بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

In the Name of Allah, the Entirely Merciful, the Especially Merciful

Al-Fatihah [1: 1], Nobel Quran

A Hotel management system project

By

1. Hafiz Huzaifa Jabbar (225209)
2. Umair Raza (225192)

The requirements for the degree of

ASSOCIATE DEGREE

IN

INFORMATION TECHNOLOGY

UNIVERSITY COMMUNITY COLLEGE

Government College University Faisalabad

To,

The Controller of Examinations,

Government College University,

Faisalabad

I, the supervisor, certifies that the contents and form of the project submitted by

1. **Hafiz Huzaifa Jabbar (225209)**
2. **Umair Raza (225192)**

Have been found satisfactory for the award of the degree.

Supervisor:

Name: ____________

Signature: ________________

Lecturer: University Community College (GCUF)

Department of Information Technology

Supervisor:

Name: ________________

Signature: ________________

To,

The Project Coordinator,

Government College University,

Faisalabad

I, the project coordinator, certifies that the contents and form of the project submitted by

1. **Hafiz Huzaifa Jabbar (225209)**
2. **Umair Raza (225192)**

Have been found satisfactory for the award of the degree.

Project coordinator

Name: ____________

Signature: ____________

Lecturer: University Community College (GCUF)

Department of Information Technology

External Examiner:

Name: ________________

Signature: ________________

Dedication

On this occasion, we dedicate our hard work to our parents, family members and friends who have encouraged and supported us in various ways. We also extend our sincere gratitude to our beloved teachers, particularly Mr. Zia Qamar, Miss Samavia and Miss Bushra for making it possible for us to design this system. They played a vital role throughout the journey of making this system. Their expertise and guidance in the field of Information Technology, as well as their innovative ideas, have been instrumental in the successful completion of our project.

Acknowledgements:

Expressing our deepest gratitude to the **Almighty Allah**, the Most Merciful, and honoring the Holy Prophet **Muhammad (peace be upon him)** as the model and guiding light for all of humanity. We begin by offering our praises and appreciation to the Almighty Allah, who has bestowed us with good health and the capacity to achieve this goal.

We would like to express our sincere appreciation to our supervisor, **Mr. Zia Qamar**, the coordinator of the Department of Information Technology at the University Community College, GC University, Faisalabad. He has consistently guided us and provided moral support throughout every step of our degree program and project.

We would also like to extend our deepest gratitude to our respected lecturer, **Miss Bushra**, from the Department of Information Technology at the University Community College, GC University, Faisalabad. He has also been a valuable mentor, offering guidance and moral support at every stage of our degree program and project.

Furthermore, we would like to thank our lecturer, **Miss Samavia** from the Department of Information Technology, University Community College, GC University, Faisalabad, for her constant scholarly guidance, skillful advice, and unwavering dedication, all of which have been invaluable for the successful completion of this research work. Her cheerful attitude and warm encouragement have consistently motivated us to work hard and achieve our goal of obtaining our degree.

We are deeply grateful to **Dr. Naeem Iqbal**, the Principal of the University Community College, GC University, Faisalabad, for providing us with a conducive environment to successfully complete our project.

ABSTRACT

We are going to present the documentation of our Online Hotel Management System designed for the purpose of booking hotel rooms online, seeing the food menu and check their validation. My system will allow the hotel to manage rooms and food orders. It will allow the customers to browse different kind of rooms to stay in and see the food menu from which they can select the food of their choice. The system will forward booking and food orders to the hotel management team through the Graphical User Interface.

In the document, we have described the model of the system, functional and non-functional requirements, and a detailed description of the user interface and the backend structure of the system. For the front end, we have used **HTLM, CSS and JavaScript**. For the backend we have used **PHP** and **mySql** database to ensure reliability and accuracy.

Table of content:

List of Tables:

List of figures:

Chapter 1 – Software Requirement Specification

1.1 – Introduction

Technology is everywhere these days, changing how we live, work, and even travel. From the first computers to the internet, technology has completely changed businesses all over the world. Hotels are no different!

Hotels are all about making guests happy and keeping things running smoothly. But with guests wanting more and more from their hotel experience, hotels need to find ways to be even more efficient and organized. That's where computers come in.

This document is about a new website that will help hotels manage their business better. It's like having a super-smart assistant that can handle lots of tasks, making everything faster, more accurate, and more enjoyable for guests.

Imagine a website that can:

- **Book rooms:** Guests can easily book rooms online, anytime, anywhere.
- **Manage reservations:** The website keeps track of all bookings, so there's no confusion about who's staying where.
- **Handle payments:** Guests can pay for their stay securely online, making check-in a breeze.
- **Communicate with guests:** The website can send important information, like directions or reminders, directly to guests' phones.
- **Track expenses and finances:** Hotels can easily see how much money they're making and spending, making it easier to manage their budget.

Before we dive into the details of this website, it's important to remember how computers have become so important for businesses today. They've helped us do things faster, smarter, and more efficiently.

This website project is a great way for hotels to use technology to improve their operations, offer personalized services, and stay ahead of the competition. Now, let's learn a little about the history of computers and why they're so important in today's world before we explore how this website will change the way hotels work.

1.2 – Stakeholders

Here is a list of possible stakeholders of my system(If it goes live) :

1. **Hotel Owners/Management**: The primary stakeholders would be the hotel owners or management team who have a vested interest in the success of the hotel management project. They will provide the overall vision, objectives, and requirements for the project.

2. **Hotel Staff**: The hotel staff, including front desk personnel, concierge, housekeeping, restaurant and kitchen staff, and maintenance team, are crucial stakeholders. They will interact directly with the hotel management system and rely on it for their daily operations.

3. **Guests/Customers**: The guests or customers of the hotel are essential stakeholders. Their experience and satisfaction are key metrics for the success of the hotel management project. The system should facilitate smooth check-ins, check-outs, reservations, and provide a seamless experience during their stay.

4. **IT Department**: The IT department within the hotel or the organization supporting the project is a stakeholder. They will be responsible for implementing and maintaining the technology infrastructure, ensuring data security, and providing technical support for the hotel management system.

5. **Sales and Marketing Team**: The sales and marketing team will use the hotel management system to manage bookings, promotions, pricing, and analyze customer data. They will rely on the system for accurate reporting and insights to drive sales and marketing strategies.

6. **Finance and Accounting Department**: The finance and accounting team will have a stake in the project to ensure accurate financial reporting, revenue management, and integration with accounting systems. The system should provide robust financial features such as billing, invoicing, and reporting capabilities.

7. **Suppliers and Vendors**: Suppliers and vendors who provide goods and services to the hotel, such as food and beverage suppliers, laundry services, and maintenance contractors, are stakeholders. The system may include features to manage inventory, procurement, and vendor relationships.

8. **Regulatory Authorities**: Regulatory authorities and government agencies responsible for overseeing the hotel industry may also be stakeholders. Compliance with regulations, such as guest registration requirements and tax reporting, may need to be integrated into the hotel management system.

9. **External Partners**: Third-party partners, such as online travel agencies (OTAs), channel managers, payment gateways, and other service providers, may be stakeholders. Integration with these partners' systems may be necessary for seamless operations and distribution.

10. **Project Team**: Lastly, the project team members themselves, including developers, designers, testers, and project managers, are stakeholders. Their successful completion of the project and meeting stakeholder expectations is crucial for the overall success of the hotel management system.

1.3 – Users

Here is the list of users of the system if it gets used:

Hotel Staff:

- **Front Desk Staff:** Handles guest check-in/out, reservations, room assignments, payment processing, guest inquiries, and managing guest requests.
- **Housekeeping Staff:** Manages room cleaning schedules, tracks room status (clean, dirty, occupied), communicates with front desk about room availability, and handles guest requests for amenities.
- **Restaurant & Bar Staff:** Takes orders, manages reservations, tracks inventory, communicates with the kitchen, and processes payments.
- **Maintenance Staff:** Receives and manages maintenance requests, tracks work orders, communicates with guests about repairs, and manages inventory of spare parts.

- **Management Staff:** (General Manager, Operations Manager, Sales & Marketing Manager, Finance Manager, HR Manager) Accesses reports, analyzes data, makes decisions, and oversees operations.

Guests:

- **Guests:** Book rooms, view room availability, manage reservations, make payments, access hotel information, request services, and provide feedback.

