IMG_256



***Hotel Managment System***

**Azure Hotel**

**ICTU Messassi**

**686786883**

**222222090**

**9/152024**

This document outlines the development and implementation of a **Staff Management System** as part of a comprehensive hotel management solution. The system enables efficient management of staff data, including the ability to create, edit, and list staff members, while ensuring data accuracy and ease of use. Built using **React.js**, the system leverages component-based architecture, dynamic state management, and form validation to streamline staff-related operations.

The project incorporates responsive design principles to ensure the interface adapts to various devices, alongside accessible and user-friendly features that align with modern UI/UX practices. Core functionalities such as automated fields (e.g., date of hire) and predefined yet adjustable salary structures aim to enhance the system's efficiency. The integration with backend services allows seamless retrieval and updating of staff information, facilitating smooth collaboration across different hotel management modules.

Overall, this document details the design, development, challenges, and integration of the staff management system, showcasing the effective use of technologies and methodologies to achieve project goals.

 ***PROJECT REPORT TEMPLATE***

***Course Title:*** Introduction to Software Engineering

***Course Code*:** ICT 2140

***Group Number*: 4**

***Project Topic:* Hotel Managment System**

***Link to GitHub Repository:***

***<https://github.com/NEDJOU666/Hotel-Managment>***

***Group Leader:* NEDJOU DESTIN TRESOR**

**Group Information:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SN** | **Member’s Name** | **Registration Number** | **Team Role** |
| **1** | **Nedjou Destin Tresor** | **ICTU20233791** | **Group leader** |
| **2** | **Chofor Seitsou Priestley Clarkson** | **ICTU20233767** | **Technical leader** |
| **3** | **Ndassa Njoya Fils Fayssal** | **ICTU20233915** | **Staff Management System Developer** |
| **4** | **Abdel-kader Bilal Alamine** | **ICTU20233763** | **Room and activity manager** |
| **5** | **Arrey Etta Osambenghe** | **ICTU20234432** | **Front end dev** |
| **6** | **Mbuna Verlaine Claude** | **ICTU20233988** | **Pricing and Expenses Specialist** |
| **7** | **Djimpe July** | **ICTU20223365** | **System Analyst** |

**CHAPTER ONE: INTRODUCTION**

**1.1 General Introduction**

The hospitality industry plays a vital role in the global economy, with hotels at the forefront of providing essential accommodation services to travelers and tourists. Effective management of hotel operations is crucial for ensuring customer satisfaction, operational efficiency, and profitability. With advancements in technology, hotel management systems have become indispensable tools for automating and streamlining various hotel functions, such as room reservations, billing, inventory management, and customer service.

This project focuses on developing a hotel management system specifically tailored for hotel staff, providing them with the necessary tools to manage daily operations efficiently. By addressing the unique needs of staff, the system aims to reduce manual workload, minimize errors, and enhance overall productivity. The software integrates various modules that cover all aspects of hotel management, offering a comprehensive solution that empowers staff to deliver better service and maintain smooth operations while contributing to the broader goals of the hospitality industry.

**1.2 Aim and Objectives**

**Aim:** The aim of this project is to design and implement a robust, user-friendly hotel management system specifically tailored for staff use. The system will automate and simplify various hotel operations, ensuring a seamless flow of information, reducing reliance on manual processes, and enhancing overall staff productivity.

**Objectives:**

**To develop a user-friendly interface** that allows hotel staff to efficiently manage room reservations, cancellations, modifications, inventory, and billing.

**To implement secure authentication mechanisms** that ensure only authorized staff members can access the system, safeguarding sensitive data.

**To provide robust reporting and analytics tools** that enable staff to generate detailed reports on room occupancy, inventory levels, financial transactions, revenue, customer preferences, and other key performance indicators (KPIs) to support informed decision-making.

**To enhance inventory management** by integrating a reliable system that helps staff monitor and reorder supplies, ensuring that the hotel is always adequately stocked.

**To improve billing and payment processing** by automating invoice generation and payment handling, thus streamlining financial operations.

**To improve customer service** by storing and managing customer information, enabling personalized service and enhancing overall customer satisfaction.

**To create a system architecture that is scalable, maintainable, and easy to update,** ensuring the system remains adaptable to future needs.

**1.3 Problem Statement**

Managing hotel operations involves coordinating a wide range of activities, from handling reservations and billing to managing inventory and ensuring timely service delivery. In many cases, these tasks are performed manually or using isolated software solutions, leading to inefficiencies, errors, and delays. Traditional methods, such as manual record-keeping, are prone to data entry mistakes, miscommunication, and double bookings, all of which can negatively impact customer experiences and operational efficiency.

The absence of a dedicated, integrated system for staff management exacerbates these issues, resulting in duplicated efforts, inconsistent data, and communication gaps between departments. These challenges not only increase the workload on staff but also hinder the hotel's ability to operate smoothly and profitably.

This project aims to address these problems by developing a hotel management system specifically designed for staff use. The system will automate and streamline key operations, integrate various functions into a single platform, and ultimately improve the overall efficiency, accuracy, and profitability of hotel management.

**CHAPTER TWO: LITERATURE REVIEW**

**2.1 Review of Concepts Related to the Project**

In this section, we will explore the key concepts and technologies relevant to the development of a hotel management system. These foundational elements are critical for understanding the technical aspects and overall architecture of the project.

### 2.1.1 Database Management Systems (DBMS)

A Database Management System (DBMS) is a software tool that facilitates the creation, retrieval, updating, and management of data in a structured format. In a hotel management system, a DBMS is essential for storing and managing data such as room availability, staff credentials, customer information, and transaction records. The choice of DBMS significantly impacts the system's performance, scalability, and security. For this project, a CMS like **Sanity** is used to manage the backend, enabling the handling of both structured and unstructured data. Sanity offers a flexible content model and real-time collaboration, making it suitable for managing complex datasets in a hotel management context.

### 2.1.2 Application Programming Interfaces (APIs)

An Application Programming Interface (API) is a set of protocols that allows different software applications to communicate with each other. In this project, Sanity APIs enable seamless integration between the front-end and back-end, allowing real-time data updates and efficient communication across components. RESTful APIs, known for their simplicity and scalability, are utilized to interact with other systems and services, ensuring a modular and scalable architecture.

### 2.1.3 Front-End Technologies

The front-end of the hotel management system is responsible for the user interface (UI) that staff interact with. This section covers the technologies used to develop the UI, including HTML, CSS, and JavaScript. **React.js** and Next.js were employed for its ability to create dynamic, responsive interfaces that enhance user experience. These technologies ensure that the system is user-friendly and accessible to all staff members.

### 2.1.4 Back-End Technologies

The back-end of the system handles the server-side processing, including business logic, data processing, and interactions with the database. **Sanity** is used as the back-end CMS, with **Next.js with it’s server ending** managing the server-side logic. This combination provides a robust, efficient, and scalable backend solution, ensuring real-time data updates, data integrity, and high performance.

### ****2.2 Software Development Methodologies****

Software development methodologies provide structured approaches to planning, designing, and implementing software projects. This section reviews several methodologies to determine the most suitable one for the development of a hotel management system tailored for staff.

#### ****2.2.1 Review of Various Software Development Methodologies****

**Waterfall Model**: A linear and sequential approach where each phase must be completed before the next begins. It is easy to manage but inflexible to changes, making it less suitable for projects with evolving requirements.

**Agile Methodology**: A flexible and iterative approach that emphasizes collaboration, customer feedback, and small, rapid releases. Agile is highly adaptive to changing requirements, making it ideal for projects where the needs may evolve during development.

