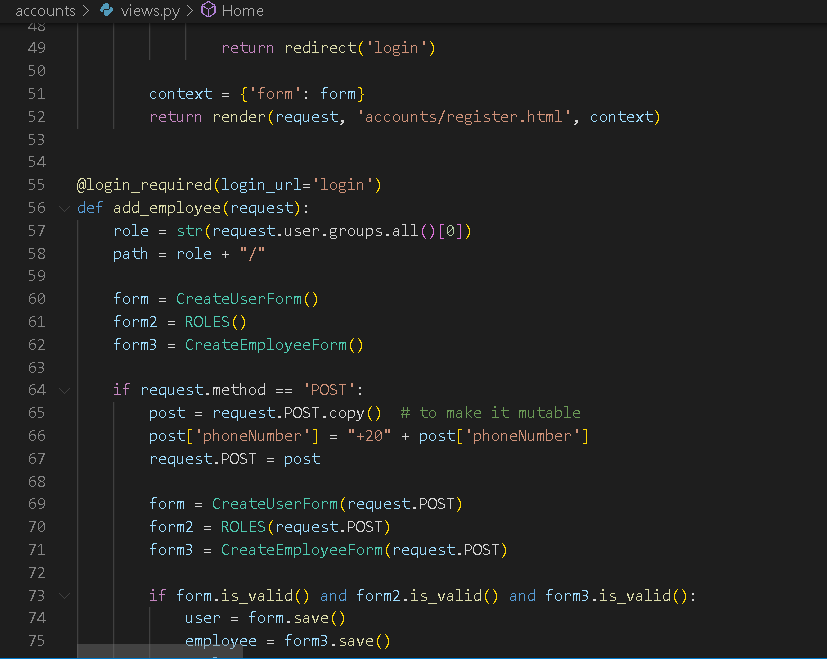
****

**Refund Request Back End**

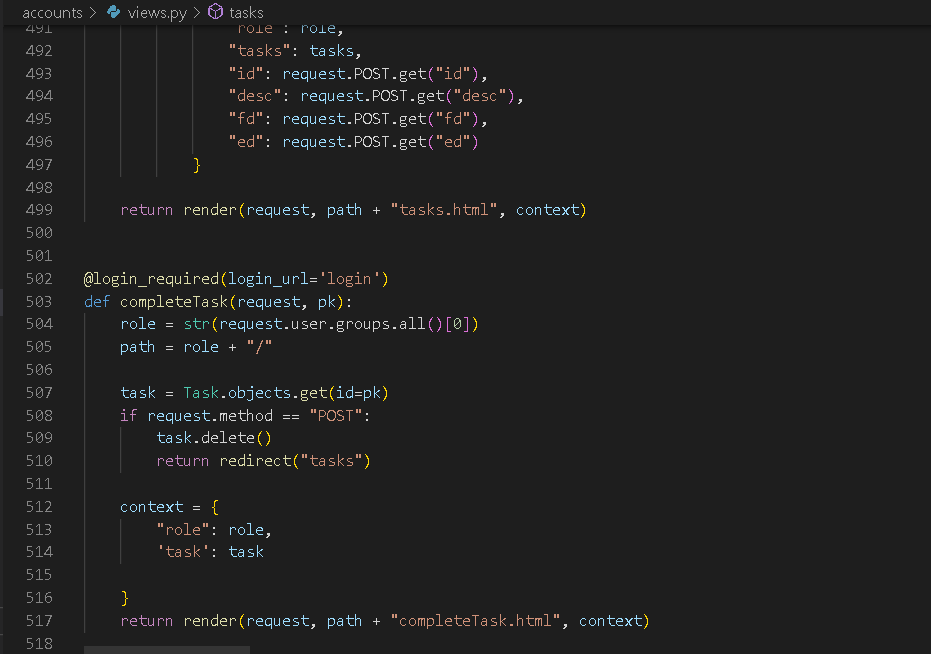
****

**Food Menu Back End**

**Add Employee Back End**

****

**Complete Task Back End**

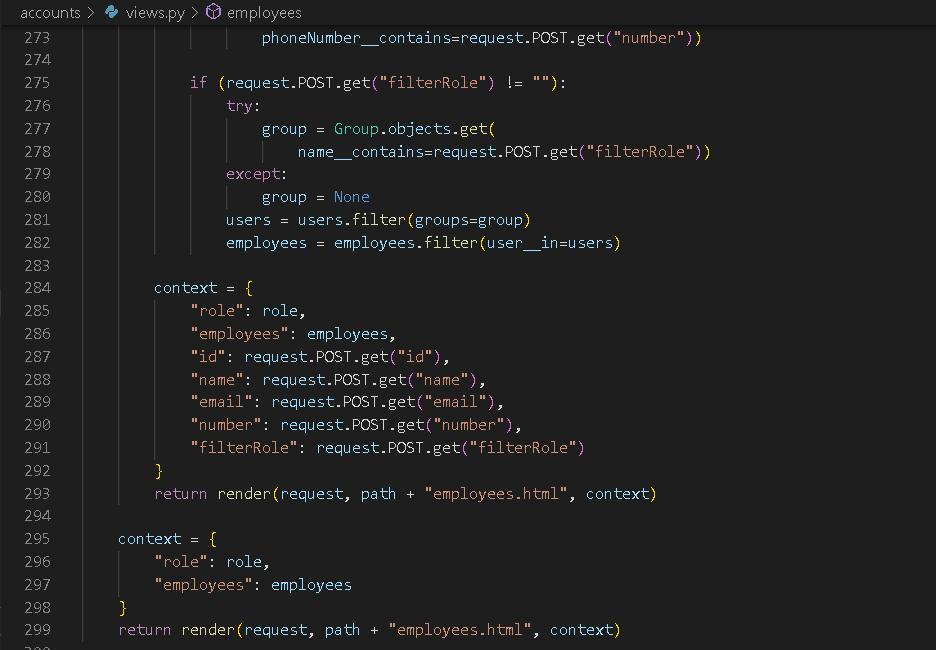