Other Users:

- **Travel Agents:** Access hotel information, book rooms for clients, manage reservations, and receive commission reports.
- **Suppliers:** Communicate with hotel staff, submit invoices, track payments, and manage inventory

1.4 – Functional requirements

Functional requirements outline the specific actions and tasks that the website must perform to fulfill its purpose. They are derived from the user needs and expectations identified in the previous section. The functional requirements for the hotel management website can be categorized as follows:

1.4.1 Customer-Facing Functionality:

- **Online Booking:** The website must allow customers to search for available rooms, filter results by criteria such as dates, price range, room type, and amenities, view detailed room information, and complete bookings securely online.
- **Room Information:** The website must provide comprehensive information about each room, including descriptions, images, amenities, pricing, and availability.
- **Payment Gateway Integration:** The website must integrate with a secure third-party payment gateway to facilitate online payments for bookings.
- **User Account Management:** The website must allow customers to create and manage user accounts, view booking history, update personal information, and manage their preferences.
- **Food Menu:** The website must display the hotel's restaurant menu, including dishes, descriptions, and prices.
- **Local Information:** The website must provide access to local information, such as attractions, transportation, and nearby amenities.

1.4.2 Staff-Facing Functionality:

- **Room Management:** The website must allow staff to manage room availability, set rates, update room information, and assign rooms to guests.
- **Booking Management:** The website must allow staff to view and manage bookings, check-in and check-out guests, handle cancellations, and manage guest requests.
- **Guest Management:** The website must provide staff with access to guest information, including contact details, booking history, and special requests.
- **Financial Management:** The website must provide tools for staff to track revenue, expenses, and generate reports for financial analysis.
- **Inventory Management:** The website must allow staff to manage hotel supplies, amenities, and services.
- **Staff Communication:** The website must facilitate communication between staff members, including messaging and task management.

1.4.3 Administration Module:

- **User Management:** The website must allow administrators to create, manage, and modify user accounts for both customers and staff.
- **System Configuration:** The website must allow administrators to configure website settings, including language, currency, and payment gateway integration.
- **Content Management:** The website must allow administrators to update website content, such as hotel descriptions, images, and promotional materials.
- **Security Management:** The website must provide administrators with tools to manage user permissions, control access to different modules, and monitor system security.

1.4.4 - Description of each requirement in Requirement Shell

Requirement Name: Manage Users

Requirement #:	Req01	**Requirement Type:**	Functional
Description:	This requirement enables authorized users to manage user accounts within the system. Administrators can add new users, delete existing users, and update user information.		

Rationale:	o effectively manage user access and permissions within the hotel management system.		
Originator:	Administrator of the system		
Fit Criterion:	• User is successfully added to the system. • Database is updated to reflect the new user. • User is successfully deleted from the system. • Database is updated to reflect the deleted user.		
Customer Satisfaction:	5	**Customer Dissatisfaction:**	10
Priority:	3	**Conflicts:**	Nil
Supporting Materials:	NA		
History:			

1.5 – Non functional Requirements

Here is a list of non functional requirements of my system:

Performance:

- **Speed:** The system should be fast and responsive, so users don't have to wait too long for things to load. Imagine a guest trying to book a room - they shouldn't have to wait forever!

- **Reliability:** The system should be reliable and work consistently, so users can trust it to do its job. We don't want the system crashing when someone is trying to check in or make a reservation!

- **Scalability:** The system should be able to handle a lot of users and data, even as the hotel grows. We want to be ready for busy seasons and more guests!

Security:

- **Data Protection:** The system should keep user information safe and secure, so guests' personal details are protected. We need to make sure no one can steal their credit card information or other sensitive data!

- **Access Control:** Only authorized users should be able to access certain parts of the system. We don't want anyone snooping around where they shouldn't be!

Usability:

- **Easy to Use:** The system should be easy to understand and use, even for people who aren't tech-savvy. We want everyone to be able to use it without getting frustrated!

- **User-Friendly Interface:** The system should have a clear and intuitive interface, so users can find what they need quickly. Think about how easy it is to navigate a website - we want the system to be just as easy!

Other Important Requirements:

- **Accessibility:** The system should be accessible to people with disabilities. We want everyone to be able to use the system, regardless of their abilities!

- **Maintainability:** The system should be easy to update and maintain, so we can keep it running smoothly for years to come. We don't want to spend a lot of time fixing problems or updating the system!

1.6 - Schedule of Project

The project is divided into 4 phases with the given timeframe for each phase. These are as follow:

Phase 1: Requirements Gathering and Analysis (1 week)

- Define project scope and objectives
- Identify stakeholders and their needs
- Create detailed functional and non-functional requirements
- Develop a system architecture and design document

Phase 2: Design and Development (2 weeks)

- Design the user interface (UI) and user experience (UX)
- Develop the database schema
- Develop the backend logic and functionality
- Create unit tests and integration tests
- Conduct code reviews and ensure code quality

Phase 3: Testing and Deployment (1 week)

- Conduct system testing (functional, integration, performance, security)
- Fix bugs and address issues identified during testing
- Develop user documentation and training materials
- Prepare the system for deployment
- Deploy the system to the production environment

Phase 4: Maintenance and Support (Still ongoing and being updated)

- Monitor system performance and address any issues
- Implement new features and updates
- Provide ongoing user support and training
- Gather feedback from users and make improvements

CHAPTER-2 – ANALYSIS

2.1 – Use case

A use case is a powerful tool for understanding how users interact with a system to achieve a specific goal. It's essentially a story that outlines the steps a user takes and the system's responses along the way. Imagine it as a blueprint for how a user will use your system, from start to finish.

The beauty of use cases lies in their user-centric focus. They prioritize what users want to accomplish, rather than the system's internal workings. By describing a complete sequence of actions, they paint a clear picture of how a user interacts with the system to achieve their goal. This makes them an invaluable tool for communication between developers, designers, and stakeholders, ensuring everyone is on the same page about the system's functionality.

Use cases are crucial for several reasons. They help you deeply understand user needs, ensuring you build the right features to meet their expectations. They also define the system's functionality by outlining the tasks users can perform and the system's responses. This process helps ensure that the system is built with the user in mind. Furthermore, use cases serve as a basis for testing the system, ensuring that it meets user requirements and works as intended.

A typical use case includes key components like the actor (the entity interacting with the system), the goal (the objective the actor wants to achieve), pre-conditions (conditions that must be true before the use case begins), post-conditions (conditions that are true after the use case is completed), and the flow of events (a step-by-step description of the interaction). By meticulously outlining these components, you gain a comprehensive understanding of how users will interact with your system.

In essence, use cases are essential for building successful systems. They help you understand user needs, define functionality, facilitate communication, and ensure that your system is tested and validated to meet those needs.

2.1.1 – Use case table

This table presents the use case model for the booking phase. The use case model is a key component of the analysis phase, as it helps us understand the system's functionality from the perspective of its users. It identifies the different actions that users can perform within the system and the interactions between the system and its users.

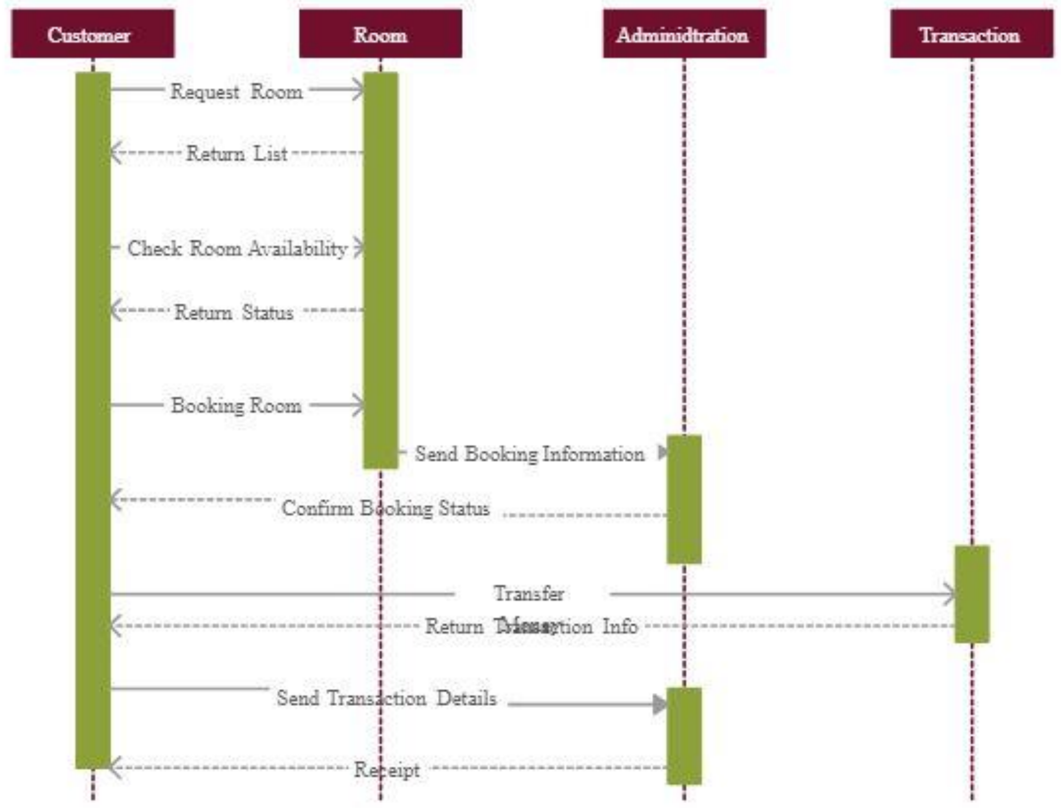
2.1.2

Use Case Name: Booking a room

UC #	01	**Ref:**	Req. #
UC Name	Booking a room		
Level	Abstract		
Description	This use case describes the process a guest follows to book a room at the hotel.		
Actor (s)	Guest		
Stakeholders	• **Internal:** Hotel Management, Front Desk Staff • **External**: Guest, Travel Agent (if applicable)		
Preconditions	• The guest has an account with the hotel or is creating a new account. • The hotel has available rooms for the requested dates. • The system is operational.		