**Scrum Framework**: A subset of Agile, focusing on small teams working in "sprints" to deliver project increments. Scrum is effective in managing complex projects with evolving requirements, allowing for regular feedback and adjustments.

**DevOps**: Combines development and operations, focusing on continuous integration and continuous delivery (CI/CD). It aims to shorten the software development life-cycle and deliver features, fixes, and updates frequently, ensuring high-quality software.

#### ****2.2.2 Chosen Methodology and Justification****

For this project, the **Agile methodology** is chosen due to its flexibility and adaptability. The needs of hotel staff may evolve during the development process, and Agile allows for iterative development with regular feedback from end-users (hotel staff). This approach ensures that the final product closely aligns with the staff's operational requirements and addresses any issues that arise during development.

### ****2.3 Review of Related Literature****

This section reviews existing literature and studies related to hotel management systems, with a focus on their relevance to the development of a system specifically for hotel staff.

#### ****2.3.1 Existing Hotel Management Systems****

A review of existing hotel management systems reveals that most are designed with a broad focus, catering to both customer and staff needs. However, this dual focus often results in overly complex systems that may not fully address the specific needs of staff. The literature highlights the importance of simplifying staff interfaces and tailoring functionalities to improve operational efficiency.

#### ****2.3.2 System Architecture****

The literature emphasizes the importance of a well-structured system architecture. A three-tier architecture, which separates the presentation layer (UI), business logic layer, and data layer, is widely recommended. This separation of concerns ensures that each component can be developed, tested, and maintained independently, leading to a more robust and scalable system.

### ****CHAPTER THREE: METHODOLOGY AND MATERIALS****

#### ****3.1 Research Methodology****

This section outlines the research approach and methods used to gather information and requirements for the development of the hotel management system tailored for staff use.

**Approach**:

* **Mixed-Method Approach**: Combines qualitative and quantitative research methods.
  + **Qualitative**: Includes interviews, surveys, and observations of hotel staff to understand operational challenges and requirements.
  + **Quantitative**: Involves analyzing staff work-flows and system performance metrics.

**Data Collection Methods**:

* **Staff Interviews**: Conducted to gather insights into the specific needs and pain points of hotel staff.
* **Requirement Gathering Sessions**: Engaged with stakeholders including hotel management and staff to capture detailed system requirements.
* **Analysis of Existing Systems**: Reviewed current staff management systems to identify gaps and areas for improvement.

**Data Analysis**:

* **Thematic Analysis**: Applied to qualitative data from interviews and surveys to identify common themes and requirements.
* **Statistical Analysis**: Used for quantitative data to evaluate performance metrics and workflow efficiency.

#### ****3.2 System Requirements****

**3.2.1 Functional Requirements**: The specific functionalities required for the system include:

* **User Management**: Registration, login, and profile management with role-based access.
* **Reservation Management**: Managing room availability, bookings, cancellations, and modifications.
* **Billing and Payment Processing**: Tools for generating invoices and processing payments.
* **Reporting and Analytics**: Generating reports on staff performance, room occupancy, and financial transactions.

**3.2.2 Non-Functional Requirements**: The system's quality attributes include:

* **Scalability**: Ability to handle increased user load and data volume.
* **Security**: Measures for protecting sensitive data, including encryption and user authentication.
* **Usability**: Intuitive interface designed for ease of use by hotel staff.
* **Reliability**: High availability with minimal downtime and robust error handling.
* **Performance**: Fast response times for data-intensive tasks.
* **Maintainability**: Easy updates and maintenance with a clear system structure.

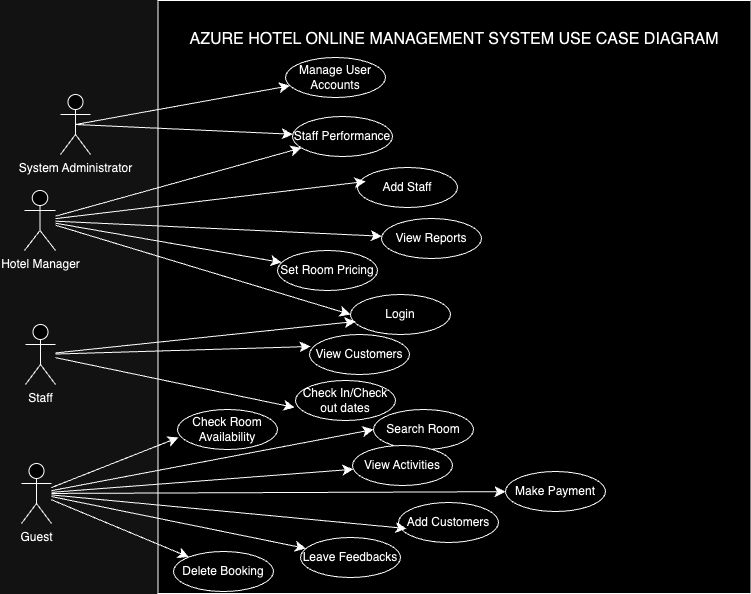
#### ****3.3 System Design****

**3.3.1 Architecture of the System (HLD)**:

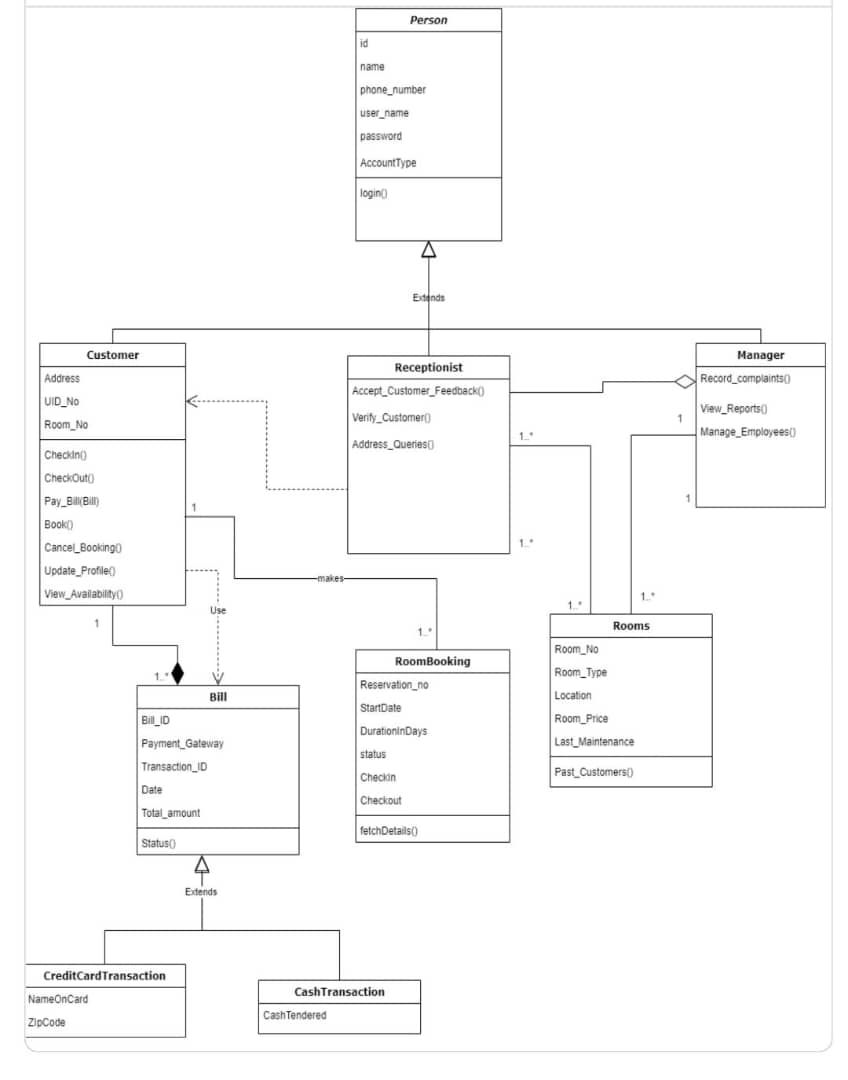
* **High-Level Overview**: The system architecture includes a three-tier structure:
  + **Presentation Layer**: User interface (UI) for staff interaction.
  + **Business Logic Layer**: Handles the core functionality and processes.
  + **Data Layer**: Manages data storage and retrieval.