****

**Employee Details**

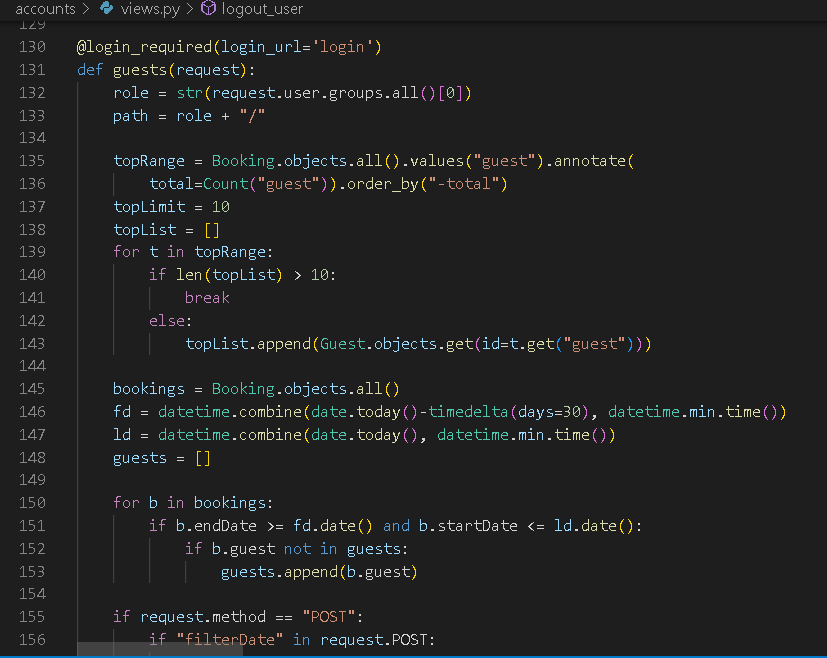
****

**Employee Page**

****

****

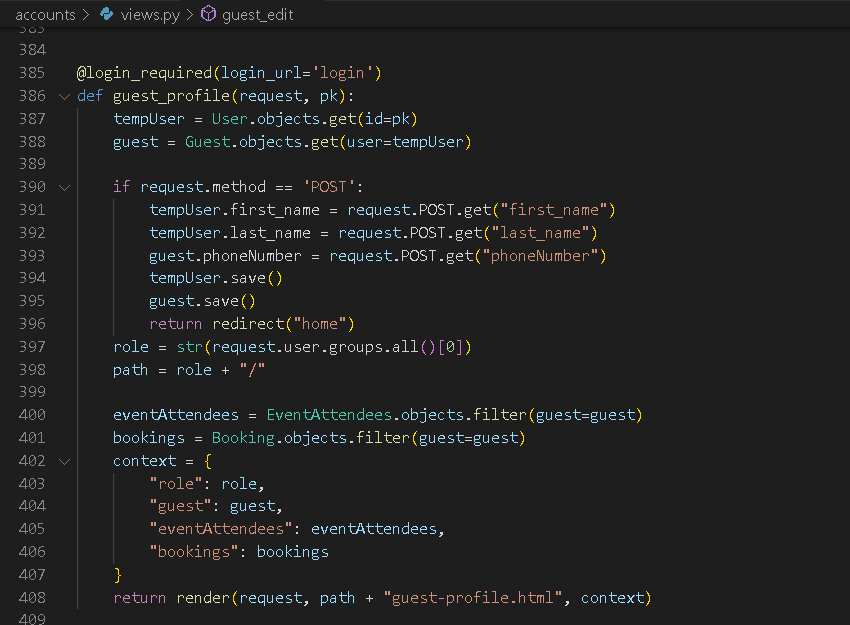
**Guest Page Back End**

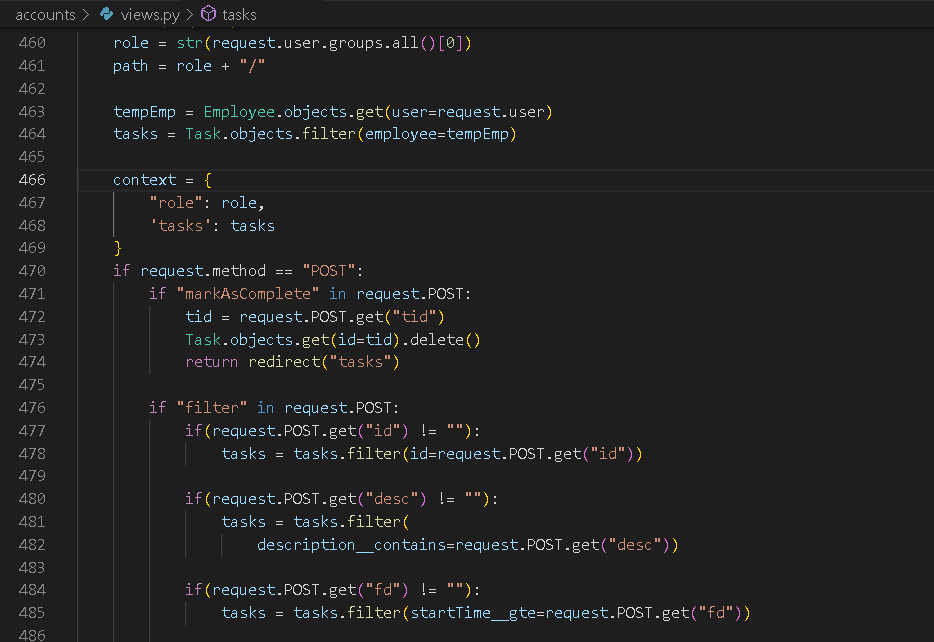