Success Guarantee	The guest receives a confirmation screen.	
Main Success Scenario	**Action**	**Response**
	1. Guest selects "Book Room" on the hotel website. | The system displays a list of available rooms.	1.The system displays a list of available rooms.
	2. Guest enters their booking details	**2.** The system generates a reservation confirmation and sends it to the guest
Special requirements		
Frequency of Occurrence	High	

1.2 System Sequence Diagram (SSD) of booking use case



2.3 – Domain model

Here is the domain model for my hotel management system, focusing on the core entities and their relationships:

Entities:

- **Hotel:** Represents a specific hotel entity.
- **Room:** Represents a room within a hotel, with attributes like room type, size, amenities, and availability.

- **Guest:** Represents a guest of the hotel, with attributes like name, contact information, and reservation history.
- **Reservation:** Represents a booking made by a guest, linking the guest to a specific room for a defined period.
- **Staff:** Represents an employee of the hotel, with attributes like name, role, and access permissions.
- **Service:** Represents a service offered by the hotel, such as room service, housekeeping, or maintenance.
- **Payment:** Represents a financial transaction made by a guest, associated with a reservation.

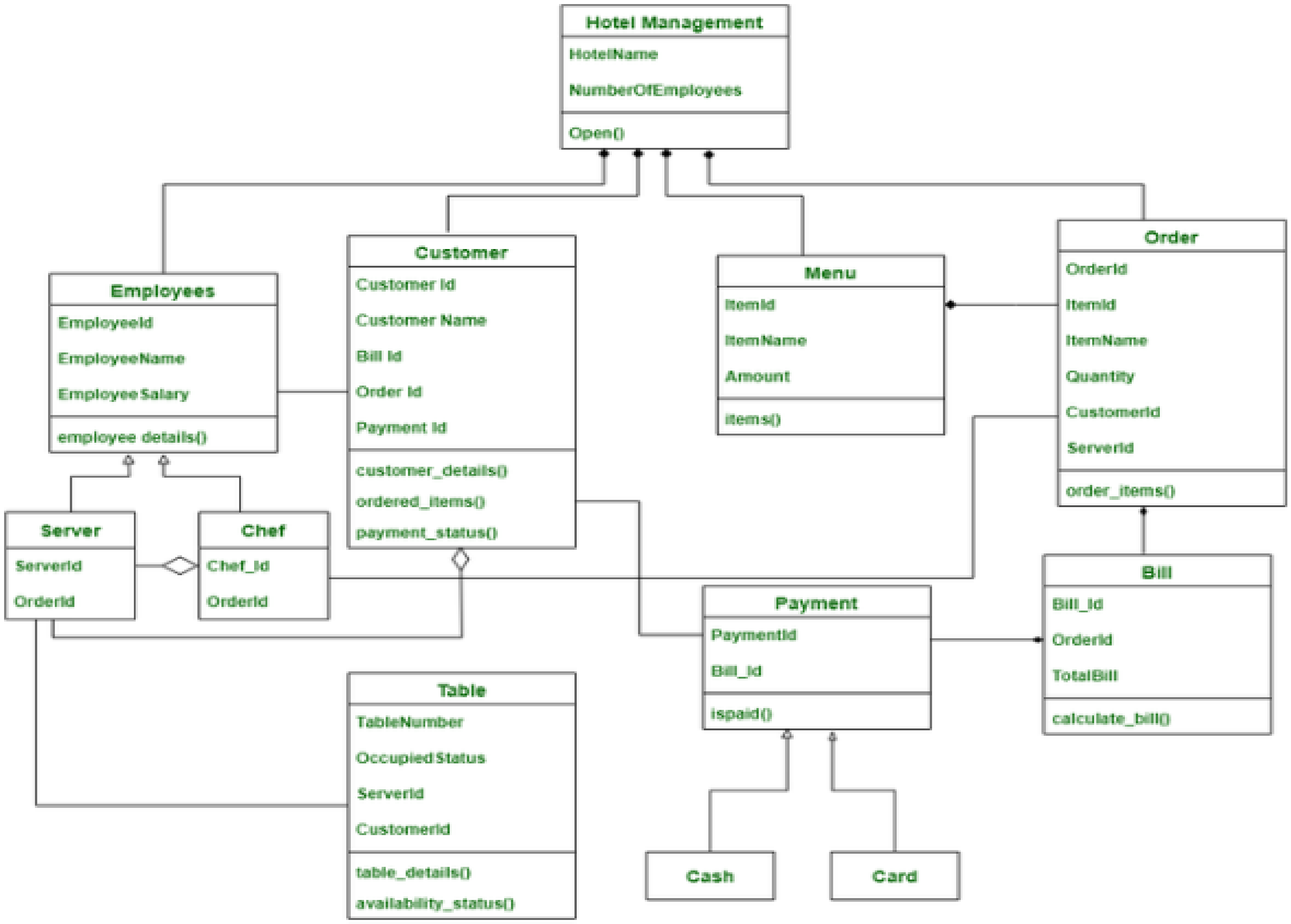
Relationships:

- **Hotel has many Rooms:** A hotel can have multiple rooms.
- **Room belongs to a Hotel:** Each room is associated with a specific hotel.
- **Guest makes Reservations:** A guest can make multiple reservations.
- **Reservation is for a Room:** Each reservation is associated with a specific room.
- **Reservation is made by a Guest:** Each reservation is linked to a specific guest.
- **Staff works at a Hotel:** A staff member can work at a specific hotel.
- **Staff can manage Services:** Staff members can manage the provision of services to guests.
- **Guest pays for Reservations:** A guest can make payments for their reservations.
- **Payment is for a Reservation:** Each payment is associated with a specific reservation.

Attributes:

- **Hotel:** Name, Address, Phone Number, Email, Website
- **Room:** Room Number, Room Type, Size, Amenities, Price, Availability
- **Guest:** Name, Address, Phone Number, Email, ID Number, Payment Information, Reservation History
- **Reservation:** Reservation ID, Check-in Date, Check-out Date, Number of Guests, Room Type, Total Cost, Status (Pending, Confirmed, Cancelled)
- **Staff:** Name, Role, Username, Password, Access Permissions
- **Service:** Service Name, Description, Price
- **Payment:** Payment ID, Amount, Payment Method, Date

Diagram:


Hotel Management
HotelName
NumberOfEmployees
Open()
Employees
EmployeeId
EmployeeName
Employee Salary
employee details()
Customer
Customer Id
Customer Name
Bill Id
Order Id
Payment Id
customer_details()
ordered_items()
payment_status()
Menu
ItemId
ItemName
Amount
items()
Order
OrderId
ItemId
ItemName
Quantity
CustomerId
ServerId
order_items()
Server
ServerId
OrderId
Chef
Chef_Id
OrderId
Bill
Bill_Id
OrderId
TotalBill
calculate_bill()
Payment
PaymentId
Bill_Id
ispaid()
Table
TableNumber
Occupied Status
ServerId
CustomerId
table_details()
availability_status()
Cash
Card

Chapter 3 – Design

3.1 Design Techniques

Front-End Design (HTML, CSS, JavaScript):

- **Responsive Web Design (RWD):** Ensure your website adapts seamlessly to different screen sizes (desktops, tablets, mobile phones). Use CSS media queries and flexible layouts to achieve this.
- **User Interface (UI) Design:**
  - **Wireframing:** Create low-fidelity sketches to map out the layout and structure of each page.
  - **Mockups:** Develop high-fidelity prototypes that show the visual design and interactions of the website. Consider using tools like Figma or Adobe XD.
  - **Visual Hierarchy:** Use typography, color, and spacing to guide the user's eye and emphasize important elements.
  - **User-Friendly Navigation:** Make it easy for users to find what they need with clear menus, breadcrumbs, and search functionality.
  - **Forms:** Design forms for booking, check-in, and other interactions that are easy to fill out and submit.
  - **Accessibility:** Ensure your website is accessible to users with disabilities by following accessibility guidelines (WCAG).
- **JavaScript for Interactivity:**
  - **Dynamic Content:** Use JavaScript to load and update content dynamically, such as room availability, menu items, and user profiles.
  - **Form Validation:** Implement JavaScript to validate user input in forms, ensuring data integrity.
  - **AJAX Requests:** Use AJAX to send and receive data from the server without reloading the entire page, creating a smoother user experience.
  - **User Experience (UX) Enhancements:** Add features like animations, transitions, and interactive elements to improve the user experience.