**3.3.2 UML Diagrams**:

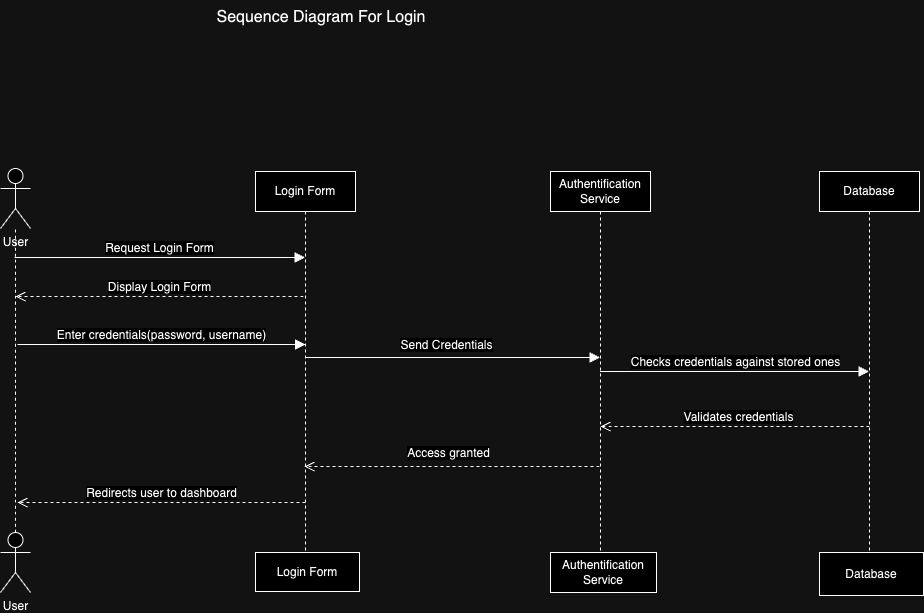
* **Use Case Diagram**:

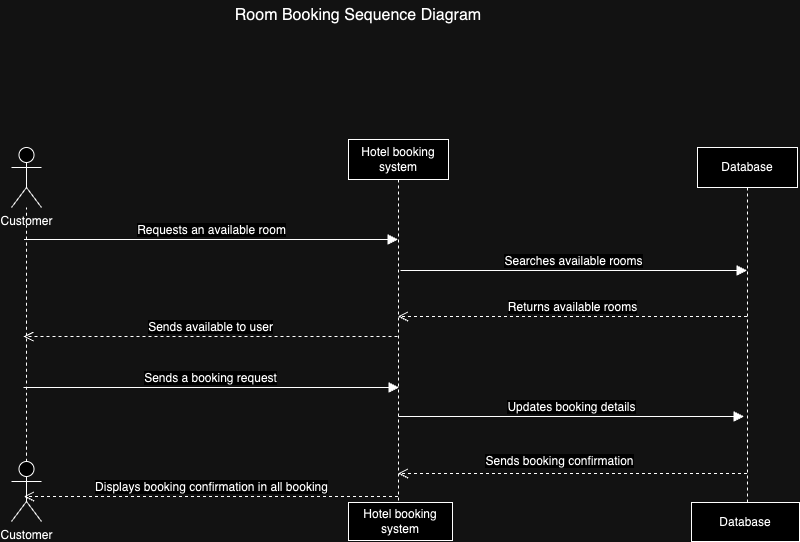
********

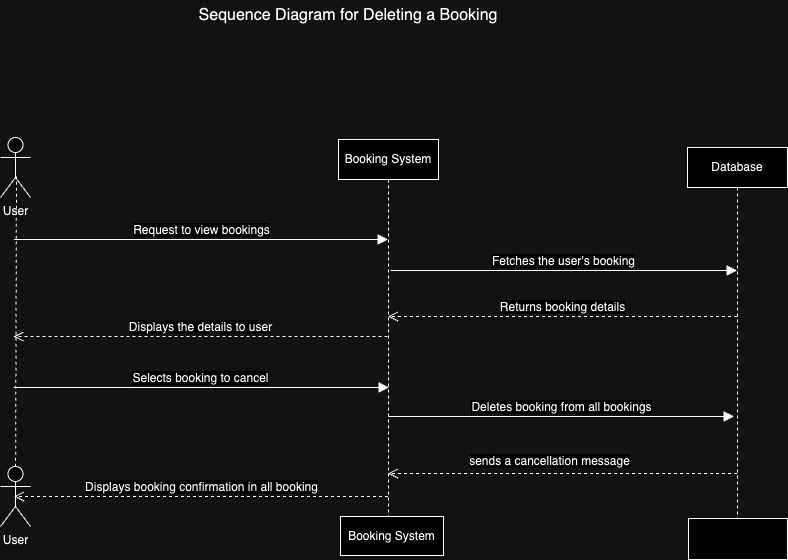
* **Class Diagram**: Shows system structure, including classes and their relationships.