****

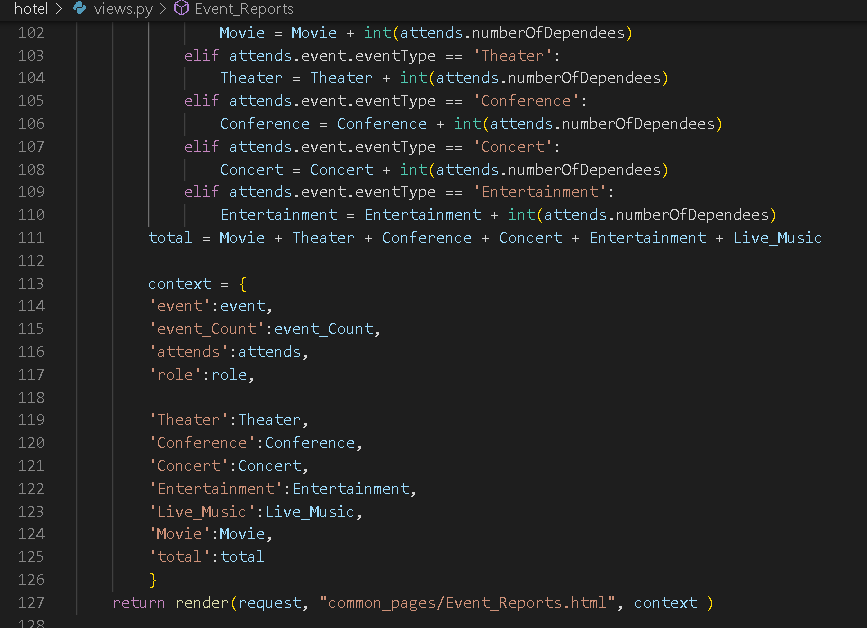
****

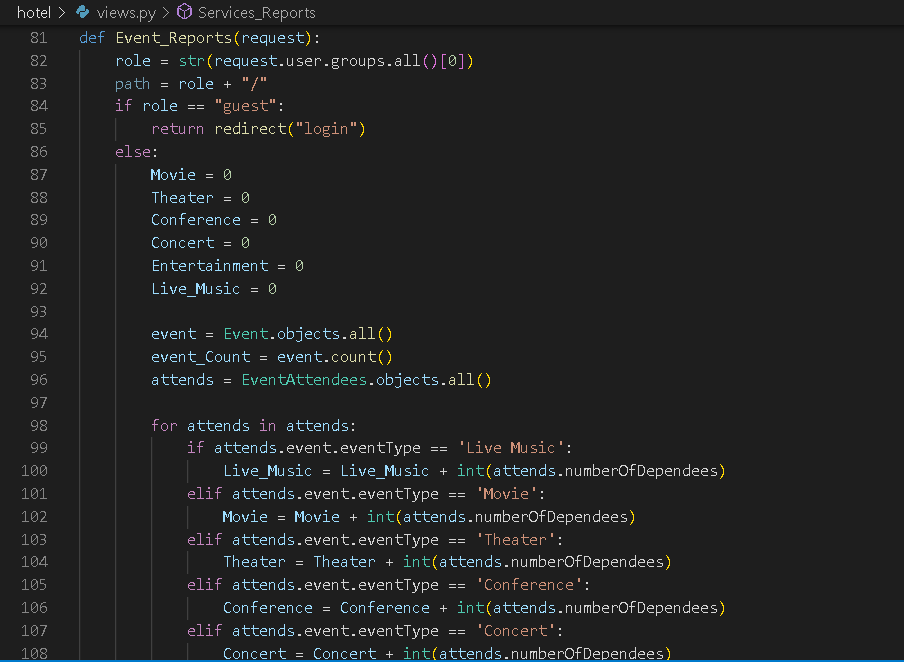
**Edit Guest**

****

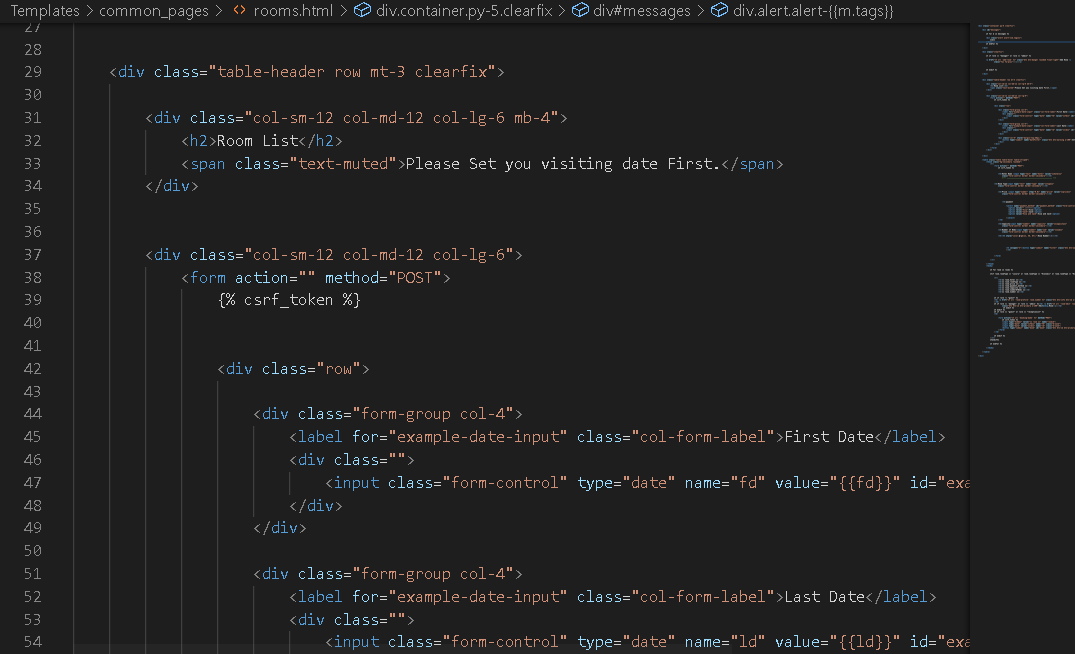
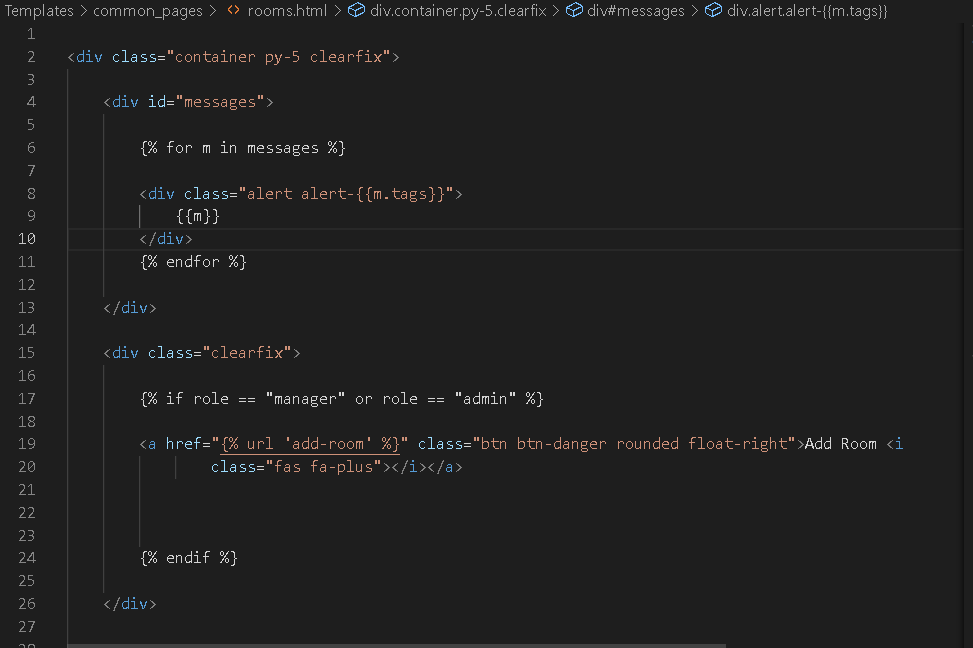
**Guest Profile **