Back-End Design (PHP, MySQL):

- **Database Design:**
    - **Entity-Relationship Diagram (ERD):** Create a visual representation of the relationships between different entities in your database (e.g., rooms, guests, bookings, staff, menus).
    - **Normalization:** Organize your database tables to reduce data redundancy and improve data integrity.
    - **Data Types:** Choose appropriate data types for each field (e.g., text, numeric, date).
- **PHP for Server-Side Logic:**
    - **Object-Oriented Programming (OOP):** Use OOP principles to structure your code, making it more modular and maintainable.
    - **Database Interaction:** Use PHP to connect to your MySQL database, retrieve data, and update information.
    - **Session Management:** Implement session management to track user logins and maintain user data.
    - **Security:** Implement security measures to protect your website and user data from attacks (e.g., input validation, SQL injection prevention, password hashing).
- **API Design:**
    - **RESTful APIs:** Consider using RESTful APIs to handle interactions between the front-end and back-end, making your system more scalable and flexible.

Additional Considerations:

- **Security:** Implement robust security measures throughout your website, including user authentication, data encryption, and input validation.
- **Performance:** Optimize your code and database queries for speed and efficiency. Use caching techniques to reduce server load.
- **Scalability:** Design your system to be scalable, so it can handle increasing traffic and data volumes in the future.
- **Testing:** Thoroughly test your website to ensure it functions correctly and meets your requirements.

3.1.1 Approach

Our solution to enhance the hotel's efficiency and guest experience is a comprehensive hotel management system, delivered through a user-friendly website. This website will streamline operations, improve communication, and provide guests with a seamless online experience. We

prioritize a simple and intuitive design because both staff and guests will need to quickly and easily navigate the system.

To develop this system, we will first analyze the hotel's current processes, documenting them in use cases. These use cases will guide the creation of a system specification, outlining the required functionalities and technical specifications. We will also analyze the different user groups, detailing their functional and non-functional requirements for the system, including both staff and guests.

With the specifications and requirements in place, we will create an information model using an E-R diagram to represent the entities and their relationships. This E-R diagram will be visualized in Visio and will serve as the foundation for the relational schema used in the final database design.

Next, we will create GUI prototypes using screen mockups in Photoshop to visualize the website's interface for both staff and guests. To support implementation, we will develop a list of functions, describing their actions and accompanying SQL statements.

During implementation, test scripts will be created and executed by a team to identify and address any implementation errors. The project will be considered complete once all implementation errors are resolved. Throughout the process, project planning and workload distribution will be essential for efficient progress.

3.1.2 Technical Approach

The technical approach for developing the hotel management system website will leverage a combination of front-end and back-end technologies to create a robust and user-friendly system.

On the front-end, we will utilize the standard web technologies of HTML, CSS, and JavaScript to build the user interface and interactive elements. This foundation will provide the structure, styling, and dynamic behavior for the website.

For the back-end, we will utilize PHP, a widely used server-side scripting language, to handle the business logic, data processing, and API interactions. PHP offers a mature ecosystem, a large community, and a wide range of frameworks that can be leveraged for this project.

For the database, we will use MySQL, a widely used open-source relational database management system known for its reliability and performance. MySQL will serve as the core data storage solution, managing all hotel information, customer data, bookings, and other relevant data.

The system architecture will follow a three-tier approach, separating the presentation layer (front-end), business logic layer (back-end), and data layer (database). Communication between the client and server will be facilitated through RESTful APIs. We will explore the use of a microservices architecture to break down the system into smaller, independent services, enhancing scalability and maintainability. Caching mechanisms will be implemented to improve website performance by storing frequently accessed data in memory.

The development process will follow an agile methodology, such as Scrum or Kanban, enabling iterative development, continuous feedback, and flexibility. Version control using Git will be employed to manage code changes, track revisions, and facilitate collaboration. Continuous Integration and Continuous Deployment (CI/CD) pipelines will be implemented to automate the build, test, and deployment processes, ensuring faster delivery and improved quality.

Security is paramount, and we will implement robust authentication and authorization mechanisms to control user access and protect sensitive data. Data encryption will be employed during transmission and storage to safeguard sensitive information. Regular security audits will be conducted to identify and address potential vulnerabilities.

Testing and quality assurance will be comprehensive, including unit testing to verify individual code components, integration testing to ensure the interaction between different system components, and user acceptance testing (UAT) with hotel staff and potential customers to validate system functionality and usability. Performance testing will be conducted to evaluate the website's performance under different traffic conditions, and security audits and penetration testing will be performed to identify and address potential vulnerabilities.

Finally, the website will be deployed to a cloud platform for scalability, reliability, and ease of maintenance. Monitoring and logging tools will be implemented to track system performance, identify issues, and gather user insights. Regular updates and security patches will be provided to address vulnerabilities and improve system performance.

By adhering to this technical approach, the hotel management system website will be developed efficiently and effectively, delivering a robust, secure, and user-friendly system that meets the hotel's needs and enhances the guest experience.

3.1.3 Needed functionalities for users

Here are some functionalities that guests using an online hotel management system would likely need:

Core Hotel Services:

- **Booking:**
    - **Search Availability:** Search for available rooms based on dates, room type, and desired amenities.
    - **View Room Details:** See detailed information about each room type, including photos, descriptions, and amenities.
    - **Select Room and Dates:** Choose the desired room and check-in/check-out dates.
    - **Guest Information:** Provide personal and contact information, including guest names and payment details.
    - **Confirm Booking:** Receive a confirmation email with booking details and payment confirmation.
- **Check-In/Check-Out:**
    - **Online Check-In:** Complete check-in formalities online, including providing identification and credit card information.
    - **Express Check-Out:** Utilize a streamlined check-out process, potentially allowing for online payment and room key return.

- **Room Management:**
    - **Room Service:** Order food and beverages directly to their room.
        - **View Food Menu:** Access the hotel's restaurant menu, including descriptions, prices, and potentially images of dishes.
        - **Add Items to Cart:** Select items from the menu and add them to a virtual shopping cart.
        - **Modify Cart:** Adjust quantities, remove items, or add special instructions for each item in the cart.
        - **View Cart Total:** See a summary of the order, including itemized costs, taxes, and any delivery fees.
        - **Checkout:** Proceed to checkout, providing contact information, payment details, and delivery or pickup options.
        - **Order Tracking:** Receive updates on the status of their order (e.g., order confirmation, preparation, delivery in progress).

Guest Communications:

- **Messaging:** Communicate with hotel staff through an in-app messaging system for inquiries or requests.
- **Notifications:** Receive notifications about important updates, such as check-in reminders, room service delivery, or special offers.

- **Account Management:**
    - **Profile Management:** Update personal information, payment details, and preferences.

Hotel Information:

    - **Hotel Policies:** Access hotel policies regarding check-in/check-out times, cancellation policies, and other relevant information.

    - **Local Attractions:** View information about nearby attractions, restaurants, and activities.

- **Feedback:**

    - **Guest Surveys:** Provide feedback on their stay through online surveys.

    - **Ratings and Reviews:** View and contribute to hotel ratings and reviews.

Mobile Optimization:

- **Responsive Design:** Ensure the website is accessible and easy to use on various mobile devices.

By providing these functionalities, the online hotel management system can offer guests a convenient and user-friendly experience, enhancing their satisfaction and loyalty.

3.1.4 Menu Management System

The menu management system will be available only to restaurant employees and will, as the name suggests, allow them to manage the menu that is displayed to users of the web ordering system. The functions afforded by the menu management system provide user with the ability to, using a graphical interface:

- Add a new/update/delete vendor to/from the menu.

- Add a new/update/delete food category to/from the menu.
- Add a new/update/delete food item to/from the menu.
- Add a new/update/delete option for a given food item.
- Update price for a given food item.
- Update default options for a given food item.
- Update additional information (description, photo, etc.) for a given food item.