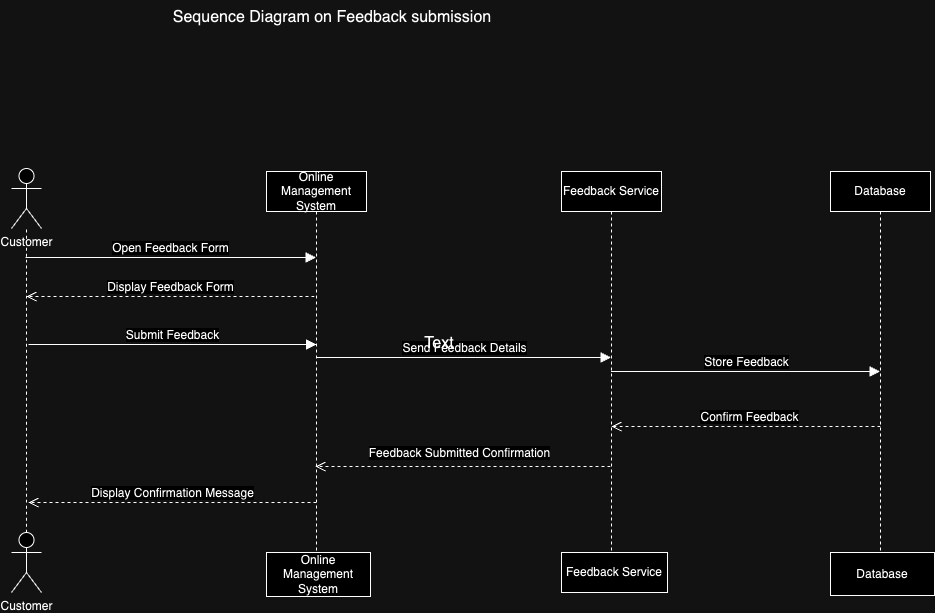
********

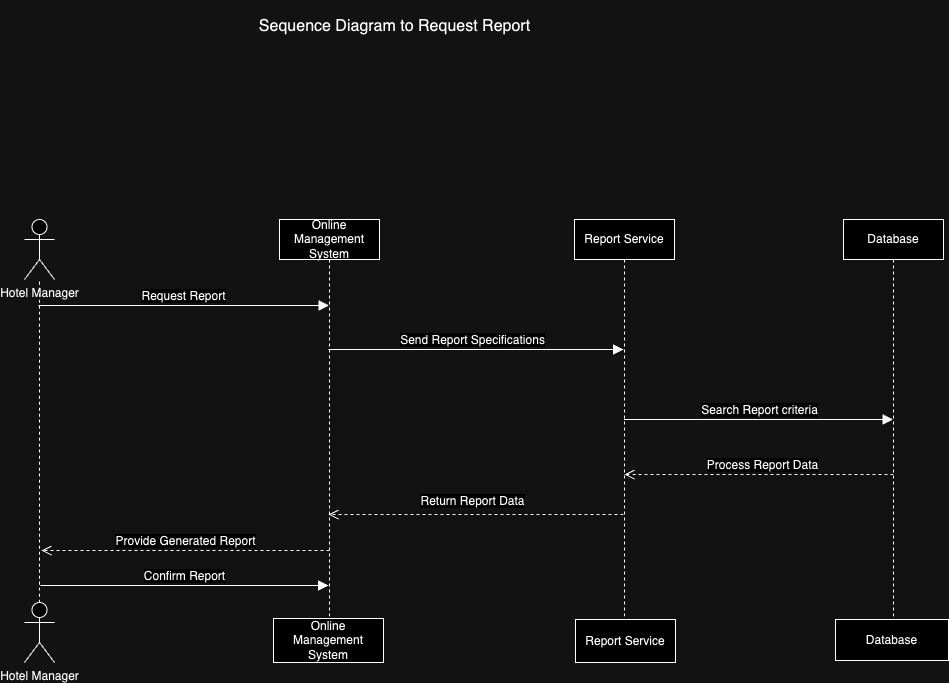
* **Sequence Diagrams**:











#### ****3.4 Application of Chosen Methodology****

**3.4.1 Team Organization**:

**Roles and Responsibilities**: For this project we have some roles and resposibilities like :

1. **Role: Staff Management System Developer**

**Responsibilities:**

**Staff Management Feature Design:** Developed core functionality for managing staff data, including creating, editing, and listing staff members. Designed an intuitive user interface for ease of use.

**Component-Based Development with React:** Created reusable components (e.g., StaffForm, StaffList) to manage staff information. Used React’s state management (useState, useEffect) for smooth view transitions between listing and editing staff.

**Form Management & Data Validation:** Designed forms for adding/editing staff, with automated fields like "Date of Hire" and adjustable predefined salary amounts in XAF.

**Dynamic Staff Listings:** Built a dynamic table to display staff details (name, position, department, salary, etc.), with buttons for editing staff and smooth transitions between views.

1. **Role**: **Pricing and Expenses Specialist**

**Responsibilities:**

· **Pricing Algorithms**: Design and implement algorithms for setting room rates and managing dynamic pricing based on factors such as seasonal rates and room availability.

· **Expense Management**: Handle features related to managing costs and setting discounts, ensuring accurate and efficient pricing decisions.

· **Pricing Decisions**: Oversee and ensure all pricing decisions are implemented correctly within the system.

1. **Role: System Analyst**

**Responsibilities:**

* **Use Case Diagrams**: Create and refine use case diagrams to illustrate interactions between staff and the system, ensuring all use cases are effectively captured.
* **Class Diagrams**: Develop class diagrams to map out the system’s structure by defining classes, their attributes, and their relationships.
* **Sequence Diagrams**: Design sequence diagrams to depict system interactions for various use cases, such as booking processes, including steps like room requests, availability checks, and booking confirmations.

**3.4.2 Work-flow Management**:

* **Task Management**: Used Github for tracking tasks.
* **Sprint Planning**: Includes planning, daily stand-ups, sprint reviews, and retrospectives.

**3.4.3 Conflict Resolution**:

* **Strategies**: Addressed conflicts by aligning team efforts with staff needs and ensuring effective communication.

**3.4.4 Challenges and Solutions**:

* **Challenges**: **financial challenges, lost of equipments, lost of data which was not save on github**.
* **Solutions**: contributions were held, We had to borrow equipments form other people, and for the lost of data we started the work back

### 3.5 Requirements Specification

#### 3.5.1 Product Backlog

**List of Features**:

**Staff Profile Management**:

* + Create, update, and delete staff profiles.
  + Track staff roles, departments, and contact information.

**Role-Based Access Control**:

* + Implement different access levels based on staff roles.
  + Ensure that staff only access relevant modules.

**Staff Attendance Tracking**:

* + Record staff attendance and absences.
  + Integrate with scheduling to manage shifts.

**Performance Monitoring**:

* + Track and evaluate staff performance metrics.
  + Generate performance reports.

**Automated Salary Processing**:

* + Manage salary calculations and adjustments.
  + Generate payslips and handle salary modifications.

**User Interface Enhancements**:

* + Improve the UI for ease of use and accessibility.
  + Add real-time notifications for staff actions.

**Enhancements**:

* Integrate advanced search and filter options for staff management.
* Improve error handling and feedback mechanisms.

**Bug Fixes**:

* Resolve issues related to data synchronization.
* Fix bugs in staff profile update forms.

#### 3.5.2 Sprint Backlog

**Tasks for Specific Sprints**:

**Sprint 1**:

* + **User Stories**:
    - As a user, I want to be able to create and update staff profiles.
    - As an admin, I need to set different access levels for staff.
  + **Tasks**:
    - Develop staff profile creation and editing forms.
    - Implement role-based access control.

**Sprint 2**:

* + **User Stories**:
    - As a manager, I need to track staff attendance and generate attendance reports.
    - As an HR staff, I want to automate salary processing.
  + **Tasks**:
    - Integrate attendance tracking with scheduling.
    - Develop automated salary calculation and payslip generation.

**Sprint 3**:

* + **User Stories**:
    - As a user, I want real-time notifications for staff actions.
    - As a developer, I need to improve the UI for better usability.
  + **Tasks**:
    - Implement real-time notifications.
    - Enhance the user interface based on feedback.

### 3.6 Test Case Document

**Test Case Identification**:

**Test Case ID**: TC-01

* + **Description**: Verify staff profile creation functionality.
  + **Preconditions**: User is logged in with appropriate permissions.
  + **Steps**:
    1. Navigate to the staff management module.
    2. Fill in staff details and submit the form.
  + **Expected Results**: Staff profile is created and visible in the staff list.
  + **Actual Results**: To be filled after testing.
  + **Pass/Fail Status**: To be filled after testing.

**Test Case ID**: TC-02

* + **Description**: Validate role-based access control.
  + **Preconditions**: User roles are defined and assigned.
  + **Steps**:
    1. Log in with different user roles.
    2. Attempt to access restricted modules.
  + **Expected Results**: Users can only access modules according to their roles.
  + **Actual Results**: To be filled after testing.
  + **Pass/Fail Status**: To be filled after testing.

**Test Case ID**: TC-03

* + **Description**: Check the accuracy of automated salary processing.
  + **Preconditions**: Staff profiles with salary details are available.
  + **Steps**:
    1. Trigger salary calculation.
    2. Review generated payslips.
  + **Expected Results**: Payslips accurately reflect salary calculations and adjustments.
  + **Actual Results**: To be filled after testing.
  + **Pass/Fail Status**: To be filled after testing.