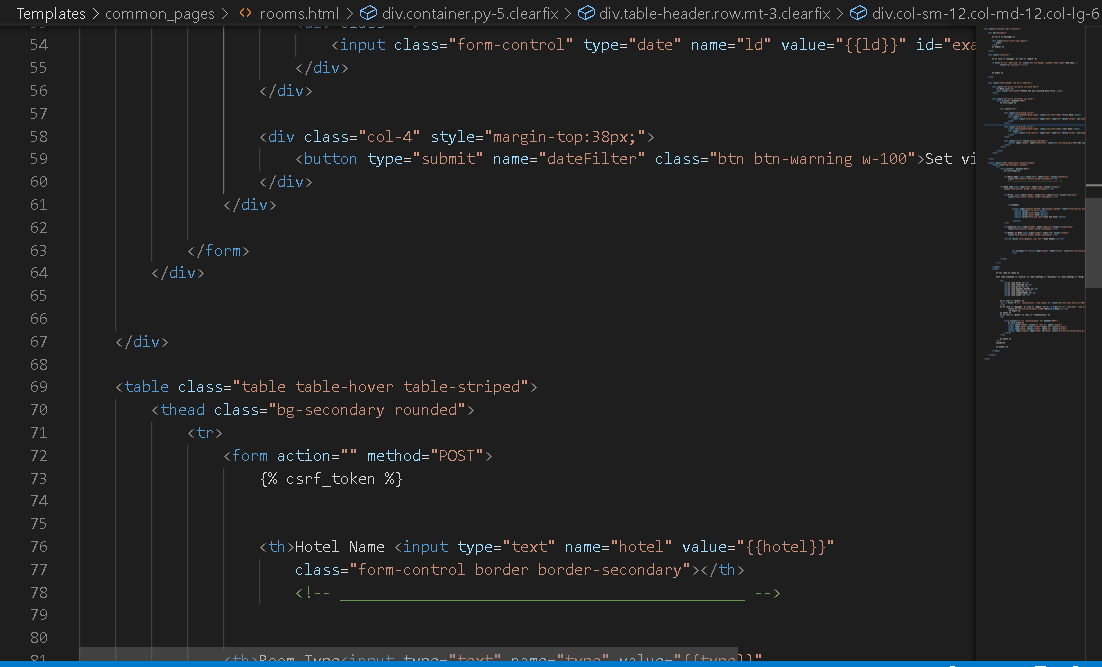
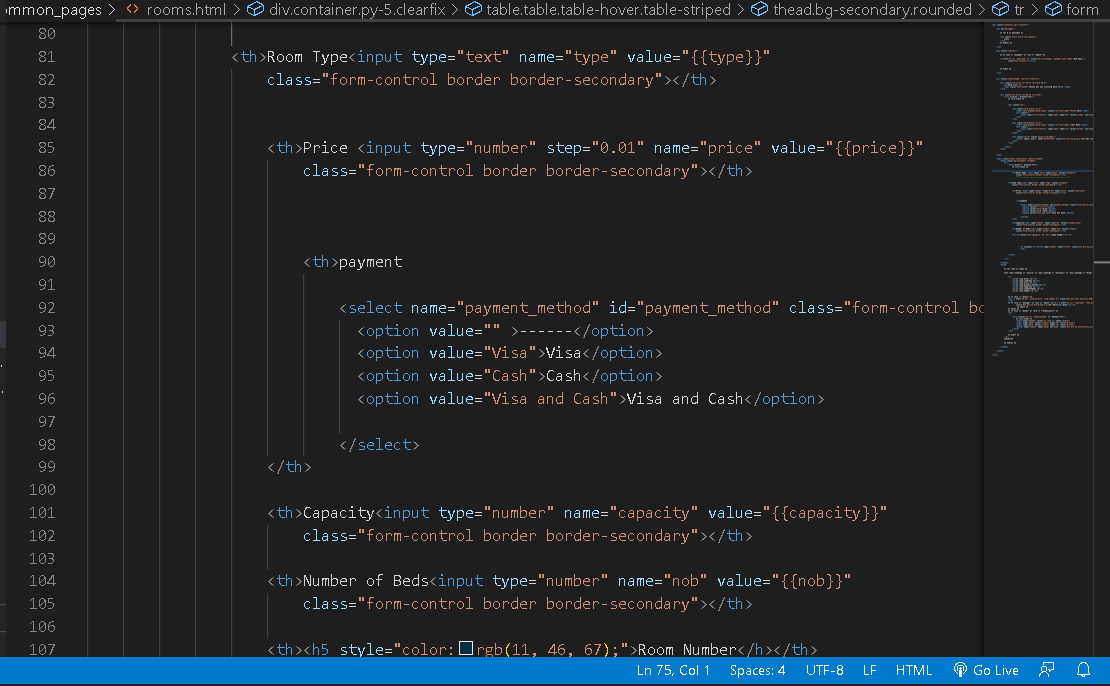
**Tasks page **