It is anticipated that the functionality provided by this component will be one of the first things noted by the restaurant user, as they will have to go through it to configure their menu, etc. before beginning to actually take orders. Once everything is initially configured, however, this component will likely be the least used, as menu updates generally do not occur with great frequency.

User interaction with the menu management system is similar to that with the web ordering system. Users navigate a tree structure to find the vendor, category, or specific food item that they would like to modify and after making their selection they are presented with a form which displays all of the current fields and values associated with that item, all of which can be modified or removed. The form also presents buttons which allow the addition of new fields and values. Unlike the web ordering system, however, most of the input here will be freeform, specifically in the form of text boxes, since there is no finite set of fields which could be added. This does not raise a major concern though, as input sanitation will be performed, and the user, who is assumed to be a restaurant employee, is less likely to be malicious than a web user.

3.1.5 Order Retrieval System

Order Retrieval System of the three components, the order retrieval system is functionally the simplest. Like the menu management system, it is designed to be used only by restaurant employees, and provides the following functions:

- Retrieve new orders from the database.

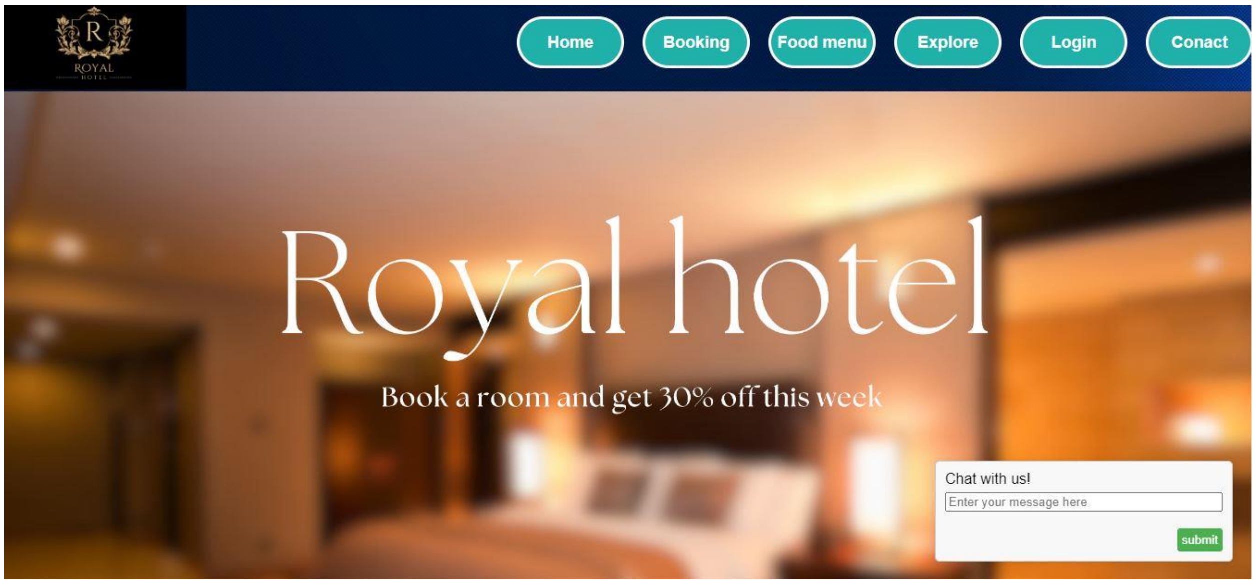
- Display the orders in an easily readable, graphical way.
- Mark an order as having been processed and remove it from the list of active orders.

User interaction with the order retrieval will be very simple. The application will automatically fetch new orders from the database at regular intervals and display the order numbers, along with delivery time, in a panel on the left hand side of the application. To view the details of an order, the user must simply click on that order number, which will populate the right-hand panel with the details, displayed in an easy to read and navigate tree structure. This structure can intuitively be expanded and collapsed to display only the desired information. Finally, once and order is processed, the user clicks a single button, labeled “Processed”, to remove it from the list of active orders.

1.2– User Interface

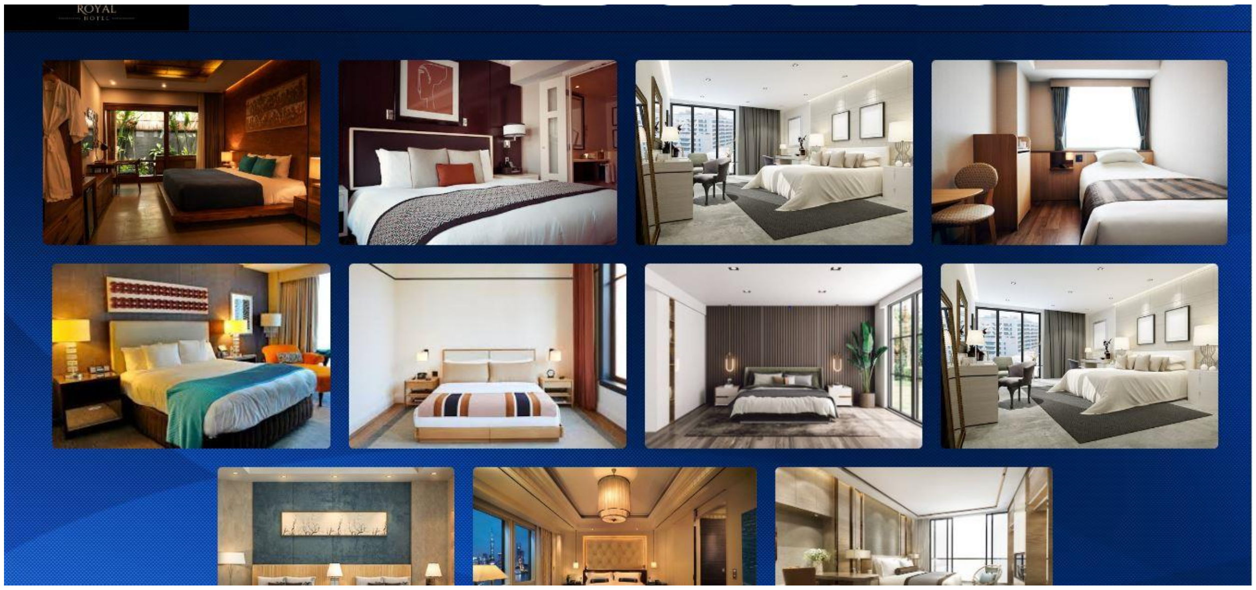
3.2.1 - Home page

This is the home page of the system. It includes a header with the logo and 6 different available options. The user can also send a message by using the message box at the bottom right corner of the page.

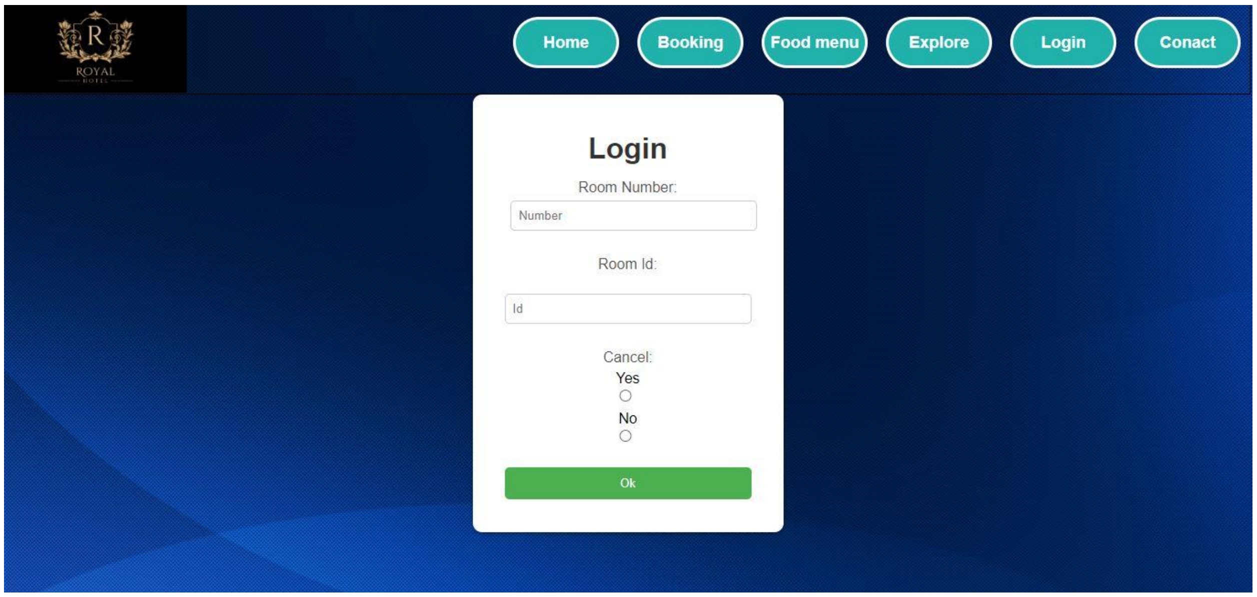


3.2.2 - Explore page

In this page the user can explore through the rooms available and chose the perfect room according to their need.