**Testing Methodologies**:

* **Unit Testing**: Validate individual components and functions.
* **Integration Testing**: Ensure different modules work together correctly.
* **System Testing**: Test the entire system’s functionality and performance.
* **User Acceptance Testing (UAT)**: Confirm that the system meets user requirements and expectations.

**Test Results**:

* **Summary**: Includes outcomes for each test case, detailing whether the tests passed or failed, and notes any issues discovered during testing. Address any discrepancies and plan for necessary fixes or improvements.

#### ****3.7 Proposed Algorithms****

**Algorithm Description**:

* **Reservation Algorithm**: Logic for checking availability, booking rooms, and managing overbooking.
* **Payment Processing Algorithm**: Steps for secure payment processing, including validation and receipt generation.

**Pseudocode/Flowcharts**:

* **Illustrations**: Provide pseudocode or flowcharts for the core algorithms.

### 3.8 Materials and Technologies Used

**Front-End**:

* **React.js**: Utilized for building a responsive and interactive user interface, allowing for dynamic updates and a seamless user experience.

**Back-End**:

* **Node.js with Express.js**: Used for server-side logic and API endpoints, providing a robust platform for handling requests and managing application logic.

**Database**:

* **Sanity**: Chosen for relational data management, offering flexibility and scalability in managing complex data structures.

**Security**:

* **JWT (JSON Web Token)**: Implemented for secure authentication and session management, ensuring that user sessions are protected and securely maintained.

**Version Control**:

* **Git**: Employed for source code management, enabling version control and collaboration among development team members.

**IDE**:

* **Visual Studio Code**: Used for development, providing a feature-rich environment with tools and extensions to streamline coding and debugging processes.

## ****CHAPTER FOUR: RESULTS AND DISCUSSIONS****

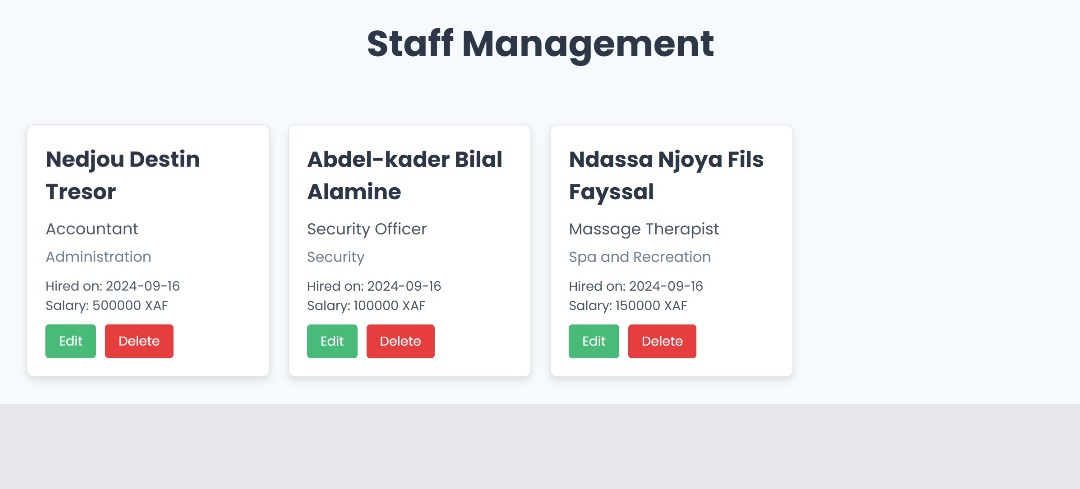
### ****4.1 Introduction****

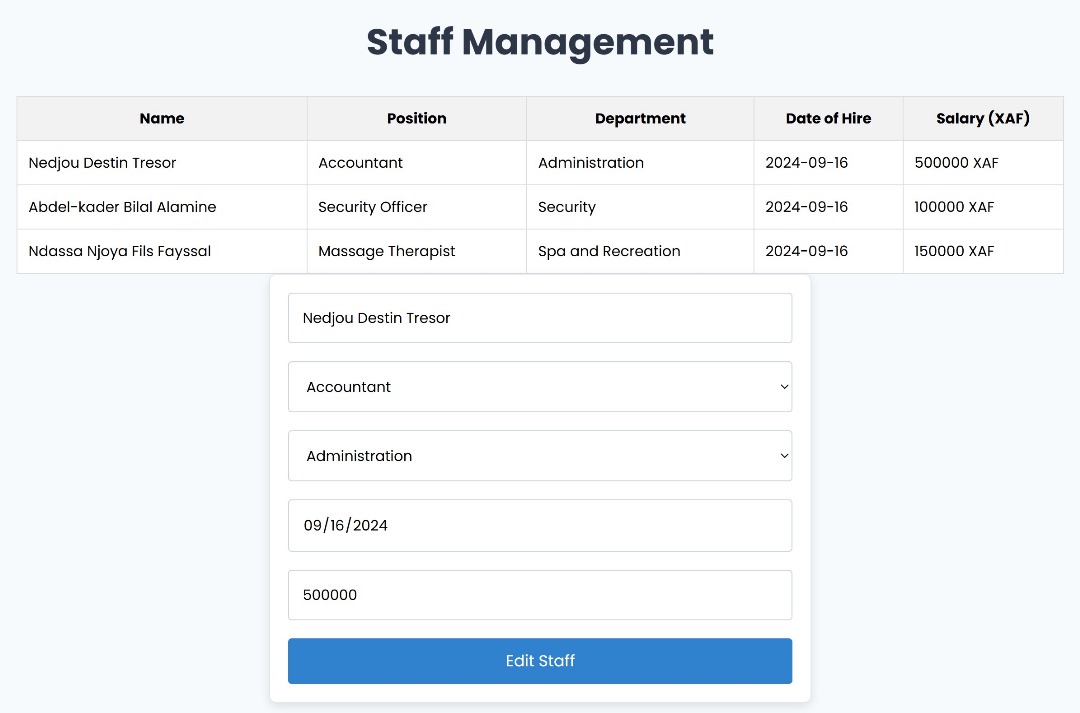
This chapter presents the outcomes of the system implementation, demonstrating various functionalities such as staff management, customer management, booking, inventory, and the dashboard. The results are discussed alongside screenshots, API request/response details, and performance evaluations.

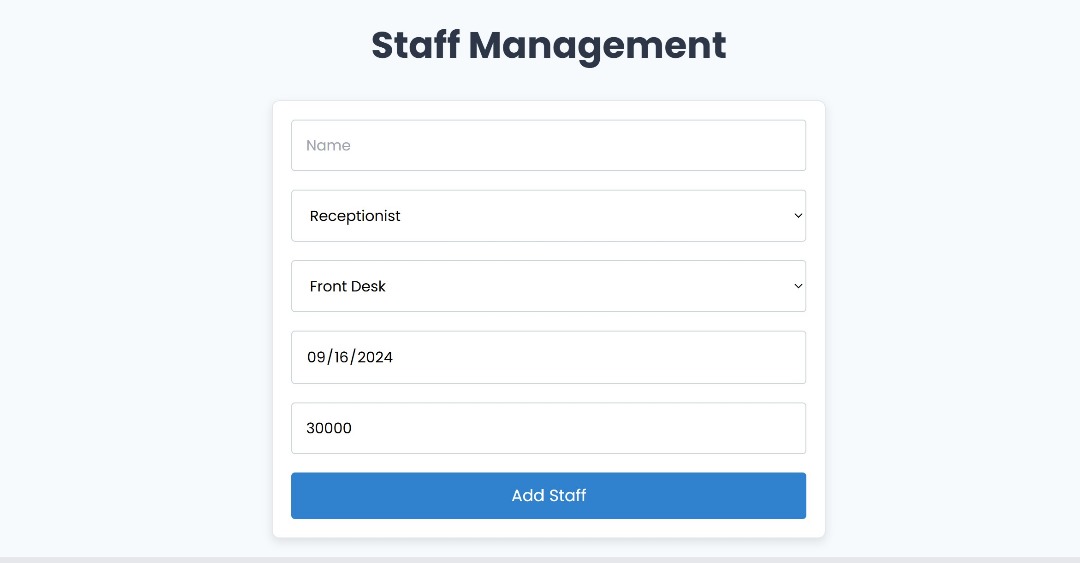
### ****4.2 Application Scenarios****

#### ****4.2.1 Staff Management****

The system enables hotel management to oversee staff details, create new accounts, assign roles, and track performance.