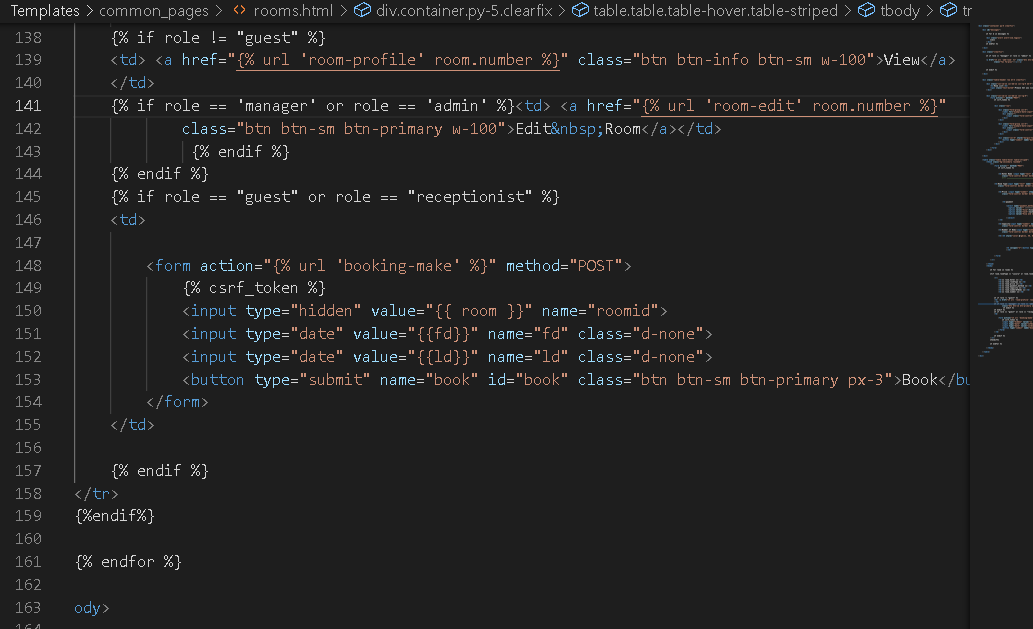
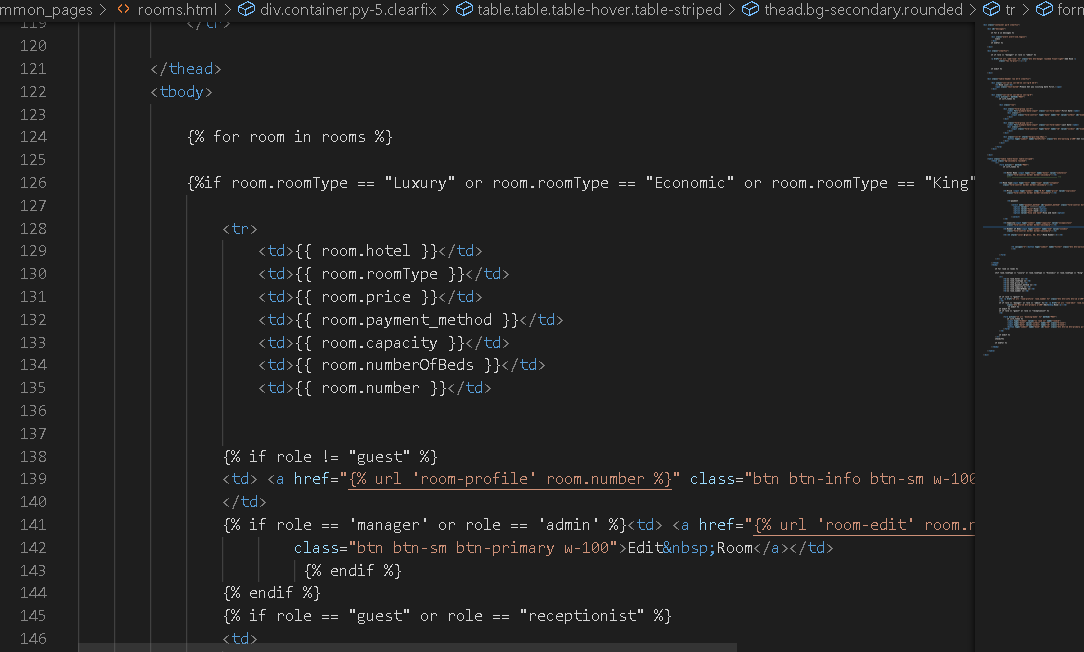
**Event Reports**