3.2.3 – Login page



Login

Room Number:

Number

Room Id:

Id

Cancel:

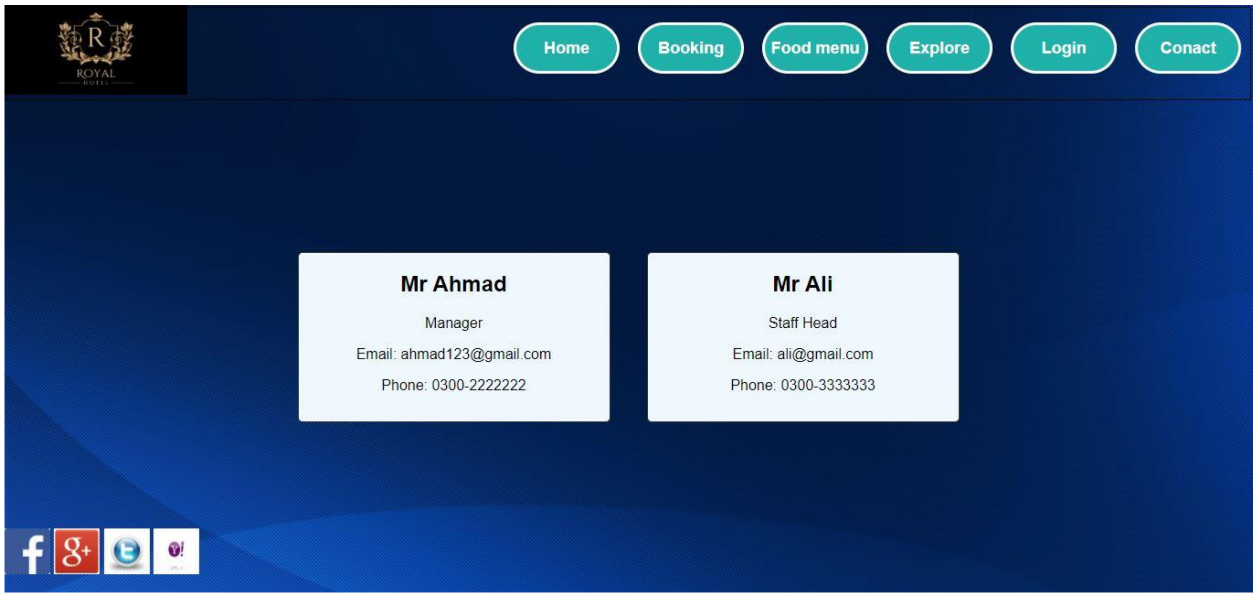
Yes

No

Ok

3.2.4 - Contact page

This page contains the contact information of the two employees of the hotel and social media links.