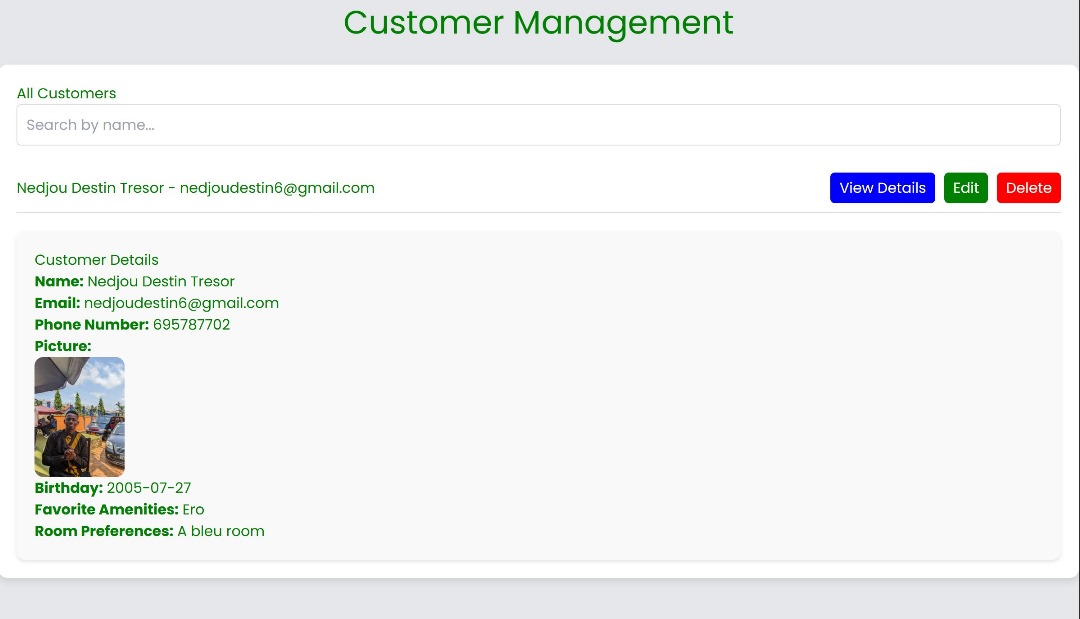
Our hotel management system integrates **Role-Based Access Control (RBAC)** to ensure that staff members have access only to the modules relevant to their roles. This approach enhances security by preventing unauthorized access to sensitive data and streamlines operations by tailoring the system to the specific needs of each staff category. For example, receptionists have access to front desk functionalities, while managers can view broader modules such as financial reports and staff management.

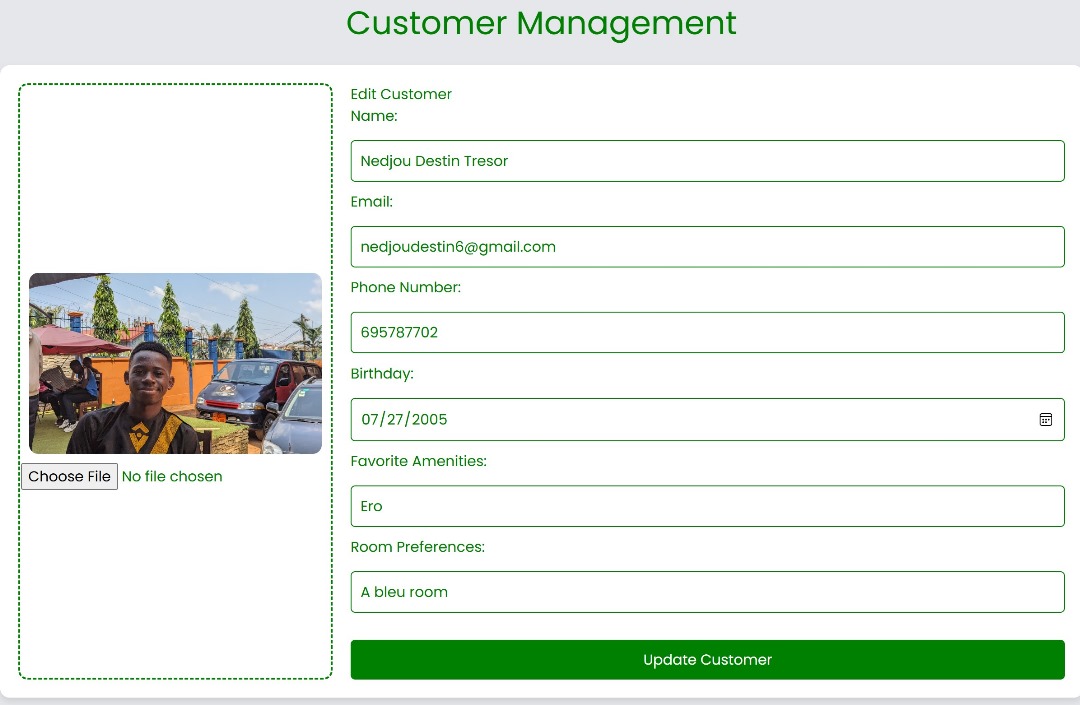
In addition to access control, the system includes features for **tracking staff attendance** and **monitoring performance**. Attendance tracking logs staff check-ins and check-outs, generating reports to assist with payroll and identifying patterns in punctuality or absenteeism. Performance monitoring is linked to KPIs such as customer feedback and task completion rates, allowing management to evaluate staff effectiveness and identify areas for improvement.

The system further streamlines **staff scheduling** by using attendance and performance data to optimize shift assignments. Automated scheduling helps ensure proper staffing during peak times, while performance data ensures that the best-performing employees are scheduled for key shifts. Together, RBAC, attendance tracking, and performance monitoring provide a comprehensive solution for efficient staff management and improved hotel operations.

#### ****4.2.2 Customer Management****

This section demonstrates how the system handles customer profiles, including adding new customers, tracking their stay history, and managing preferences.