****

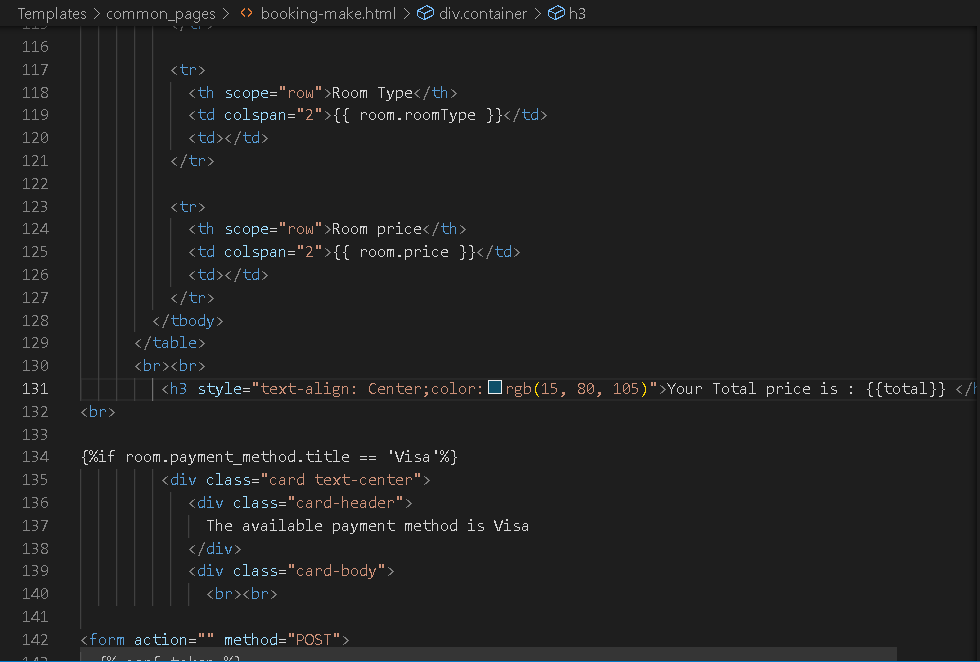
**Rooms Template (HTML, CSS)**

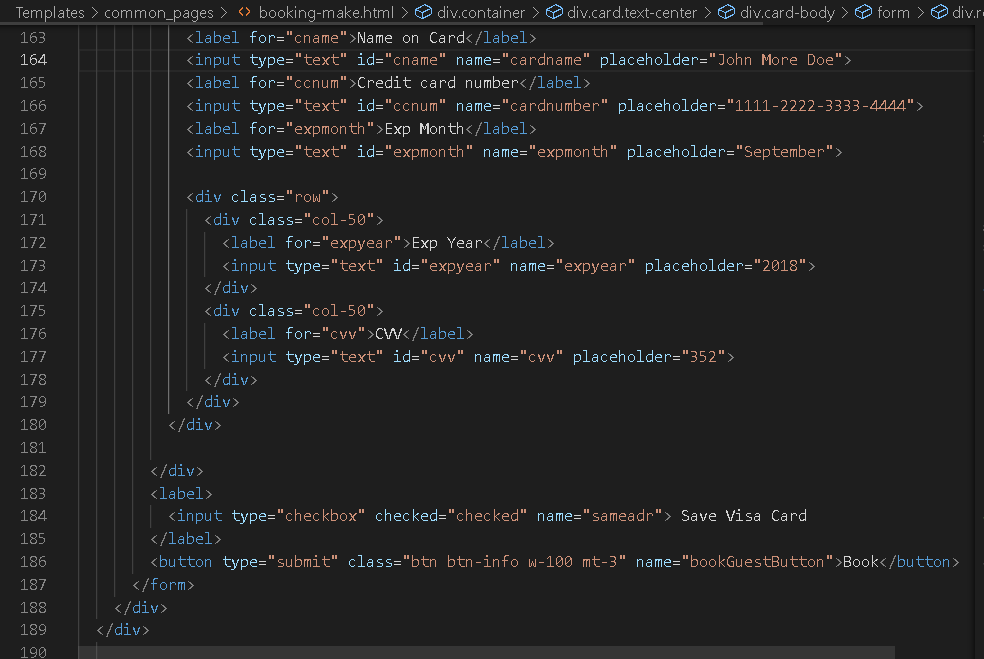
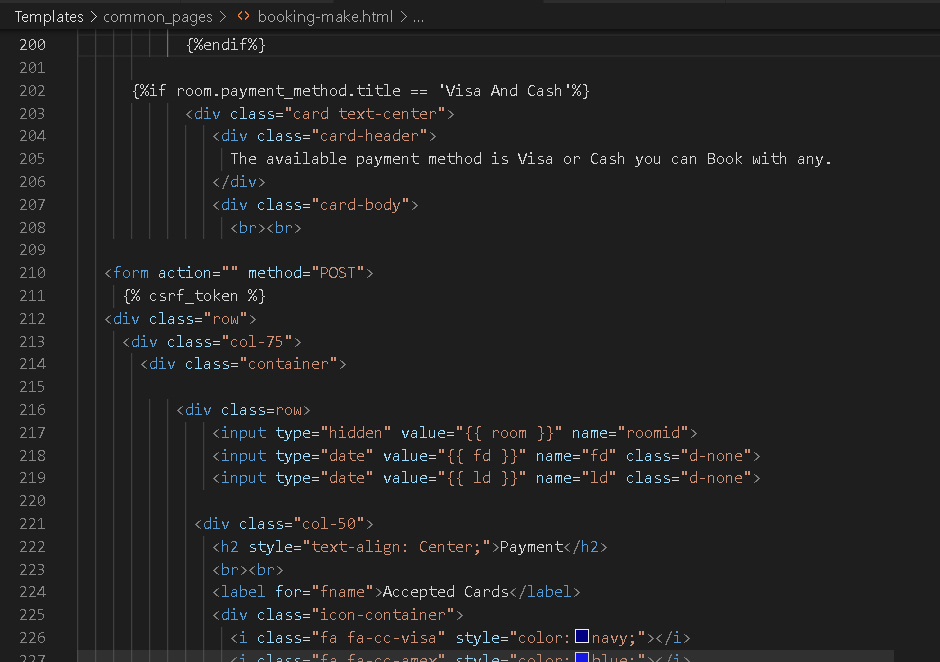
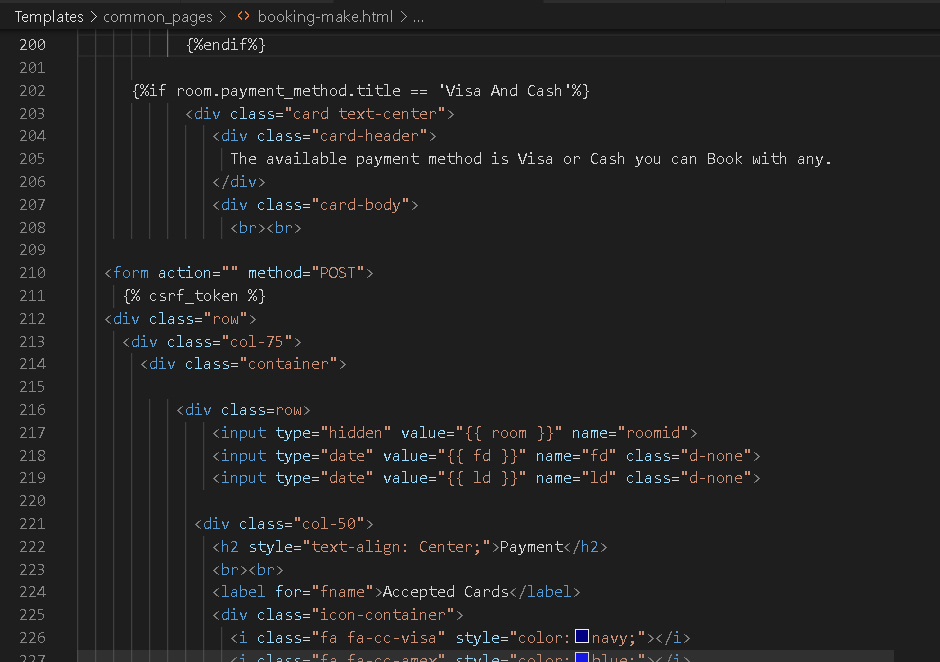
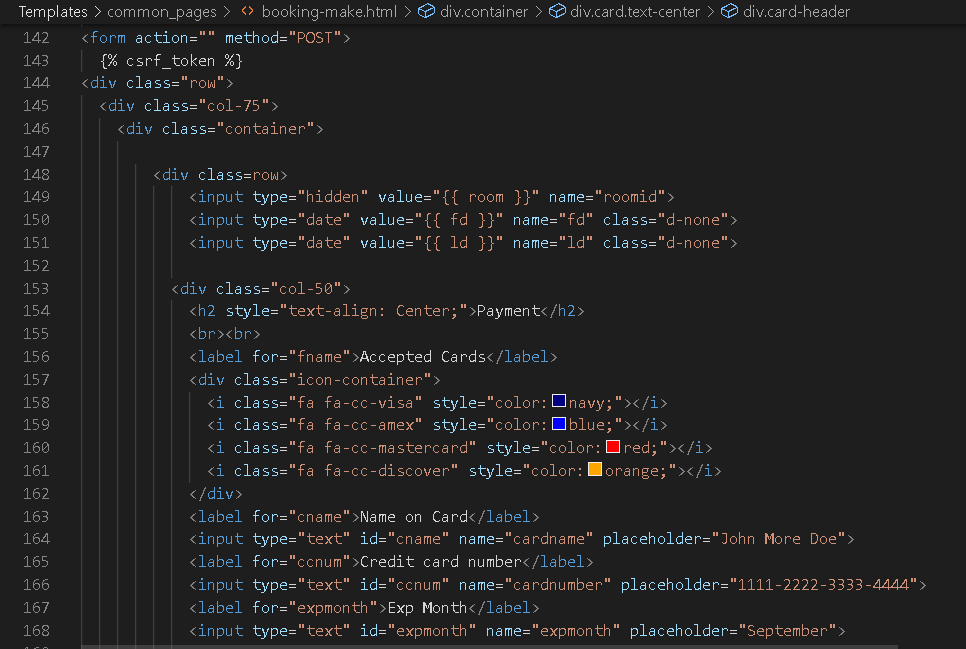
****