Mr Ahmad

Manager

Email: ahmad123@gmail.com

Phone: 0300-2222222

Mr Ali

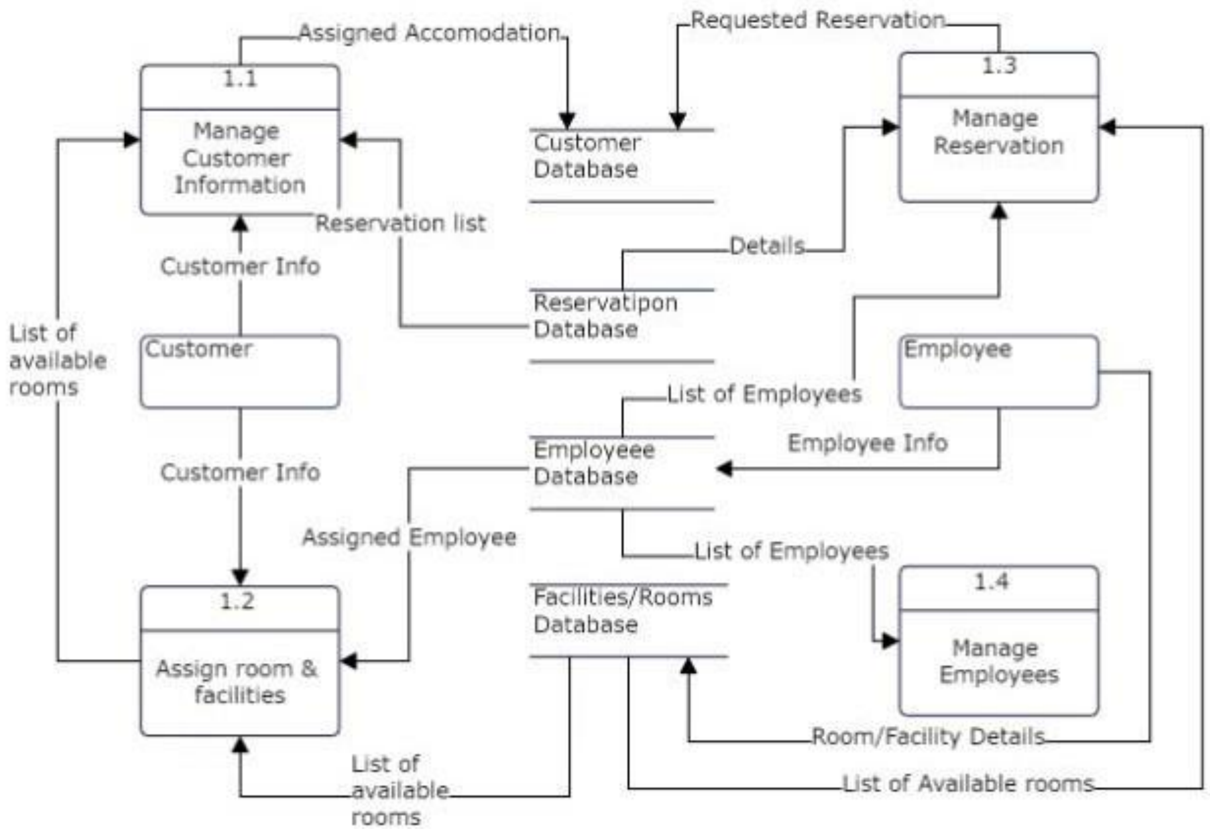
Staff Head

Email: ali@gmail.com

Phone: 0300-3333333

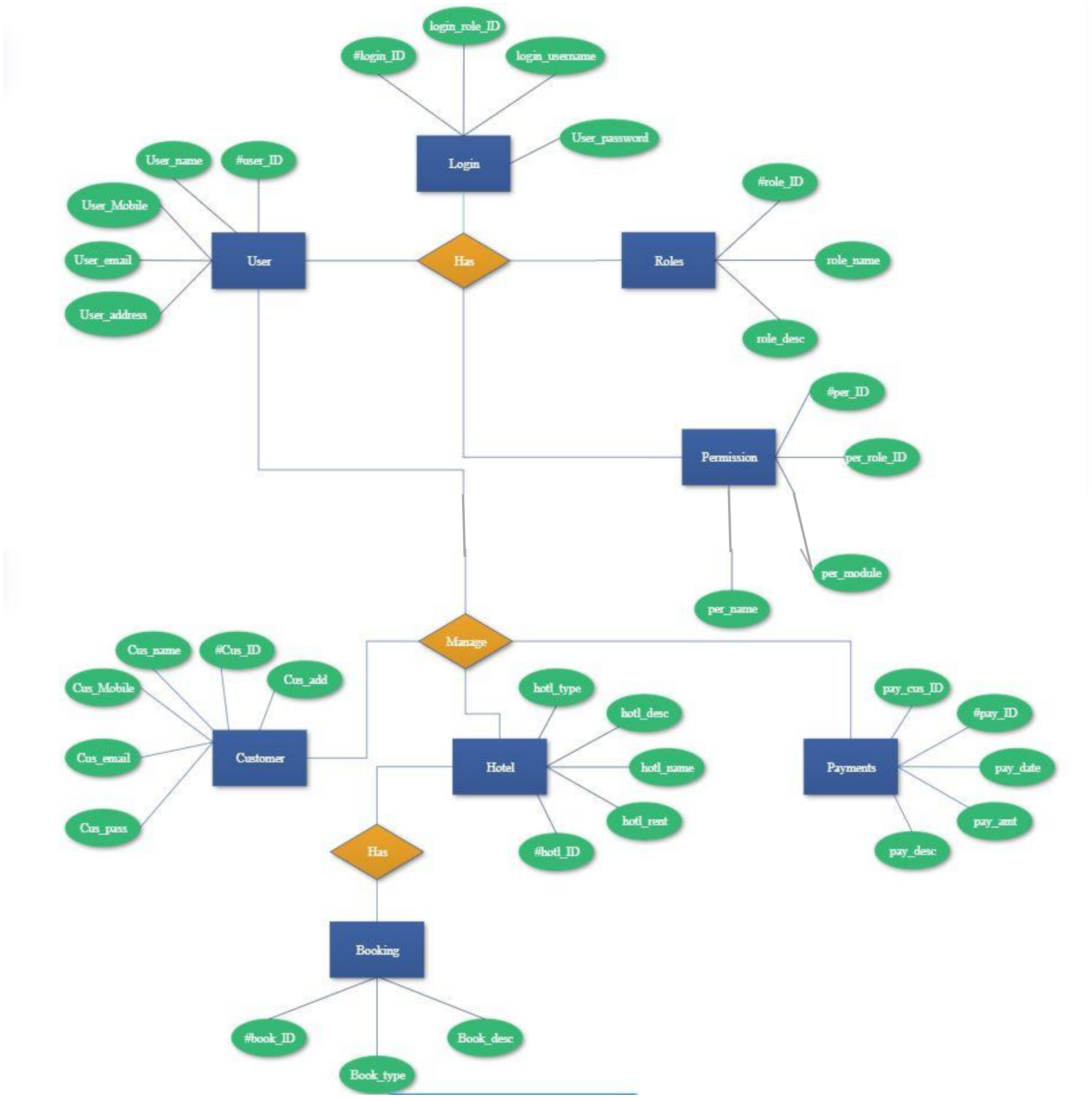
1.3– Data flow diagram

The following diagram shows the data flow of the system:



1.4– ER Diagram

The following diagram shows the Entity relationship model of the system:



CHAPTER 4 – Implementation

This chapter delves into the implementation details of the Hotel Management System, outlining the key functionalities, technologies used, and design considerations.

4.1 System Architecture

The Hotel Management System was designed with a client-server architecture, separating the front-end user interface from the back-end data processing and logic. The system comprises the following key components:

- **Front-end (Client-side):**
  - **HTML:** Provides the structure and content of the web pages.
  - **CSS:** Defines the visual style and layout of the web pages.
  - **JavaScript:** Handles user interactions, dynamic content updates, and communication with the back-end.
- **Back-end (Server-side):**
  - **PHP:** Processes user requests, interacts with the database, and generates dynamic content.
  - **MySQL:** Stores and manages the hotel's data, including guest information, room details, reservations, and financial records.

4.1.2 Technologies Used

The Hotel Management System was built using a combination of technologies, chosen for their suitability and efficiency:

- **Front-end:**

    - **HTML5:** Provides the foundation for the web pages, including semantic elements and multimedia support.
    - **CSS3:** Defines the visual styling, layout, and responsiveness of the web pages.
    - **JavaScript:** Handles user interactions, form validation, dynamic content updates, and AJAX requests to the server.
    - **jQuery:** Simplifies JavaScript development by providing a concise and efficient way to manipulate the DOM and handle events.
- **Back-end:**
    - **PHP:** A server-side scripting language used to process user requests, interact with the MySQL database, and generate dynamic content.
    - **MySQL:** A relational database management system used to store and manage the hotel's data.
- **Other Tools:**
    - **Apache Web Server:** Hosts the website and serves web pages to users.
    - **Visual Studio Code:** Used as the primary code editor for both front-end and back-end development.
    - **Git:** A version control system used to track changes in the codebase and facilitate collaboration.

4.1.3 Design Considerations

Several design considerations guided the development of the Hotel Management System:

- **User-friendliness:** The system was designed with a simple and intuitive interface, ensuring ease of use for both hotel staff and guests.

- **Security:** Robust security measures were implemented to protect sensitive guest data and financial information. This includes:
    - **Input Validation:** Sanitizing user inputs to prevent SQL injection and other attacks.
    - **Password Hashing:** Storing user passwords securely using hashing algorithms.
    - **Session Management:** Implementing secure session handling to prevent unauthorized access.
- **Scalability:** The system was built to accommodate future growth in hotel operations and guest volume. This was achieved through:
    - **Database Optimization:** Implementing efficient database queries and indexing to handle large amounts of data.
    - **Code Optimization:** Writing efficient code to minimize server load and improve performance.
    - **Load Balancing:** Utilizing multiple servers to distribute traffic and improve system responsiveness.
- **Data Integrity:** Data validation and error handling mechanisms were incorporated to ensure data accuracy and consistency.
- **Integration:** The system was designed to integrate with existing hotel systems, such as property management systems and payment gateways.

4.1.4 Implementation Details

This section provides a detailed overview of the implementation process for each module:

Reservation Module:

- **Functionality:**

    - Online booking: Allows guests to book rooms directly through the website.
    - Room availability check: Displays real-time room availability based on selected dates.
    - Guest registration: Collects guest information, including contact details and payment information.
    - Payment processing: Integrates with a payment gateway to process online payments securely.
- **Implementation:**
    - **Front-end:**
        - HTML forms for booking requests, including date selection, room type selection, and guest details.
        - JavaScript for form validation, dynamic content updates, and AJAX calls to the server.
        - jQuery library for simplified DOM manipulation and event handling.
    - **Back-end:**
        - PHP scripts to handle booking requests, validate data, update database records, and generate confirmation emails.
        - MySQL database tables to store reservation details, room availability, and guest information.

Room Management Module:

- **Functionality:**

    - Room type management: Allows staff to define different room types, pricing, and amenities.
    - Availability tracking: Updates room availability in real-time based on reservations and check-in/check-out dates.
    - Maintenance scheduling: Allows staff to schedule room maintenance and track maintenance history.

- **Implementation:**
    - **Front-end:**
        - HTML forms for managing room types, pricing, and availability.
        - JavaScript for dynamic content updates and AJAX calls to the server.
    - **Back-end:**
        - PHP scripts to handle room management requests, update database records, and generate reports.
        - MySQL database tables to store room details, availability, and maintenance information.

Guest Management Module:

- **Functionality:**
    - Guest registration: Collects guest information, including contact details, preferences, and booking history.
    - Profile management: Allows guests to view and update their profile information.
    - Communication tools: Enables staff to send messages and notifications to guests.

- **Implementation:**
    - **Front-end:**
        - HTML forms for guest registration and profile management.
        - JavaScript for form validation and dynamic content updates.
    - **Back-end:**
        - PHP scripts to handle guest management requests, update database records, and send notifications.
        - MySQL database tables to store guest information, booking history, and preferences.

Billing Module:

- **Functionality:**
    - Invoice generation: Generates invoices for guest stays, including room charges, additional services, and taxes.
    - Payment tracking: Records payments made by guests and tracks outstanding balances.
    - Financial reporting: Provides reports on revenue, expenses, and other financial metrics.
- **Implementation:**
    - **Front-end:**
        - HTML templates for generating invoices and displaying payment history.
        - JavaScript for dynamic content updates and AJAX calls to the server.

- **Back-end:**
    - PHP scripts to handle billing requests, generate invoices, process payments, and generate financial reports.
    - MySQL database tables to store billing information, payment records, and financial data.