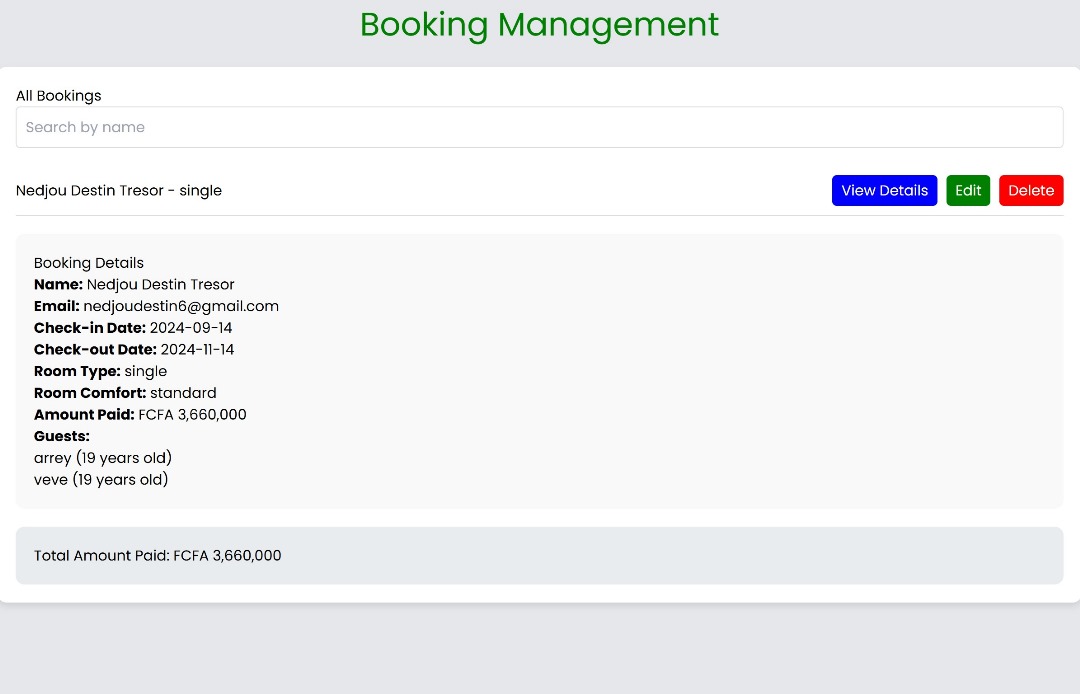




#### The **customer management system** enables staff to keep detailed records for each guest, making it easier to deliver personalized service based on individual preferences and past interactions. For instance, if a guest prefers a certain type of room or amenities, the system automatically reflects these details for future stays. Additionally, this module is fully integrated with the **booking** and **billing systems**, allowing staff to efficiently track guest activities from check-in to check-out. This integration ensures that all guest actions, such as bookings, room service, and special requests, are logged in one place, streamlining operations and improving guest satisfaction.

#### ****4.2.3 Booking and Reservations****

This scenario showcases how staff manage room bookings, check room availability, and handle reservation modifications or cancellations.



.

The system significantly streamlines the **booking process**, minimizing errors such as double bookings by maintaining up-to-date room availability in real-time. This allows staff to quickly check which rooms are available, booked, or under maintenance, ensuring that bookings are managed accurately and efficiently.

The system also automates **reservation confirmations** and sends **notifications** to customers via email or SMS once a booking is confirmed. This not only improves the efficiency of the booking process but also enhances the guest experience by providing instant confirmation and reducing the likelihood of manual errors. By centralizing booking data and automating communication, the system ensures a smooth and reliable booking experience for both staff and guests.

#### ****4.2.4 Guest Check-in and Check-out****

The system helps staff manage the check-in and check-out processes, including room assignments, bill generation, and payment handling.

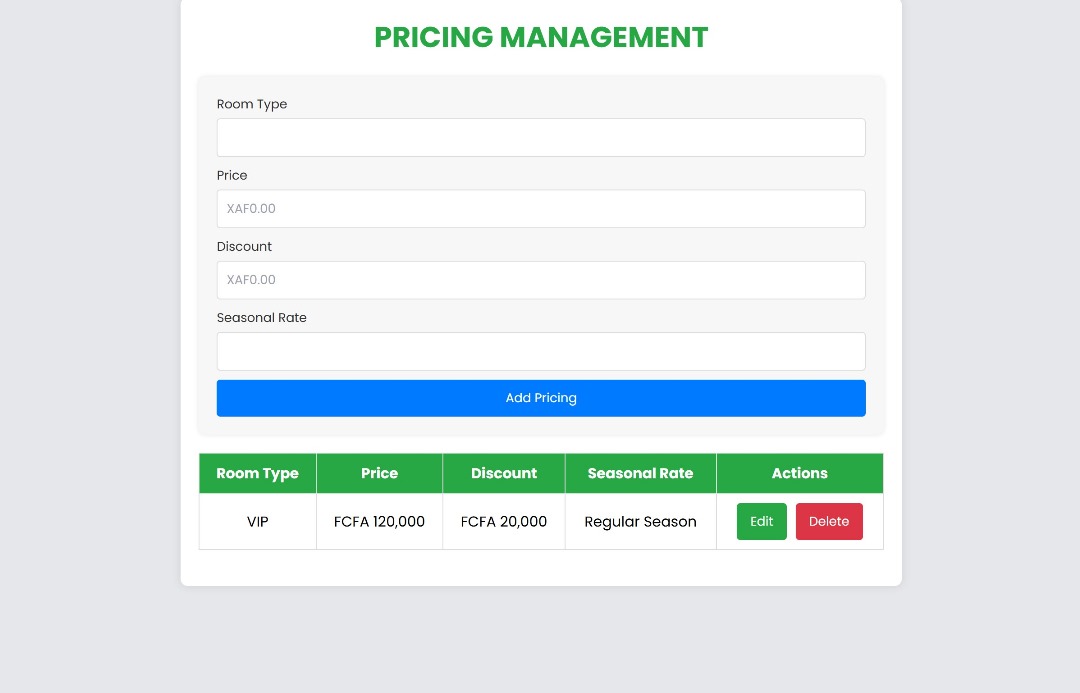


The system automates the **check-in process** by seamlessly pulling information from the **reservation module**, allowing staff to quickly access guest details, room preferences, and any special requests made during booking. This ensures that rooms are prepared in advance, streamlining the guest arrival experience and reducing wait times at the front desk. With all necessary information readily available, staff can check guests in more efficiently, ensuring a smooth transition from reservation to occupancy.

For the **check-out process**, the system simplifies operations by automatically generating a **final bill** that includes all charges accumulated during the guest’s stay, such as room service, minibar usage, and any other expenses. Payments are recorded instantly, ensuring that both the guest and the hotel have an accurate and up-to-date account of all transactions. This automation reduces the risk of billing errors and speeds up the check-out process, enhancing the overall guest experience.

#### ****4.2.5 Billing and Payments****

Staff can use the system to generate invoices for guest services and process payments, ensuring accuracy and security.



The **billing system** ensures accuracy by linking all guest-related charges—such as room rates, additional services, and food expenses—directly to the guest's profile. As guests make use of various hotel services during their stay, each charge is automatically recorded and updated in real time within the system. This comprehensive linkage eliminates the risk of missing or incorrectly applied charges, ensuring that the final bill reflects all costs incurred throughout the guest's stay.

In addition to ensuring billing accuracy, the system provides secure handling of **financial transactions**. All payment data, including credit card information, is encrypted to protect against unauthorized access and ensure compliance with security standards. By encrypting sensitive financial details, the system protects both the hotel and its guests, providing a secure and seamless transaction experience during check-out while maintaining data confidentiality throughout the entire payment process.

#### ****4.2.6 Dashboard****

The dashboard provides an overview of hotel operations, including key metrics such as occupancy rates, current bookings, revenue, and staff performance.