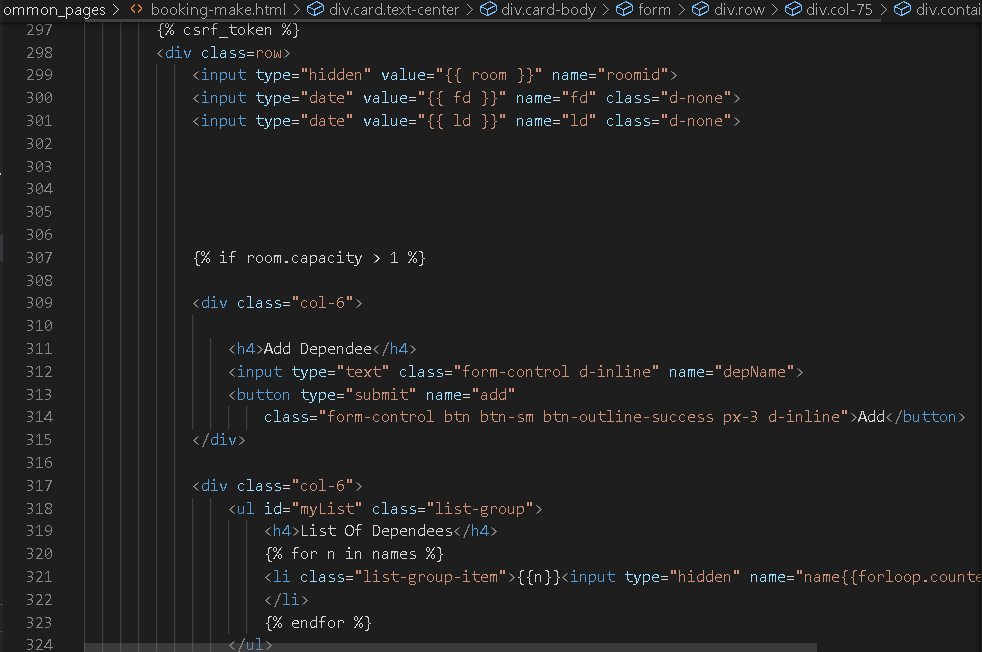
****

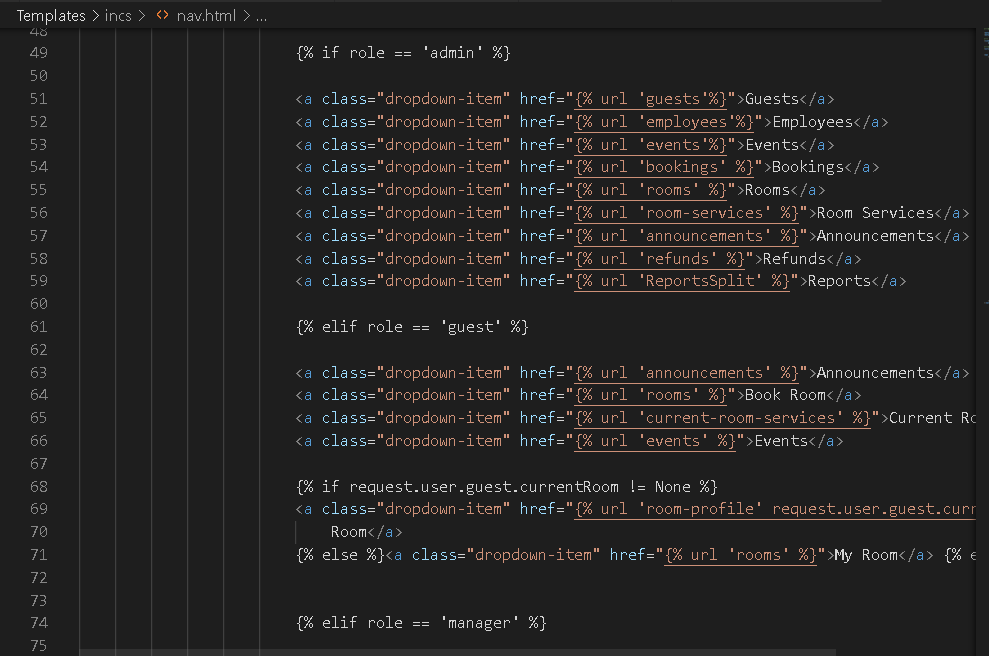
****

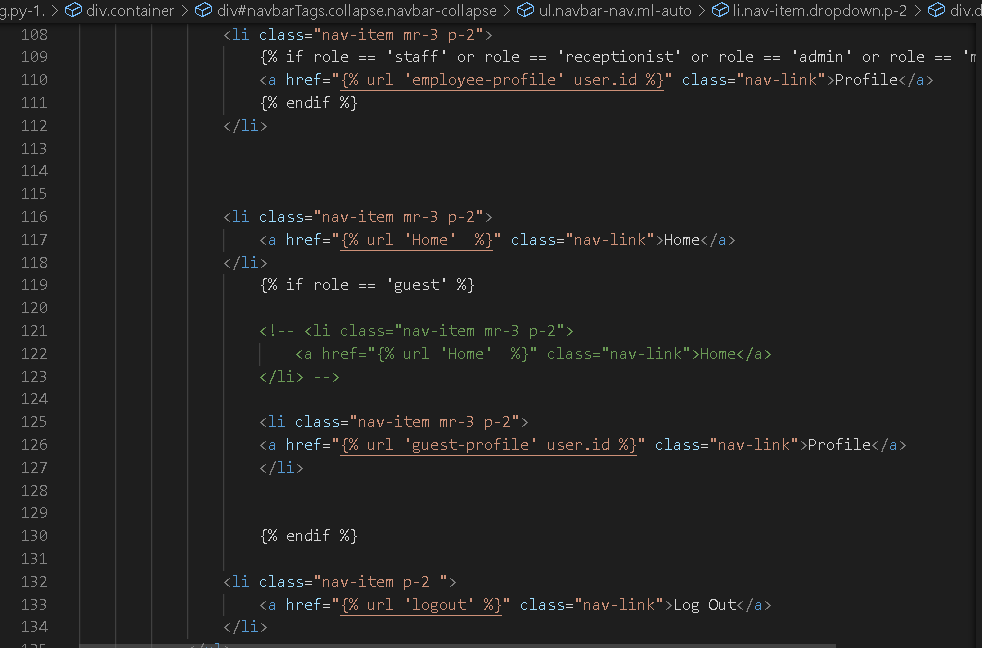
**Booking Template**

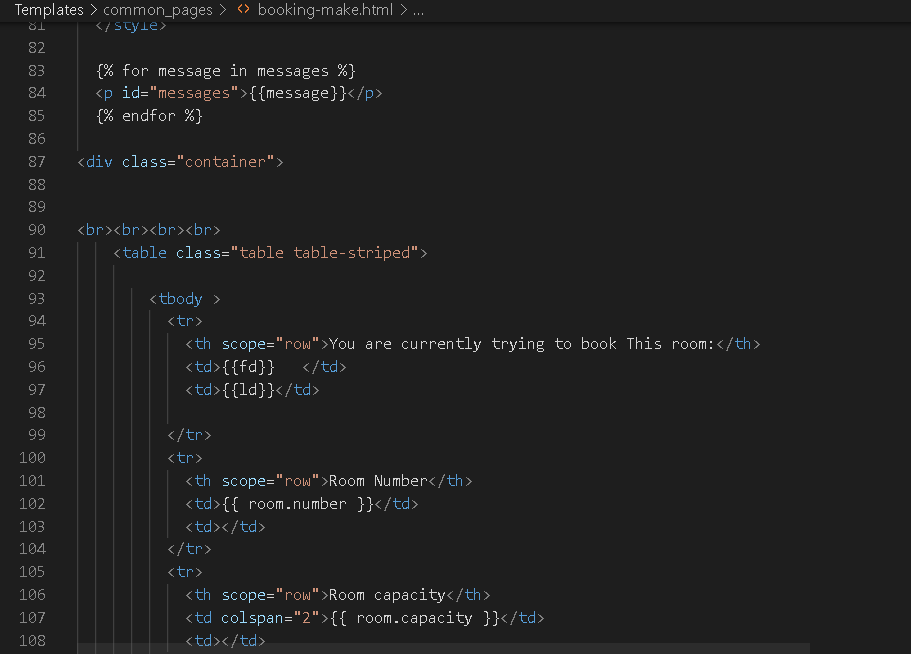