4.1.5 Challenges and Solutions

During the implementation process, several challenges were encountered and addressed:

- **Challenge 1: Data Security:** Ensuring the security of sensitive guest data, including personal information, payment details, and booking history.
    - **Solution:** Implemented robust security measures, including input validation, password hashing, secure session management, and encryption for sensitive data.
- **Challenge 2: Scalability:** Designing the system to handle a growing number of guests, bookings, and hotel operations.
    - **Solution:** Implemented database optimization techniques, code optimization strategies, and load balancing to ensure the system can handle increased traffic and data volume.
- **Challenge 3: Integration with Existing Systems:** Integrating the new system with existing hotel systems, such as property management systems and payment gateways.
    - **Solution:** Designed the system to be modular and flexible, allowing for seamless integration with existing systems through APIs and data exchange protocols

4.2 – Coding

Html code (Home page) :

```
<!DOCTYPE html>

<html>

<head>

<title>Hotel Management</title>

<link rel="stylesheet" type="text/css" href="index.css"/>

<link rel="stylesheet" type="text/css" href="dhtmlxcalendar.css"/>



<script src="dhtmlxcalendar.js"></script>



<style>

#calendar

#calendar2

{

border: 1px solid #909090;

font-family: Tahoma;

font-size: 12px;
}

</style>

<script>

var myCalendar;

function doOnLoad()

{

myCalendar = new dhtmlXCalendarObject(["calendar","calendar2",]);

}

</script>

</head>

<body onload="doOnLoad();">

<div id="holder"><a href="index.html"><img src="logo.jpg" width="200px" height="100px"/></a>

<div id="header">



<ul>

<li><a href="login.html">Login</a></li>

<li><a href="overview.html">Overview</a></li>

<li><a href="restraunt.html">Restaurant</a></li>
```

```
<li><a href="book.html">Booking</a></li>

<li><a href="index.html">Home</a></li>

</ul>

</div><!--header -->

<br>

<div id="ab">

<div id="check">

<br>

<form action="http://localhost/Hotel management system/check.php" method="post">

<div style="position:relative;height:80px;">

<label for="in">Check in:</label> &nbsp;&nbsp;&nbsp;&nbsp;

<input id="calendar" name="in" placeholder="2024-05-02" required="" type="text"> <br><br>



<label for="in">Check out:</label> &nbsp;&nbsp;

<input id="calendar2" name="out" placeholder="2024-05-05" required="" type="text">
<br><br>



</div>

Room Type:
```

```
<select name="room">

<option value="A/C">A/C</option>

<option value="Non-A/C">Non-A/C</option>

</select>

<select name="type">

<option value="single">Single</option>

<option value="double">Double</option>

<option value="three">Three</option>

</select>

<br><br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;&nbsp;&nbsp;

<input class="hover22" type="submit" name="submit" value="submit" />

<input class="hover22" type="reset" name="reset" value="Reset" />

</form>

</div><!--chack -->

&nbsp;

<div id="chat">

<form action="" method="post">

<label for="in">Chat</label>
```

```
<input id="chat" name="chat" placeholder="Enter your message here" required="">

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input type="submit" name="submit" value="submit" />

</form>

</div><!--chat-->

<br>

<br>

<div id="fd">

<form action="http://localhost/Hotel management system/feedback.php" method="post">

<label for="in">Feedback</label>

<input id="fd" name="feedback" placeholder="Enter your feedback here" required="">

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input type="submit" name="submit" value="submit" />

</form>
```

```
<img id="img1" src="picture.jpg" width="1065px">

<img id="img2" src="room.jpg" width="1370px">


</div>

</div>

</body>

</html>
```

Css:

```
body{

margin:0px;

padding:0px;

background-color: #333;

}

#holder{

border:0px solid;

width:1348px;

height:auto;

margin-left:0px;
```

```
margin-right:0px;

margin-top:0px;

margin-bottom:0px;

background-color: black;

}

#header{

background-image:url(images/11.jpg);

background-repeat:no-repeat;

border:1px solid;

width:1347px;

height:100px;

margin-top:-100px;

margin-right:0px;

margin-left:0px;

background-color: #333;

}

#header ul{

list-style-type:none;
```

```
}

#header ul li a{

border:3px solid;

font-size:18px;

float:right;

background-color:#FFD700;

width:110px;

height:50px;

text-align:center;

margin:0px 10px 0px 10px;

font-weight:bold;

line-height:50px;

border-radius:50px;

transition-property:all;

transition-duration:0.4s;

transition-timing-function:linear;

}

#header ul li a{
```

```
text-decoration:none;

color:white;

}

#header ul li a:hover{

background-color:#00FF7F;

color:#4b388a;



}

#header ul li a:active{

background-color:#00FFFF;

}

#ab{

background-image:url("images/17.jpg");

border:0px solid;

width:285px;

height:750px;

margin-top:-21px;

}
```

```
#check{

border:0px solid;

color:#333;

width:285px;

height:200px;

margin-left:6px;

margin-right:0px;

margin-bottom:auto;

margin-top:0px

}


#chat{

margin-left:10px;

width:180px;

height:100px;

color:black;

}

#fd{
```

```
margin-left:10px;

width:180px;

height:100px;

color: black;

}

#ac{

background-color:purple;

border:0px solid;

width:276px;

height:551px;

text-align:center;

margin-left:288px;

margin-right:auto;

margin-top:-591px;

margin-bottom:auto;

}

#img1 {

margin-left: 275px;
```

```
margin-top: -500px;

}


.hover22 {

/* Base styles */

background-color: #FFFFE0; /* Light yellow */

padding: 10px 20px;

border-radius: 5px;

color: #333; /* Dark gray text */

cursor: pointer;

transition: background-color 0.3s ease; /* Smooth transition */

}


.hover22:hover {

/* Hover effect */

background-color: #FFD7D7; /* Light red */

}

#img2 {
```

```
margin-left: -20px;

margin-top: -20px;

}

#img3 {

margin-top: -750px;

margin-left: 680px;

}

#d1 {

width: 120px;

}
```

Php :

```
<?php

include_once('connection.php'); // Include your database connection file



function reg_in() {

global $mysqli; // Make the $mysqli connection object available inside the function



if (isset($_POST['submit'])) {
```

```
$check_in = filter_var($_POST['in'], FILTER_SANITIZE_STRING);

$check_out = filter_var($_POST['out'], FILTER_SANITIZE_STRING);

$room = filter_var($_POST['room'], FILTER_SANITIZE_STRING);

$room_type = filter_var($_POST['type'], FILTER_SANITIZE_STRING);



// Use prepared statements to prevent SQL injection

$stmt = $mysqli->prepare("INSERT INTO `check` (`check_in`, `check_out`, `room`,
`room_type`) VALUES (?, ?, ?, ?)");

$stmt->bind_param("ssss", $check_in, $check_out, $room, $room_type);

if ($stmt->execute()) {

header('Location: thankyou.html');

exit();

} else {

echo "Error: " . $stmt->error;

}

$stmt->close();

}

}
```

```
if (isset($_POST['submit'])) {

reg_in();

//echo ' succesfully inserted';

?>
```

CHAPTER-5 - Testing

5.1 Test case :

Test Case #:	001	**Test Case Name:**	System testing
System:	Hotel management	**Sub System:**	Login and check in
Designed By:	Huzaifa	**Design Date:**	16-05-2024
Executed By:	Huzaifa	**Execution Date:**	29-05-2024
Short Description:			

Pre-Condition:	

Operating System:	Windows 10	**Environment:**	LocalHost
Software Tools & Technologies with version	Google chrome **125.0.6422.112**		

Steps	**Strategy (Test-to-Pass) / (Test-to-Fail) (T-T-P) Or (T-T-F)**	**Action**	**Input**	**Actual Result / System Response**	**Expected Result/ Expected System Response**	**Status (Pass/Fail)**	**Remarks**

1	T-T-P	Login	Id and password	Submitted	Submitted	Pass	Data received in the database
2	T-T-P	Booking	Room no	Checked	Checked	Pass	

Feelings of completion:

At the completion of this project, we are extremely thankful to our teachers who helped us in the process. We got to learn a lot of lessons in this project and it was a really good experience. We faced a lot of uncountable problems but we solved them and it helped us build our problem solving skills. It feels really good to make a project by yourself and we'll have to do that in future so we got some experience about it. Thanks to our teachers and college to provide us a chance to enhance our skills.

Conclusion:

This report details the development of our Hotel management system, outlining the challenges encountered, the solutions implemented, and the final outcome. Despite facing initial delays, the project ultimately achieved a satisfactory level of completion within the allotted timeframe.

The report provides a comprehensive overview of the project's journey, from problem identification in Chapter 1 to system evaluation in Chapter 4. While the system functions effectively, there is always room for further improvement.

The initial stages of development progressed at a slower pace than anticipated. However, through focused effort and increased collaboration, we successfully accelerated development in the final weeks to meet the project deadline.

This report aims to provide a clear understanding of the project's process rather than simply summarizing results. Each chapter delves into specific aspects of the project, offering detailed explanations of the challenges, design choices, and implementation strategies.

Chapter 1 establishes the problem definition, outlining the project's objectives and the challenges we aimed to address. Chapter 2 details our approach to tackling these challenges, outlining the methodology and planning strategies employed. Chapter 3 focuses on the system's design and development, showcasing the various features and specifications implemented. Finally, Chapter 4 presents the evaluation process, where we tested the system's functionality and gathered user feedback.

While we did not conduct formal interviews or in-depth evaluations, we sought feedback from potential users to gain insights into their experience with the system. This user feedback proved invaluable in identifying areas for improvement and refining the system's design.

In conclusion, this report serves as a chronicle of our project's journey, highlighting the challenges faced, the solutions implemented, and the final outcome. While the system currently meets our objectives, we recognize the potential for further enhancements. The report provides a comprehensive understanding of the project's process, offering detailed insights into each stage of development.