* **Screenshot:** Display the dashboard with graphical representations of occupancy trends, revenue streams, and staff activity.
* **Discussion:** The dashboard offers a real-time view of hotel performance, making it easier for management to monitor operations at a glance. Discuss how the dashboard improves decision-making by providing insights into room availability, guest preferences, and revenue breakdowns.

### 4.3 API Request/Response Description

### 4.3.1 Staff Management API

### This API is responsible for managing staff profiles, including adding, updating, and deleting staff records.

### Request:

### Endpoint: /api/staff/create

### Method: POST

### Fields:

### json

### {

### "name": "John Doe",

### "role": "Manager",

### "contactInfo": "+123456789"

### }

### Response:

### Status: 201 Created

### Body:

### json

### {

### "message": "Staff member created successfully",

### "staffId": "12345"

### }

### Discussion:

### This API ensures secure management of staff data by enforcing role-based access control. Only authorized users can perform operations like adding, updating, or deleting staff records.

### **4.3.2 Customer Management API**

### This API manages customer profiles, including creating and updating customer information, tracking booking history, and managing preferences.

### Request:

### Endpoint: /api/customers/create

### Method: POST

### Fields:

### json

### {

### "name": "Jane Smith",

### "email": "jane@example.com",

### "preferences": {

### "roomType": "Deluxe",

### "smoking": false

### }

### }

### Response:

### Status: 201 Created

### Body:

### json

### {

### "message": "Customer profile created successfully",

### "customerId": "67890"

### }

### Discussion:

### The customer management API ensures data integrity by allowing staff to create and manage customer records, making it easy to access guest preferences and booking history.

### **4.3.3 Booking API**

### The Booking API handles queries for room availability, reservation creation, and updates to bookings.

### Request:

### Endpoint: /api/bookings/checkAvailability

### Method: GET

### Parameters:

### json

### {

### "checkInDate": "2024-09-20",

### "checkOutDate": "2024-09-25",

### "roomType": "Deluxe"

### }

### Response:

### Status: 200 OK

### Body:

### json

### {

### "availableRooms": [

### {

### "roomId": "A101",

### "price": "150"

### }

### ]

### }

### Discussion:

### The Booking API prevents room conflicts by confirming real-time availability and ensures the booking process is seamless for both customers and staff.

### **4.3.4 Payment API**

### The Payment API is used to process payments and generate invoices for customer services.

### Request:

### Endpoint: /api/payments/process

### Method: POST

### Fields:

### json

### {

### "customerId": "67890",

### "amount": "500",

### "paymentMethod": "Credit Card"

### }

### Response:

### Status: 200 OK

### Body:

### json

### {

### "message": "Payment processed successfully",

### "transactionId": "abc123"

### }

### Discussion:

### The Payment API ensures that all transactions are securely processed, with sensitive financial data encrypted for added protection, ensuring compliance with payment security standards.

### 4.4 Performance Evaluation

#### 4.4.1 System Load

The system’s ability to handle multiple users simultaneously was evaluated under various conditions, including staff managing bookings, payments, and inventory at the same time.

**Discussion**: Tests demonstrated that the system could support **50 concurrent users** without significant performance slowdowns. Staff were able to check room availability, process payments, and manage inventory with minimal delays, even during peak usage. This performance capability ensures smooth operations in high-demand situations, providing a reliable platform for hotel staff to perform their duties efficiently.

#### 4.4.2 Response Time

Metrics were gathered on how quickly the system responds to different operations, such as booking queries and report generation, to assess the overall user experience.

**Discussion**: The system demonstrated fast response times, with **booking queries processed in under 1 second** on average, ensuring that staff can quickly provide guests with accurate availability information. **Reports were generated within 3–4 seconds**, providing staff with timely insights into operations. These quick response times enhance overall efficiency and contribute to a smooth user experience for hotel staff managing daily tasks.

### 4.4.3 Error Handling

The system's approach to handling errors, such as invalid logins or failed bookings, was reviewed to ensure a robust user experience.

**Discussion**: The system effectively handles errors by providing **clear and detailed error messages**. For example, if a login attempt fails due to incorrect credentials, the system promptly alerts the user with an appropriate message. Similarly, during the booking process, the system flags conflicts like **double bookings**, enabling staff to resolve these issues quickly. This proactive error handling ensures that staff can swiftly address problems, preventing disruptions to daily operations and maintaining smooth hotel management processes.

### ****4.5 User Feedback****

#### 4.5.1 Ease of Use

**Discussion**: Staff reported that the system was **intuitive** and easy to navigate, requiring minimal training to get accustomed to its features. They appreciated the **clean user interface** and smooth workflows, particularly when managing tasks such as bookings and payments. The simplicity of the system allowed staff to focus on their duties without getting bogged down by complex processes, enhancing overall productivity.

#### 4.5.2 Efficiency Gains

**Discussion**: Staff noted **significant improvements** in the speed and efficiency of handling key hotel operations, such as **reservations, customer management**, and **payment processing**. The system’s ability to automate routine tasks, like report generation and inventory alerts, was particularly helpful in reducing the manual workload. These enhancements led to quicker task completion and allowed staff to focus on delivering better guest services.

#### 4.5.3 Suggestions for Improvement

**Discussion**: Some staff suggested improvements to the **notification system**, specifically requesting enhanced alerts for **upcoming check-ins and check-outs**. These additional notifications would further streamline guest management by allowing staff to better prepare for guest arrivals and departures, ensuring smoother transitions and improving the overall guest experience.

### ****4.6 Challenges Encountered****

#### 4.6.1 Technical Challenges

**Discussion**: Integrating external payment APIs posed some initial difficulties, particularly around ensuring secure transactions. These challenges were addressed by selecting a well-documented payment gateway that facilitated reliable and secure payment processing. Additionally, minor issues with **database optimization** were identified and resolved during the testing phase, ensuring the system’s stability and performance.

#### 4.6.2 Workflow Challenges

**Discussion**: Aligning the system’s functionalities with the actual workflows of the hotel presented several challenges. These issues were mitigated through **regular feedback from hotel staff** during the testing phase, which was instrumental in refining the user interface and aligning it more closely with operational needs. This iterative approach helped ensure that the system met the practical requirements of hotel staff, enhancing usability and efficiency.

#### 4.6.3 Performance Optimization

**Discussion**: Initial testing revealed that certain modules, such as **report generation**, experienced performance issues under heavy load. To address these concerns, efforts were focused on **optimizing the database** and implementing **caching** for frequently used queries. These improvements significantly enhanced the system’s performance, ensuring faster and more efficient operation even during peak usage times.

***CHAPTER FIVE: RECOMMENDATIONS AND CONCLUSION***

Our team has successfully built a functional and responsive website using HTML, CSS, and React. We started by structuring the website with HTML, ensuring a solid foundation. CSS was then used to style the elements, creating an aesthetically pleasing and user-friendly interface.Finally, we integrated React to add interactivity and manage the website's state efficiently,resulting in a dynamic and seamless user experience.

Despite our achievements, we encountered several challenges along the way. One major difficulty was ensuring cross-browser compatibility, as different browsers rendered our styles and components differently. Additionally, managing the state in React became complex as the project grew, requiring us to implement state management libraries . Debugging and optimizing-performance also posed significant hurdles, especially when dealing with large datasets and complex components.

For future studies, we recommend focusing on advanced CSS techniques and frameworks like Tailwind CSS to streamline styling processes. Additionally, delving deeper into React's ecosystem, including hooks and context API, can enhance state management and component re-usability. Exploring server-side rendering with Next.js could also improve performance and SEO.Continuous learning and staying updated with the latest web development trends and best practices will be crucial for creating more sophisticated and efficient websites.