****

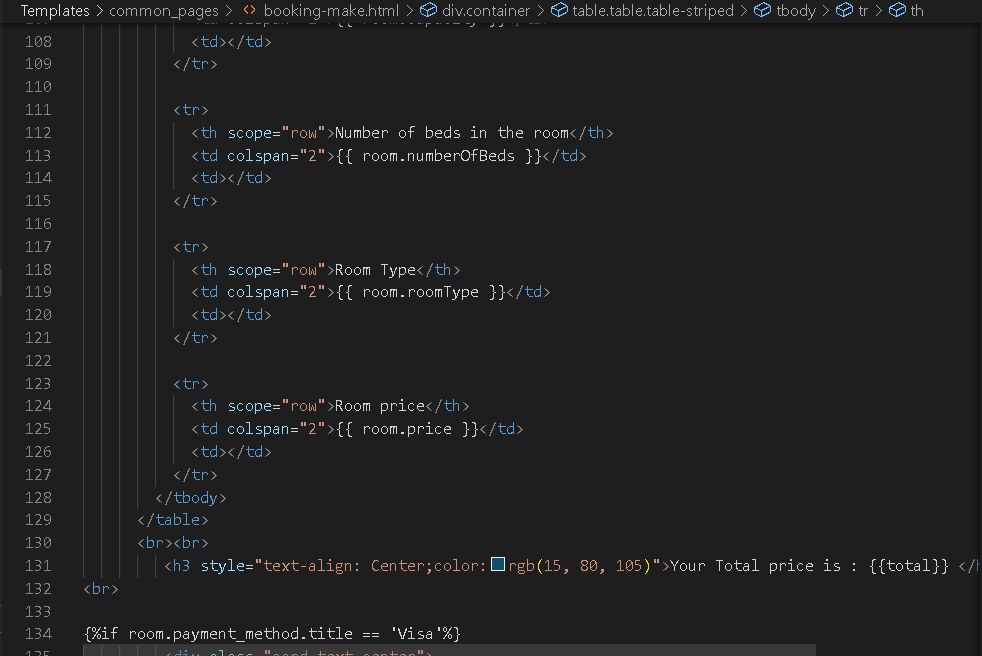
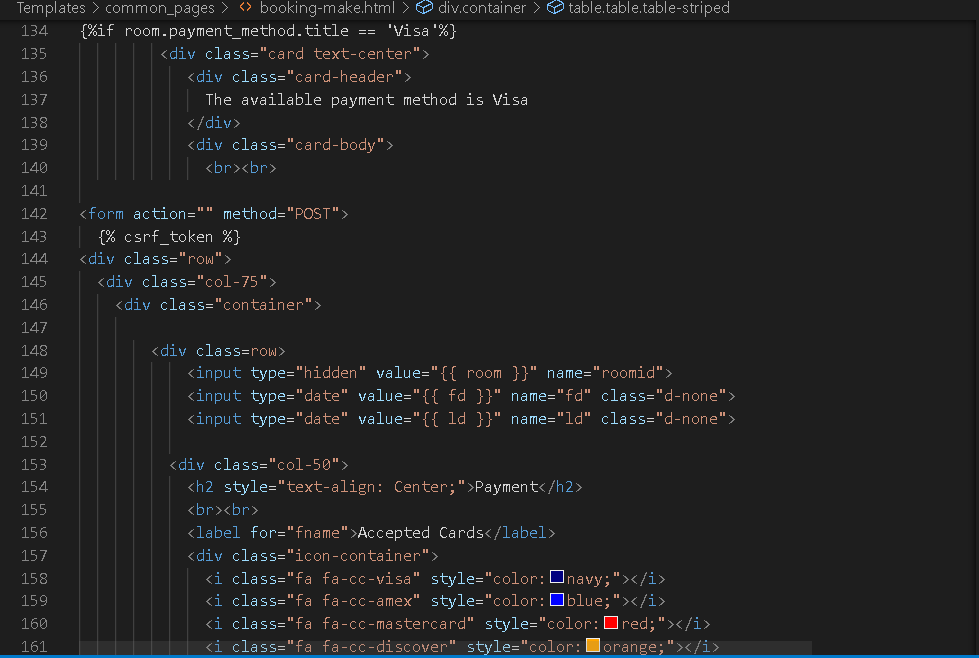
****

****

**Vav Bar Template **

****

**Booking make Template**

****

****

****